SESSION

SOFTWARE ENGINEERING AND WEB BASED SYSTEMS + OBJECT ORIENTED TECHNOLOGY + SOA

Chair(s)

TBA

A Web-based Visual Analytic System for Understanding the Structure of Community Land Model

Yang Xu [1], Dali Wang [2] *, Tomislav Janjusic [3], Xiaofeng Xu [2]

[1] Department of Geography
University of Tennessee, Knoxville
TN 37996, USA
yxu30@utk.edu

[2] Climate Change Science Institute
Oak Ridge National Laboratory
TN 37831, USA
{wangd, xux4}@ornl.gov

[3] Computer Science and Mathematics Division
Oak Ridge National Laboratory
TN 37831, USA
janjusict@ornl.gov

* Corresponding author, Tel +1 8652418679, Fax +1 865 5749501. Email: wangd@ornl.gov

Abstract - *Development of high fidelity earth system models is important to the understanding of earth system science. Along with several decades of active developments, the complexity of the model's software structure became a barrier that hinders model interpretation and further improvements. In this paper, a web-based visual analytic system is introduced to better understand the software structure of Community Land Model (CLM) within an earth system modeling framework. First, the software structure is decomposed from source codes and we use a graph structure to represent the interrelationships among different CLM components. Second, a web-based front end is developed to demonstrate the CLM software structure in a visual analytical context. Finally, we present a pilot case study to discuss how an improved understanding of CLM software structure can be achieved from three different perspectives, namely CLM structure overview, visualization of submodel structure and CLM inter-version comparison. We believe the approaches and visualization tools can be beneficial to CLM model interpretation and improvements as well as other large-scale modeling systems across different research domains.*

Keywords - Community Earth System Model, Community Land Model, Software Structure Decomposition, Graph Visualization.

1 Introduction

Researchers have made great progress over the past decades in developing high fidelity earth system models [1]. The Community Earth System Model (CESM) is one of the leading earth system models funded by National Science Foundation (NSF) and U.S. Department of Energy (DOE). The Community Land Model (CLM) is the land model of CESM that simulates surface energy, water, carbon, and nitrogen fluxes and state variables for the land surfaces [2-4]. The model formalizes and quantifies concepts of ecological climatology under an interdisciplinary framework to understand how natural and human changes in vegetation affect climate. As a scientific application for the earth system simulation, it is important to get the fundamental processes correct [5]. This requires a good understanding of CLM ecosystem functions as well as the interplay among them within the context of ecosystem science.

The CLM contains several submodels related to land biogeophysics, biogeochemistry, hydrologic cycle, human dimensions and ecosystem dynamics. The structure of each submodel is generally organized by software modules or subroutines based on natural system functions such as carbon-nitrogen cycles, soil temperature, hydrology and photosynthesis [6]. Each module or subroutine interacts with a list of variables which are globally accessible or subroutine explicit. Several efforts have been made to better understand CLM and the ecosystem processes through software structure profiling [6], functional unit testing [7] and memory pattern analysis [8]. The whole CLM modeling system consists of more than 1800 source files and over 350,000 lines of source code. New CLM software analysis methods are much needed for rapid model interpretation and improvements.

In this paper, a web-based visual analytic system is introduced to gain an improved understanding of CLM software structure. First, we decompose the CLM software from source codes and propose a CLM graph structure that summarizes the interrelationships among all the function calls and variables. Second, a web-based front end with three different views is developed to demonstrate the CLM software structure in a visual analytical context. A pilot case study is then presented to gain insights into the structure using the three views, namely CLM structure overview, visualization of submodel structure, and CLM inter-version comparison. We believe the visualization tools can be beneficial to the understanding of CLM software structure. The approaches can also be applied in other large-scale modeling systems across different research domains.

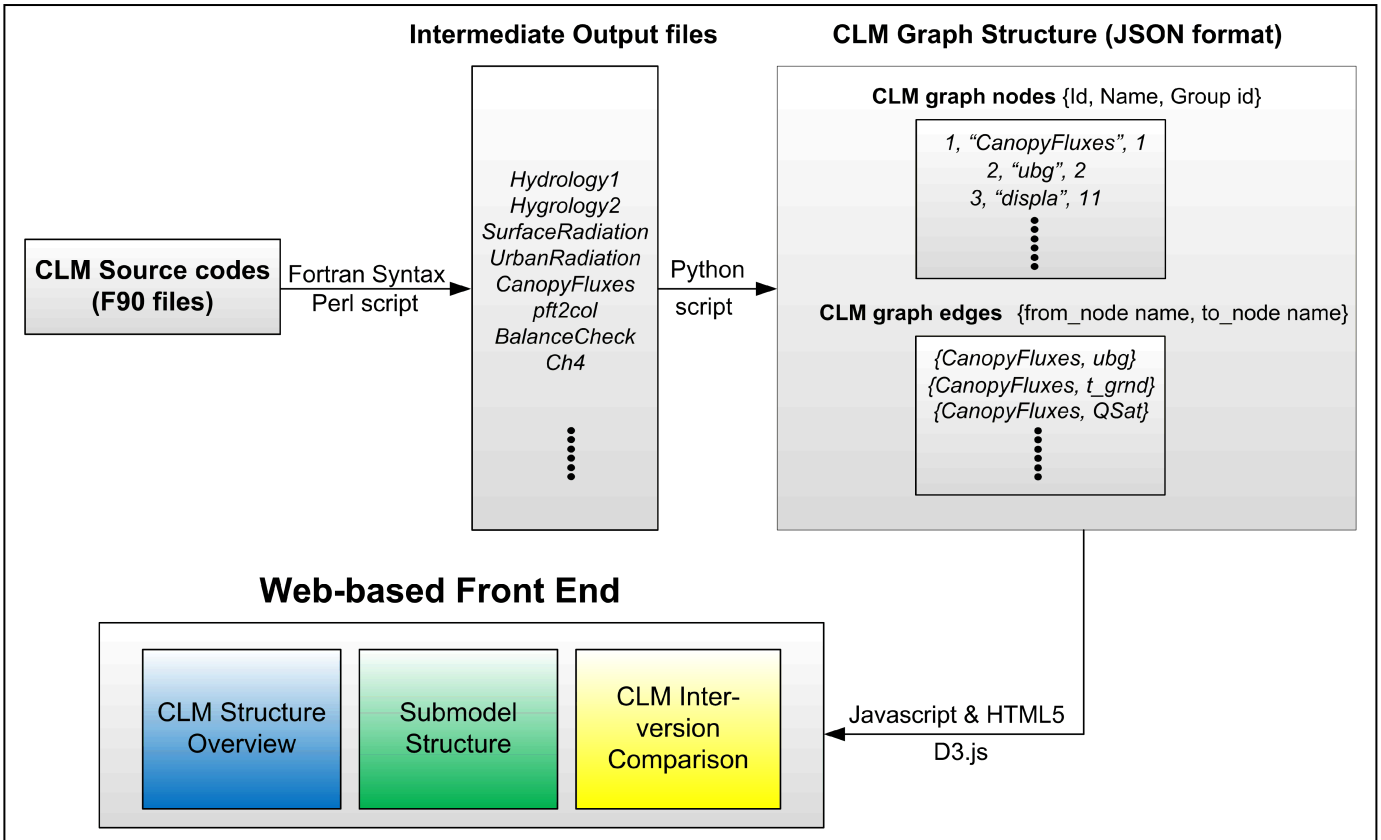


Fig. 1. Work flow and key components of the web-based visual analytic system for understanding the structure of Community Land Model (CLM)

2 Methodology and key components

In this section, we introduce the key components and work flow of our web-based visual analytic system. As shown in Fig. 1, a CLM Fortran-syntax specific Perl script was developed to decompose the CLM software structure into tokens of function calls, subroutine explicit parameters and global variables. Definition of tokens will be further explained in section 2.1. Then, a Python script builds a CLM graph structure, which summarizes the interrelationships among all the tokens. Finally, the graph structure is visualized in the web-based front end using Javascript and D3.js (http://d3js.org).

2.1 Decomposition of CLM structure

Understanding complex codes such as CLM undoubtedly requires tools to facilitate code decomposition into simpler forms. This allows users to use visualization tools to further understand the code structure. For this purpose, we developed a CLM Fortran-syntax specific Perl script that categorizes key variables and data structures into tokens. Herein, we refer to a token as any source-code identified function call or variable, which includes name of subroutines, global visible variables, as well as all the variables used in subroutine definitions *(subroutine-in, subroutine out)*. *Subroutine-in* variables are all tokens identified in the subroutine's signature*. Subroutine-out* variables are a subset that was identified to be written to, i.e. these tokens were used to store a value. Globally visible variables are identified using the pointer assignment syntax during source-code scanning. This means that any token found in the source-code line that adheres to the general pointer assignment syntax is treated as globally visible variable. We further break this category into *Read-only, Write-only*, or *Modified* variables. Specifically, during scanning the script stores any pointer to derived member values into a hash of tokens. The source code lines are decomposed into left-hand (lHand) and right-hand (rHand) statements and further broken down if more assignments are present. Every token found on the lHand side is a write category and similarly every token on the rHand side is a read category token. If a token falls into both categories, we will assign that token into the modified category. There are, of course, special cases that require further statement breakdown using special-case rules. For example, statements that use pointers to access other derived types are often found in the lHand/rHand statement syntax, thus the script will decompose tokens to identify the correct category for the globally visible variable. In addition, tokens found in source-code statements that are identified by special-keywords (e.g., call) are used to build static call-graphs, which is a list of calle subroutine names originating from the current caller subroutine. The Perl scanning process outputs a list of files named after the subroutine's name. Each file records the variables and function calls (Calle Subroutines) that a particular subroutine has accessed. Table 1 gives an example of the output file of *CanopyFluxes* subroutine.

Table 1. Tokens of Cannopyfluxes Subroutine

Category	**Tokens**
Subroutine-In	*ubg, ubc, lbg, lbp, num_nolakep,* ......
Subroutine-Out	*Null*
Global Read Only	*t_grnd, psnsun_wc, alphapsnsun, psnsun,* ......
Global Write Only	*cgrnd, psnsun, rb1, ulrad, dlrad,* ......
Global Modified	*displa, rc13_psnsun, z0qv, z0hv,* ......
Global None	*watopt, watdry*
Function Calls	*QSat, FrictionVelocity, Photosynthesis,* ......

2.2 Graph construction of CLM software and submodels

Based on the output files generated by the Perl script, we developed a Python script to organize the CLM components into a graph structure with nodes and edges. The nodes refer to all the identified tokens, and the edges are used to describe how these tokens access or are accessed by others. We use this graph structure to summarize the interrelationships among all the function calls, subroutine explicit parameters and global variables. As described in section 1, CLM consists of several submodels and each submodel is usually organized by particular subroutines. In order to incorporate this information into the graph, the Python script also records which submodel each subroutine belongs to. Then this graph structure is used in the web-based front end to facilitate the understanding of the CLM structure from multiple perspectives. For example, users could get an overall idea of CLM structure by exploring the submodels and subroutines contained, or look into the structure of particular CLM submodels. The graph structure can also be compared across different CLM versions.

Due to the complexity of CLM software, some tokens (nodes) belong to multiple categories in the modeling context. For example, in Table 1, the token *displa* marked as *Global Modified* variable for the subroutine *CanopyFluxes* while its category becomes *Subroutine-in* for another subroutine *FrictionVelocity*. In order to maintain this information, we generate a CLM node group list that enumerates all possible combinations of the token categories. As shown in Table 2, each Group id corresponds to a particular combination of token categories. By introducing the list, we are able to label each node with group information during graph construction. For example, as shown in Table 1, all *Function Calls* (e.g., *Qsat*) after graph construction will have a Group id of 1. The *Subroutine-in* variables (e.g., *ubg*) will have a Group id of 2. While variables like *displa* as described above will have a Group id of 11. The group information can be used to understand the CLM software structure with respect to token category, i.e. its function.

Table 2. CLM Node Group Information

Group id	**Combination of Token Categories**
1	*Function Calls*
2	*Subroutine-in*
3	*Subroutine-out*
4	*Global Read-only*
5	*Global Write-only*
6	*Global Modified*
7	*Global-None*
8	*Subroutine-in & Subroutine-out*
9	*Subroutine-in & Global Read-only*
10	*Subroutine-in & Global Write-only*
11	*Subroutine-in & Global Modified*
12	*Subroutine-in & Global-None*
⋮	⋮

2.3 Web-based front end based on Javascript and D3.js

One of the important components in our system is the web-based front end, which is designed to facilitate the exploration and investigation of CLM software structure in a visual analytical context. The web-based front end has three major views: CLM structure overview, visualization of submodel structure and CLM inter-version comparison. The CLM structure overview aims to provide users with an overall picture about different CLM software versions and submodels. The visualization of submodel structure summarizes the inter-relationships among all the function calls, subroutine explicit parameters and global variables related to that submodel (e.g., *CanopyFluxes* shown in the next section). The inter-version comparison is used to demonstrate what changes and improvements have been made from one CLM software version to another. The web-based front end is developed based on Javascript and D3.js (http://d3js.org). D3.js is a Javascript library which allows developers to bind their data to a Document Object Model (DOM) and then transfer the data information into interactive visualizations. As shown in Fig. 1, the Python scanning process generates a list of node and edge files in JSON format (http://json.org). These files that record the CLM software structures are used to create interactive visualizations using Javascript and D3.js.

3 A pilot case study

As we mentioned before, visualizing and analyzing the software structure of large-scale modeling system such as CLM is very important to model interpretation and provides opportunities for further improvements of model structures. In this section, a pilot case study is introduced to describe how an improved understanding of CLM software structure can be achieved with the web-based visual analytic system. First, a collapsible tree is used to demonstrate the overall structure of CLM software from a hierarchical perspective. This effort will allow users to explore the submodels and subroutines included in each CLM version. Second, a directed graph is used to visualize the *CanopyFluxes* submodel within CLM. This effort

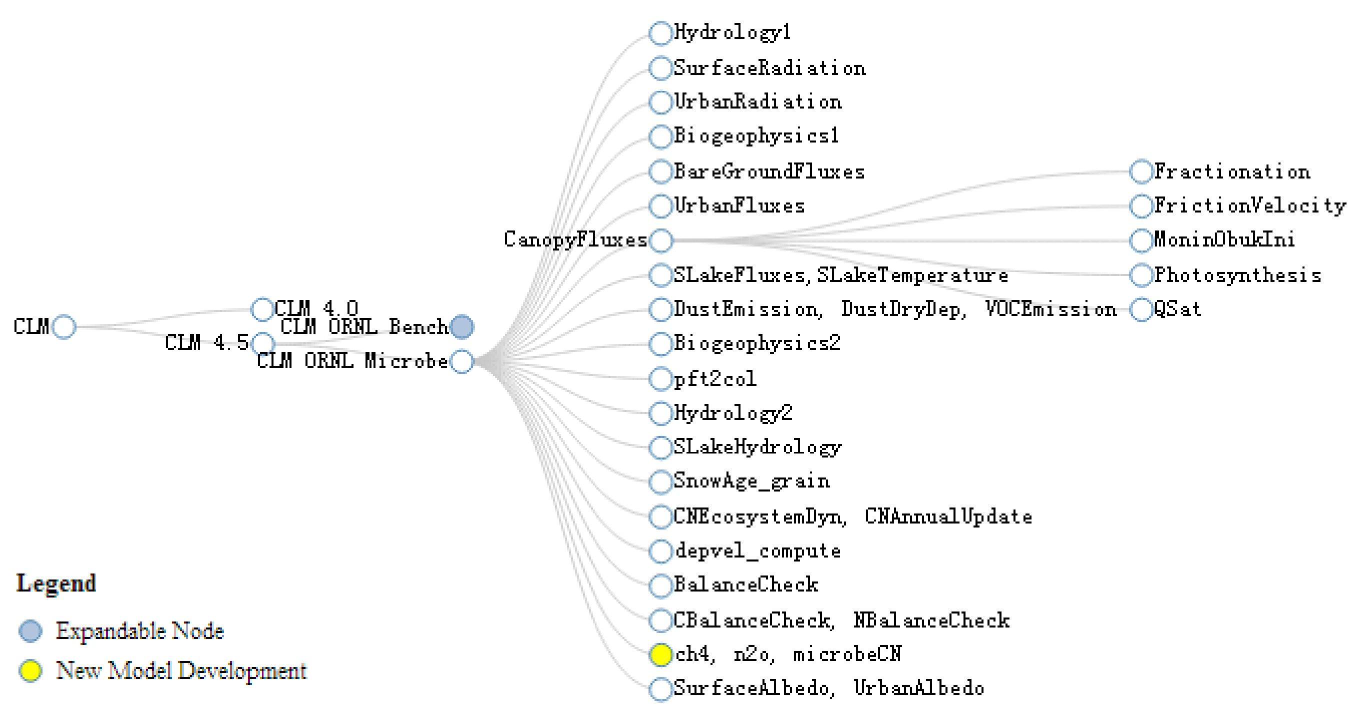


Fig. 2. Visualization of CLM software structure using a hierarchical collapsible tree. The nodes can be clicked to expand or collapse. The nodes in light blue denote the ones that can be expanded. The nodes in yellow represent new model development (e.g., ch4, n2o, microbeCN) based on the bench version (e..g, CLM ORNL Bench).

allows users to further look into the sturcture of particular submodels based on their research interests. Finally, we present our case study for the CLM inter-version comparison (*CLM ORNL Bench* vs. *CLM ORNL Microbe*). This effort enables users to trace the changes between two CLM versions.

3.1 CLM structure overview

The CLM model has several public releases such as CLM 4.0 and CLM 4.5. At Oak Ridge National Laboratory (ORNL), we have our own code repository, which use the official release CLM 4.5 as our bench case. Based on that, several new modules (e.g., the Microbe module [9]) have been developed. In our web-based visual analytic system, a collapsible tree is used to demonstrate the CLM software structure from a hierarchical perspective. As mentioned in section 2.1, after the decomposition of CLM structure, we generate a list of files named after the subroutine's name, which can be used to construct the overview software structure. Fig. 2 shows a hierarchical tree, which allows users to explore the CLM software structure by expanding or collapsing particular nodes. For example, when users click the node "*CLM*", it will expand and show several CLM major release such as "*CLM 4.0*" and "*CLM 4.5*". The node "*CLM 4.5*" can be then expanded to view different versions of CLM source codes such as "*CLM ORNL Bench*" and "*CLM ORNL Microbe*". Each CLM2 version can be further expanded to view its submodels (e.g., "*CanopyFluxes*") as well as the corresponding subroutines. The visualization also highlights the nodes in yellow to illustrate the submodels which are newly developed based on the CLM bench version. For example, the node "*ch4, n2o, microbeCN*" in Fig. 2 shows that this submodel is newly incorporated in "*CLM ORNL Microbe*" as compared with its bench version "*CLM ORNL Bench*".

The hierarchical visualization provides users with an overall picture of CLM software release, submodel components as well as the model improvements. The interactive visualization can be found at (http://web.ornl.gov/~7xw/CLM_Overview.html).

3.2 Visualization of submodel structure

Visualizing the structure of CLM submodels can be useful when users want to understand particular CLM components at the micro level. As described in section 2.2, the Python script generates a comprehensive graph structure recording all the submodels and their corresponding subroutines. Within each subroutine, the interrelationships among all the function calls and variables are recorded. Hence, the web-based front end is able to visualize a subset of the whole graph in order to demonstrate the structure of a particular CLM submodel.

Fig. 3 shows the structure of *CanopyFluxes* submodel within CLM. Nodes with different colors and sizes are used to denote the types of the tokens. Nodes with bigger size and the color of yellow stand for all the function calls (subroutines) for the *CanopyFluxes* submodel. The nodes with smaller size denote all the variables and among these variables: (1) green nodes stand for subroutine explicit parameters; (2) blue nodes stand for global variables; (3) the nodes in red denote the ones that are used as both subroutine explicit and global variables.

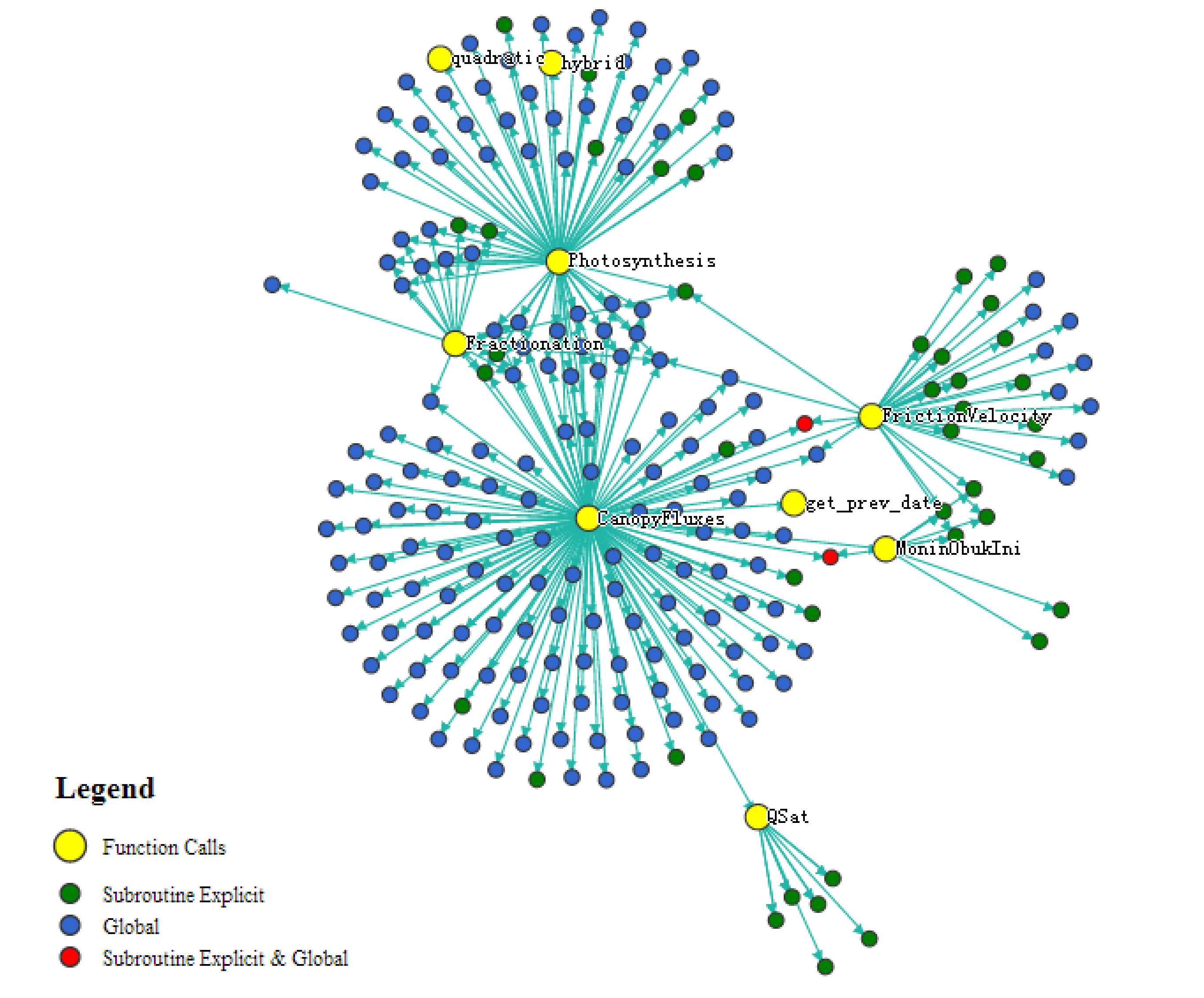


Fig. 3. Visualization of *CanopyFluxes* submodel of CLM ORNL Microbe. Nodes in yellow stand for the function calls (subroutines). Nodes in green denote subroutine explicit variables. Nodes in blue represent all the global variables. While nodes in red denote the variables which are used as both subroutine explicit/global variables. The edges are used to describe how the variables are accessed by particular function call(s).

The visualization using directed graphs provide users with an intuitive way of investigating how function calls and variables access or are accessed by others within the context of particular submodel. By exploring the submodel structure, users could better understand how the tokens are connected together as well as the specific role(s) that each of them is playing. As shown in Fig. 3, several red nodes exist in the structure of *CanopyFluxes*, which means that these variables serve as both global variables for the submodel and explicit parameters for the function calls (subroutines). The information allows users to further explore the scientific meanings of these variables.

The case study of this visualization can be found at (http://web.ornl.gov/~7xw/CanopyFluxes/CanopyFluxes.html) . When a user puts the mouse over a certain node, the name as well as the node group id will pop up. The group id, as described in Table 2, offers users detailed information about the specific category of an token. For example, the pop up information will help users to distinguish if a global variable belongs to the category of *Read-only, Write-only* or *Modified*. Our web page contains a hyperlink (i.e., "View Group Information") which leads to a file similar to Table 2 whenever users want to view the group information of the tokens.

3.3 CLM inter-version comparison

The CLM is a community model which is open for any contributions and usages across the scientific community. For example, the current release of CLM 4.5 (i.e., *CLM ORNL Bench* in our case study) consists of four key components: biogeophysics, hydrologic cycle, biogeochemistry and dynamic vegetation. For the biogeochemistry component in the CLM, the carbon and nitrogen cycling in the soil and vegetation under the influence of environmental factors are simulated. The microbial controls on carbon and nitrogen processes are implicitly represented as a few empirical equations, which are one of the primary uncertainties for model improvements targeting better predicting biogeochemistry-climate feedbacks. Thus, a more advanced model with explicit representation of microbial processes which contributes to the soil biogeochemical processes is needed. Targeting this need, a new modeling structure, *CLM ORNL Microbe*, is developed to improve the *CLM Bench* version by [9, 10].

Our web-based front end uses a rendered directed graph to describe the changes between two CLM versions. As described in section 2, the CLM graph structure (nodes and edges) that summarizes the interrelationships among all the tokens is generated using Python script. By comparing the graph structures between two different versions, we are able to uncover the changes in term of: (1) which function calls and variables are newly added, modified or no longer existing; (2) the interplay between function calls and variables that are newly added or no long existing. Fig. 4 shows an example of our web-based visualization for comparing *CLM ORNL Bench* and *CLM ORNL Microbe*. The *CLM ORNL Bench* is chosen as the base version. The blue nodes denote the newly added function calls and the yellow nodes stand for the function calls that are modified. For example, we can see several new function calls that are added into the *CLM ORNL Microbe* such as *n2o*, *microbeCN* and *microbeRest*.

The nodes with smaller size denote all the variables. We use red to represent newly added variables and green to represent the ones that are modified. We also use solid and dash lines to represent the changing relationships among all the tokens through the two CLM software versions. For example, as shown in Fig. 4, there are many solid as well as dashed links associated with the function call *ch4*. It means that as compared with the based version, the *ch4* submodel for *CLM ORNL Microbe* is modified to access some new variables (red nodes connected with solid lines). Meanwhile, some of the variables (green nodes) and function calls (yellow nodes) are connected

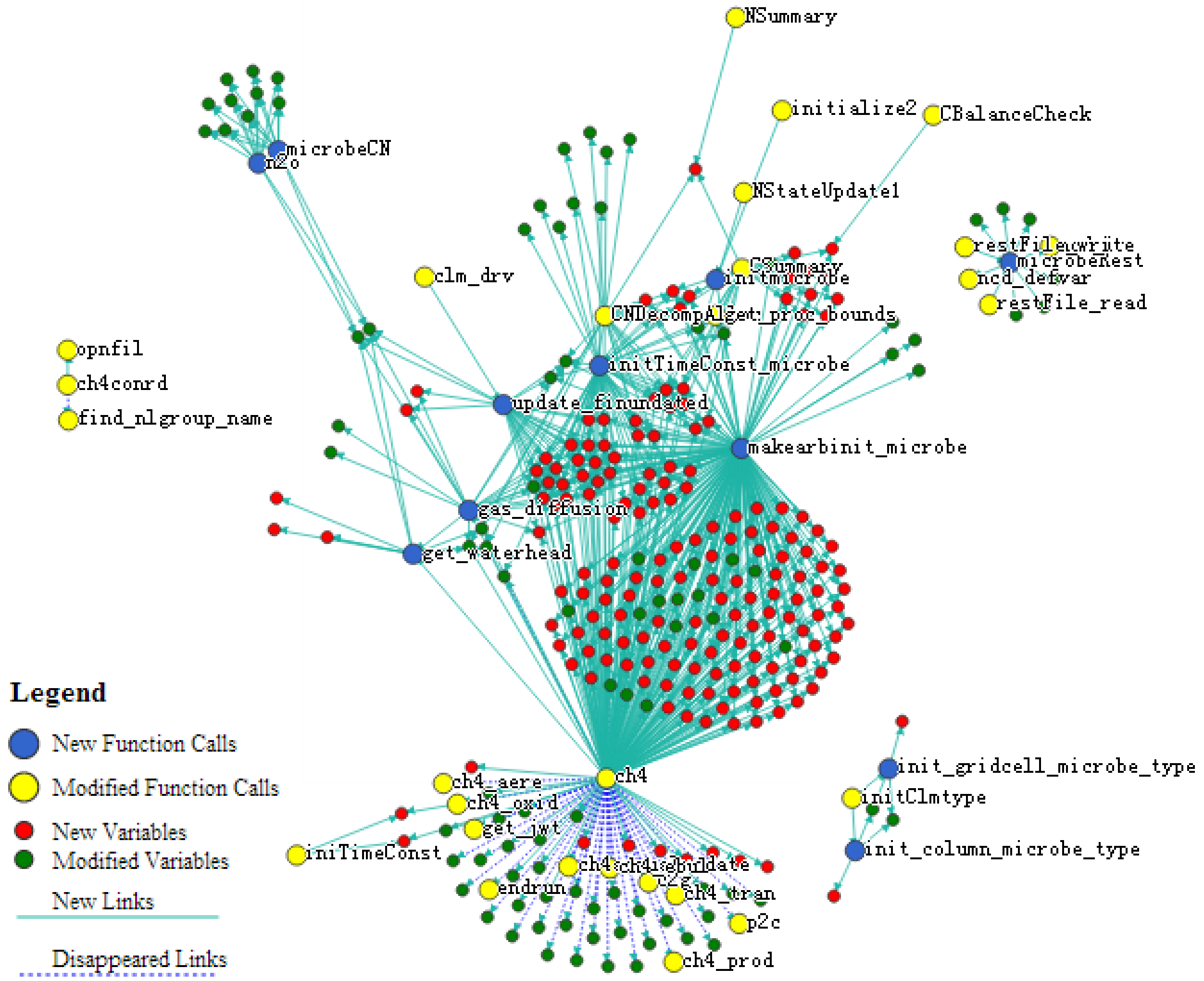


Fig. 4. Graph visualization of CLM inter-version comparison (*CLM ORNL Bench* vs. *CLM ORNL Microbe*). The *CLM ORNL Bench* is chosen as the base version. Blue nodes represent newly added function calls and the yellow nodes represent the modified function calls. Red nodes denote the newly added variables and the onw in green stand for modified variables. The links rendered with solid and dash lines are used to represent newly added and disappeared links respectively.

to *ch4* with dash lines, which means that those variables and function calls are accessed by *ch4* in the *CLM ORNL Bench*, but are no longer accessed by *ch4* in *CLM ORNL Microbe*. The graph structure allows users to easily trace the changes between two CLM software versions in a visual analytical context. The web-based visualization for this case study can be found at (http://web.ornl.gov/~7xw/CLM_Microbe_Comparison/CLM_Comparison.html).

4 Conclusions and future work

In this paper, we present our approaches for better understanding the structure of Community Land Model within the Earth System Modeling framework. A web-based visual analytic system is developed to allow users to gain insights into software and data structure from different perspectives. The CLM structure overview provides an overall picture of different CLM release and the submodels. It helps users to explore major components as well as new module development for a particular CLM version (e.g., *CLM ORNL Microbe*). The system also enables users to look into the structure of a particular submodel by visualizing the interrelationships among all the function calls and variables. Moreover, a deeper understanding can be obtained by exploring their names and categories, which is important for model interpretation and further improvements. The CLM inter-version comparison allows users to trace the changes between two CLM versions in a visual analytical context. Users can easily identify the function calls and variables which are added, modified or removed from one CLM version to the other. We believe the approaches and visualization tools can be beneficial to the understanding of CLM software structure as well as other large-scale modeling systems across different research domains.

The future work will focus on two directions. First we will develop an online database system hosting software structure as well as performance data from other advanced tools such as Vampir [11] and Valgrind (www.valgrind.org). Users will be able to query the database to get detailed information of CLM submodels, function calls and variables. Meanwhile, visualizations of software and submodel structures will be generated on the fly based on the customized queries. Second, we will incorporate a web-based functional testing platform over the cloud computing infrastructure to facilitate the understanding of CLM ecosystem processes.

Acknowledgment

This research was partially funded by the U.S. Department of Energy (DOE), Office of Science, Biological and Environmental Research (BER). This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the Department of Energy under Contract DE-AC05-00OR22725. Yang Xu's work is partially supported through a subcontract to the University of Tennessee, Knoxville. Oak Ridge National Laboratory is managed by UT-Battelle LLC for the Department of Energy under contract DE-AC05-00OR22725.

References

[1] W.M. Washington and C.L. Parkinson, *An Introduction to Three-Dimensional Climate Modeling*, 2nd ed. University Science Books, 2005.

[2] G.B. Bonan, "The Land Surface Climatology of the NCAR Land Surface Model Coupled to the NCAR Community Climate Model", *J. of Climate*. vol. 11(6), pp.1307-1326, 1998.

[3] R.E. Dickinson, K.W. Oleson, G. Bonan, F. Hoffman, P. Thornton, M. Vertenstein, et al. The Community Land Model and Its Climate Statistics as a Component of the Community Climate System Model, *J. of Climate*. Vol. 19(11), pp.2302-2324, 2006.

[4] K. Oleson, D. Lawrence, B. Gordon, M. Flanner, E. Kluzek, J. Peter , et al. "Technical Description of Version 4.0 of the Community Land Model (CLM)", 2010.

[5] D. Wang, D. Ricciuto, W. Post and M. Berry. "Terrestrial Ecosystem Carbon Modeling", *Encyclopedia for parallel Computing*, pp.2034-2039, 2011.

[6] D. Wang, J. Schuchart, T. Janjusic, F. Winkler and Y. Xu, Toward Better Understanding of the Community Land Model within the Earth System Modeling Framework. *International Conference on Computational Science*. 2014. In press.

[7] D. Wang, Y. Xu, P. Thornton, A. King, C. Steed, L. Gu and J. Schuchart. "A Functional Test Platform for the Community Land Model", Environment Modeling & Software, vol. 55, pp. 25-31, 2014.

[8] T. Janjusic, K. Kavi and B. Potter. "A Memory Analysis Tool", *Procedia Computer Science*. Vol. 4, pp. 2058-2067. 2011.

[9] X. Xu, J.P. Schimel, P. Thornton, X. Song, F. Yuan, S. Goswami. "Substrate and Environmental Controls on Microbial Assimilation of Soil Organic Carbon: A Framework for Earth System Models". *Ecology Letters*, DOI: 10.1111/ele.12254. 2014.

[10] X. Xu, D.A. Elias, D.E. Graham, T.J. Phelps, S.L. Carrol and P. Thornton. "A Microbial Functional Group Based Model for Simulating CO_2 and CH_4 Dynamics". unpublished.

[11] M.S. Müller, A. Knüpfer, M. Jurenz, M. Lieber, H. Brunst, H. Mix, et al. "Developing Scalabe Applications with Vampir, VampirServer and VampirTrace". *Parallel Computing: Architectures, Algorithms and Applications*. Vol. 15, pp. 637–644. 2008

Adding Semantic Web Technology to the Object-Oriented Method for Interoperability

P. Anisetty[1], P.Young[2]
[1]Acxiom Corporation, Conway, AR, USA
[2]Computer Science Department, University of Central Arkansas, Conway, AR, USA

Abstract—*Computer communications advances, functional similarities in related systems, and enhanced information description mechanisms have led to the widespread interconnection of existing systems to meet current and future system requirements in such diverse fields as healthcare, e-commerce, and military applications. Full realization of the potential offered by such interconnections can only be achieved if systems are fully interoperable. Interoperability among independently developed heterogeneous systems is difficult to achieve: systems often have different architectures, hardware platforms, operating systems, host languages and data models.*

The Object-Oriented Method for Interoperability (OOMI) introduced by Young offers the capability for resolving modeling differences among heterogeneous systems, thereby enabling system interoperation. The OOMI resolves modeling differences by first identifying corresponding entities among systems using semantic and syntactic correlation approaches involving keyword search and neural network techniques. This paper describes how Semantic Web technologies can be used to improve the accuracy of the OOMI correlation methodology by adding Web Ontology Language (OWL) and reasoning capabilities to the OOMI model.

Keywords: Interoperability, Object-Oriented, Semantic Web, Information Exchange, XML, Web Ontology Language

1 Introduction

Until fairly recently, most software-intensive systems were developed as special-purpose, stand-alone applications. Recent advances in computer communications technology, the recognition of common areas of functionality in related systems, and an increased awareness of how enhanced information access can lead to improved capability have led to an increased interest in integrating current stand-alone systems to meet future system requirements.

One example of where independently developed systems might be interconnected in the healthcare domain would be the integration of your primary care physician's healthcare management system with the local hospital's Magnetic Resonance Imaging (MRI) system to provide your physician with near-real-time results of the scan he ordered on your knee. In business, connecting a company's inventory management system with its retail outlet point-of-sale (POS) system can result in reduced inventory requirements and faster re-supply of needed items. Finally, a military application might connect a squadron's mission planning suite with the theatre intelligence center to provide the latest location of the targeted adversary.

In addition to being able to exchange information, full system interoperability requires that systems share tasks as well as information [1, 2]. Such an interconnected collection of independently developed heterogeneous systems or components is referred to as a system federation. This differs from the concept of an integrated system, where homogeneous components are connected by a development team that has common objectives and shares a common view of the problem environment being modeled.

Attempts to interconnect independently developed systems that were never intended to interoperate are often faced with difficulty. Typically such existing systems were developed without any of the constructs normally included when forward-fitting a system to support interoperability. Any modification to existing systems to enable them to interoperate is costly and time-consuming. Therefore, methodologies that will enable systems to interoperate without requiring modification to existing software are highly desirable.

In the past, establishing communication between different software systems involved manually resolving differences for each system interface, which required time and cost to customize. In the Object-Oriented Method for Interoperability (OOMI) [3] Young's first step in creating an interoperable federation of independently developed heterogeneous systems or components was to develop an object model of the entities involved in the interoperation between systems in the federation, termed a Federation Interoperability Object Model (FIOM) under the OOMI, shown in Fig. 1. Then, the FIOM is used by a translator at run-time to reconcile any representational differences among a component's attributes and methods, as seen in Fig. 2. The translator serves as an intermediary between component systems. It can be implemented as part of a software wrapper enveloping each system, as shown in Fig. 2, or as a stand-alone module that lies between interconnected systems.

Correspondences are established among entities shared by systems in the federation in order to enable the resolution of any differences that might be found. The current method used to establish correspondences among shared entities uses a combined keyword and neural network search. Inaccuracies tied to each of these correlation approaches have led to the introduction of Semantic Web technologies to assist in the cor-

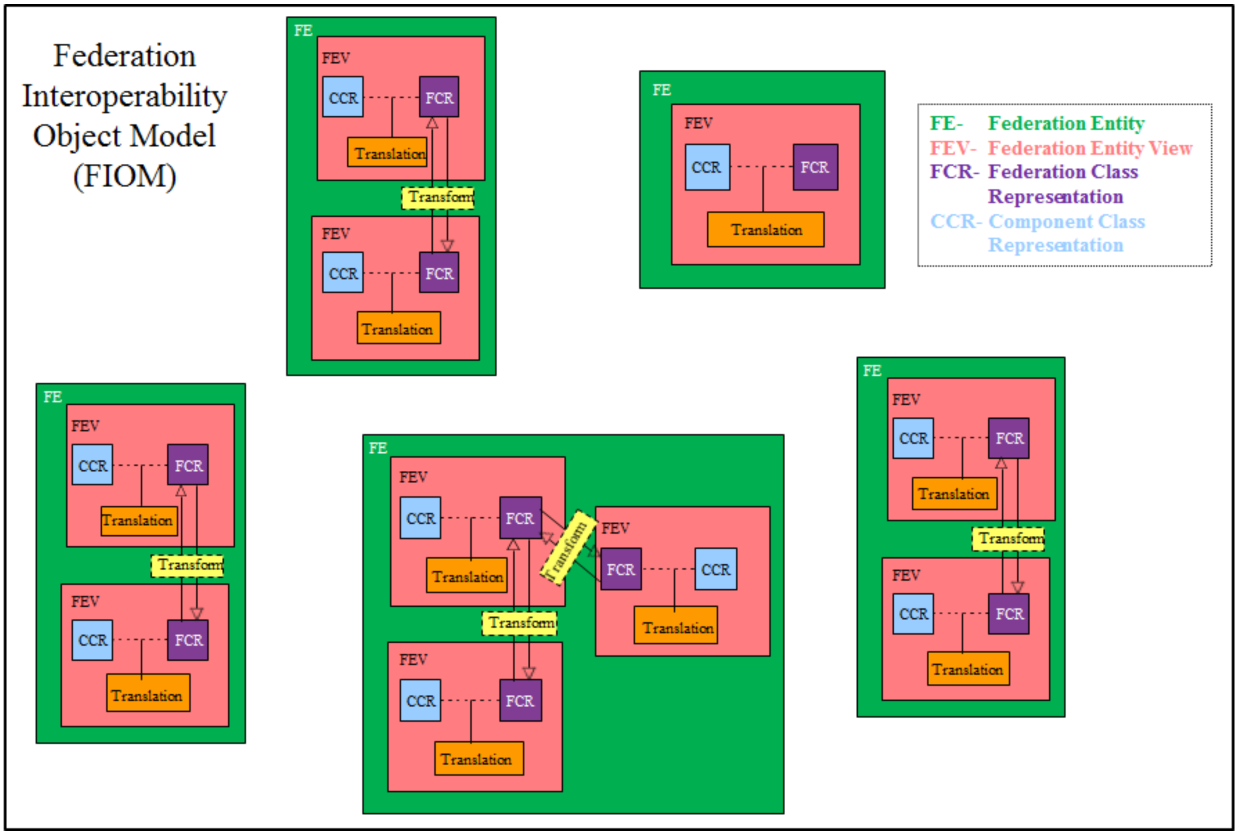

Federation Interoperability Object Model (FIOM)
FE
FEV
CCR
FCR
Translation
Transform
FE- Federation Entity
FEV- Federation Entity View
FCR- Federation Class Representation
CCR- Component Class Representation

Fig. 1. Federation Interoperability Object Model (FIOM)

relation process and to improve search precision and recall. This has resulted in a modification of the methodology used for creating an FIOM for a federation of component systems and a projected improvement in the correlation process.

The remainder of the paper first provides background information on the types and categories of heterogeneities that can be found among systems, how the OOMI utilizes those categories in creating the FIOM, how the FIOM uses correspondences among information shared among systems in a federation to resolve system heterogeneities, and how the Semantic Web can be utilized to improve the ability to establish correspondences among shared information. The paper next discusses how Semantic Web technology can be integrated into the FIOM, comparing the original OOMI correlation model with the Semantic Web-OOMI correlation model. Status of the Semantic Web-OOMI correlation model implementation is provided next, followed by a summary of insights gained from adding Semantic Web capabilities to the OOMI, and finally a recommendation for future work.

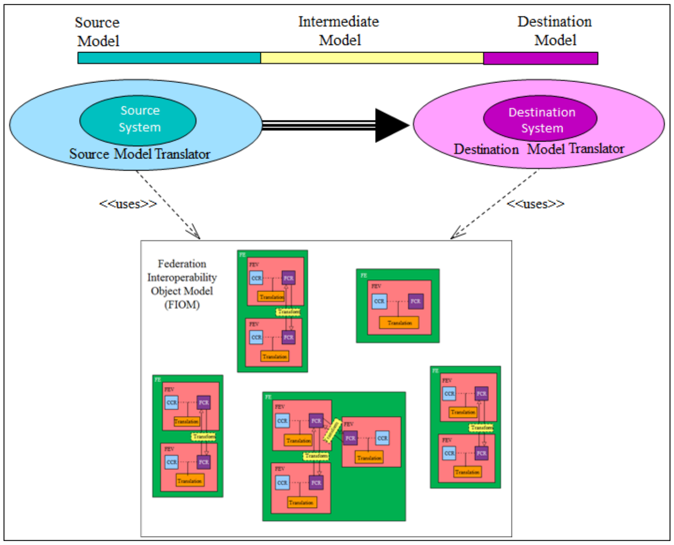


Fig. 2. OOMI Translator-FIOM Interaction

2 Background

2.1 Heterogeneities among systems

When two systems are not designed to interoperate from the start, there can be a number of differences between them that arise when trying to include them in a system federation. There can be differences due to being developed and deployed on different hardware and software architectures, differences in the conceptual models used in their creation, differences in the structure of how data and information are organized, differences in application domain, units of measure, or data type, or differences caused by the existence of synonyms, homonyms, abbreviations or spellings used in the systems' data models. In addition there can be differences in the attributes which are considered important to model about an entity, or in how atomic data elements are aggregated between systems.

Finally there can be disparities in the timeframe or scale used to model an entity [4] [5] [6] [7] [8].

Resolving heterogeneities among systems involves first constructing a model of the external interface of each system involved in the interoperation. Modeling differences among systems can be classified as differences in view or in representation [3]. Differences in view exist when different information is shared about an entity as a result of differences in scope, level of abstraction, or temporal validity. Fig. 3 provides an example of a difference in view where Patient Record A contains information related to the patient's medical history, whereas Patient Record B contains medical billing information. Both model different aspects of a Patient's Record, which is modeled as a Federation Entity (FE) in the FIOM.

Even when two systems share the same view of an entity, representational differences can also exist, such as those caused by different systems of measurement. In Fig. 4, attributes shown in italics in System A are measured using the metric system of measure whereas the same attributes in System C use the U.S. system. Differences in attribute naming, a common occurrence in computing, also results in representational differences among systems.

2.2 Introduction to the Object-Oriented Method for Interoperability (OOMI)

The OOMI [9, 3] has been developed as a means for resolving differences among software systems that were not designed to interoperate in order to enable them to exchange information and share tasks. The foundation for the OOMI is built upon the FIOM, an object model of the information shared by the external interfaces of the systems connected in a system federation.

2.2.1 Federation Interoperability Object Model (FIOM)

The FIOM consists of a number of Federation Entities (FEs), each used to represent an item shared among systems, such as a Patient Record or Lab Report. Entities will normally be shared among systems utilizing some sort of messaging standard, preferably represented using the eXtensible Markup Language (XML). These entities will subsequently be represented as Java classes in the OOMI in order to facilitate functional transformations required to resolve differences in view and representation among entities.

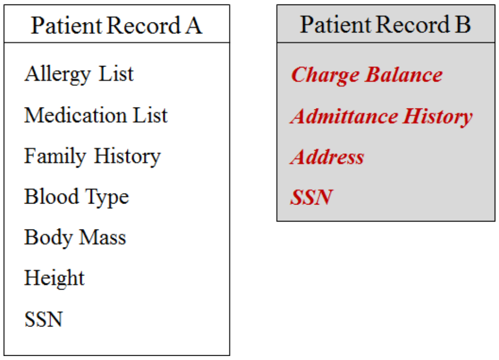


Fig. 3. Differences in View

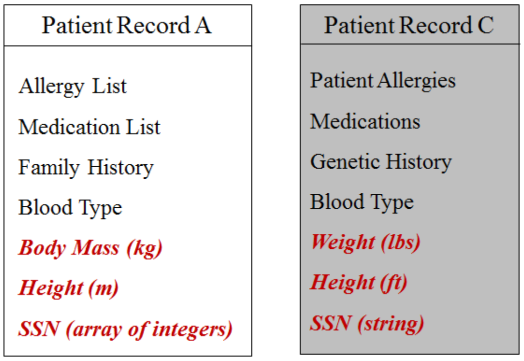


Fig. 4. Differences in Representation

For each FE modeled by the systems in the FIOM, differences in view are captured by one or more Federation Entity Views (FEVs). A Federation Class Representation (FCR) provides a standard representation for each view. Creation of a standard representation for each FEV facilitates the use of a two-step process for resolving differences among software systems. In a two-step process the component representation of one system is first translated into a common standard representation before being converted into the representation used by the second system. Use of a two-step process over the more traditional direct system-to-system translation enables a reduction in the number of translations required from *n*(*n-1*) to *2n* for a federation of *n* systems.

Associated with each FCR are one or more Component Class Representations (CCRs) which are used to provide component system specific representations of a view. Fig. 1 provides an overview of the FIOM's top-level components.

A translation class is provided for each FCR-CCR pair to resolve differences between component system and standard representations of a view. The OOMI includes an Integrated Development Environment (OOMI IDE) which contains a Translation Generator that provides computer aid in creating the translation class in three areas. First, the IDE uses correspondences between a CCR's and FCR's attributes and operations identified by the user to provide a framework for translation definition. Second, it provides facilities for creation and maintenance of a library of pre-defined translation definitions for insertion into this translation framework. Finally, the OOMI IDE provides the user tools to facilitate customization of the translations used for representational difference resolution [3].

In the current OOMI, resolution of differences in view of an entity between component systems is handled by means of a transformation class used by a translator at run time. A component system's view of an entity is represented by the FCR to which it is registered. Multiple FCRs shall be defined for an entity having different views. Each of the views for an entity are related by means of an inheritance hierarchy whereby all FCRs defined for the view extend from a common superclass. Differences in view are resolved using the transformation class through exploitation of the information contained in this FCR inheritance hierarchy using Liskov and Wing's notion of behavioral subtyping [10]. These classes and the relationships among component systems are used by a translator application to resolve system heterogeneities at runtime as previously seen in Fig. 2.

2.2.2 Creating a Model of System Interoperation

The first task in resolving heterogeneities among systems in the OOMI is the creation of the FIOM. In the OOMI, it is assumed that a component system's external interface can be expressed as a series of messages by which information and task requests are exported and imported. These messages represent the entities that are to be shared among systems in the federation. In the OOMI, these messages are further presumed to be in the form of a series of XML messages, the content and composition of which are constrained by a corresponding series of XML Schema documents. Thus, communication between systems is done via XML messaging.

The first step in creating an FIOM in the current OOMI IDE is to take each XML Schema document used to represent a message exported or imported by a particular component system and convert it to an equivalent Java class, referred to in the FIOM as a Component Class Representation (CCR). This CCR is added to a CCR Library to be used by the translator during run-time resolution of system differences.

Either as each CCR is added to the FIOM, or after all CCRs have been created, an FCR is selected which holds the same view as the CCR. In order for two systems to have the same view of an entity, "there must be no difference in scope, level of abstraction, or temporal validity between the two systems' models of the entity … this means that at some level of aggregation each attribute set and each operation set of each system must be in one-to-one correspondence" [3]. Furthermore, "corresponding operations must be behaviorally equivalent for two systems to have the same view of a real-world entity" [3]. This FCR may already exist in the FIOM or if not, it must be created. Creating an FCR can follow either a step-by-step process whereby the FCR is defined and attributes and operations added, or by following a similar procedure used to create a CCR from an XML Schema document. The current implementation of the OOMI IDE provides only the latter method for FCR creation.

2.2.3 Original OOMI Correlation Model

When adding a new component system CCR to a federation, effort must be expended to determine if there are existing Federation Entities and FEV's in the FIOM to which the new system's entities correspond. If corresponding FE's and FEV's are found, translation and transformation classes can be created to resolve any differences in representation or view, respectively, between the component system and standard representations of those entities. If such FE's and FEV's cannot be found, the FIOM is expanded to include FEV's and / or FE's to which the new CCR can be added.

Determining whether an FCR already exists in the FIOM which corresponds to a CCR being added can be done by manually browsing the FIOM or by using a more automated approach. The Component Model Correlator is responsible for locating corresponding FE's and FEV's in the FIOM if they exist. The original Correlator used in the OOMI utilized a combined keyword and neural network-based approach for establishing correspondences among CCRs and FCRs in the FIOM. While providing a foundation for demonstrating the correlation process in the OOMI, this approach is believed to not provide sufficient values for precision and recall in the correlation process. Therefore, an investigation was conducted to determine what added promise the Semantic Web might offer to the correlation problem.

Fig. 5 summarizes how the components in the FIOM are used in resolving differences in representation and view between a source and destination system. First, an XML Schema, to which a source system XML document conforms, is used to create a CCR class using XML data binding [11]. Then, the source system XML document is converted to a CCR object which is an instance of the CCR class created initially using the same data binding process. The CCR object is converted to an FCR object using the CCR-FCR translation class created by the OOMI IDE Translation Generator described previously. The FCR object, in turn, is an instance of an FCR class created by the OOMI IDE. The source system FCR object is converted to a destination system FCR object using Liskov and Wing's notion of behavioral subtyping [10].

2.3 Introduction to Semantic Web technologies

As discussed in the previous section, the process of constructing an FIOM includes creating a standard representation, called an FCR, for every view of an entity that is present in the object model. The terminology used to represent the information in the FCR should be based on terms which reflect the federation-sanctioned representation of an entity's state and behavior [3].

An ontology defines the terms used to represent an area of knowledge. There have been many languages developed to de-

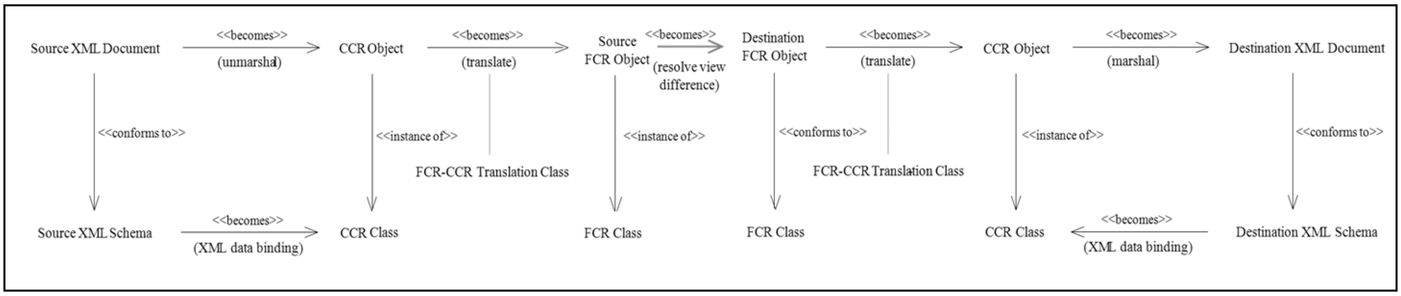


Fig. 5. Original OOMI Modeling Difference Resolution Process

scribe the contents of an ontology, the most predominant of which is the Web Ontology Language (OWL) which was selected by the World Wide Web Consortium (W3C) as its standard ontology language [12]. OWL is a Resource Description Framework (RDF) language developed by the W3C for defining classes and properties. It can also be used for enabling more powerful reasoning and inference over relationships between classes and properties as well as between objects and classes. OWL is based on XML-authoring tools, used mainly to express the needs of computer applications to deal with knowledge and information presented in the internet [13].

Soon after the W3C released the OWL language, it has “rapidly become a de facto standard for ontology development in general” [14]. Since OWL became a standard for creating ontologies it started influencing additional research such as investigation of reasoning techniques for ontology designs [14]. This research led to the creation of numerous reasoners for the OWL language, referred to as OWL reasoners. Some of the most popular open-source reasoners are Pellet [15], FaCT++ [16], and HermiT [17].

3 Adding Semantic Web technology to the OOMI

3.1 Overview

Since an FCR is defined to embody the standard representation of an entity, the information or terminology used to model an FCR should be taken from an ontology of federation-sanctioned terms and components. As the first step in adding Semantic Web technology to the OOMI, a Federation Ontology Library is added to the FIOM. The Federation Ontology Library defines the ontology classes, properties and individuals needed to specify the standard representation of the entities involved in system interoperation.

Recalling from earlier, even though two component systems share the same view of an entity, they may not represent that view in the same manner. A CCR is used to capture the unique way each component system may choose to represent its view of an entity. The next step in constructing the FIOM is to determine the view each component system represents. This can be accomplished by determining the FCR to which each CCR corresponds.

Locating an FCR which corresponds with a particular CCR is treated as a class correlation problem in the OOMI. CCR-FCR correlation in the original OOMI is handled using keyword and neural network techniques to provide semantic and syntactic matching between the classes used to represent a CCR and FCR. In subsequent sections we describe how the use of ontologies and reasoners can be used to improve upon the results returned by the original OOMI correlation process.

3.2 Semantic Web-OOMI Correlation Model

In the Semantic Web-OOMI Correlation Model, correlation is conducted between OWL classes using reasoning capabilities provided by an OWL reasoner such as Pellet [15], FaCT++ [16], or HermiT [17]. First, a Federation Ontology Library is constructed for the federation from federation-sanctioned OWL classes, properties and individuals. This library can either be imported from an existing ontology or ontologies or created using tools included with the OOMI Integrated Development Environment (IDE).

In the Semantic Web model it is assumed that the external interfaces of component systems are available as XML Schema Documents (XSD). Then, in order to utilize OWL to establish a correspondence between a component system and standard representation of an entity, each XSD which represents an entity from the component system external interface is converted into an OWL Ontology class, termed a CCR OWL class in Fig. 6. Finally, the Federation Ontology Library is queried using an OWL reasoner to find a class in the library corresponding to the CCR OWL class. If such a corresponding class is found, termed an FCR OWL class, an interoperability engineer creates an object model of that class with selected properties by converting the candidate FCR OWL class into a corresponding FCR class.

There may be some situations where the interoperability engineer cannot find a matching class in the current ontology library. Then, with permission of the ontology library custodian, the interoperability engineer can use the OOMI IDE’s OWL editor capabilities to add a new class to the ontology library. The search for an FCR OWL class which corresponds to the desired CCR OWL class is regenerated, this time with success in finding a match almost certain. This FCR OWL class is then converted to a corresponding FCR which is used to create the desired CCR-FCR translation. This paper is mainly focused on providing the capabilities of creating and managing the OWL ontology and converting classes and their properties from an OWL ontology into object models so that it would be easy for an interoperability engineer to create corresponding FCRs.

The process to search for a corresponding OWL class in the Federation Ontology library as well as the creation of a CCR object model from an XSD representation of a system’s external interface is left as future work. Creating the translation class to resolve the representational differences between the component CCR and standard FCR has been done previously by Young [3].

4 Implementation status

At its core, this work defines a new correlation methodology for the OOMI as depicted in Fig. 6. The original OOMI relied on a keyword search and neural network-based approach for locating existing standard representations of a component system’s entities which are shared via its external interface in the FIOM. The new methodology incorporates Semantic Web technology to assist in the correlation process used to locate standard entity representations.

The foundation for the Semantic Web technology used in redefining the FIOM creation process is the creation of a Federation Ontology Library containing OWL classes, properties and individuals used to define the standard representation of entities to be shared by components in the

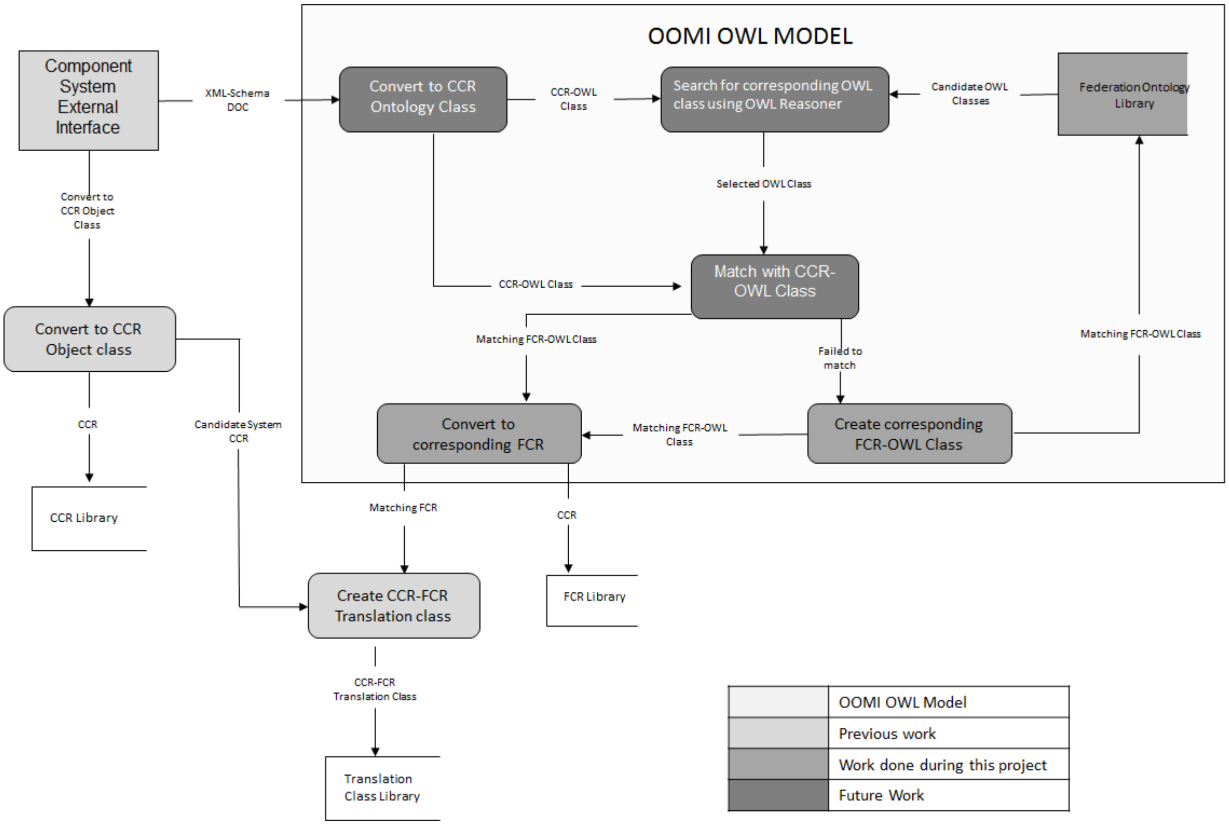

OOMI OWL MODEL
Component System External Interface
XML-Schema DOC
Convert to CCR Ontology Class
CCR-OWL Class
Search for corresponding OWL class using OWL Reasoner
Candidate OWL Classes
Federation Ontology Library
Selected OWL Class
Convert to CCR Object Class
CCR-OWL Class
Match with CCR-OWL Class
Convert to CCR Object class
Matching FCR-OWL Class
Failed to match
Matching FCR-OWL Class
CCR
Candidate System CCR
Convert to corresponding FCR
Matching FCR-OWL Class
Create corresponding FCR-OWL Class
CCR Library
Matching FCR
CCR
Create CCR-FCR Translation class
FCR Library
CCR-FCR Translation Class
Translation Class Library
OOMI OWL Model
Previous work
Work done during this project
Future Work

Fig. 6. Semantic Web-Based FIOM Creation Process

federation. Included with the addition of a new Federation Ontology Library are tools which enable an interoperability engineer to 1) import an existing library or libraries from other systems, federations or domains; 2) create a new library, adding classes, properties, and individuals to define the entities shared among components in the federation, and 3) modify an existing library by adding, deleting or changing existing items in the library.

As depicted in Fig. 6 and discussed previously, an entity imported to or exported from a component system is represented by an XML Schema document. This document is to be converted into an equivalent CCR OWL Class which is used to locate an equivalent standard representation of the entity, termed an FCR OWL class, in the Federation Ontology Library using an OWL-based reasoner. While not currently developed for the OOMI, such a reasoner should be able to be constructed using currently available technology such as that provided by Pellet [15], FaCT++ [16], or HermiT [17]. This portion of the new correlation methodology has not been implemented and is left for future work.

If a corresponding FCR OWL class is found in the library, it is converted into an equivalent FCR and used to create the CCR-FCR translation class used to resolve representational differences between the component and standard versions of the entity. The ability to convert an FCR OWL class into an equivalent FCR has been partially completed; it lacks only the user interface necessary to select which elements from the FCR OWL class are to be used in creating the FCR. The functionality required to create a CCR-FCR translation class from a matching CCR and FCR had been completed as part of the original OOMI IDE development effort.

Should an existing FCR OWL class not be found which matches the CCR OWL class of interest, a new FCR OWL class can be created using the existing Federation Ontology Library tools discussed previously. Repeating the process for locating a matching standard representation for a given component representation should result in this newly created FCR OWL class being found, with the CCR-FCR translation process proceeding as previously described.

5 Conclusion and recommendation for future work

Transition of the correlation process for matching component and standard representations of entities involved in the interoperation among systems from a keyword and neural network based approach to one using Semantic Web technologies promises to increase the precision and recall results of such search efforts. In addition, introducing the ontology-based approach for maintaining the standard representation of such entities provides an accepted list of terms from which to define the vocabulary for a federation, while limiting the disadvantages such a standards-based approach brings.

Ontologies and standards-based approaches for achieving interoperability enable syntactic and semantic differences among systems to be addressed by dictating a common structure and meaning for entities in a domain. An ontology defines the set of terms, entities, objects, classes, and their relationships, providing formal definitions and axioms that constrain the interpretation of these elements [18]. Adopting an ontology as a standard provides a clear, unambiguous structure and meaning to the elements in the domain to which the standard applies.

While an ontology and standards-based approach to interoperability has many benefits, one of the primary limitations to such an approach is the number of standards that have been defined. In the Health Information Technology (HIT) domain, Eichelberg, Aden, Riesmeier, Dogac, and Laleci review seven competing standards used in creating electronic health records (EHRs) with no clear winners found among them [19]. In fact, Wendt points out that Microsoft is involved in more than 100 standards-based organizations worldwide [20]. Another difficulty with standards-based approaches lies in the conformance of systems to the standards- when systems deviate from a standard they erode its value [21]. A final limitation with standards is their resistance to change. As standards are generally overseen by some organization to ensure their applicability, the time required for adapting an existing standard to changing technology or circumstances can be excessive.

The OOMI-Semantic Web approach provides the benefits of such an ontology-based methodology for maintaining the standard representation of such entities while minimizing its liabilities. By providing the capability to custom define the ontology to be used for a specified federation, the OOMI-Semantic Web method eliminates the problem of competing or conflicting standards by creating a specific ontology for the federation in question. While this ontology may be derived from an existing ontology or ontologies, it is specific to the federation being defined. This approach also virtually eliminates non-conformance among federation systems by providing the capability to customize the ontology to meet the demands of its component systems and is more flexible than traditional standards to changes in technology or requirements.

This paper lays the foundation for adapting the OOMI to utilize Semantic Web technology in constructing the FIOM. While this work goes a long way in taking advantage of the benefits a Semantic Web based approach brings, there is still work remaining to realize the full advantage of this capability. Such future work includes: 1) implementing an XML Schema to CCR Ontology Class converter; 2) implementing and integrating an OWL class reasoner to match component and standard ontology representations of an entity, 3) generating an FCR OWL class from the Ontology Library class matching the selected component representation of an entity, and 4) converting the generated FCR OWL class to an equivalent FCR. Full integration of Semantic Web technology into the OOMI will enable further exploration of areas where the OOMI approach may be adopted to improve system interoperability.

6 References

[1] *Levels of Information Systems Interoperability (LISI)*, C4ISR Architecture Working Group, 30 March 1998.

[2] E. Pitoura, "Providing Database Interoperability through Object-Oriented Language Constructs", *Journal of Systems Integration*, Volume 7, No. 2, August 1997, pp. 99-126.

[3] P. Young, "Heterogeneous Software System Interoperability Through Computer-Aided Resolution of Modeling Differences," Ph.D. Dissertation, Comp. Sci. Dept., Naval Postgraduate School, Monterey, CA, June 2002.

[4] G. Wiederhold, "Intelligent Integration of Information", *ACM-SIGMOD 93*, Washington, DC, May 1993, pp. 434-437.

[5] J. Hammer, D. McLeod, "Resolution of Representational Diversity in Multidatabase Systems", *Management of Heterogeneous and Autonomous Database Systems*, Morgan Kaufman, 1999.

[6] J. Kahng, J., McLeod D., "Dynamic Classificational Ontologies: Mediation of Information Sharing in Cooperative Federated Database Systems", *Cooperative Information Systems, Trends and Directions*, Academic Press, 1998.

[7] Holowczak, R., Li, W., "A Survey on Attribute Correspondence and Heterogeneity Metadata Representation", [Online]. Available: [http://www.computer.org/conferences/meta96/li/paper.html], 1996.

[8] Kim, W., Seo, J., "Classifying Schematic and Data Heterogeneity in Multidatabase Systems", *IEEE Computer*, Vol. 24, No. 12, pp. 12-18, December 1991.

[9] P. Young, V. Berzins, J. Ge, and Luqi, "Using an Object Oriented Model for Resolving Represenataional Differences between Heterogenous Systems," in *Proc. 17th ACM Symp. on Applied Computing*, Madrid, Spain, March 10-14, 2002, pp. 976-983.

[10] B. Liskov, J. Wing, "A Behavioral Notion of Subtyping," *ACM Transactions on Programming Languages and Systems*, Vol. 16, No. 6, November 1994, pp. 1811-1841.

[11] "Java Architecture for XML Binding," [Online]. Available: http://www.oracle.com/technetwork/articles/javase/index-40168.html

[12] M. Taye, "Web-Based Ontology Languages and its Based Description Logics," *Int. Journal of ACM Jordan*, vol. 2.

[13] P. Anisetty, "Adding Semantic Web Technology to the Object-Oriented Method for Interoperability (OOMI)," Masters Project Report, Comp. Sci. Dept., Univ. of Central Arkansas, Conway, 2012.

[14] T. Gardiner, D. Tsarkov, and I. Horrocks, "Framework for an Automated Comparison of Description Logic Reasoners," *Proc. 2006 Int. Semantic Web Conf.-ISWC 2006*, pp. 654-667, 2006.

[15] *Pellet: OWL 2 Reasoner for Java*. [Online]. Available: [http://clarkparsia.com/pellet/]

[16] *FaCT++* [Online]. Available: [http://owl.man.ac.uk/factplusplus/]

[17] *HermiT OWL Reasoner* [Online]. Available: [http://hermit-reasoner.com/]

[18] L. Pouchard, A. Cutting-Decelle, "Ontologies- and Standards-Based Approaches to Interoperability for Concurrent Engineering," in *Concurrent Engineering in Construction Projects*. Chimay Anumba, John Kamara, Anne-Francoise Cutting-Decelle, eds. pp.118-161. January 2007.

[19] Eichelberg, Aden, Riesmeier, Dogac, and Laleci, "A Survey and Analysis of Electronic Healthcare Record Standards," Journal of ACM Computing Surveys (CSUR), Vol. 37, Issue 4, Dec. 2005, pp. 277-315, ACM New York, NY, USA

[20] M. Wendt, "A Standards-Based Approach to Cloud Interoperability," [Online]. Available: [http://ow2.org/xwiki/bin/download/Events/OW2_Berlin_Day_2012/MicrosoftStandardsBasedApproachToCloud.pdf]

[21] Mahugh, D., "Standards-Based Interoperability," [Online]. Available: [http://blogs.msdn.com/b/dmahugh/archive/2009/06/05/standards-based-interoperability.aspx]

Clustering Analysis for Object Formation in Software Modeling

Dusan Sormaz[1], Danyu You[1], and Arkopaul Sarkar[2]
[1]Department of Industrial and Systems Engineering, Ohio University, Athens, OH, USA
[2]Institute for Corrosion and Multiphase Technologies, Ohio University, Athens, OH, USA

Abstract - *Object-oriented software modeling (OOM) is a widely used approach to provide structured way of designing software architecture. The object in OOM is a group of 'functions' and 'attributes' which perform related functionalities. The identification of objects is challenging in some application domains where functionalities are difficult to be partitioned/grouped, for instance, chemical calculations. This paper conducts the clustering analysis to partition those 'functions' and 'attributes' and form preliminary objects for OOM. The objective of clustering algorithm is to minimize the number of functions and attributes shared by more than one object. In this study, a function attribute matrix is first developed to represent the requirement of attributes for each function. And then a clustering algorithm is proposed to continuously restructure the function attribute matrix until the termination conditions are met or no more improvement can be found. The experimentation is carried based on the corrosion prediction software which contains up to 200 nested functions and attributes. And the results show that the promising approach can capture the homogeneous functions and attributes and form preliminary objects.*

Keywords: software clustering, legacy software, object oriented modeling

1 Introduction

Object-oriented software modeling (OOM) is a matured software design function and widely applied in the early phase of software development [1]. Clustering algorithms are commonly functions of discovering intrinsic patterns from a collection of raw data. It has been applied in many domains such as group technology, cellular manufacturing, organizational behavior and others.

This paper intends to apply the clustering technology on reference from functions to variables, and to reveal potential combination pattern of them, which then is used to find objects.

The rest of this paper is organized as following: Section 2 describes the background of the current problem. Section 3 reviews the development in the software modeling and the clustering technology. Section 4 explains the construction of Functions-Variables matrix, the clustering algorithm, the adjustments to result and the creation of classes. Section 5 shows results of applying clustering analysis on an existing software application. At last, in section 6, the conclusion is drawn and the reference papers are listed.

2 Problem Description

This paper focuses on identification of potential objects, which is the preparation of OOM; in existing software applications. In our case, the target system is a software application for predicting corrosion rate of oil pipeline [1]. Since it was written many years ago when OO features were not supported by Fortran 77 language, it is not an OO designed application, which means a large portion of its functions and variables were defined globally to simplify passing arguments through functions, but its functional modules are coupled tightly. An obvious shortcoming of such structure is: whenever any modification is introduced into this system, corresponding global influences should be considered. It makes development and maintenance become increasingly harder as the growth of application size. To solve that problem, a loosen-coupled structure should be built to replace that out-dated structures. As OOM is a very common and matured function to construct flexible software structure, and OO mannered programming starts being supported by Fortran 90, it is applied to guide the process of reorganization. In order to solve the problem, well-known function of clustering is it's applies from is chosen and the goal has been set to explore its performance in a new domain.

Using OOM to build system models, the initial stage is defining system objects and understanding relationship between them in a high enough level. Usually, developers have complete freedom to create necessary objects as long as they are consistent with customer's requirements. But in this case, in order keep existing modules, which are well functioned, OOM has to start from the lowest lever – the level of codes, and objects needs to be extracted from codes. In another words, the task is to observe potential groups of interrelated functions and variables from current codes to build objects. This is the opposite of the early process of OOM; however, it is what clustering technology is designed for – finding groups in a collection of unstructured data.

3 Previous Work

3.1 Software Reorganization

Software reorganization is quite common in software engineering. Studies of software engineering [1] reveal that a great amount of code changes are requested in the later stage of development and even after development is finished. Implementation of those changes is proved not only costly and time-consuming, but also negatively impacts existing system architecture, especially when the existing structure is not well designed to accommodate changes. Under some serious situation, the existing design even needs to be reorganized to upgrade to a better structure.

OOM, because of its successful application in software engineering, becomes the most common technology being used in process software design and redesign. The core concept of OOM is the object [3]which is a group of functionally related 'functions' / 'functions' and 'attributes' / 'variables', called members. Other concepts, such as inheritance, interface and polymorphism, help to construct a flexible network of objects. Inside objects, members are coupled tightly; outside objects, models are loosen-coupled [7].

3.2 Clustering Algorithms

Clustering algorithm is a function in observing structures of data so that interrelated members can be organized into groups [8]. It has been used in diverse domains, such as network optimization [11], information retrieval [12], market research [9], and quality control [11].

4 Methodology

The clustering analysis of the existing software structure is performed through several steps as shown in figure 1. First, a function-variable matrix is built for the existing software package. For clustering, King's ROC algorithm [5] is applied. Then, adjustments are implemented to simplify matrix. By dividing the matrix into segments and treating each segment as a class, the function-variable matrix is changed into a class-function matrix. Through analysis against the matrix, functions are distributed into the class having the strongest connection to them. At the end, the logic of cross reference between classes is established focus the existing relations.

4.1 Function-Variable Matrix

Function-variable matrix is a binary matrix constructed to explore the relationship between function funtions and variables, which can be expressed by equation (1):

$$X = matrix(r \times c) \qquad (1)$$

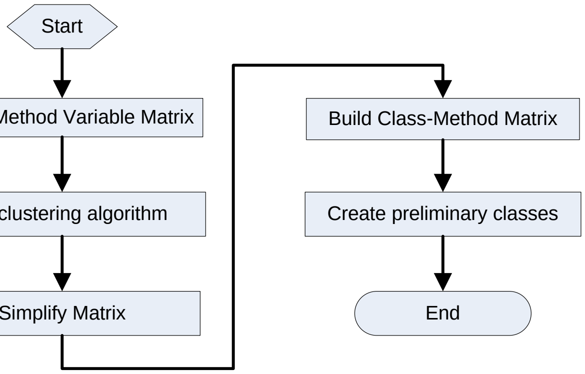

Start
Define Method Variable Matrix
Run clustering algorithm
Simplify Matrix
Build Class-Method Matrix
Create preliminary classes
End

Figure 1: The procedure of methodology

Where, r is the row of matrix, equals to number of functions; c is the column of matrix, and equals number of variables.

$$X_{ij} = 0 \text{ or } 1 \qquad (2)$$

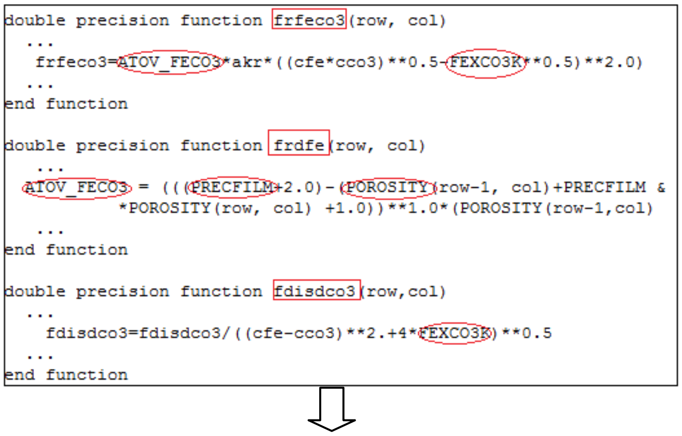
Where, iE [1, r] and jE[1, c]. If Xi, j=1, it means function with index i uses variable with index j; if Xi, j=0, there is no connection between function i and a variable j.

Following is an example of the function variable matrix, which is for an application having 8 variables and 4 functions. So the matrix is a two dimensional array with 4 rows and 8 columns. In that matrix, X3, 3=1, which means variable v3 is used in function f3.

	v1	v2	v3	v4	v5	v6	v7	v8
f1	0	0	1	0	0	0	0	0
f2	0	1	0	0	1		0	0
f3	1	0	1	1	0	1	1	1
f4	1	1	0	1	0	0	0	0

Figure 2. An Example of function-variable Matrix

Another example gives a demonstration of how to build such a matrix from codes. A segment of codes is taken from MULTICORP and shown in Figure 2, where functions names and global variables are marked by rectangles and ellipses separately (repeatedly used variables are only marked once). There are 4 different global variables and 3 functions. Using the method discussed in this section, the corresponding function-variable matrix having 4 columns and 3 rows is built and show in figure 3



```
double precision function frfeco3(row, col)
  ...
   frfeco3=ATOV_FECO3*akr*((cfe*cco3)**0.5-FEXCO3K**0.5)**2.0)
  ...
end function

double precision function frdfe(row, col)
   ...
  ATOV_FECO3 = (((PRECFILM+2.0)-(POROSITY(row-1, col)+PRECFILM &
           *POROSITY(row, col) +1.0))**1.0*(POROSITY(row-1,col)
   ...
end function

double precision function fdisdco3(row,col)
  ...
   fdisdco3=fdisdco3/((cfe-cco3)**2.+4*FEXCO3K)**0.5
  ...
end function
```

	ATOV_FECO3	FEXCO3K	POROSITY	PRECFILM
frfeco3	1	1	0	0
frdfe	1	0	1	1
fdisdco3	0	1	0	0

Figure 3: An example of building function-variable matrix from codes.

4.2 Run Algorithm

King's POC algorithm is applied a function–variable matrix connected focus the software. To apply the algorithm for each function and variable, a score is assigned by using following equations. Each row and column of the matrix is given a weight factor equal to 2i-1 or 2j-1 (for example, third row has weight 23 = 8).

Then for each method and variable, a score is assigned by using following equations.

Score of the function is,

$$Si = \sum_{j=1}^{c} X_{ij} * W_j \qquad (3)$$

Similarly, score of the variable is,

$$S\,j = \sum_{i=1}^{r} X_{ij} * W_i \qquad (4)$$

For example, using equation (3), in figure 2, the score for function 3 (the fourth row) is = 21 + 23 + 24+ 26 + 27 + 28=474.

The clustering algorithm applied by this paper. Using score for rows and columns, King's ROC algorithm is applied on the following sequence:

(1) Calculate score for each row and each column,

(2) Sort columns basing on their scores from low to high,

(3) Sort rows by the same way,

(4) After each sorting, compare with the latest matrix with the previous one, if there is no obvious change after sorting, stop algorithm; otherwise, go to (1), continue the loop.

For example, if applying previous steps to matrix in figure 2, following result (figure 4) should be generated.

	POROSITY	PRECFILM	FEXCO3K	ATOV_FECO3
fdisdco3	0	0	1	0
frdfe	1	1	0	1
frfeco3	0	0	1	1

Figure 4: Result of applying the algorithm against the matrix in figure 3

4.3 Adjustments

If there is a complete separation of functions and variable result of clustering, figure 5 would be an ideal combination of variables and functions. Such a result, however, cannot be expected in a real case is usually not obtained.

	v 1	v 2	v 3	v 4	v 5	v 6	v 7	v 8
f 1	1	1	0	0	0	0	0	0
f 2	1	1	0	0	0	0	0	0
f 3	0	0	1	1	1	0	0	0
f 4	0	0	0	0	0	1	1	1

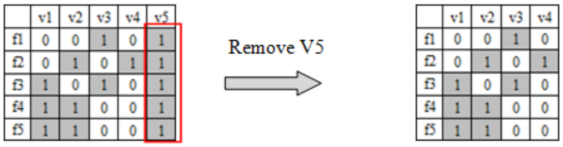
Figure 5: An example of ideal result

For a realistic matrix, which does not have such a clean pattern, such as the result shown in figure 5, some adjustments are necessary for its simplification. Rules are developed for adjustments and described as following:

Rule 1: Separate global variables

If $\sum_{i=1}^{r} X_{ij} \geq r * p$, $j \in [1, c]$, $p \in (0,1]$, in our case, p = 20%.

This Rule indicates, a variable will keep staying as a global one and be taken out of the matrix if it is used by a relatively large collection of functions, say 20% of total functions. An extreme situation is shown in figure 6, where variable 5 is used by every function. So V5 should be global and not be considered in this matrix.



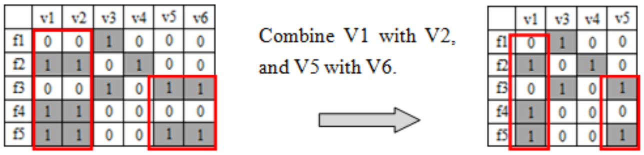
	v1	v2	v3	v4	v5
f1	0	0	1	0	1
f2	0	1	0	1	1
f3	1	0	1	0	1
f4	1	1	0	0	1
f5	1	1	0	0	1

	v1	v2	v3	v4
f1	0	0	1	0
f2	0	1	0	1
f3	1	0	1	0
f4	1	1	0	0
f5	1	1	0	0

Figure 6: An example application of Rule1

Rule 2: Combine variables (columns) with the same functions callers.

If there are existing n columns, for , is true, then keep only one column of them, and remove the rest. An example is shown in figure 7.



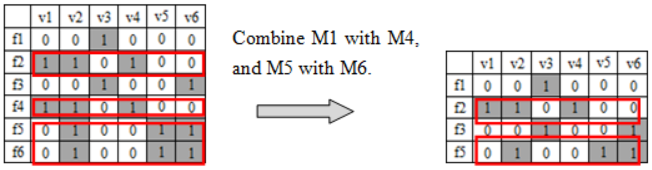
	v1	v2	v3	v4	v5	v6
f1	0	0	1	0	0	0
f2	1	1	0	1	0	0
f3	0	0	1	0	1	1
f4	1	1	0	0	0	0
f5	1	1	0	0	1	1

	v1	v3	v4	v5
f1	0	1	0	0
f2	1	0	1	0
f3	0	1	0	1
f4	1	0	0	0
f5	1	0	0	1

Figure 7: An example application of Rule2

Rule 3: Combine functions which has the same set of variables.

If there are existing n rows, for , is true, then keep only one row, and remove the rest. An example is shown in figure 8.



	v1	v2	v3	v4	v5	v6
f1	0	0	1	0	0	0
f2	1	1	0	1	0	0
f3	0	0	1	0	0	1
f4	1	1	0	1	0	0
f5	0	1	0	0	1	1
f6	0	1	0	0	1	1

	v1	v2	v3	v4	v5	v6
f1	0	0	1	0	0	0
f2	1	1	0	1	0	0
f3	0	0	1	0	0	1
f5	0	1	0	0	1	1

Figure 8: An example application of Rule3

Rule 4: Eliminate variables which are used in only one function.

After applying those rules, it may be necessary to run POC algorithm once again to obtain additional clusters.

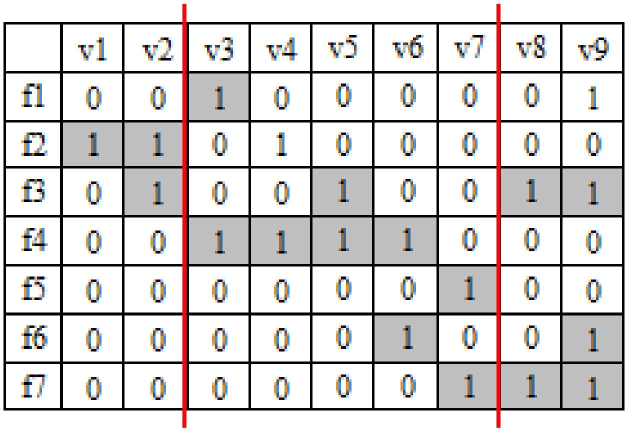
4.4 Create Classes

After the clustering completed, the final matrix serves the purpose of defining the classless. Class creation is performed in three steps: indentify divisions, assign methods to classes, setup cross references between the classes as shown in this section.

4.4.1 Identify Divisions

Classes are formed by identifying divisions as shown Figure 6 gives an example of the matrix that might be formed after previous steps.

The matrix needs to be divided into divisions horizontally, and each of them becomes a class. The basic rule of drawing division is trying to put as much as possible usages (represented by cells with number 1) of the same variable into one area. One possible plan of division is given by red lines in figure 9.

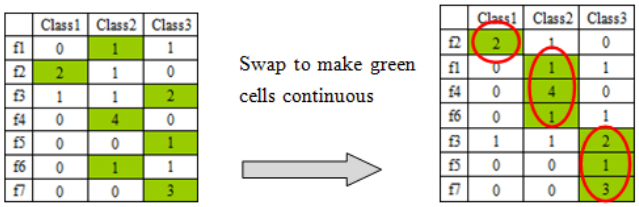


	v1	v2	v3	v4	v5	v6	v7	v8	v9
f1	0	0	1	0	0	0	0	0	1
f2	1	1	0	1	0	0	0	0	0
f3	0	1	0	0	1	0	0	1	1
f4	0	0	1	1	1	1	0	0	0
f5	0	0	0	0	0	0	1	0	0
f6	0	0	0	0	0	1	0	0	1
f7	0	0	0	0	0	0	1	1	1

Figure 9: An example of possible division of a matrix

4.4.2 Assign Functions to Classes

Given divisions from figure 9, another form of matrix, the function-class matrix, is developed. The function-class matrix shown is similar to the function-variable matrix built in Section 4.1 except of following two differences:
(1) The first row contains name of preliminary classes (excluding first cell to the left);
(2) Cells contain integer numbers instead of Boolean value to indicate how many variables a function uses in the class.
An example is shown in figure 10, which is generated from the matrix in figure6.



	Class1	Class2	Class3
f1	0	1	1
f2	2	1	0
f3	1	1	2
f4	0	4	0
f5	0	0	1
f6	0	1	1
f7	0	0	3

	Class1	Class2	Class3
f2	2	1	0
f1	0	1	1
f4	0	4	0
f6	0	1	1
f3	1	1	2
f5	0	0	1
f7	0	0	3

Figure 10 Matrix improvement

As shown in figure 10-(a), some functions are shared by classes, but, in OOM, one specific function usually only belong to one class. Other classes that need to call that function only need to keep an instance of that class. A function should be put into the class that has the strongest connection with it. Assuming that, the more one function is used in one class, the stronger their connection is. In figure 10, the first maximum number of each row is marked with green color to tell which class uses a function the most. Then, rows are swapped to make green cells in the same column stay as continuous as possible. Swapped matrix is shown in figure 10-(a). In column of each class, there is a strip of continuous green cells (circled by red rectangle), which stands for a collection of functions that are strongly depended by the class. Thus, those functions are assigned to their depending class.

4.4.3 Set up cross references between classes

The last step is to construct a bridge between classes so that a function is able to be called by other classes. This bridge is the reference. In OOM, a reference is an instance of a class. In order to highlight references between classes, the matrix in figure 10-(b) is simplified into a new matrix and shown in figure 11.

	Class1	Class2	Class3
f2	🟩	1	0
f1	0	🟩	2
f4			
f6			
f3	1	1	🟩
f5			
f7			

Figure 11: Function–class matrix.

In figure 11, a green rectangle is considered as a class; integer number inside a cell means how many variables a class uses functions form another one. Specifically, for Xi, j = N, Xi, j is the value of cell in row i and column j, i>0, j>0, N ≥ 0, (1) If i > j, it means class i calls function from class j N times; (2) If i<j, it means class j calls function from class i N times.

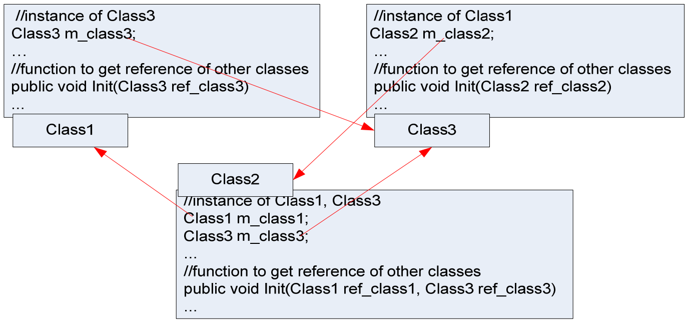
In figure 11, for example, X2, 3 = 2, which means Class3 uses functions from Class2 two times; and X3, 1 = 1 means class3 uses functions from Class1 one time.

The matrix can be interpreted by table 1 to indicate if a class needs an instance of another one.

Table 1 Establishing references between classes

Class Name	Reference to Class1	Reference to Class2	Reference to Class3
Class1	--	**No**	**Yes**
Class2	**Yes**	--	**Yes**
Class3	**No**	**Yes**	--

Based on table 1, a solution for calling functions between classes is proposed in figure 12.



```
//instance of Class3
Class3 m_class3;
…
//function to get reference of other classes
public void Init(Class3 ref_class3)
…
```

```
//instance of Class1
Class2 m_class2;
…
//function to get reference of other classes
public void Init(Class2 ref_class2)
…
```

```
//instance of Class1, Class3
Class1 m_class1;
Class3 m_class3;
…
//function to get reference of other classes
public void Init(Class1 ref_class1, Class3 ref_class3)
…
```

Figure 12: calling methods between classes

5 Case Study

Methodology discussed in section 4 is applied on a collection of functions and variables from a real software application, Multicorp/Corrsim, originally implemented in Fortran 77, and results are shown. AS a result the application has been converted into Fortran 90, which supports abstract data types and allows application of object-oriented modeling.

5.1 Build function-variable matrix

Analysis of the existing application reveals that there are 33 functions and more 90 global variables. Related function-variable matrix is built and shown in Figure 13.

5.2 Apply clustering algorithm and adjust

After the first clustering, matrix is changed and shown in figure 14, which is not excellent but expected. Though, there is some clustering achieved, the structure of the code requires further analysis.

Through adjustments discussed in section 4.2, the matrix is simplified to variables and functions and shown in figure 15.

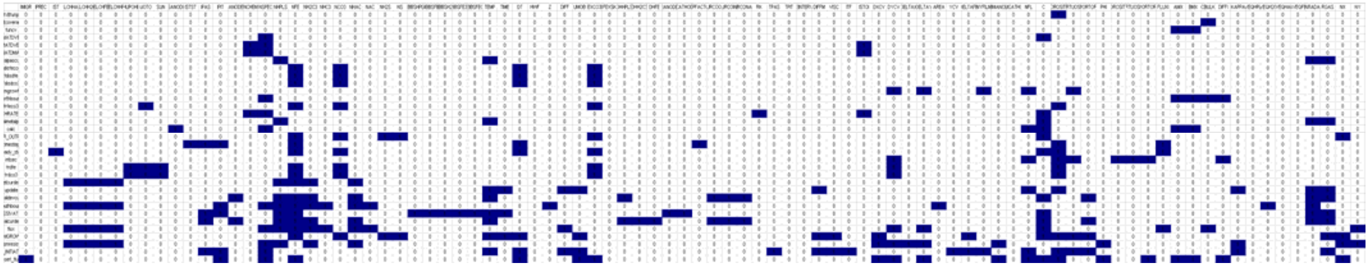


Figure 13: Initial function-variable matrix

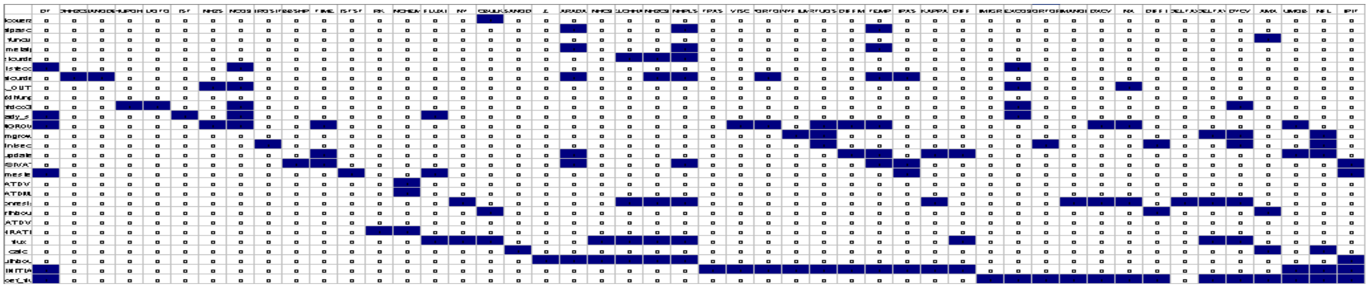


Figure 14: Preliminary result of running algorithm

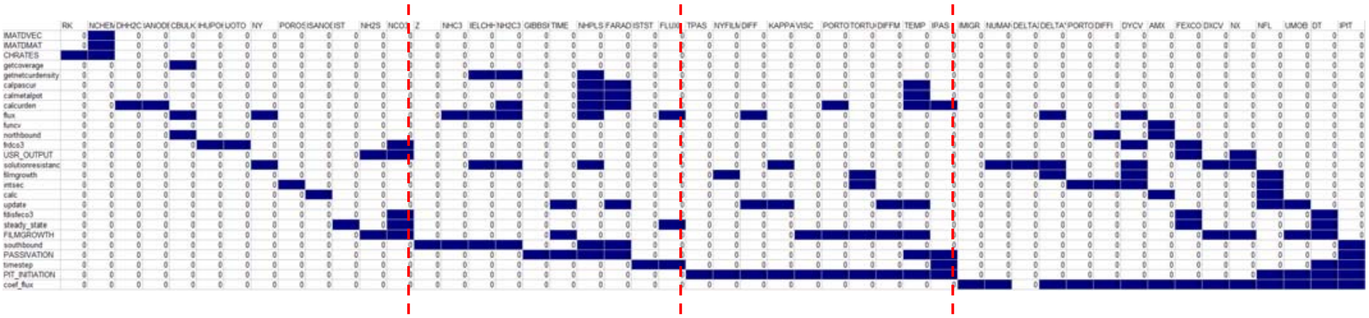


Figure 15: Divisions of matrix

5.3 Identify classes

In figure 15, four divisions are drawn via function discussed in section 4.4.1 and variables are grouped into the following classes:

Class1:

RK NCHEM DHH2C3 IANODE CBULKIHUPOHE
IJOTO NY POROSITYI ISANODE IST NH2S
NCO3

Class2:

Z NHC3 IELCHHACNH2C3 GIBBSHPLS TIME
NHPLS FARADAY ISTST FLUXI

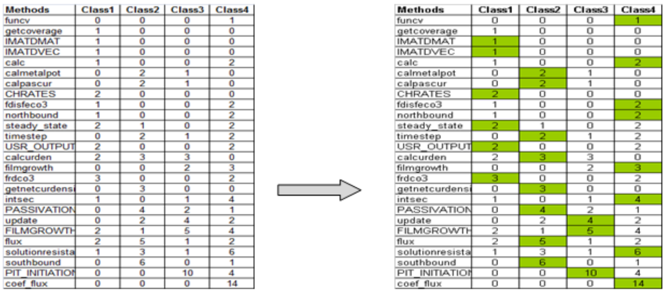
Class3:

TPAS NYFILM DIFF KAPPA VISC PORTOR
TORTUOSITY DIFFM TEMP IPAS

Class4:

IMIGR NUMANODE DELTAX DELTAY PORTORI
DIFFI DYCV AMX FEXCO3K DXCV NX NFL
UMOBDT IPIT

Then, the matrix is modified into a class-function matrix, which is shown in figure 16-(a). The maximum number of each row is marked as green to show connections between functions and classes.

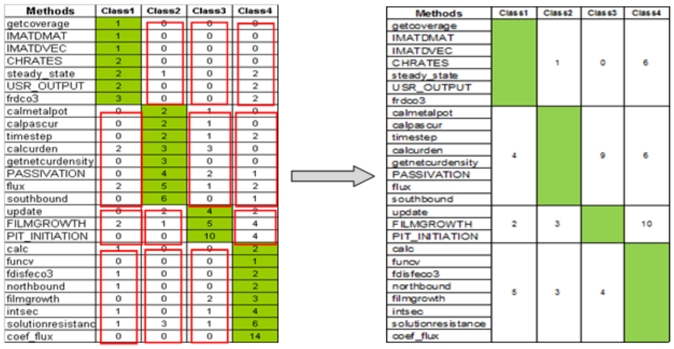


Methods	Class1	Class2	Class3	Class4
funcv	0	0	0	1
getcoverage	1	0	0	0
IMATDMAT	1	0	0	0
IMATDVEC	1	0	0	0
calc	1	0	0	2
calmetalpot	0	2	1	0
calpascur	0	2	1	0
CHRATES	2	0	0	0
fdisfeco3	1	0	0	2
northbound	1	0	0	2
steady_state	2	1	0	2
timestep	0	2	1	2
USR_OUTPUT	2	0	0	2
calcurden	2	3	3	0
filmgrowth	0	0	2	3
frdco3	3	0	0	2
getnetcurdensi	0	3	0	0
intsec	1	0	1	4
PASSIVATION	0	4	2	1
update	0	2	4	2
FILMGROWTH	2	1	5	4
flux	2	5	1	2
solutionresista	1	3	1	6
southbound	0	6	0	1
PIT_INITIATION	0	0	10	4
coef_flux	0	0	0	14

Methods	Class1	Class2	Class3	Class4
funcv	0	0	0	🟩 1
getcoverage	1	0	0	0
IMATDMAT	🟩 1	0	0	0
IMATDVEC	🟩 1	0	0	0
calc	1	0	0	🟩 2
calmetalpot	0	🟩 2	1	0
calpascur	0	🟩 2	1	0
CHRATES	🟩 2	0	0	0
fdisfeco3	1	0	0	🟩 2
northbound	1	0	0	🟩 2
steady_state	🟩 2	1	0	2
timestep	0	🟩 2	1	2
USR_OUTPUT	🟩 2	0	0	2
calcurden	2	🟩 3	3	0
filmgrowth	0	0	2	🟩 3
frdco3	🟩 3	0	0	2
getnetcurdensi	0	🟩 3	0	0
intsec	1	0	1	🟩 4
PASSIVATION	0	🟩 4	2	1
update	0	2	🟩 4	2
FILMGROWTH	2	1	🟩 5	4
flux	2	🟩 5	1	2
solutionresista	1	3	1	🟩 6
southbound	0	🟩 6	0	1
PIT_INITIATION	0	0	🟩 10	4
coef_flux	0	0	0	🟩 14

Figure 16 Mark maximum in each row of Class-function Matrix

By swapping rows in the matrix, green cells in the same column are connected together. Each strip of green cell is a collection of functions that should belong to the same class.

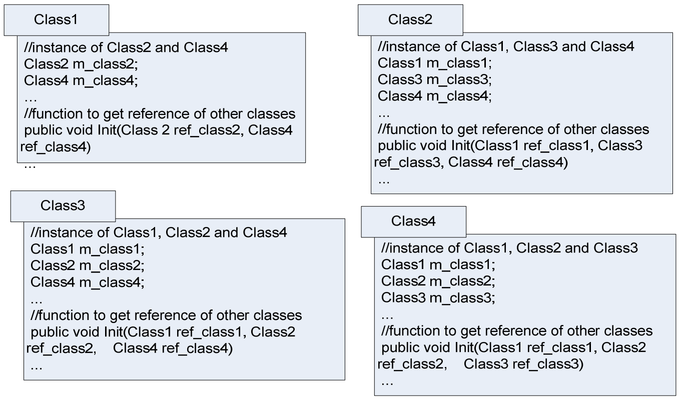


Methods	Class1	Class2	Class3	Class4
getcoverage	🟩 1	0	0	0
IMATDMAT	🟩 1	0	0	0
IMATDVEC	🟩 1	0	0	0
CHRATES	🟩 2	0	0	0
steady_state	🟩 2	1	0	2
USR_OUTPUT	🟩 2	0	0	2
frdco3	🟩 3	0	0	2
calmetalpot	0	🟩 2	1	0
calpascur	0	🟩 2	1	0
timestep	0	🟩 2	1	2
calcurden	2	🟩 3	3	0
getnetcurdensity	0	🟩 3	0	0
PASSIVATION	0	🟩 4	2	1
flux	2	🟩 5	1	2
southbound	0	🟩 6	0	1
update	0	2	🟩 4	2
FILMGROWTH	2	1	🟩 5	4
PIT_INITIATION	0	0	🟩 10	4
calc	1	0	0	🟩 2
funcv	0	0	0	🟩 1
fdisfeco3	1	0	0	🟩 2
northbound	1	0	0	🟩 2
filmgrowth	0	0	2	🟩 3
intsec	1	0	1	🟩 4
solutionresistanc	1	3	1	🟩 6
coef_flux	0	0	0	🟩 14

Methods	Class1	Class2	Class3	Class4
getcoverage	🟩	1	0	6
IMATDMAT				
IMATDVEC				
CHRATES				
steady_state				
USR_OUTPUT				
frdco3				
calmetalpot	4	🟩	9	6
calpascur				
timestep				
calcurden				
getnetcurdensity				
PASSIVATION				
flux				
southbound				
update	2	3	🟩	10
FILMGROWTH				
PIT_INITIATION				
calc	5	3	4	🟩
funcv				
fdisfeco3				
northbound				
filmgrowth				
intsec				
solutionresistance				
coef_flux				

Figure 17 Matrix with classified functions further simplified

Using function discussed in 4.3.2.3, continuous green cells and cells inside red rectangle are merged into single cells, a simple and clean matrix about connection between classes is formed. Based on structure indicated by this figure, a possible solution for building four classes are proposed as following.



```
//instance of Class2 and Class4
Class2 m_class2;
Class4 m_class4;
…
//function to get reference of other classes
public void Init(Class 2 ref_class2, Class4
ref_class4)
…
```

```
//instance of Class1, Class3 and Class4
Class1 m_class1;
Class3 m_class3;
Class4 m_class4;
…
//function to get reference of other classes
public void Init(Class1 ref_class1, Class3
ref_class3, Class4 ref_class4)
…
```

```
//instance of Class1, Class2 and Class4
Class1 m_class1;
Class2 m_class2;
Class4 m_class4;
…
//function to get reference of other classes
public void Init(Class1 ref_class1, Class2
ref_class2,    Class4 ref_class4)
…
```

```
//instance of Class1, Class2 and Class3
Class1 m_class1;
Class2 m_class2;
Class3 m_class3;
…
//function to get reference of other classes
public void Init(Class1 ref_class1, Class2
ref_class2,    Class3 ref_class3)
…
```

Figure 18: One possible implementation of collaboration between classes

6 Conclusions and future studies

The results shown in section 5 demonstrate that, in the process of redesigning the structure of an existing software application, it is possible to draw a rough picture of potential classes and their interrelation by using clustering algorithms, without fully understanding the logic structure of that application. Although this function does not guarantee the correctness and reasonability of the results, it at least can server as preprocess of OOM to help discover some patterns inside codes. Future work will be related to how to use results from this paper to assist semantic analysis, and also to distinguish between reading and writing variables. Implementation of the improved version of the software based on the findings is under way.

7 References

[1] S. Nesic, Jiyong Cai, Shihuai Wang, Ying Xiao and Dong Liu, Ohio University Multiphase Flow and Corrosion Prediction Software Package MULTICORP V3.0, Ohio University(2004).

[2] Chandra K. S., 2007, “Identifying Software Patterns Approaches to Cognitive Modeling”, The *ICFAI Journal of Computer Sciences*, 56 (1), 54-60.

[3] Aksit M., 1996, “Composition and Separation of Concerns in the Object-Oriented Model”, *ACM Computing Surveys*, Vol. 28A, No. 4.

[4] Siva Balan R. V., Punithavalli M., 2010, “Software Architecture, Scenario and Patterns”, *IJCSI International Journal of Computer Science Issues*, 7 (5), 418-423

[5] Kamrani A. K., 1998, *Group Technology and Cellular Manufacturing: Methodologies and Applications*, Taylor & Francis

[6] Landauer T. K., 1995, *The Trouble with Computers: Usefulness, Usability and Productivity*., MIT Press., Cambridge.

[7] Li W. and Henry S., 1993, “OO Metrics that Predict Maintainability”, *Journal of Systems and Software*, 23(2), 111-122.

[8] Wu Z., and Leahy R., 1993, “An Optimal Graph Theoretic Approach to Data Clustering: Theory and Its Application to Image Segmentation”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15, 1101–1113.

[9] Punj G., and Stewart D.W., 1983, “Cluster analysis in marketing research: review and suggestions for application, *Journal of Marketing Research*, 20, 134–148.

[10] Lew M.S., Sebe N., Djeraba C., and Jain R., 2006, “Content-based multimedia information retrieval: state of the art and challenges”, *ACM Transactions on Multimedia Computing, Communication, and Applications*, 2, 1–19.

[11] Zarandia M. H. F., and Alaeddinib A., 2010, “A general fuzzy-statistical clustering approach for estimating the time of change in variable sampling control charts”, *Information Sciences*, 180, 3033–3044.

[12] Deng Y., and Wang F., 2006, “Optimal Clustering Size Of The Small File Access In Network Attached Storage Device”, *Parallel Processing Letters*, 16 (4), 501-502.

Domain-Driven Model Inference Applied To Web Applications

A. Sébastien Salva[1], **B. William Durand**[2]
[1]LIMOS - UMR CNRS 6158, Auvergne University, France
[2]LIMOS - UMR CNRS 6158, Blaise Pascal University, France

Abstract—*Model inference methods are attracting increased attention from industrials and researchers since they can be used to generate models for software comprehension, for test case generation, or for helping devise a complete model (or documentation). In this context, this paper presents an original inference model approach which recovers models from Web application HTTP traces. This approach combines formal model inference with domain-driven expert systems. Our framework, whose purpose is to simulate this human behaviour, is composed of inference rules, translating the domain expert knowledge, organised into layers. Each yields partial IOSTSs (Input Output Symbolic Transition System), which become more and more abstract and intelligible.*

Keywords: Model inference, formal model, IOSTS, rule-based system

1. Introduction and Contribution

In the Industry, legacy applications are often problematic as they are hard to maintain, poorly documented, and usually not covered by tests. When it comes to this situation, there is a high risk of introducing a bug, and options left to developers are weak. The only way to ensure stability while fixing a bug is to learn how the application behaves.

A first classic solution is to express these behaviours with formal models, for instance Input/Output Symbolic Transition Systems (IOSTS) [3]. Such models are particularly interesting to automatically generate test suites using Model- based testing techniques. But, the complete model writing is often an heavy task, and is error prone; hence the need for model inference approaches.

Model inference is a relatively recent research field aiming at recovering the application behaviours captured by a model. Zong et al. [9] proposed to infer specifications from API documentations to check whether implementations match them. Such specifications do not reflect the implementation behaviours though. In [6], specifications, which are extremely detailed, show the method calls observed from a related set of objects. Some works [4], [5], [1], [8] proposed to derive models by automatically testing an application. These are often based upon crawling techniques, which can produce either basic models or too detailed models. In both cases, it is not suitable for test case generation. For instance, Memon et al. [4] initially presented GUITAR, a tool for scanning desktop applications which produces event flow graphs and trees showing the GUI execution behaviours. The generated models are quite simple and many false event sequences have to be weeded out later. Mesbah et al. [5] proposed the tool Crawljax specialised in AJAX applications, which produces state machine models that are too complex and unreadable.

In this paper, we leverage model inference techniques in order to obtain a model from an existing application, running in a production environment. We decided to record incoming and outgoing data (traces) by monitoring applications, rather than crawling the entire application to prevent the limitations described above. Our proposal takes another direction to infer models. We do not suppose that the application being analysed is event-driven but at least yields traces. It emerges from the following idea: a domain expert, which is able to conceive specifications, is also able to diagnose the behaviour of the corresponding implementation by reading and interpreting its execution traces. His knowledge can be formalised and exploited to automatically infer models. Our approach is based upon this notion of domain knowledge, implemented with an expert system which includes inference rules. The originality of our approach also resides in the incremental production of several models, expressing the behaviour of the same application at different abstraction layers. This approach can be applied on any application producing traces, i.e. not only event-driven applications.

Below, we describe the architecture of our model inference framework. Then, we recall some definitions on the IOSTS formalism used throughout the paper in Section 3. We concretely describe and define this framework in the context of Web applications in Section 4. We give some experimentation results in Section 5. Conclusions are drawn in Section 6 together with directions for further research and improvements.

2. Architecture of our framework

Our framework is divided into several modules as depicted in Figure 1. The *Models generator* is the centrepiece of the framework. It takes traces as inputs, which can be sent by a *Monitor* collecting them on the fly. But it is worth mentioning that the traces can also be sent by any tool or even any user, as far as they comply to a chosen standard format. The Models generator is based upon an expert system, which is an artificial intelligence engine emulating acts of a domain expert by inferring a set of rules representing the expert knowledge. This knowledge is organised into a hierarchy with several layers. Each gathers a set of inference rules written with a first order predicate logic. Typically, each layer creates two IOSTSs (except the first one), and the higher the layer is, the more abstract the IOSTSs become. These models are then successively stored and can be later analysed by experts, verification tools, etc. This number of layers is not strictly

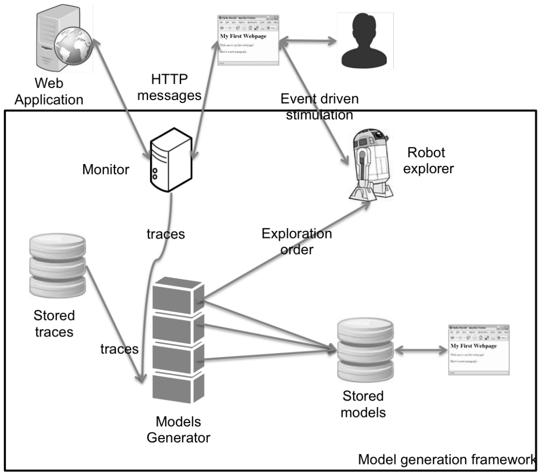


Fig. 1: Model generation framework

bounded even though it is manifest that it must be finite. The domain knowledge encapsulated in the expert system can be used to cover trace sets coming from several applications of the same category thanks to generic rules. But, rules can also be specialised and refined for a given application in order to yield more precise models, easing application comprehension.

Our approach allows to take a predefined set of traces collected from any kind of applications producing traces. For event-driven applications, traces could be produced using automatic testing techniques. We provide a Robot explorer which will not be presented here because of lack of room. It generates traces and find new application states in an efficient manner using strategies, that are expressed with inferences rules as well, tackling the issue related to the amount of traces needed to decently cover an application.

Our proposal is both flexible and scalable. It does not produce one model but several ones, depending on the number of layers of the Models generator, which is not limited and may evolve in accordance to the application type. Each model, expressing the application behaviours at a different level of abstraction, can be used to ease the writing of complete formal models, to apply verification techniques, to check the satisfiability of properties, to automatically generate functional test cases, etc.

In the following, we detail the different framework parts in the context of Web applications, except for the Monitor, which is here a classical proxy.

3. Model Definition and Notations

We consider the Input/Output Symbolic Transition System (IOSTS) formalism [3] for describing the functional behaviour of systems or applications. An IOSTS is a kind of automata model which is extended with two sets of variables, internal variable to store data, and parameters to enrich the actions. Transitions carry actions, guards, and assignments over variables. The action set is separated with inputs beginning with ? to express actions expected by the system, and with outputs beginning with ! to express actions produced by the system. An IOSTS does not have states but locations.

Definition 1 (IOSTS) *An IOSTS $\mathcal{S}$ is a tuple $< L, l0, V, V0, I, \Lambda, \rightarrow>$, where:*

- *L is the finite set of locations, $l0$ the initial location,*
- *V is the finite set of internal variables, I is the finite set of parameters. We denote D_v the domain in which a variable v takes values. The assignment of values of a set of variables $Y \subseteq V \cup I$ is denoted by valuations where a valuation is a function $v : Y \rightarrow D$. $v_\emptyset$ denotes the empty valuation. D_Y stands for the valuation set over the variable set Y. The internal variables are initialised with the assignment $V0$ on V, which is assumed to be unique,*
- *Λ is the finite set of symbolic actions $a(p)$, with $p = (p_1, ..., p_k)$ a finite list of parameters in $I^k (k \in \mathbb{N})$. p is assumed unique. $\Lambda = \Lambda^I \cup \Lambda^O \cup \{!\delta\}$: Λ^I represents the set of input actions, (Λ^O) the set of output actions,*
- *$\rightarrow$ is the finite transition set. A transition $(l_i, l_j, a(p), G, A)$, from the location $l_i \in L$ to $l_j \in L$, denoted $l_i \xrightarrow{a(p),G,A} l_j$ is labelled by: an action $a(p) \in \Lambda$, a guard G over $(p \cup V \cup T(p \cup V))$ which restricts the firing of the transition. $T(p \cup V)$ is a set of functions that return boolean values only (a.k.a. predicates) over $p \cup V$, an assignment function A which updates internal variables. A is of the form $(x := A_x)_{x \in V}$, where A_x is an expression over $V \cup p \cup T(p \cup V)$.*

An IOSTS is also associated with an IOLTS (Input/Output Labelled Transition System) to formulate its semantics. Intuitively, IOLTS semantics correspond to valued automata without symbolic variables, which are often infinite: IOLTS states are labelled by internal variable valuations while transitions are labelled by actions and parameter valuations. The semantics of an IOSTS $\mathcal{S} =< L, l_0, V, V_0, I, \Lambda, \rightarrow>$ is the IOLTS $[\![\mathcal{S}]\!] =< Q, q_0, \Sigma, \rightarrow>$ composed of valued states in $Q = L \times D_V$, $q_0 = (l_0, V_0)$ which is the initial one, Σ which is the set of valued symbols, and $\rightarrow$ which is the transition relation. The IOLTS semantics definition of can be found in [3]. In short, for an IOSTS transition $l_1 \xrightarrow{a(p),G,A} l_2$, we obtain an IOLTS transition $(l_1, v) \xrightarrow{a(p),\theta} (l_2, v')$ with v a set of valuations over the internal variable set, if there exists a parameter valuation set θ such that the guard G evaluates to true with $v \cup \theta$. Once the transition is executed, the internal variables are assigned with v' derived from the assignment $A(v \cup \theta)$. Runs and traces of an IOSTS can now be defined from its semantics.

Definition 2 (Runs and traces) *For an IOSTS $\mathcal{S} =< L, l0, V, V0, I, \Lambda, \rightarrow>$, interpreted by its IOLTS semantics $[\![\mathcal{S}]\!] =< Q, q_0, \Sigma, \rightarrow>$, a run of $\mathcal{S}$, $q_0\alpha_0 q_1 ... q_{n-1}\alpha_{n-1}q_n$ is a sequence of terms $q_i\alpha_i q_{i+1}$ with $\alpha_i \in \Sigma$ a valued action and q_i, q_{i+1} two states of Q. $Run(\mathcal{S}) = Run([\![\mathcal{S}]\!])$ is the set of runs found in $[\![\mathcal{S}]\!]$.*

It follows that a trace of a run r is defined as the projection $proj_\Sigma(r)$ on actions. $Traces_F(\mathcal{S}) = Traces_F([\![\mathcal{S}]\!])$ is the set *of traces of all runs finished by states in $F \times D_V$.*

4. Model inference

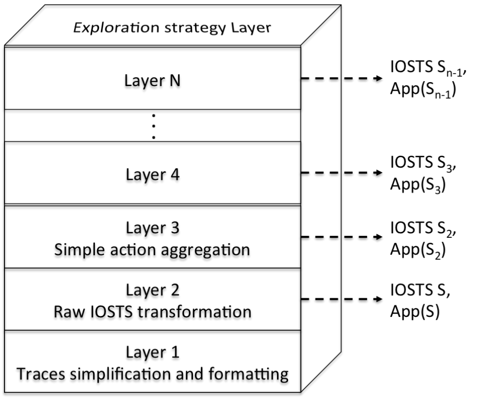


Fig. 2: Models generator layers

The Models generator is mainly composed of a rule-based system, adopting a forward chaining. Such a system separates the knowledge base from the reasoning: the former is expressed with data a.k.a. facts and the latter is realised with inference rules that are applied on the facts. Our Models generator initially takes traces as an initial knowledge base and owns inference rules organised into layers for trying to match the human expert behaviour. These layers are depicted in Figure 2.

Usually, when a human expert has to read traces of an application, he often filters them out to only keep those that are relevant to him. This step is done by the first layer whose role is to format the received raw traces into sequences of valued actions, and to delete those considered as unnecessary. The resulting structured trace set, denoted ST, is then given to the next layer. This process is incrementally done, i.e. every time new traces are given to the Models generator, these are formatted and filtered before being given to Layer 2. The remaining layers yield two IOSTSs each: the first one $\mathcal{S}_i (i \geq 1)$ has a tree structure derived from the traces. The second IOSTS, denoted $App(\mathcal{S}_i)$, is an approximation which captures more behaviours than $\mathcal{S}_i$. Both IOSTSs are minimised with a bisimulation minimisation technique. The role of Layer 2 is to carry out a first IOSTS transformation from the structured traces. The obtained IOSTSs are not re-generated each time new traces are received but are completed on the fly. The next layers 3 to N (with N a finite integer) are composed of rules that emulate the ability of a human expert to simplify transitions, to analyse the transition syntax for deducing its meaning in connection with the application, and to construct more abstract actions that aggregate a set of initial ones. Theses deductions are often not done in one step. This is why the Models generator supports a finite but not defined number of layers. Each of these layers i takes the IOSTS $\mathcal{S}_{i-1}$ given by the direct lower layer. This IOSTS, which represents the current base of facts, is analysed by the rules to infer another IOSTS whose expressiveness is more abstract than the previous one. The lowest layers (at least Layer 3) are composed of generic rules that can be reused on several applications of the same type. In contrast, the highest layers own the most precise rules that may be dedicated to one specific application.

In the following, and for readability purpose, we chose to represent inference rules using this format: *When conditions on facts Then actions on facts* (format borrowed from the Drools inference engine [1]). Independently on the application type, the Layers 2 to N handle the following fact types: *Location* which represents an IOSTS location, and *Transition*, which represents an IOSTS transition, composed of two Locations Linit, Lfinal, and two data collections Guard and Assign. Now, it is manifest that the inference of models has to be done in a finite time and in a deterministic way. To reach that purpose, we formulate the following hypotheses on the inference rules:

1) (finite complexity): a rule can only be applied a limited number of times on the same knowledge base,
2) (soundness): the inference rules are Modus Ponens,
3) (no implicit knowledge elimination): after the application of a rule r expressed by the relation $r : T_i \rightarrow T_{i+1} (i \geq 2)$, with T_i a Transition base, for all transition $t = (l_n, l_m, a(p), G, A)$ extracted from T_{i+1}, l_n is reachable from l_0.

In the following, we detail these layers in the context of Web applications while giving some rule examples.

4.1 Layer 1: Trace filtering

Traces of Web applications are based upon the HTTP protocol, conceived in such a way that each HTTP request is followed by only one HTTP response. Consequently, the traces, given to Layer 1, are sequences of couples (HTTP request, HTTP response). This layer begins formatting these couples so that these might be analysed in a more convenient way.

For a couple (HTTP request, HTTP response), we extract the following information: the HTTP verb, the target URI, the request content which is a collection of data (headers, content), and the response content which is the collection (HTTP status, headers, response content). An header may also be a collection of data or may be null. Contents are texts e.g., HTML texts. Since we wish translating such traces into IOSTSs, we turn these textual items into a structured valued action $(a(p), \theta)$ with a the HTTP verb and θ a valuation over the variable set $p = \{URI, request, response\}$. This is captured by the following proposition:

Definition 3 (Structured HTTP Traces) *Let $t = req_1$, $resp_1$, ..., req_n, $resp_n$ be a raw HTTP trace composed of an alternate sequence of HTTP request req_i and HTTP response $resp_i$. The structured HTTP trace σ of t is the sequence $(a_1(p), \theta_1)...(a_n(p), \theta_n)$ where:*

- *a_i is the HTTP verb used to make the request in req_i,*
- *p is the parameter set $\{URI, request, response\}$,*
- *θ_i is a valuation $p \rightarrow D_p$ which assigns a value to each variables of p. θ is deduced from the values extracted from req_i and $resp_i$.*

[1] http://www.jboss.org/drools/

The resulting trace set derived from raw HTTP traces is denoted ST.

Now, the structured traces can be filtered. Given a main request performed by a user, many other sub-requests are also sent by a browser in order to fetch images, CSS and JavaScript files. Generally speaking, these do not enlighten a peculiar functional behaviour of the application. This is why we propose to add rules in Layer 1 to filter these sub-requests out from the traces. Such sub-requests can be identified by different ways, e.g., by focussing on the file extension found at the end of the URI, or on the Content-Type value of the request headers. Consequently, we created a set of rules, constituted of conditions on the HTTP content found in an action, that remove valued actions when the condition is met.

After the instantiation of the Layer 1 rules, we obtain a formatted and filtered trace set ST composed of valued actions. Now, we are ready to extract the first IOSTSs.

Completeness, soundness, complexity: HTTP traces are sequences of valued actions modelled with positive facts. Typically, they form Horn clauses. Furthermore, inference rules are Modus Ponens (soundness hypothesis). Consequently, Layer 1 is sound and complete. Keeping in mind the (finite complexity) hypothesis, its complexity is proportional to $\mathcal{O}m(k+1)$ with m the valued action number and k the rule number. (at worst, every action is covered $k+1$ times).

4.2 Layer 2: transformation of the traces into IOSTSs

Intuitively, the IOSTS transformation relies upon the IOLTS semantics transformation that is achieved in a backward manner. Two IOSTSs are built: the former, structured as a tree, represents the original traces modelled with an IOSTS formalism. The latter is an over approximation of the former. These IOSTSs are generated by performing the following steps:
1. the associated runs are computed from the structured traces by injecting states between valued actions,
2. the first IOSTS denoted $\mathcal{S}_1$ is derived from these runs and minimised,
3. a second IOSTS, denoted $App(\mathcal{S}_1)$, is obtained from $\mathcal{S}_1$ by merging some of its locations, and by also applying a minimisation technique.

These steps are detailed below:

4.2.1 Traces to runs

Given a trace σ, a run r is firstly derived by constructing and injecting states on the right and left sides of each valued action of σ. Keeping in mind the IOLTS semantics definition, a state shall be modelled by the couple $((URI,k),v_\emptyset)$ with $v_\emptyset$ the empty valuation. (URI,k) is a couple composed of a URI and of an integer $(k \geq 0)$. Typically, a couple (URI,k) shall be a location of the future IOSTS. Since we wish to preserve the sequential order of the actions found in the traces, when a URI previously encountered is once more detected, the resulting state is composed of the URI accompanied with an integer, which is incremented to yield a new and unique state. Due to lack of room, the algorithm translating the structured traces into a run set is not provided in this paper but can be found in [7].

4.2.2 IOSTS generation

The first IOSTS $\mathcal{S}_1$ is derived from the run set SR. It corresponds to a tree composed of paths, each expressing one trace starting from the same initial location.

Definition 4 *Given a run set SR, the IOSTS $\mathcal{S}_1$ is called the IOSTS tree of SR and corresponds to the tuple $< L_{\mathcal{S}_1}, l0_{\mathcal{S}_1}, V_{\mathcal{S}_1}, V0_{\mathcal{S}_1}, I_{\mathcal{S}_1}, \Lambda_{\mathcal{S}_1}, \rightarrow_{\mathcal{S}_1} >$ such that:*
- *$L_{\mathcal{S}_1} = \{l_i \mid \exists r \in SR, (li, v_\emptyset)$ is a state found in $r\}$,*
- *$l0_{\mathcal{S}_1}$ is the initial location such that $\forall r \in SR$, r starts with $(l0_{\mathcal{S}_1}, v_\emptyset)$,*
- *$V_{\mathcal{S}_1} = \emptyset$, $V0_{\mathcal{S}_1} = v_\emptyset$,*
- *$\Lambda_{\mathcal{S}_1} = \{a_i(p) \mid \exists r \in SR, (a_i(p), \theta_i)$ is a valued action in $r\}$,*
- *$\rightarrow_{\mathcal{S}_1}$ is defined by the following inference rule applied on every element $r \in SR$:*

$$\boxed{\begin{array}{l} s_i(a_i(p),\theta_i)s_{i+1} \textit{ is a term of } r, s_i = (l_i, v_\emptyset), \\ s_{i+1} = (l_{i+1}, v_\emptyset), G_i = \bigwedge_{(x_i=vi)\in\theta_i} x_i == vi \\ \vdash \\ l_i \xrightarrow{a_i(p),G_i,(x:=x)_{x\in V}}_{\mathcal{S}_1} l_{i+1} \end{array}}$$

From an IOSTS tree $\mathcal{S}_1$, an over-approximation IOSTS can now be straightforwardly deduced by merging together all the locations of the form $(URI,k)_{k\geq 0}$ into a single location (URI). This possibly cyclic IOSTS usually expresses more behaviours and should be strongly reduced in term of location size. But this is also an approximation in the sense that new action sequences, which do not exist into the initial traces, may appear. This model may be particularly interesting to help establish a complete model or to increase the coverage of specific testing methods e.g., security testing, since more behaviours are represented. In contrast, it is manifest that a conformance testing method must not take this model as a reference to generate test cases.

Definition 5 *Let $\mathcal{S}_1$ be an IOSTS tree of SR. The approximation of $\mathcal{S}_1$, denoted $App(\mathcal{S}_1)$, is the IOSTS $< L_{App}, l0_{App}, V_{App}, V0_{App}, I_{App}, \Lambda_{App}, \rightarrow_{App} >$ such that:*
- *$L_{App} = \{(URI) \mid (URI,k) \in L_{\mathcal{S}_1}, k \geq 0\}$,*
- *$l0_{App} = l0_{\mathcal{S}_1}$, $V_{App} = V_{\mathcal{S}_1}$, $V0_{App} = V0_{\mathcal{S}_1}$, $\Lambda_{App} = \Lambda_{\mathcal{S}_1}$,*
- *$\rightarrow_{App} = \{(URI_m) \xrightarrow{a(p),G,A} (URI_n) \mid (URI_m,k) \xrightarrow{a(p),G,A} (URI_n,l) \in \rightarrow_{\mathcal{S}_1}\} \cup \{l0_{App} \xrightarrow{a(p),G,A} (URI_n) \mid l0_{\mathcal{S}_1} \xrightarrow{a(p),G,A} (URI_n,l) \in \rightarrow_{\mathcal{S}_1}\} \cup \{(URI_m) \xrightarrow{a(p),G,A} l0_{App} \mid (URI_m,k) \xrightarrow{a(p),G,A} l0_{\mathcal{S}_1} \in \rightarrow_{\mathcal{S}_1}\} (k \geq 0, l \geq 0)$.*

4.2.3 IOSTS minimisation

Both IOSTSs are reduced in term of location size by applying a bisimulation minimisation technique which still preserves the functional behaviours expressed in the original model. Intuitively, this minimisation constructs the state sets (blocks) that are bisimilar equivalent. Two states are said bisimilar equivalent, denoted $q \sim q'$ iff they simulate each other and go to states from where they can simulate each other again. A bisimulation minimisation algorithm can be found in [2].

Completeness, soundness, complexity: Layer 2 takes any structured trace set obtained from HTTP traces. If the trace set is empty then the resulting IOSTS $\mathcal{S}_1$ has a single location l_0. A structured trace set is translated into an IOSTS in finite time: every valued action of a trace is covered once to construct states, then every run is lifted to the level of one IOSTS path starting from the initial location. Afterwards, the IOSTS is minimised with the algorithm presented in [2]. Its complexity is proportional to $\mathcal{O}(mlog(m+1))$ with m the number of valued actions. The soundness of Layer 2 is based upon the notion of traces: an IOSTS $\mathcal{S}_1$ and its approximation are composed of transition sequences derived from runs in SR, itself obtained from the structured trace set ST. As defined, the behaviours encoded in ST and $\mathcal{S}_1$ are equivalent since (ordered) runs are transformed into ordered IOSTS sequences. On the other hand, the approximation of $\mathcal{S}_1$ shares the behaviours found in $\mathcal{S}_1$ and ST but also describes new behaviours. This is captured by the following Proposition:

Proposition 6 *Let ST be a trace set and SR be is corresponding run set. If $\mathcal{S}_1$ is the IOSTS tree of SR, we have $Traces(\mathcal{S}_1) = ST$ and $Traces(App(\mathcal{S}_1)) \supseteq ST$.*

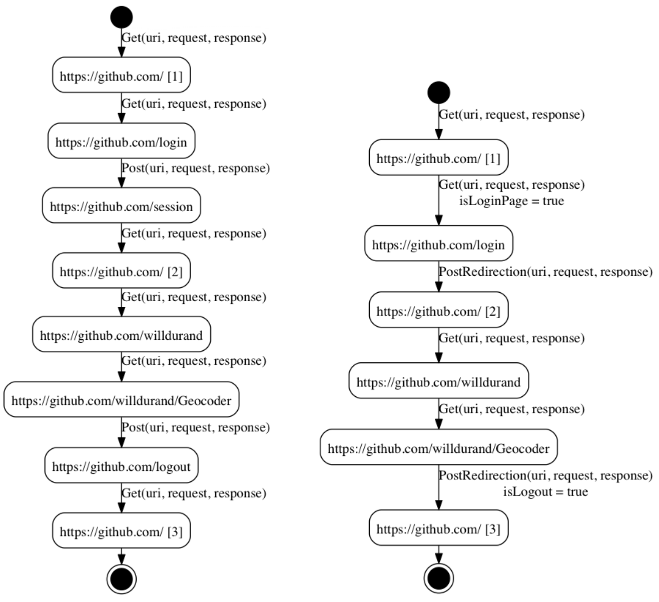
The proof is this proposition is Given in [7]. For sake of readability, we do not provide the rules of Layer 2, which match the above definitions and algorithms. Instead, we illustrate an IOSTS generation example below:

Example 4.1

We take as example a trace obtained from the Github Web site [2] after having executed the following actions: login with an existing account, choose an existing project, and logout. These few actions already produced a large set of requests and responses. The trace filtering from this example returns the following structured traces where the request and response parts are concealed for readability:

```
GET(https://github.com/)
GET(https://github.com/login)
POST(https://github.com/session)
GET(https://github.com/)
GET(https://github.com/willdurand)
GET(https://github.com/willdurand/Geocoder)
POST(https://github.com/logout)
GET(https://github.com/)
```

After having applied rules of Layer 2, we obtain the IOSTS of Figure 3(a). Locations are labelled by the URI found in the request plus an integer to keep the tree structure of the initial traces. Actions are composed of the HTTP verb enriched with the variables URI, request, and response. This IOSTS exactly reflects the trace behaviour but is still difficult to interpret. More abstract actions shall be deduced by the next layers.


Get(uri, request, response)
https://github.com/ [1]
Get(uri, request, response)
https://github.com/login
Post(uri, request, response)
https://github.com/session
Get(uri, request, response)
https://github.com/ [2]
Get(uri, request, response)
https://github.com/willdurand
Get(uri, request, response)
https://github.com/willdurand/Geocoder
Post(uri, request, response)
https://github.com/logout
Get(uri, request, response)
https://github.com/ [3]
Get(uri, request, response)
https://github.com/ [1]
Get(uri, request, response)
isLoginPage = true
https://github.com/login
PostRedirection(uri, request, response)
https://github.com/ [2]
Get(uri, request, response)
https://github.com/willdurand
Get(uri, request, response)
https://github.com/willdurand/Geocoder
PostRedirection(uri, request, response)
isLogout = true
https://github.com/ [3]

(a) IOSTS tree $\mathcal{S}_1$

(b) IOSTS $App(\mathcal{S}_2)$

Fig. 3: Approximation models

4.3 Layers 3-N: IOSTS Abstraction

As stated earlier, the rules of the upper layers analyse the transitions of the current IOSTS for trying to enrich its semantics while reducing its size. Given an IOSTS $\mathcal{S}_1$, every next layer carries out the following steps:

1. apply the rules of the layer and infer a new knowledge base (new IOSTS $\mathcal{S}_i$, $i \geq 2$),
2. derive $App(\mathcal{S}_i)$ and apply a bisimulation minimisation on both,
3. store the two IOSTSs.

Without loss of generality, we now restrict the rule structure to keep a link between the generated IOSTSs. Thereby, every rule of Layer i ($i \geq 3$) either enriches the sense of the actions (transition per transition) or aggregates transition sequences into one unique new transition to make the obtained IOSTSs more abstract. It results in an IOSTS $\mathcal{S}_i$ exclusively composed by some locations of the first IOSTS $\mathcal{S}_1$. Consequently, for a transition or path of $\mathcal{S}_i$, we can still retrieve the concrete path of $\mathcal{S}_1$. This is captured by the following proposition:

Proposition 7 *Let $\mathcal{S}_1$ be the first IOSTS generated from the structured trace set ST. The IOSTS $\mathcal{S}_i (i > 1)$ produced by Layer i has a location set $L_{\mathcal{S}_i}$ such that $L_{\mathcal{S}_i} \subseteq L_{\mathcal{S}_1}$.*

Completeness, soundness, complexity: the knowledge base is exclusively constituted by (positive) Transition facts that have an Horn form. The rules of these layers are Modus Ponens (soundness hypothesis). Therefore, these inference rules are sound and complete. Furthermore, a behaviour encoded in an IOSTS $\mathcal{S}_i$ cannot be lost in $\mathcal{S}_i$. With regards to the (no implicit knowledge elimination) hypothesis and to Proposition 7, the transitions of $\mathcal{S}_i$ are either unchanged, enriched or combined

[2] https://github.com/

together into a new transition. The application of these layers ends in a finite time ((finite complexity) hypothesis) and the complexity of each is proportional to $\mathcal{O}m(k)$ with m the transition number and k the rule number.

In the following, we detail two layers specialised for Web applications:

4.3.1 Layer 3

As stated above, Layer 3 corresponds to a set of generic rules that can be applied on a large set of applications belonging to the same category. This layer has two roles:

- the enrichment of the meaning captured in transitions. In this step, we have chosen to mark the transitions with new internal variables. These shall help deduce more abstract actions in the upper layers. For example, the rule depicted in Figure 4 aims at recognising the receipt of a login page: if the response content, which is received after a request sent with the GET method, contains a login form, then this transition is marked as a "login page" with the assignment on the variable isLoginPage,
- the generic aggregation of some successive transitions. Here, some transitions (two or more) are analysed in the conditional part of the rule. When the rule condition is met then the successive transitions are replaced by one transition carrying a new action. The rule of Figure 5 corresponds to a simple transition aggregation, introducing a new $PostRedirection$ action.

```
rule "Identify Login Page"
when
    $t: Transition(Action == GET, Guard.
      response.content contains('login-form'))
then
    modify ($t) { Assign.add("isLoginPage:=true") }
end
```

Fig. 4: Login page recognition rule

```
rule "Identify Redirection after a Post"
when
    $t1: Transition(Action == POST and
       (Guard.response.status = 301 or Guard.response.
       status = 302) and $l1final := Lfinal)
    $t2: Transition(Action == GET, linit == $l1final,
       $l2linit:=Linit)
    not (Transition (Linit == $l2linit))
then
    insert(new Transition("PostRedirection", Guard(
       $t1.Guard, $t2.Guard), Assign($t1.Assign,
       $t2.Assign), $t1.Linit, $t2.Lfinal );
    retract($t1);
    retract($t2);
end
```

Fig. 5: Redirection recognition rule

Example 4.2 When we apply these rules on the IOSTS example of Figure 3(a), we obtain a new IOSTS, illustrated in Figure 3(b), which has 6 transitions instead of 9 initially. However, it does not reflect clearly the initial scenario yet. Rules deducing more abstract actions are required. These are found in the next layer.

4.3.2 Layer 4

This layer allows to infer a more abstract model composed of more expressive actions. Its rules may have different forms:

- they can be applied on a single transition. In this case, the rule replaces the transition action to add more sense,
- the rules can also aggregate several successive transitions up to complete paths into one transition labelled by a more abstract action. For instance, the rule illustrated in Figure 6 recognises a user authentication thanks to the variable "isLoginPage" added by Layer 3.

```
rule "Identify Authentication"
when
    $t1: Transition(Action == GET,
       Assign contains "isLoginPage:= true",
    $t1final:=Lfinal)
    $t2: Transition(Action == PostRedirection,
       Linit == $t1lfinal, $t2linit:=Linit)
    not (Transition (Linit == $t2linit))
then
    insert(new Transition("Authentication",
       Guard($t1.Guard,$t2.Guard), Assign($t1.Assign,
       $t2.Assign), $t1.Linit, $t2.Lfinal );
    retract($t1);
    retract($t2);
end
```

Fig. 6: Authentication recognition rule

Other rules can also be application-specific, so that these bring specific new knowledge to the model. For instance, the GitHub Web application has a dedicated URL grammar (a.k.a. routing system). GitHub users own a profile page that is available at: $https : //github.com/username$ where $username$ is the nickname of the user. However, some items are reserved e.g., *edu* and *explore*. The rule given in Figure 7 is based upon this structure and produces a new action $Showprofile$ offering more sense. We did the same for projects as well, introducing a $Showproject$ action.

```
rule "GitHub profile pages"
when
    $t: Transition(action == GET, (
      Guard.uri matches "/[a-zA-Z0-9]+$",
      Guard.uri not in [ "/edu", "/explore" ]))
then
    modify ($t) (SetAction("Showprofile"));
end
```

Fig. 7: User profile recognition rule

Example 4.3 The application of the previous rules leads to the final IOSTS depicted in Figure 8, owning actions that have a precise meaning, and now clearly describing the application behaviour.

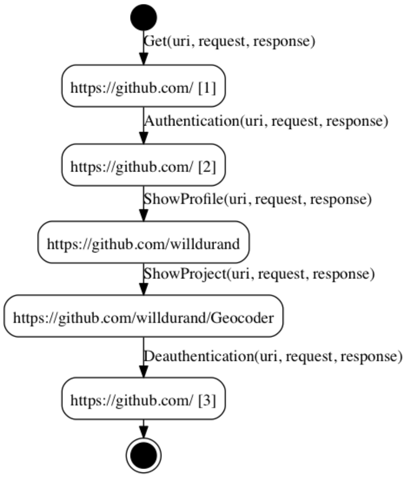

Get(uri, request, response)
https://github.com/ [1]
Authentication(uri, request, response)
https://github.com/ [2]
ShowProfile(uri, request, response)
https://github.com/willdurand
ShowProject(uri, request, response)
https://github.com/willdurand/Geocoder
Deauthentication(uri, request, response)
https://github.com/ [3]

Fig. 8: IOSTS $App(\mathcal{S}_3)$ obtained from Layer 4

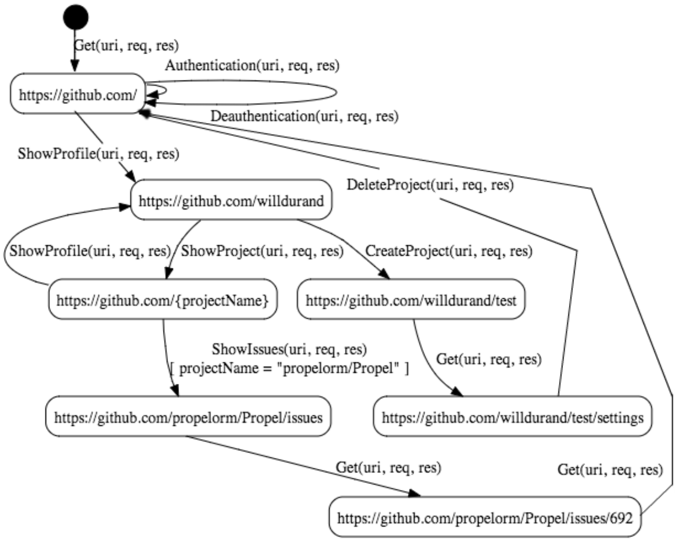

Get(uri, req, res)
Authentication(uri, req, res)
https://github.com/
Deauthentication(uri, req, res)
ShowProfile(uri, req, res)
DeleteProject(uri, req, res)
https://github.com/willdurand
ShowProfile(uri, req, res)
ShowProject(uri, req, res)
CreateProject(uri, req, res)
https://github.com/{projectName}
https://github.com/willdurand/test
ShowIssues(uri, req, res)
[projectName = "propelorm/Propel"]
Get(uri, req, res)
https://github.com/propelorm/Propel/issues
https://github.com/willdurand/test/settings
Get(uri, req, res)
Get(uri, req, res)
https://github.com/propelorm/Propel/issues/692

Fig. 9: IOSTS $App(\mathcal{S}_4)$ obtained from the Github Web site

5. Experimentation

The framework presented earlier has been implemented in a prototype tool called *Autofunk* (Automatic Functional model inference). A user interacts with Autofunk through a Web interface and either gives a URL or a file containing traces formatted with the HTTP Archive (HAR) format, the defacto standard for describing HTTP traces, used by various HTTP related tools (many HTTP monitoring tools, and Web browsers such as Mozilla Firefox and Google Chrome). The JBoss Drools Expert tool has been chosen to implement the rule-based system. Such an engine leverages Oriented Object Programming in the rule statements and takes knowledge bases given as Java objects (Location, Transition, GET, POST objects in this work).

From the Github Web site, we recorded a trace set composed of 840 HTTP requests / responses. Then, we applied Autofunk on them with a Models generator composed of 5 layers gathering 18 rules whose 3 are specialised to Github. After the trace filtering (Layer 1), we obtain a first IOSTS tree composed of 28 transitions. The next 4 layers automatically infer a last IOSTS tree $\mathcal{S}_4$ composed of 13 transitions whose 7 have a clear and intelligible meaning. Its approximation $App(\mathcal{S}_4)$ is illustrated in Figure 9. Most of its actions have a precise meaning reflecting the user interactions while the trace recording. Now, one can easily deduce that the user created, chose, deleted some projets and read the issues of others.

6. Conclusion

This paper presents an original approach combining model inference and expert systems to derive IOSTSs models. Our proposal yields several models, reflecting different levels of abstractions of the same application with the use of inference rules that capture the knowledge of an expert. Our approach can be applied on all applications that are able to produce traces.

We applied our framework on Web applications as a premise. In the future, we intend to apply it on industrial systems to ease their diagnostics. But this kind of system brings several issues not yet addressed in the model inference area. For instance, industrial systems may include asynchronous actions and timed properties. At the moment, our solution does not yet support this kind of properties. Furthermore, writing rules may be as tough as writing models in some cases. This is why we are working on a human interface which helps design rules from a trace set example. We also plan to add a test case generation module for regression testing.

References

[1] V. Dallmeier, M. Burger, T. Orth, and A. Zeller. Webmate: a tool for testing web 2.0 applications. In *Proceedings of the Workshop on JavaScript Tools*, JSTools '12, pages 11–15, New York, NY, USA, 2012. ACM.

[2] J.-C. Fernandez. An implementation of an efficient algorithm for bisimulation equivalence. *Science of Computer Programming*, 13:13–219, 1989.

[3] L. Frantzen, J. Tretmans, and T. Willemse. Test Generation Based on Symbolic Specifications. In J. Grabowski and B. Nielsen, editors, *FATES 2004*, number 3395 in Lecture Notes in Computer Science, pages 1–15. Springer, 2005.

[4] A. Memon, I. Banerjee, and A. Nagarajan. Gui ripping: Reverse engineering of graphical user interfaces for testing. In *Proceedings of the 10th Working Conference on Reverse Engineering*, WCRE '03, pages 260–, Washington, DC, USA, 2003. IEEE Computer Society.

[5] A. Mesbah, A. van Deursen, and S. Lenselink. Crawling Ajax-based web applications through dynamic analysis of user interface state changes. *ACM Transactions on the Web (TWEB)*, 6(1):3:1–3:30, 2012.

[6] M. Pradel and T. R. Gross. Automatic generation of object usage specifications from large method traces. In *Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering*, ASE '09, pages 371–382, Washington, DC, USA, 2009. IEEE Computer Society.

[7] S. Salva and W. Durand. Model inference combining expert systems and formal models. Technical report, LIMOS, http://sebastien.salva.free.fr/RR-14-04.pdf, 2014. LIMOS Research report RR-14-04.

[8] W. Yang, M. R. Prasad, and T. Xie. A grey-box approach for automated gui-model generation of mobile applications. In *Proceedings of the 16th international conference on Fundamental Approaches to Software Engineering*, FASE'13, pages 250–265, Berlin, Heidelberg, 2013. Springer-Verlag.

[9] H. Zhong, L. Zhang, T. Xie, and H. Mei. Inferring specifications for resources from natural language api documentation. *Autom. Softw. Eng.*, 18(3-4):227–261, 2011.

Web Accessible for the Deaf and Blind People

AbdulRahman Saud Alsaif
Department of Software Engineering, Florida Institute of Technology
Melbourne, Florida, 32901
alsaif789@hotmail.com

Abstract—the World Wide Web is used nowadays by a variety of normal users and users with disabilities in order to gather information look for resources and access services. Many web sites are designed and improved to be easy for the majority of people who are without disability. However, the percentage of disable people in the society has been quickly increasing today. Researchers and governmental leader have to pay little attention to their requirements and needs when planning, design, and improve Web sites. Therefore, this paper is a collection of some research papers representing by some scientists or researchers in Web System Evolution conferences and other conferences to give some suggestions and advice on how to access the Internet and improve the accessibility of hearing impaired and sight impaired people. Thus, people with disabilities can understand, navigate, perceive, communicate with other people and interact with the web.

Keywords— hearing impaired, sight Impaired, W3C, WAI.

I. INTRODUCTION

Web accessibility today is one of the essential issues for the improvement and usability of web sites and applications. Web sites today are mostly inaccessible to the disabled people. According to what Geoff Freed, the director of the Web Access Project for Boston-based, has said, "Only 1% of web developers have taken any action to make their sites more accessible to the disabled [1]." Therefore, this shows a critical issue. The proportion of disabled people today has been quickly growing in the world. For example, In the United States of America, the population of people with different kinds of disabilities (the hearing impaired, the sight impaired or others) has been evaluated to be about 40s millions. In addition, the number of people who face challenges to see words, letters, or to distinguish between colors on computer screens has been evaluated to be in range of 4 millions [1].While, web sites and applications today continue to improve, and evolve, disabled people are increasingly finding themselves at a disadvantage.

As a graduate student and developer, I would like to present their problems in how to deal with computers. In addition, developers may need to be aware, and to pay attention to those disabled people in order to make their life easier. Disabled people have to feel comfortable and a member of the world's society. Thus, web accessibility has been reported and regulated by the World Wide Web consortium, which has boosted the Web Accessibility Initiative in order to improve strategies, resources, guidelines in order to support disabled people to make their lives better [2]. Moreover, in the last ten years, there are some researchers and developers have discussed the issues related to the usability of Web Sites and how to make them accessible to disabled people.

In this paper, I divided it to two major topics. The first major is how to make Web Sites accessible for hearing impaired or deaf people. The second major is how to improve usability of Web Sites for blinds or sight impaired. In addition, the paper is organized as follows: in section 2 a synopsis paragraph about Web Accessibility. Then, in section 3 I talk about the hearing impaired and the sight impaired in general. After that, in section 4 and 5 I go deep to define each of them, find some solutions, and define some tools that being used today.

II. WEB ACCESSIBILITY

Web accessibility is the way of making Web Sites easy, available, and accessible for all users, including people with disabilities (e.g., blind and deaf people) [3]. In addition, Web Sites need to be improved and designed using easy tools and techniques, so that all users could have the equal right to browse, access, and search over the Internet. Therefore, people with disabilities can navigate, perceive, understand, and interact with Web, and that they can communicate with other people, contribute to the Web and share knowledge with others.

Many disabled people today have disabilities that could affect their lives and ways of using Web. Moreover, most Web sites nowadays are inaccessible which make it complicated and hopeless for many people with disabilities to access the Web. As more accessible Web sites available, people with disabilities are eligible to use, share, and contribute to the Web more effectively.

A. Why to make the Web Accessible

It is very imperative that the Web Sites be accessible for all kinds of users, including users with disabilities in order to provide equal rights and equal occasion. Accessible Web Sites can also help people with disabilities more lively contribute in the society. In addition, access to information makes opportunities and empowers all people to participate. Therefore, this will help people with disabilities more than others, so that they can benefit from huge services, information, resources available on the Internet. In fact, there is one of five people disabled [1]. This percentage will increase as the number of population has obviously increased.

Copyright © 2014 CSREA Press, ISBN: 1-60132-286-0; Printed in the United States of America

B. How to make the Web Accessible

There are some limitations and barriers may make moving to accessible Web Sites more difficult. The World Wide Web Consortium lists seven common accessibility barriers [1] which are:

1. Pictures with no alternate text.
2. Image map without alternative text.
3. Do not know how to reorganize the element on pages.
4. Uncaptioned audio or undescribed video.
5. Scarcity of information for users who cannot access frames or scripts.
6. Complicated tables.
7. Building sites with poor color contrast.

However, the World Wide Web Consortium (W3C) has countered the issue of the accessibility of Web sites by regulating some rules and guidelines to follow and creating the Web Accessibility Initiative (WAI) [4]. Therefore, WAI guidelines and rules are considered the international standard for Web accessibility. In addition, to make your Web Site accessible, WAI states that accessibility must be achieved and considered when Web Site designed. Web sites' content has to be practicable, functional, perceivable, easy and understandable by the majority possible range of users and convenient with a wide range of assistive technologies, now and in the future [4].

Furthermore, WAI sets some rules and guidelines related to Web Sites accessibility [1] which are Web content accessibility guidelines, authoring tool accessibility guidelines, and user agent accessibility guidelines. Web content accessibility guidelines, which may benefit people with disabilities, builds of two things. First, the Web Site's content should be accessible and clear despite a user's disability and the limitation of any hardware or software he/she uses. Second, the Web Site's content should be understandable and navigable by using plain and simple language and navigation so that people with disabilities could quickly and easily understand the content and able to the orient themselves [1].

Developers and designers should be aware when design and implement Web Sites and follow the fourteen rules and guidelines that set by W3C [1] which are:

1. Provide sensible alternatives solutions to auditory users and visual context.
2. Using alternative and do not base on color alone.
3. Use markup to highlight and style sheets.
4. Try to use easy natural language.
5. Make tables easy to convert.
6. Ensure also that pages featuring are easy to transform.
7. Ensure user control of time-sensitive content changes.
8. Ensure embedded user interfaces are direct to access.
9. Design independent devices.
10. Use temporary solutions.
11. Use W3C technologies and rules.
12. Provide context and all information for accessibility.
13. Provide obvious navigation techniques.
14. Ensure that the written documents are evident and simple to comprehend.

III. HEARING IMPAIRED AND SIGHT IMPAIRED

In this section, I will give a brief written essay about the difference between hearing impaired and sight impaired in how to contribute, anticipate, share, and deal with the Internet. For more details, techniques, and case study skips this section and move to the next section. According to what (*Daniel M. Berry,* [5]) has said, It should be obvious what is perfect for the hearing impaired is not pretty for the sight impaired and vice versa. Nowadays, both HI and SI people complain and ask for their right. It is rare that to find a Web Site offers the most features for both to be in textual and graphical interfaces and at the same time has the features of speech and read the content. It is obvious that the number of Wes Sites that offers pictures, images, graphics, and written plain English language more than the aural Web Sites. However, both of them are enfranchising.

Daniel is a hearing impaired and has a good experience of a blind student who took one of his courses gives us some recommendation on how to make Wes Sites accessible. The first recommendation is when designers and developers implement the Web Sites; they should make it possible in two ways sounds and text or pictures. In addition, the sound and text should be synchronized together in order to diminish cognitive interference. The second recommendation is that the computer is to agree input from the user as voice and textual input. Because, many HI people are not capable to speak well, and many SI people find the visual content difficult [5].

To complete what he has said, the output from the machine or computer or any smart electronic devices should be in both sound so that the blind or sight impaired can get the result, and text, graphics or picture so that deaf or hearing impaired can get the output. In addition, if developers do the same what Daniel has said, it will be an original source and other media that generate from the source. If the web content is text, then the sound can be created by a voice synthesizer which is working on the text. For example, as I mention that Daniel has a good experience that one of his students is blind. So that student could use lip readable or text to read the text and at the same time listen to the generated sound. There is a technique called lip synching; it is a very useful technique that could be used to the synchronized sound with lip reading or text or graphics. On the other hand, if the source is a text in a phonetic alphabet character. Then this text should be clear to read and displayed [5].

One of the most things that Daniel has talked about it is the input. The machine should be built to accept different kinds of input. Some inputs that computers can accept are voice, which powered by voice recognition technology, keyboard, mouse, clicking on a button, typing a direct response, menu entries or making hand gestures. If the user has difficulty speaking clearly or cannot speak, as many HI people do. Then, voice input may not be the convenient tools, and the other input will be needed [5].

IV. HEARING IMPAIRED AND DEAF

Daniel is an academic teacher, and describes some difficulties that could face hearing impaired people in his paper (Requirements for maintaining Web access for Hearing Impaired Individuals). He is a hearing impaired person from birth and he used to understand the English language by reading lips. Then, he explains how it was his life and how he faces challenges in using a telephone. Moreover, reading lips in the telephone is so hard for him. He is not satisfied with the quality of the telephone. It will be more distortion if the sound is amplified. In addition, he says," the increased use of answering machines, voice mail, and voice-directed menu selection have taken away the possibility of my asking the person on the other end of a call if I understood her or of my requesting her to repeat what she just said. In essence, I have become disenfranchised from the telephone, so much so that I do not give out my phone number anymore [5]."

Therefore, he feels more comfortable with written communication. He used to use the Internet for communication since 1979. However, he was panic after he knew that the computer will have built to accept a different kind of inputs which is a voice interface. At the end, he felt that it is very important for him and hearing impaired people to participate to prevent disenfranchisement form the computer and the Internet and make them accessible.

A. Classification of Hearing Impaired people

First of all, a person who is not able to hear as well as normal hearing person or hearing thresholds of 25dB or better in both ears is considered to be a hearing loss person. In fig 1, Daniel describes his situation.

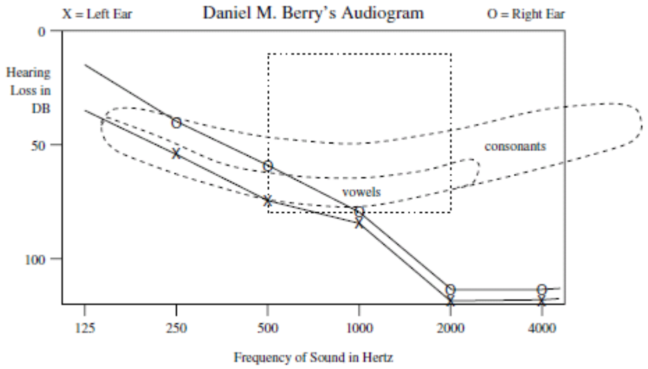


Figure 1

He says," An audiogram shows two plots, one for each ear. The plot for an ear shows for each frequency, the hearing loss of the ear at the frequency. The loss of an ear at a frequency is measured by determining the minimum volume required for the ear to hear a tone of the frequency. The more of the speech-understanding rectangle that lies below the plots for an ear is the more that the ear can help understand human speech [5]."

There are many ways to classify hearing impaired people to. The first way depends on the severity of the hearing loss. The second way is the length of time he has had the hearing loss. The last one is what kind of inputs hearing loss person requires in place of pure voice.

1. The severity of loss classification:

The author divided them to three groups [5]:

a) The first group is a person who has less than a 50db loss in all frequencies; that is, he can hear some frequencies.
b) The second group is a person who has greater than 100 db loss in all frequencies; that is, he is considered completely deaf.
c) The third group is a person who is neither in the first group nor in the second group. He has usable hearing in some ranges of frequencies and is totally deaf in other ranges of frequencies.

So that, people who are in the first group speak well and wear some hearing aid tools that amplify all frequencies. Nevertheless, people who are in the second group always cannot speak and they only sign and do not wear any aid tools since aids are useless. People who are in the third group sometimes they use aid tools to help them to hear or could use lip reading or only sign.

2. Length of time of Hearing loss:

Here we can classify Hearing impaired people depend on the long of time of being hearing loss [5]:

a) The first group is a person who has loss his hearing since before he could talk.
b) The second group is a person who has loss his hearing after he learned to talk.

So that, a person in the second group has already learned how to speak since he became hearing loss after he has learned to talk. In addition, he could make the sound correctly. However, a person in the first group could not understand the speech and they use to sign.

3. Kind of input classification

Here, the author classified hearing loss into three groups depend on what kind of inputs they used [5]:

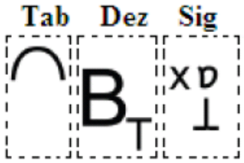
a) The first group is a person who requires signing.
b) The second group is a person who wears residual hearing aids and uses lip reading to understand speech as it is spoken.
c) The third group is a person who wears only residual hearing.

The most group that is considered to be hearing loss is first group. Signers are the largest group of hearing impaired how uses the Internet. Moreover, it is very hard for non hearing impaired to understand them.

B. How to help Hearing Impaired People:

One of the common techniques that are being used today is signed language. Signing language will be a significant contribution to the development of the Web accessible to deaf people. In addition, it would mean a great deal to their daily uses to the Internet. There are different variants of sign language American Sign Language (ASL), Greek Sign Language (SYNENNOESE), South African Sign Language (SASL-MT), Arabic Sign Language (ArSL-TS), Spanish Sign Language, Italian Sign Language, Japanese Sign Language, British Sign Language [7].

There are different well-known writing systems for singing language. The first one called Stokoe Notation. The stokoe notation system [8] is the world’s first phonemic script that has been developed by William Stokoe for writing American Sign Language (ASL). The original language notation contains of 55 symbols. In which it divided into three groups. Moreover, each group represents one of the important aspects of sign: ("tab" or sign location), ("dez" or handshape & orientation), and ("sig" or movement). For example, Stokoe notation for the American Sign Language for the term “don’t know” can be seen in Fig 2



Stokoe Notation of “don’t know” .

Figure 2

The second one called HamNoSys Notation [8]. The Hamburg Sign Language Notation System or HamNoSys is a phonetic transcription system which has its root in the Stokoe notation. The hamNoSys notation system includes about 210 iconic characters to represent the different sign aspects. It is more accurate that the Stokoe notation system. The third one called Sign Writing Notation, [8] which was originated from a choreographic notation system called Dance Writing. There are intuitive graphical symbols to record every sign and to represent hand shapes, palm orientation, movements, body locations, facial expressions and punctuation.

One of the most effective techniques that could be used to help to communicate is Singing Avatar System. An avatar system gives us a visual language alternative to displaying spoken massages within educational settings or workplaces that include deaf, and hearing loss. The first step is a speech recognition. This step is to convert an audio stream containing spoken words to a stream of text. Then, the text is converted into sign language phonetics, which consists of a combination of manual and non-manual signals including hand shape, position, orientation, as well as facial expression and body motion [7]. After that, the avatar system is instructed. Finally, all user motions are captured, and stored.

V. SIGHT IMPAIRED

According to the World Health Organization [9], the proportions of people who are estimated to be complete blind are 39 million people, and 246 million people are considered to be low vision. In addition, the percentage of blind or visual impaired people who lives in modern countries is rated to be about 90%. Moreover, blindness is defined to be unable to see, look for or lacking the sense of sight. There are many reasons to cause blindness which are uncorrected refractive errors, which is considered the main cause of visual impairment, cataract, and glaucoma.

As I mentioned before, researchers and governmental leader pay little attention to the visual impaired people needs when planning, design, and improve Web sites. They tend to design the Web Sites and application to be visual interface. Web interface in general comprises of three different kinds [10]. The first one is content. Content means all stuffs that are considered to be looked at such as images, graphics, text, videos. In addition, all these contents may involve together in one page. The second is interface semantics. There are different layouts for every Web Sites and different graphics; moreover, users may need to move between Web Site pages and using linking. The third one is navigation. So that, mobile from one page to another page is rely on the activation of links, and the user should be capable to visually identify links, guessing their meaning, and moving the pointer of the mouse directly over one of them.

In the last ten years, "many government and other international organizations such as the United Nations and European Union stat that the accessibility to services and information on the web is a fundamental right for any citizen, so the needs of people with disabilities must be taken into account by Web site and application developers [2]." So that, W3C sets some rules and guidelines in order to make accessible Web Sites. This will include the needs of sight impaired. W3C guidelines are eager to cover all technical aspects in order to make content and functionality can reach and access by users.

A. Requirements for Aural Web Sites:

There are three requirements [10] which are needed in order to make accessible Web Sites. The first one is the information architecture requirement. These requirements will help a user to comprehend and memorize the overall structure of Web Sites. The second is related to page navigation requirements. So that, users may need to comprehend, navigate, and access the content of the Web sites. The third is related to how the Web's contents interact. To accomplish these entire requirements, there are fourteen rules [10] will help developers and designers to make Web Sites accessible for all:

1. At the beginning and whenever necessary, giving a user quick aural glance of the web sites.
2. Besides that, give a user an aural semantic map of the whole application.

3. Provide a synopsis summary of any list.
4. Partition long lists in smaller meaningful chunks.
5. Do not use the sequence of physical pages. Provide a tool to emphasize the history of visited pieces of content.
6. Provide a semantic navigation button to allow a user to go “up” to the last visited list of items.
7. Creating aural page templates.
8. Minimize the number of templates.
9. Give a user a brief talk in how the page is organized and it is structure.
10. Read the first key message of the page.
11. Making accesses to section available whenever needed.
12. Allow a user to move freely (forward and backward) across page sections.
13. Allow the user to pause, and resume the dialogue flow.
14. Allow the user to re-play an item or an entire section.

B. Tools and Techniques to help Sight Impaired people:

W3C recommends some assistive technology software for blind users to use in order to provide highly accessible content. Home Page Reader (HPR) [11] is a computer program. HPR was developed by IBM from the work of Chieko Asakawa at IBM Japan. It was design for blind users, and developers to experience the blind users’ usability. The first step, all text can be spoken loudly so that they can make sure that the visually information in the graphics view corresponds to the information in the text view. It reads aloud the text on Web pages. However, it has some problems and defects [2] as following:

1. There are some contents that cannot be transferred as speech, such as images, flash animations, and other multimedia contents.
2. There are some contents that are difficult to proceed, such as hyperlinks or tables.
3. The listening of the contents of a Web page is very slow, with respect to the cognitive speech of the user so that the usability of Web pages can be very poor.

There is also another technology program called aDesigner. ADesigner is a disability simulator, which was developed for Web developers to help them ensure that their pages are accessible and usable for people with vision disabilities. ADesigner have complete set of tools for color contrast of the page, the font size. In addition, tools also check the page's compliance with accessibility guidelines. Moreover, each Web page is given an overall score. With this information, Web content developers get immediate feedback and can make the necessary modifications to address these problems before the content is published [11].

In addition, there are several softwares [12] such as JAWS and NVDA for visually impaired people. Most of the software uses screen reading technique. JAWS is one of the common softwares which is used mainly for documents. So that, JAWS read information displayed on a computer monitor loudly. Moreover, it reads aloud text within a document, information within dialog boxes and error messages. It also reads aloud menu selections, text with the graphical icons on the desktop. However, we may face another issue which is related to bilingual language. Are those screen reader softwares able to understand different language such as Arabic or not? It is a perfect idea for researchers to find solutions for this kind of challenge.

VI. CONCLUSION

In summation, making web sites accessible for all people including deaf and blind people is vital. All users should have the same equal rights so that they can access, browse, and research over the Internet. In this novel, I talk about the importance of making web sites accessible for disabled people and how to help them to make their lives easy. W3C sets some rules and guidelines for web sites' designers and developers to help disabled people to access web sites. After that, I explain the differences between hearing impaired and visual impaired people. In addition, I mention how to make the web accessible for both of them. Then, I talk about both of them. Moreover, there are some tools and techniques that will help disable people to make the Internet accessible and available.

For future work, researchers and scientists may need to think about bilingual tools. These tools just support some languages around the world. We need these tools to aid all different people with different languages. So that, there is a default language which a disable person speaks or knows and there is a popular international language which is English.

VII. ACKNOWLEDGMENT

I appreciate Dr. Scott Tilley from the Florida Institute of Technology (Software Engineering Dept.) to help me to stand and explain the idea of writing my first research paper. Also, I thank all researchers and scientists who spend their lives to help disable people to make their lives easy.

VIII. REFERENCES

[1] J. Carter and M. Markel, “Web accessibility for people with disabilities: an introduction for web developers,” Professional Communication, IEEE Transactions on, vol. 44, no. 4, pp. 225–233, 2001.

[2] C. Cesarano, A. R. Fasolino, and P. Tramontana, “Improving usability of web pages for blinds,” in Web Site Evolution, 2007. WSE 2007. 9th IEEE International Workshop on. IEEE, 2007, pp. 97–104.

[3] S. Sandhya and K. S. Devi, “Accessibility evaluation of websites using screen reader,” in Next Generation Web Services Practices (NWeSP), 2011 7th International Conference on. IEEE, 2011, pp. 338–341.

[4] G. A. Di Lucca, A. R. Fasolino, and P. Tramontana, “Web site accessibility: Identifying and fixing accessibility problems in client page code,” in Web Site Evolution, 2005.(WSE

2005). Seventh IEEEInternational Symposium on. IEEE, 2005, pp. 71–78.

[5] D. M. Berry, “Requirements for maintaining web access for hearing impaired individuals,” Software Quality Journal, vol. 12, no. 1, pp. 9–28, 2004.

[6] A. Cavender and R. E. Ladner, “Ntid international symposium on technology and deaf education: a review,” ACM SIGACCESS Accessibility and Computing, no. 97, pp. 3–13, 2010.

[7] Y. Bouzid and M. Jemni, “An animated avatar to interpret signwriting transcription,” in Electrical Engineering and Software Applications (ICEESA), 2013 International Conference on. IEEE, 2013, pp. 1–5.

[8] M. Muhammad, Y. Thoo, and S. Masra, “Sound navigation aid system for the vision impaired,” in Humanities, Science and Engineering (CHUSER), 2012 IEEE Colloquium on. IEEE, 2012, pp. 288–293.

[9] D. Bolchini, S. Colazzo, and P. Paolini, “Requirements for aural web sites,” in Web Site Evolution, 2006. WSE’06. Eighth IEEE International Symposium on. IEEE, 2006, pp. 75–82.

[10] H. Takagi, S. Kawanaka, M. Kobayashi, D. Sato, and C. Asakawa, “Collaborative web accessibility improvement: challenges and possibilities,” in Proceedings of the 11th international ACM SIGACCESS conference on Computers and accessibility. ACM, 2009, pp. 195– 202.

[11] M. R. Amin, B. Sylhet, B. Paul, and F. A. Khan, “Bi-lingual audio assistance supported screen reading software for the people with visual impairments.”

Comparative Evaluation of Executable Modeling Languages for Object-Oriented Modeling Education

S. Akayama[1], **K. Hisazumi**[2], **S. Kuboaki**[3], **S. Hiya**[1], **and A. Fukuda**[4]
[1]Graduate School of Information Science and Electrical Engineering, Kyushu University, Fukuoka, Japan
[2]System LSI Research Center, Kyushu University, Fukuoka, Japan
[3]Afrel Co.,Ltd., Tokyo, Japan
[4]Faculty of Information Science and Electrical Engineering, Kyushu University, Fukuoka, Japan

Abstract— *In this study, we investigated modeling education for novices that is based on executable modeling languages and model-driven development(MDD). MDD can verify the accuracy of models and generate the source code, which allows programmers to reduce the development time required for evaluating the software so that they can focus on the modeling process. Thus, modeling should be taught using MDD, because it allows students to acquire modeling skills in a short time.*

In this paper, two executable modeling languages for object-oriented modeling education, Executable UML and domain-specific modeling (DSM), are compared. We conducted two trial courses in which we used the Executable UML and DSM languages, respectively. The results show the effectiveness of using MDD in modeling education.

Keywords: MDD (Model-driven development), Executable UML, object-orientation, DSM (Domain-specific modeling), learning support

1. Introduction

Object-oriented modeling is widely used during embedded software development and is taught in many institutes of higher education. Since, modeling ensures good quality and productivity during software engineering [1], software development is shifting from manual programming to model-driven development (MDD) [2]. However, it is difficult to teach modeling. In the early stages of the learning process, students ask questions such as “How do we model?” and “Is this model equivalent to the specification?” MDD is used to verify the accuracy of models and generate source code, which allows programmers to minimize the development time so that they can focus on the modeling process. Therefore, modeling should be taught using MDD, because it allows students to acquire modeling skills quickly.

Previous studies on MDD and education can be divided into three categories: MDD education [3][4], including subjects such as the meta-model and mechanisms of MDD; developing MDD in system development exercises [5][6][7]; and software modeling education using MDD [8][9].

The purpose of this study was to demonstrate the effectiveness of using MDD in modeling education for novices.

In this study, we used two MDD languages. One was Executable UML [10], which has the following advantages. (1) It uses a small, well-defined subset of UML to represent domain models. (2) It provides a framework for the systematic development of software. (3) Executable UML models can be systematically translated into deployable software using tool support or by manual coding. Executable UML also has many advantages even when used for purposes other than automatic code generation. Thus, it has been used in the teaching of software analysis and design [6]. It has also been used to introduce the abstract thinking processes involved in modeling, before the more concrete thought processes involved in programming with the standard textual imperative programming languages used at high school level are introduced [8].

The second MDD language that we used was the domain-specific modeling (DSM) language, which has two aims: first, to raise the level of abstraction beyond programming by specifying the solution in a language that directly uses concepts and rules from a specific problem domain, and second, to generate from these high-level specifications final products in a chosen programming language or other form [11].

In the study presented in this paper, we compared the effectiveness of these two executable modeling languages for object-oriented modeling education. The results show the effectiveness of using MDD for object-oriented modeling education.

The remainder of this article is organized as follows. In Section 2, the learning objective is described. In Section 3, MDD is presented. In Sections 4 and 5, the contents and results of two trial courses are presented, while in Section 6, the support of modeling education using MDD is discussed. We provide our conclusions in Section 7.

2. Learning objective

The aim of the educational courses used in this study was to facilitate the acquisition of the minimum skill set required to create an object-oriented model in a short time. The educational subjects were software novices. We used class diagrams as a static model and state machine diagrams as a dynamic model.

The learning-specific goals were as follows.

Class diagrams
Assign responsibility to each class where the name represents the responsibility of the class.

State machine diagrams
Define the appropriate state name or event name and represent the dynamic behavior of each class using a state machine diagram.

3. MDD

Instead of directly coding software using programming languages, MDD developers model software systems using intuitive, highly expressive, graphical notations that provide a higher level of abstraction than native programming languages. In this approach, generators automatically create the code and implement the system functionalities [2].

MDD methods have the following advantages. (1) MDD facilitates the separation of design and implementation, so that developers can focus on modeling tasks. (2) MDD can simulate the model and therefore developers can conduct validation early in the development process. (3) Testing and improvement can be achieved quickly by automating the implementation process to repeat the design cycle.

Modeling education based on MDD allows students to repeat the model refinement process quickly. In addition, it provides a development environment that can be called by the programs, except the modeling target of the model, and therefore, students can focus solely on modeling the target application. Thus, they can learn modeling methodology more easily.

During the early stages of learning, students also gain experience in the software development process and the modification of existing models, which can be learned using MDD methods.

3.1 Executable UML

Executable UML is an extension of UML that allows models to be executed and translated into code via model compilers [5]. The basic elements of Executable UML are class diagrams, state machine diagrams, and action languages. The steps of the modeling process using Executable UML are as follows.

1) Define the subject matter, or domain, of the system.
2) Create class diagrams for the domain.
3) Model the life cycle using state machine diagrams.
4) Describe the procedure of each state machine using action languages.
5) Validate the models.
6) Compile the models and generate the source code.

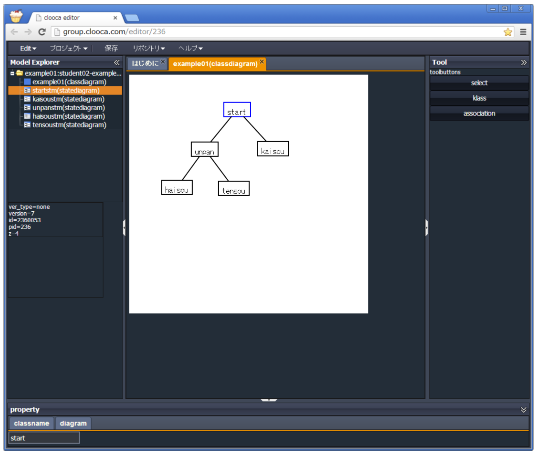
In this study, we used BridgePoint (Mentor Graphics) as the MDD tool.

3.2 DSM

Domain-specific modeling has two main objectives: first, to raise the level of abstraction beyond programming by specifying the solution in a language that directly uses concepts and rules from a specific problem domain, and second, to generate final products in a chosen programming language or other form from these high-level specifications [11].

When creating and using a DSM language, (domain-specific language (DSL))tools that can are required. We developed a domain-specific modeling (DSM) language for the purpose of teaching modeling using the social DSL platform “clooca [12].” This platform allows the user to make class diagrams and state machine diagrams. A class diagram consists of classes and relations, while state machine diagrams consist of states (including an initial state), event transmission states, events, and actions.

Fig. 1 shows the editing screen of the MDD tool.



```
ver_type=none
version=7
id=2360053
pid=236
z=4
```

classname: start

Fig. 1: Editing screen of the DSL tool.

4. Modeling education using Executable UML

4.1 Content of practical exercises

We conducted an Executable UML course (xUML course) for first-year college students. The framework of this course was hierarchical. It comprised three classes: basic, advanced, and project-based learning (PBL). Each class addressed several subjects. In the basic and advanced classes, the subjects were fundamental techniques, fundamental exercises, and integrated exercises. In the PBL class, the subjects were kick-off, PBL, and result presentation.

It was a characteristic of this course that the same material was used in all the classes. The materials used in the basic class and the advanced class were shared, although the development process was different. The material used in the PBL class was an extension of the material used in the basic and advanced classes. These common and different aspects helped students to learn throughout the development process.

The basic and advanced classes contained a set of a fundamental exercise and an integrated exercise. The educational items used in the basic and advanced classes are shown in Table 1. In addition, the C language was used as the implementation language, Executable UML as the modeling language, and BridgePoint (Mentor Graphics) as the MDD tool.

In fundamental exercise A, the students created a simple program, such as a line trace. In fundamental exercise B, the students learned the relationship between the model and the code by reverse modeling and refactoring the program. In the integrated exercise, the students could also acquire the skills to help them utilize the knowledge acquired during each step via programming and modeling a more complex system. The students developed the system by gradually increasing the level of abstraction, as follows. The basic class exercise involved only implementation, while the first half of the advanced class involved design and implementation by hand coding and the second half involved design and automatic transformation for coding using the MDD tool.

The goal of the integrated exercise was to develop an automated transport robot for a fictitious transportation company. The development objective was a vehicle robot developed using LEGO Mindstorms NXT (Fig. 2). The automated operations were transportation, forwarding, and sending round. Three types of operations were affected by the presence or absence of the delivery destinations or cargoes. A wall detector (sonar sensor) also monitored the delivery destinations, while the bumper (touch sensor) detected the forwarding destination. The challenge for the robot was to trace a black line in the course using a line monitor (light sensor) to make stops at the delivery destination, forwarding destination, or garage. The robot should behave appropriately at each point and deliver the cargo. Fig. 3 shows the exercise robot course.

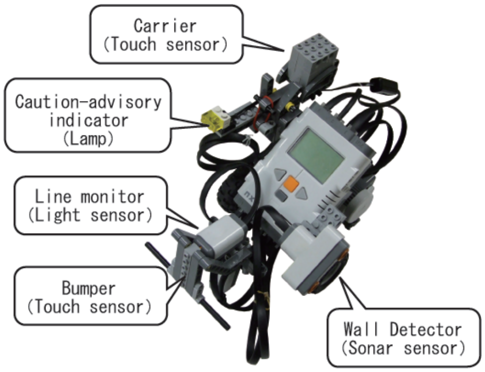


Fig. 2: Automated transport robot.

In integrated exercise B, we provided part of the class diagram and a development environment, which made it easy for the students to call external devices from the model, such as sensors and motors. Thus, they could focus on the modeling of the target application and learn the modeling methodology.

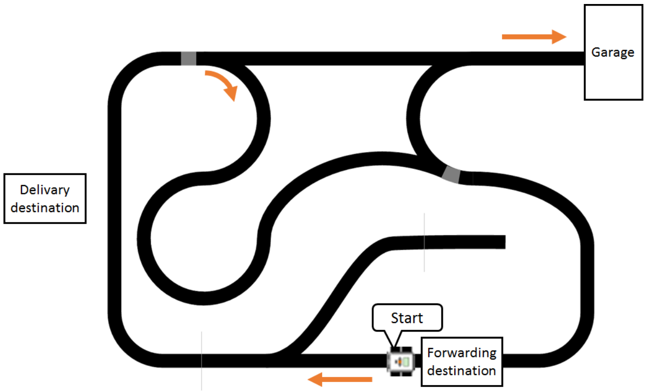


Fig. 3: Course layout for integrated exercise in the xUML course.

Fig. 4 shows the class diagram. This class diagram also shows the BridgePoint, additional key letters, and the relationship specifiers[1].

The classes in the unit layer are distributed as executable models in the lower layers of the class diagram, which are used to configure the auto transport robot, such as the Bumper class and Carrier class. The main challenges were to create state machine diagrams focused on the responsibilities of the two classes, i.e., the AutoTransporter (control of the automatic transport system) and the LineTracer (tracing the line). This helped the students learn how to describe dynamic behavior using state machine diagrams by focusing on the responsibilities of only two classes.

We presented a list of major events in order to help students create a model of the appropriate level of abstraction. The event list is shown in Table 2.

The subjects taking this course had background knowledge of C language grammar, excluding pointers and structures. The outline of the xUML course is shown in Table 3.

4.2 Results of practical exercises

In integrated exercise B, the achievement rate was evaluated according to ten milestones. The achievement rates for integrated exercise B are shown in Table 4. Most students were able to reach Milestone No. 4, but failed to reach No. 5. Thus, they were not able to attempt Nos. 6 to 10. Difficulties in detecting the marker, rather than in modeling,

[1] The alphanumeric characters in the lower right hand corner of the class (example: TP_ATP) are known as key letters, which are used to describe the access of the class. The alphanumeric characters are added to the relationships between classes (example: R1), which are known as relationship specifiers and are used to describe the access relationships.

Table 1: Educational Items used in Basic and Advanced Classes

Basic	Fundamental techniques A	Fundamentals of embedded systems, RTOS, cross development
	Fundamental exercise A	Sensor and motor programming, line trace programming
	Integrated exercise A	Implementation of a series of operations
Advanced	Fundamental techniques B	UML, MDD methods, modeling techniques, implementation techniques
	Fundamental exercise B	Reverse modeling of fundamental exercise A, Manual translation of the code from models based on the translation rule
	Integrated exercise B	Development of the same exercise as the integrated exercise using the MDD methodology

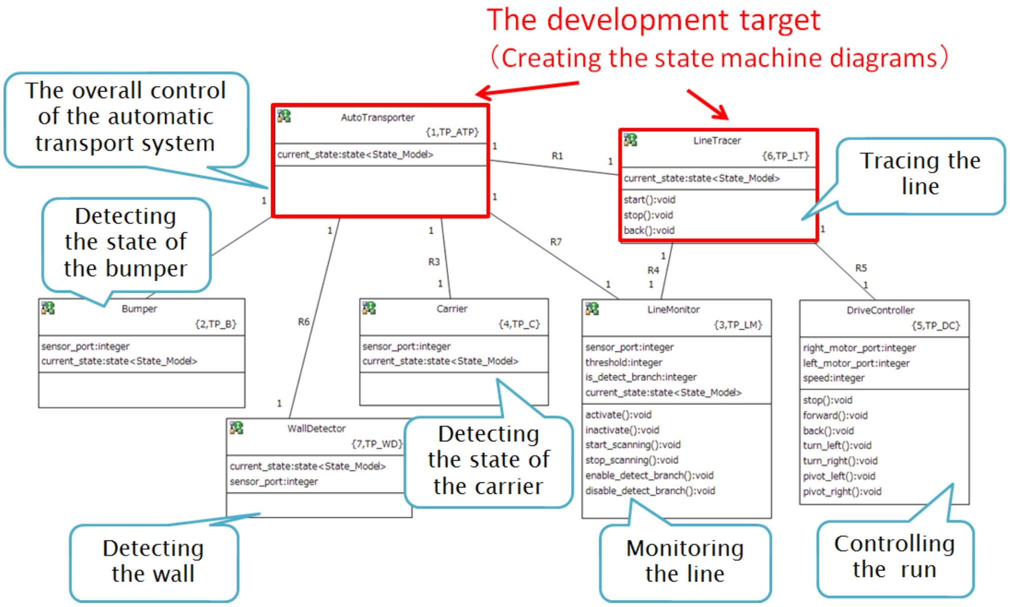


Fig. 4: Class diagram of the integrated exercise in the advanced class.

Table 2: Event List of the xUML course

class name	event name
Auto Transporter	bumper_touched bumper_released cargo_loaded cargo_unloaded end_of_turning wall_detected marker_detected
LineTracer	step_into_the_line go_out_from_the_line

caused failure at Milestone No. 5. Therefore, in our opinion the students were able to appreciate the responsibility of each class and model state change in the class using state machine diagrams.

5. Modeling education using DSM

5.1 Content of practical exercises

We conducted the DSM language course for six second- and third-year college students. This was an eight-hour course. The subjects were well versed in Java and UML. This course covered:

- Using the MDD tool (clooca).
- MDD methods and modeling techniques.
- Developing the auto transport system (a system similar to that used in the programming class of the Executable UML course) using a DSML and MDD.

The outline of the DSM course is shown in Table 5.

In the integrated exercise, we changed the exercise robot course such that there was no marker, because the students found marker detection difficult in the xUML course. Fig. 6 shows the exercise robot course.

Table 6 shows the action and event list of the DSM course.

5.2 Results of practical exercises

In the DSM course, the achievement rates were evaluated according to only eight milestones, because we changed the robot course in the exercise. The achievement rates for the integrated exercise are shown in Table 4. Approximately

Table 3: Outline of the xUML Course

Basic	Students	21 first-year college students
	Background knowledge	C Language grammar, excluding pointers and structure
	Exercise style	Pair (one PC and one robot per pair)
	Term	3 days (7 hours per day)
	Test	Nothing
	Questionnaire	Course content, knowledge, and skills
Advanced	Students	20 first-year college students
	Background knowledge	Acquired in the basic course
	Exercise style	Pair (1 PC and 1 robot per pair)
	Term	4 days (7 hours per day)
	Test	Knowledge of UML (day 2), knowledge of MDD (day 4)
	Questionnaire	Course content
PBL	Students	17 first-year college students
	Background knowledge	Acquired in the basic and advanced courses
	Exercise style	4 or 5 students per team
	Term	14 days (kick-off: 2 days; presentation of results: 2 days)
	Test	Nothing
	Questionnaire	Course content, knowledge, and skills

Table 4: Achievement Rates for Integrated Exercise in xUML and DSM Courses

Milestone	xUML course	DSM course
1) Trace a line	100 %	100 %
2) Detect a wall and stop	78 %	67 %
3) Detect unloading and send round	78 %	83 %
4) Stop in the garage	89 %	67 %
5) Change behavior to detect a marker	56 %	- %
6) Change right course	44 %	- %
7) Trace the opposite line edge	33 %	83 %
8) Ignore the marker	33 %	- %
9) Invert	33 %	67 %
10) Complete all patterns	11 %	17 %

Table 5: Outline of the DSM Course

Students	6 second and third-year students
Background knowledge	C Language grammar excluding pointers and structure
Exercise style	Single (one PC and one robot per person)
Term	2 days (4 hours per day)
Test	Nothing
Questionnaire	course content

Table 6: Action and Event List of the DSM course

Action	Event
Stop	Bumper touched
Go forward	Bumper released
Go forward with turn right	Cargo loaded
Go forward with turn left	Cargo unloaded
Go backward	Wall detected
Go backward with turn right	Marker detected
Go backward with turn left	Step into the line
	Go out from the line

Table 7: The Total Number of Classes with Errors

Errors	Number
Relations are not drawn	2
No relations names	4
Class names do not represent the responsibilities	2
Relation names are inappropriate	2
There are unnecessary classes	1
One class has several responsibilities	1

70% of the students met the challenges, except for Milestone No. 8 (complete all patterns).

Typical examples of class diagrams are shown Fig. 7.

Table 7 shows the number of errors in the class diagrams. Each subject created one class and therefore the maximum number of errors is six.

A typical example of a state machine diagram is shown in Fig. 8. Five students used the same name to describe multiple states in the state machine diagrams. Two students made "No states names" and "State names inappropriate" errors, such as a or b. However, such an error in state name was made in only a few states and the remaining states were given appropriate state names.

6. Discussion

6.1 Are the languages useful for modeling education?

Executable UML can be used to draw a model of many areas, because it is a generic language.

However, it was found that when learners create a model using Executable UML, it takes them a long time to learn the action language that is required to define the state actions. Therefore, they find it difficult to focus on creating state machine diagrams and class diagrams. Because all the names in the models, such as class names and state names, must be

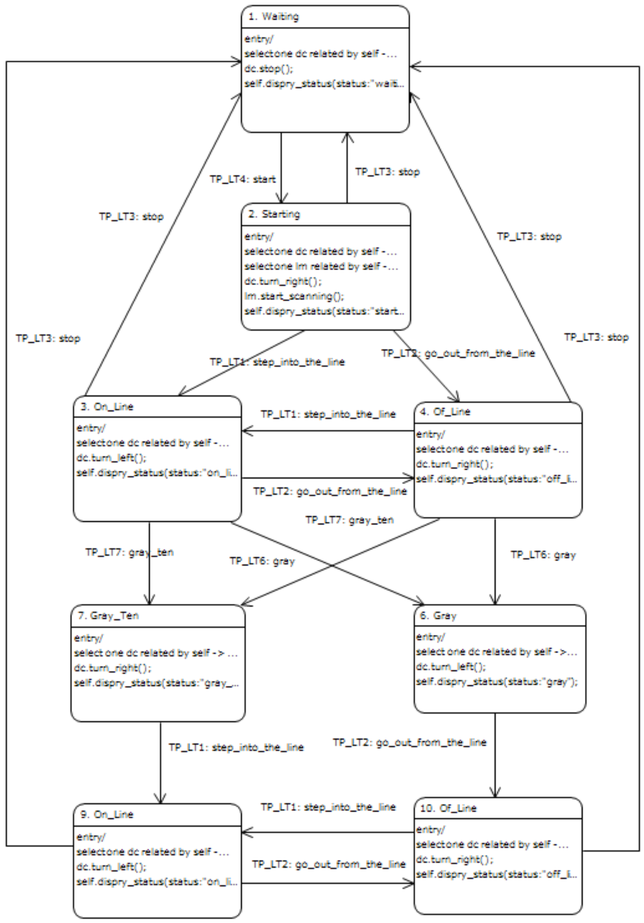

1. Waiting
entry/
selectone dc related by self -...
dc.stop();
self.dispry_status(status:"waiti...
TP_LT4: start
TP_LT3: stop
2. Starting
entry/
selectone dc related by self -...
selectone lm related by self -...
dc.turn_right();
lm.start_scanning();
self.dispry_status(status:"start...
TP_LT3: stop
TP_LT3: stop
TP_LT3: stop
TP_LT3: stop
TP_LT1: step_into_the_line
TP_LT2: go_out_from_the_line
3. On_Line
entry/
selectone dc related by self -...
dc.turn_left();
self.dispry_status(status:"on_li...
TP_LT1: step_into_the_line
TP_LT2: go_out_from_the_line
4. Of_Line
entry/
selectone dc related by self -...
dc.turn_right();
self.dispry_status(status:"off_li...
TP_LT7: gray_ten
TP_LT7: gray_ten
TP_LT6: gray
TP_LT6: gray
7. Gray_Ten
entry/
select one dc related by self -> ...
dc.turn_right();
self.dispry_status(status:"gray_...
6. Gray
entry/
select one dc related by self ->...
dc.turn_left();
self.dispry_status(status:"gray");
TP_LT1: step_into_the_line
TP_LT2: go_out_from_the_line
9. On_Line
entry/
selectone dc related by self -...
dc.turn_left();
self.dispry_status(status:"on_li...
TP_LT1: step_into_the_line
TP_LT2: go_out_from_the_line
10. Of_Line
entry/
selectone dc related by self -...
dc.turn_right();
self.dispry_status(status:"off_li...

Fig. 5: Example of a state machine diagram in the xUML course.

written in English, it was difficult for the students who were not proficient in English to determine appropriate names. This also led to a decrease in the students' motivation.

In contrast, the DSM course required that a modeling language be created for the development domain. In this practical exercises, we developed a DSM language based on the results of the xUML course. Because the design goal of the DSM language was to enable the students to focus on modeling without having to consider the strict rules of grammar, this language allowed a low degree of freedom. However, the students were able to develop this system smoothly in the practical exercises.

The total development time of the integrated exercise in xUML was nine hours, whereas in the DSM course it was three hours. Even taking into account the change in the robot course in the exercise, the students were able develop the system in a relatively short time in the DSM course. Some students in the DSM course stated that "It is good that I can check my model using a executable model," "It is good to be able to attempt the challenge many times," and "I am able to keep my motivation high." Thus, the DSM course was able to keep students' motivation at a high level.

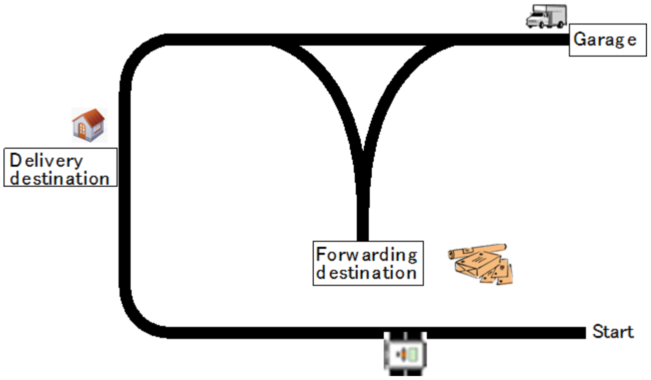


Fig. 6: Course layout for integrated exercise in the DSM course.

6.2 Learning support using MDD

6.2.1 Class diagrams

The class name must be stated in alphanumeric form in both Executable UML and DSM language, and thus, in this characteristic there is no difference between the two languages. The advantage of using MDD is that it is very effective for making structural changes in the second half of development. In the DSM course, in order to analyze the process of creating the model (examples: classes added, modified, and deleted), the exercise time was divided into three parts: first, middle, and last. According to the results, students also added and deleted classes in the last part. Therefore, the students changed the class structure until they reached the last part. Thus, they took time to address the model refinement process. We found that these educational courses based on MDD gave the students the necessary experience to improve their modeling skills.

6.2.2 State machine diagrams

The state name must be written in English in Executable UML. In contrast, in the DSM language Japanese can be used. Thus, since time is not spent on translating them into English, it is possible to determine the state names more smoothly. Because the event and action can be chosen from a pull-down menu, the time it takes to create a state machine diagram is shorter.

In the xUML course, however, the procedure for each state machine must be described using action languages. Thus, the time it takes to create a state machine diagram in Executable UML is longer.

A comparison of Fig. 5 and Fig. 8 shows that in Executable UML the action scripts in the states are so long that it takes time to understand the content of the action. In

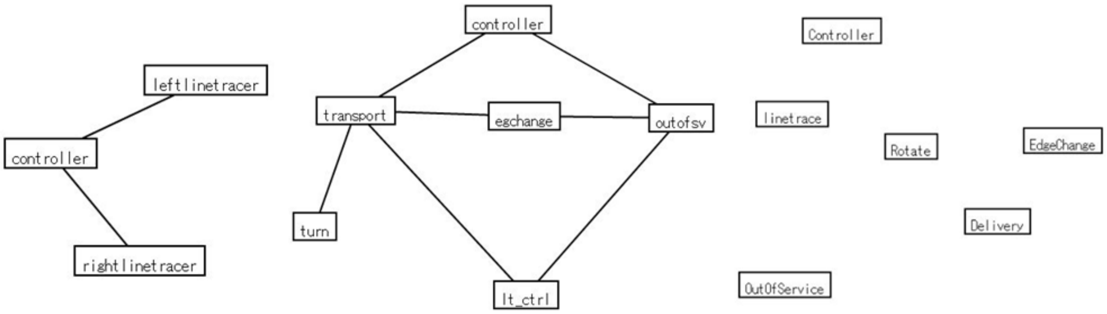


Fig. 7: Examples of class diagrams in the DSM course.

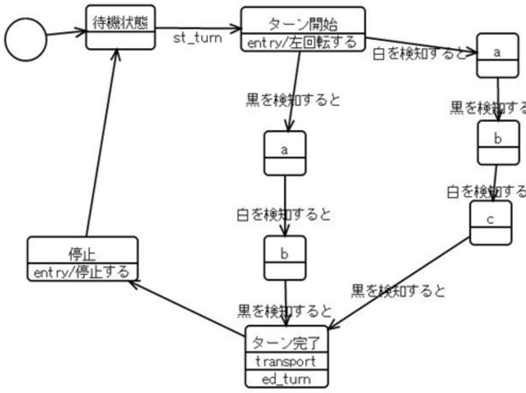


Fig. 8: Example of a state machine diagram in the DSM course.

contrast, in the DSM language, the state can be understood immediately, because the action is defined in one sentence.

Thus, creating a DSM tailored to the educational subject may help improve the understanding of the students.

7. Conclusion

In this paper, we described a study in which we compared two executable modeling languages for object-oriented modeling education: Executable UML and DSM language. We conducted two trial courses for modeling education for novices in which the two languages were used. In development using Executable UML, problems related to drawing the models were found: formulating action scripts is difficult for students and the fact that all the names in the models, such as class names and state names, must be written in English, hinders the students and lowers their motivation.

It was found that the students spent less time on development in the exercise in the DSM course than in the xUML course. Therefore, in the introductory teaching of modeling, it is effective to use the DSM language, which is customized for educational targets and items.

One problem arose when using the MDD method for teaching modeling: the students neglected the quality of the model because they were focused on completing the functional aspects that could be evaluated with the MDD. In these courses, we addressed this issue by conducting a review with the participation of the teachers. It is necessary to consider a supporting method for enhancing the quality of the model for the future.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 26730174.

References

[1] T. C. Lethbridge, G. Mussbacher, A. Forward, and O. Badreddin, “Teaching uml using umple: Applying model-oriented programming in the classroom,” in *24th IEEE-CS Conference on Software Engineering Education and Training*, 2011, pp. 421–428.

[2] P. Liggesmeyer and M. Trapp, “Trends in embedded software engineering,” *IEEE Software*, vol. 26, no. 3, pp. 19–25, 2009.

[3] T. Gjøsater and A. Prinz, “Teaching model driven language handling,” *Electronic Communications of the EASST*, vol. 34, pp. 1–10, 2010.

[4] B. Tekinerdogan, “Experiences in teaching a graduate course on model-driven software development,” *Computer Science Education*, vol. 21, no. 4, pp. 363–387, 2011.

[5] H. Burden, R. Heldal, and T. Siljamaki, “Executable and translatable uml - how difficult can it be?” in *18th Asia Pacific Software Engineering Conference*, 2011, pp. 114–121.

[6] S. Flint, H. Gardner, and C. Boughton, “Executable/translatable uml in computing education,” in *Sixth Australasian Computing Education Conference*, vol. 30. ACS, 2004, pp. 69–75.

[7] Y. Khmelevsky, G. Hains, and C. Li, “Automatic code generation within student’s software engineering projects,” in *Proc. of the Seventeenth Western Canadian Conference on Computing Education*. ACM, 2012, pp. 29–33.

[8] C. Starrett, “Teaching uml modeling before programming at the high school level,” in *Seventh IEEE International Conference on Advanced Learning Technologies*. IEEE Computer Society, 2007, pp. 713–714.

[9] S. Akayama, S. Kuboaki, K. Hisazumi, T. Futagami, and T. Kitasuka, “Development of a modeling education program for novices using model-driven development,” in *Proc. 2012 Workshop on Embedded and Cyber- Physical Systems Education*. ACM, 2012.

[10] S. J. Mellor and M. J. Balcer, *Excutable UML -A Foundation for Model-Driven Architecture*. Addison-Wesley, 2002.

[11] J.-P. T. Steven Kelly, *Domain-Specific Modeling: Enabling Full Code Generation*. Wiley-IEEE Computer Society Press, 2008.

[12] Technical Rockstars, “clooca,” http://www.clooca.com.

Effectiveness of Coupling Metrics in Identifying Change-Prone Object-Oriented Classes

Mahmoud O. Elish[1] **and Ali A. Al-Zouri**[2]
[1]Information and Computer Science Department, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia, elish@kfupm.edu.sa
[2]IT Development Department, IT Head Office, SABB, Riyadh, Saudi Arabia, ali.alzouri@sabb.com

Abstract - *This paper empirically evaluate the effectiveness of a set of coupling metrics, identified in a literature survey, as early indicators of change-prone classes from one release to the next in object-oriented software evolution process. Several hypotheses were tested, and different logistic regression models were constructed for predicting change-prone classes. Coupling metrics were found to be statistically correlated with change-proneness of classes. The results also indicate that a prediction model based on coupling metrics is generally more accurate than a model based on cohesion metrics. Furthermore, a prediction model based on import coupling metrics is more accurate than a model based on export coupling metrics. We also found that the coupling metrics in the C&K suite are not necessarily more accurate than other metrics in the suite in identifying change-prone classes in evolving object-oriented software. Moreover, there is no confounding effect of class size in the validity of some of the investigated coupling metrics.*

Keywords: Coupling metrics; object-oriented software; software evolution.

1 Introduction

Identification of change-prone classes in an object-oriented software system is an important activity especially in large and complex systems. A great majority of changes is rooted in a small proportion of classes [14]. In other words, around 80% of the changes are actually rooted in around 20% of the classes. This phenomenon has been known as Pareto's Law (also as the 80:20 rule) [14]. Identifying change-prone classes can therefore be very useful in guiding software maintenance and evolution; distributing resources more efficiently and effectively; and thus enabling the project manager and his team to focus their effort and attention on the change-prone classes during the evolution process.

Many coupling metrics have been proposed in the literature. Most of them have been empirically validated by exploring the relationships between them and certain software quality attributes such as fault-proneness [3, 4, 8, 13, 16], testability [5], reusability [12], and maintenance effort [15]. Results from the previous empirical studies indicate that coupling metrics are good indicators of several quality attributes.

This paper aims to empirically evaluate a set of coupling metrics, identified in a literature survey [1], as early indicators of change-prone classes from one release to the next in evolving object-oriented software. This set of metrics covers comprehensively different type of interaction and relationships within a class and between classes. Metrics that are correlated with class change-proneness will be essential for objective and quantitative identification and characterization of change-prone classes and for effective prediction of change-proneness during software evolution throughout the releases. In addition, these metrics will provide useful guidance to practitioners involved in development and maintenance of evolving large-scale software.

The rest of this paper is organized as follows. Section 2 reviews related works. Section 3 describes the empirical study and discusses its results. Section 4 concludes the papers and suggests directions for future work.

2 Related Work

In the literature, the relationships between coupling metrics and several software quality attributes have been explored. Briand et al. [4] explored the relationships between design measures and fault-proneness of classes. They found that most of the coupling metrics are good predictors of fault-proneness. Gyimothy et al. [13] found that CBO (coupling between object classes) and RFC (response set for class) metrics are good predictors of fault-proneness. El-Emam et al. [8] found that OCMEC (export coupling based on class-method interaction) and OCAEC (export coupling based on class-attribute interaction) metrics have strong association with fault-proneness.

Bruntink and Deursen [5] empirically studied the relationship between several object-oriented metrics and software testability. RFC metric was one of them. Testability was measured in terms of test efforts which were taken from the size of test suites. Results showed that there is significant relationship between RFC and testability.

Gui and Scott [12] empirically studied the relationship between CBO, RFC, MPC (message passing coupling) and DAC (data abstraction coupling) metrics and software reusability. Reusability was measured by the number of lines of code that were added, modified or deleted in order to extend some function in the system. Their results indicated that there is strong relationship between coupling metrics and reusability, especially CBO and RFC.

Li and Henry [15] studied coupling metrics (CBO, RFC, MPC, DAC) with respect to maintenance effort, which was measured by the number of lines changed per class. A line change could be an addition or a deletion. Their results showed that there is strong relationship between coupling metrics and maintenance effort.

Wilkie and Kitchenham [18] studied two versions of CBO metric according to direction of coupling, CBO(backward) to count export coupling and CBO(forward) to count import coupling, with respect to how the final version of the system differs from the first version. Changes were counted as the number of changes per class. They found that CBO(forward) is a good predictor of change-prone classes. CBO(backward) had slightly lower correlation with change-proneness and found to be not significant.

Koru and Liu [14] tested and validated the Pareto's Law which implies that a great majority (around 80%) of changes are rooted in a small proportion (around 20%) of the classes. They also identified and characterized the change-prone classes in two products (KOffice and Mozilla) by producing tree-based models. Their results from both systems strongly supported Pareto's law. The resulting tree-based model consists of several metrics and OCMEC and OCMIC (import coupling based on class-method interaction) coupling metrics were part of the model.

In addition to the coupling metrics, other metrics and approaches have been proposed in the literature to predict change-proneness. For example, Tsantalis et al. [17] proposed a probabilistic approach to estimate the change proneness of an object-oriented design. Moreover, Elish and Al-Khiaty [9] proposed a suite of metrics for quantifying historical changes to predict future change-prone classes in object-oriented software.

This study explores the relationships between a comprehensive set of coupling metrics and class change-proneness, whereas previous studies have been limited to few coupling metrics. In addition, we are assessing these metrics in a software evolution context throughout the releases.

3 Empirical Study

The objective of this study is to empirically investigate the relationships between a set of coupling metrics and the change-proneness of classes in evolving object-oriented software. In other words, we want to evaluate the capability of these metrics as early indicators of change-prone classes from one software release to the next.

3.1 Independent and Dependent Variables

The main independent variables are 22 coupling metrics, which were identified in a literature survey on object-oriented design measures [1]. All of them are static and language independent. For comparison purposes, cohesion metrics (identified in a literature survey on object-oriented design measures [2]) and C&K metrics (Chidamber and Kemerer [6]) were used as other independent variables to build other prediction models. Definitions of the coupling, cohesion and C&K metrics are provided in [1], [2] and [6] respectively.

As a dependent variable, we used a dichotomous variable (named CHANGE) that indicates whether or not a class was changed from one software release to the next release. A class is considered changed if at least one of its lines of source code was changed or deleted, or at least one new line of code was added to it. Comment and blank lines were excluded.

3.2 Hypotheses

- Hypothesis 1: There is a statistically significant correlation between each of the investigated coupling metrics and change-proneness of classes in evolving object-oriented software.
- Hypothesis 2: A prediction model based on the investigated coupling metrics is more accurate than a model based on cohesion metrics in identifying change-prone classes in evolving object-oriented software.
- Hypothesis 3: A prediction model based on import coupling metrics is more accurate than a model based on export coupling metrics in identifying change-prone classes in evolving object-oriented software.
- Hypothesis 4: Coupling metrics in the C&K suite are more accurate than other metrics in the suite in identifying change-prone classes in evolving object-oriented software.

3.3 Software Systems Analyzed

Two multi-release object-oriented software systems of different size and from different application domains were analyzed in this study: Stellarium[1] and LabPlot[2]. Both systems are open source systems and written in C++ programming language. Stellarium is an educational system for astronomy, and the goal of the system is to render 3D photo-realistic skies in real time with OpenGL. It displays stars, constellations, planets, nebulas and others things like ground, landscape, atmosphere, etc. LabPlot is a desktop environmental system for visualization data. The goal of the system is data plotting and function analysis.

All releases of both systems were analyzed from the first release to the most recent release at the time of this study. Stellarium system has seven releases, whereas LabPlot system has six releases. Release numbers and size measures of these two systems are provided in Table 1. The percentage of changed and unchanged classes from one release to the next are shown in Figure 1 and Figure 2 for Stellarium and LabPlot systems respectively.

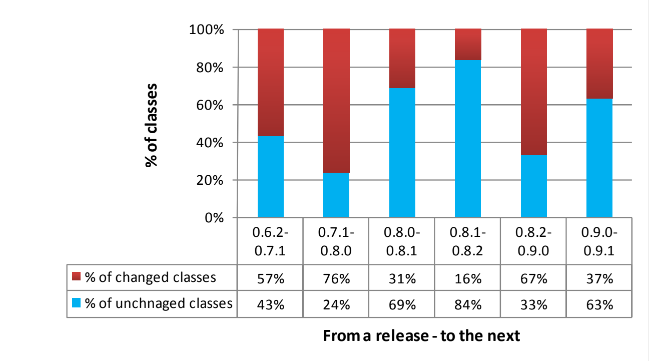
[1] www.sourceforge.net/projects/stellarium/

[2] www.sourceforge.net/projects/labplot/

Table 1. Release numbers and size measures

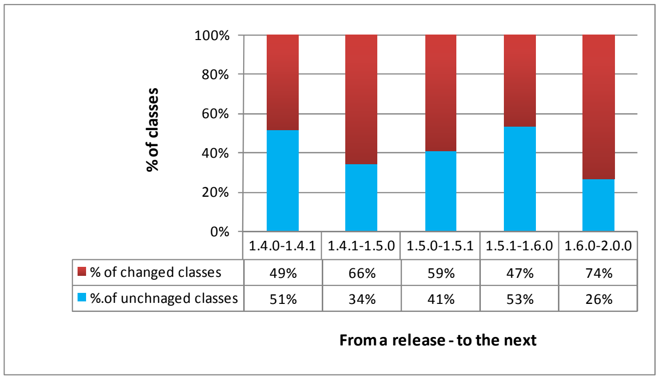
Stellarium system			LabPlot system		
Release Number	Number of classes	Lines of code (LOC)	Release Number	Number of classes	Lines of code (LOC)
0.6.2	102	9499	1.4.0	35	1399
0.7.1	117	12774	1.4.1	38	1794
0.8.0	140	16103	1.5.0	44	1519
0.8.1	147	17004	1.5.1	49	2173
0.8.2	154	20141	1.6.0	53	2902
0.9.0	191	22334	2.0.0	24	852
0.9.1	190	12140			

It can be observed that the size of the Stellarium system is bigger than the LabPlot system in terms of the number of classes and lines of code (LOC). Moreover, the number of classes and LOC are increasing from one release to the next in both systems, from the first release to the release before the last one, which is most likely due to the addition of new features and requirements. However, there is a significant drop in the number of classes in the LabPlot system, and a significant drop in LOC in both systems in the last release. This suggests that a major refactoring was performed. The percentages of changed classes that are provided in Figure 1 and Figure 2 include deleted classes. The differences in the percentages of changed classes between releases and between the two systems will be helpful in evaluating the accuracy of the prediction models and preventing biased results.



	0.6.2-0.7.1	0.7.1-0.8.0	0.8.0-0.8.1	0.8.1-0.8.2	0.8.2-0.9.0	0.9.0-0.9.1
🟥 % of changed classes	57%	76%	31%	16%	67%	37%
🟦 % of unchnaged classes	43%	24%	69%	84%	33%	63%

Figure 1. Percentage of changed and unchanged classes from one release to the next in Stellarium system



	1.4.0-1.4.1	1.4.1-1.5.0	1.5.0-1.5.1	1.5.1-1.6.0	1.6.0-2.0.0
🟥 % of changed classes	49%	66%	59%	47%	74%
🟦 %.of unchnaged classes	51%	34%	41%	53%	26%

Figure 2. Percentage of changed and unchanged classes from one release to the next in LabPlot system

3.4 Data Collection

Columbus[3] tool [11] was used in this study to collect all independent variables, i.e., coupling, cohesion and C&K metrics. ExamDiff Pro[4] tool was used to collect the dependent variable by comparing classes from one release to the next. Comment and blank lines were excluded in class comparison. Our approach is release by release prediction where we train the prediction model using all releases from the first release to release i-1 and then test it using release i. Since there are seven releases in the Stellarium system and six in the LabPlot system, five and four pairs of training and testing datasets were produced from each system respectively; no training dataset for the first release and no testing dataset for the last release.

3.5 Descriptive Statistics

Table 2 and Table 3 provide descriptive statistics of the 22 coupling metrics under investigation, which were collected from Stellarium and LabPlot systems, respectively. In general, the LabPlot system has lower coupling than the Stellarium system. Only coupling metrics that have more than five non-zero values were considered for further analysis since those metrics that have less than five non-zero values have low variance (almost zero) and they may mislead the analysis. This strategy was also applied in the previous studies [4, 8]. Metrics with less than five non-zero values are highlighted in both tables.

3.6 Correlation Analysis

We performed Spearman's rank-order correlation analysis, at 99% confidence level, between the dependent variable (CHANGE) and each of the investigated coupling metrics. Table 4 reports the results over all the releases of Stellarium and LabPlot systems, respectively, in terms of the correlations coefficients and p-values. All metrics in both systems, except OCAEC in Stellarium, were found to be significantly correlated (p-value < 0.01) with CHANGE. However, OCAEC metric in Stellarium was significantly correlated with CHANGE but at 95% confidence level (p-value < 0.05). Hypothesis 1 is therefore accepted.

It can be observed that all coupling metrics, except IH-ICP (information-flow-based inheritance coupling) metric, are positively correlated with class change-proneness. This indicates that the more the coupling of a class with the rest of the system the higher the probability that the class will change. In case of IH-ICP metric, the negative correlation between it and class change-proneness suggests that the more the coupling of a class with its ancestors through methods invocations the less likely the class will change. This observation can be explained; ancestor classes are expected to be highly stable [10], and thus change propagations from them to their descendant classes are less likely to occur.

[3] http://www.frontendart.com
[4] http://www.prestosoft.com/edp_examdiffpro.asp

Table 2. Descriptive statistics of coupling metrics in Stellarium system

Metric	Mean	Std Dev.	Max	Min
CBO	2.24	3.91	47	0
CBO1	1.63	3.63	45	0
RFC	16.4	37.49	447	0
RFC1	14.85	34.02	424	0
MPC	16.38	74.08	1309	0
DAC	2.19	12.04	165	0
DAC1	1.09	3.22	34	0
ICP	29.28	141.74	2357	0
IH-ICP	3.13	10.57	109	0
NIH-ICP	26.15	141.51	2357	0
ACAIC	0	0	1	0
DCAEC	0	0	1	0
ACMIC	0	0	2	0
DCMEC	0	0	3	0
AMMIC	0	0	0	0
DMMEC	0	0	0	0
OMMIC	0	0	0	0
OMMEC	0	0	0	0
OCAIC	2.11	11.36	165	0
OCAEC	1.89	5.8	117	0
OCMIC	1.93	4.05	32	0
OCMEC	1.81	7.45	71	0

Table 3. Descriptive statistics of coupling metrics in LabPlot system

Metric	Mean	Std Dev.	Max	Min
CBO	0.32	0.75	4	0
CBO1	0.31	0.75	4	0
RFC	9.31	17.18	124	0
RFC1	9.31	17.18	124	0
MPC	4.13	15.45	124	0
DAC	0.97	2.21	13	0
DAC1	0.73	1.43	7	0
ICP	4	18.29	168	0
IH-ICP	0	0	0	0
NIH-ICP	4	18.29	168	0
ACAIC	0	0	0	0
DCAEC	0	0	0	0
ACMIC	0	0	0	0
DCMEC	0	0	0	0
AMMIC	0	0	0	0
DMMEC	0	0	0	0
OMMIC	0	0	0	0
OMMEC	0	0	0	0
OCAIC	0.97	2.21	13	0
OCAEC	0.95	2.37	12	0
OCMIC	0.97	2.28	14	0
OCMEC	0.93	2.59	18	0

Table 4. Spearman correlation results

Metric	Stellarium		LabPlot	
	Corr. Coef.	p-value	Corr. Coef.	p-value
CBO	0.14	<0.01	0.26	<0.01
CBO1	0.20	<0.01	0.24	<0.01
RFC	0.19	<0.01	0.48	<0.01
RFC1	0.22	<0.01	0.48	<0.01
MPC	0.20	<0.01	0.21	<0.01
DAC	0.27	<0.01	0.43	<0.01
DAC1	0.27	<0.01	0.44	<0.01
ICP	0.18	<0.01	0.24	<0.01
IH-ICP	-0.16	<0.01	---	---
NIH-ICP	0.21	<0.01	0.24	<0.01
OCAIC	0.28	<0.01	0.43	<0.01
OCAEC	0.07	0.03	0.30	<0.01
OCMIC	0.26	<0.01	0.43	<0.01
OCMEC	0.17	<0.01	0.28	<0.01

3.7 Prediction Models

Different logistic regression models, standard models based on maximum likelihood estimation, were constructed in this study for predicting change-prone classes from one software release to the next. In the following subsections, we compare the accuracy (correct classification rate) of coupling-based model vs. cohesion-based model; import coupling-based model vs. export coupling-based model; and among the models that are based on each metric in the C&K suite.

3.7.1 Coupling-based Model vs. Cohesion-based Model

In a modular design, each individual class should have high cohesion within the class and low coupling with other classes. It is interesting to investigate which one of these two class characteristics (coupling and cohesion) are better predictors of its change-proneness. Accordingly, two prediction models were constructed for identifying change-prone classes for each system (Stellarium and LabPlot). One model was based on the coupling metrics as independent variables; and the other was based on the cohesion metrics as independent variables. The prediction was performed release by release where the models were trained using all releases from the first release to release i-1 and then tested using release i. The accuracy (correct classification rate) of each model for each release was calculated as well as the average accuracy over all the releases. Figure 3 and Figure 4 illustrate the accuracy curve of these models on Stellarium and LabPlot systems respectively. The horizontal axis represents the release number and the vertical axis represents the accuracy for identifying change-prone classes in that release.

Out of the five releases of Stellarium system, the accuracy of the coupling-based model was better than the accuracy of the cohesion-based model in two releases (0.8.0 and 0.8.1), and the accuracy of the cohesion-based model was better in two releases as well (0.7.1 and 0.8.2). In release 0.9.0, both models have almost the same accuracy. However, the best achieved accuracy by the coupling-based model throughout the releases was 70.1% compared to 68.6% which was achieved by the cohesion-based model. In LabPlot System, the coupling-based model had better accuracy than the cohesion-based model in three releases, and in release 1.5.1 in which the accuracy of the cohesion-based model was better, the difference was only 2%. The highest accuracy of the coupling-based model reached 79.5% in release 1.5.0, while the highest accuracy of the cohesion-based model was 75.5%.

It was observed that the models based on coupling metrics outperform the models based on cohesion metrics in both systems in terms of the average accuracy across the releases. In Stellarium system, the average accuracy of the coupling-based model was 62.2%, whereas it was 58.8% by the cohesion-based model. In LabPlot System, the average accuracy of the coupling-based model was 72.2%, whereas it was 68% by the cohesion-based model. These results suggest the acceptance of hypothesis 2.

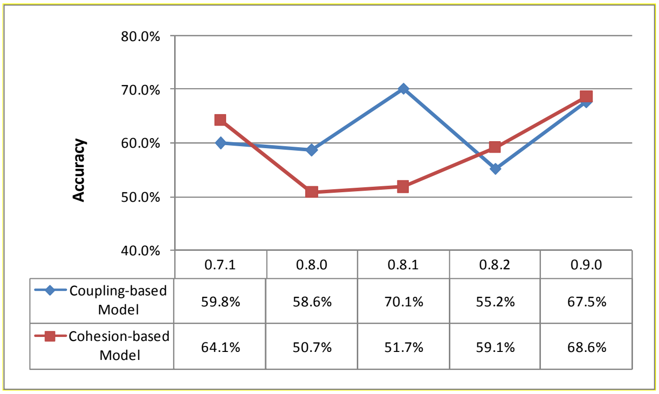


Figure 3. Accuracy of coupling-based model and cohesion-based model for Stellarium system

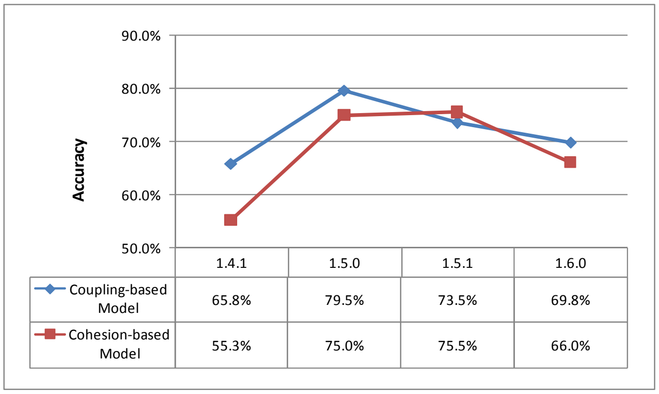


Figure 4. Accuracy of coupling-based model and cohesion-based model for LabPlot system

3.7.2 Import Coupling-based Model vs. Export Coupling-based Model

A class could depend on some other classes, and some classes could depend on it. Import and export coupling metrics measure these dependencies respectively. Table 5 lists import and export coupling metrics. We investigated which one of these two types of coupling metrics (import and export) is better predictor of class change-proneness. Two prediction models were constructed for identifying change-prone classes for each system (Stellarium and LabPlot). One model was based on the import coupling metrics as independent variables; and the other was based on the export coupling metrics as independent variables. The prediction was also performed release by release where the models were trained using all releases from the first release to release i-1 and then tested using release i. The accuracy (correct classification rate) of each model for each release was calculated as well as the average accuracy over all the releases. Figure 5 and Figure 6 illustrate the accuracy curve of these models on Stellarium and LabPlot systems respectively.

Table 5. Import and export coupling metrics

Import Coupling Metrics	CBO, CBO1, RFC, RFC1, MPC, DAC, DAC1, ICP, IH-ICP, NIH-ICP, OCAIC, OCMIC
Export Coupling Metrics	CBO, CBO1, OCAEC, OCMEC

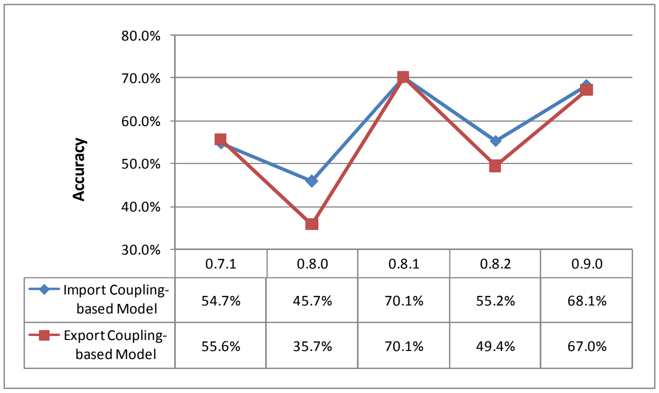


Figure 5. Accuracy of import coupling-based model and export coupling-based model for Stellarium system

In Stellarium system, the import coupling-based model outperformed the export coupling-based model in three releases (0.8.0, 0.8.2 and 0.9.0). In release 0.8.1, both models had the same accuracy (70.1%), and in release 0.7.1, the accuracy of the export coupling-based model was better than the accuracy of the import coupling-based model by 1% only. The average accuracy of the import coupling-based model was 58.7%, whereas it was 55.5% by the export coupling-based model. In LabPlot system, the import coupling-based model outperformed the export coupling-based model in all releases. The highest accuracy of the import coupling-based model was 81.8%. The average accuracy of the import coupling-based model was 71.9%, which is very high compared to the average accuracy of the export coupling-based model (55.5%).

These results indicate that the models based on import coupling metrics outperform the models based on export coupling metrics in both systems in terms of the average accuracy. We therefore accept hypothesis 3.

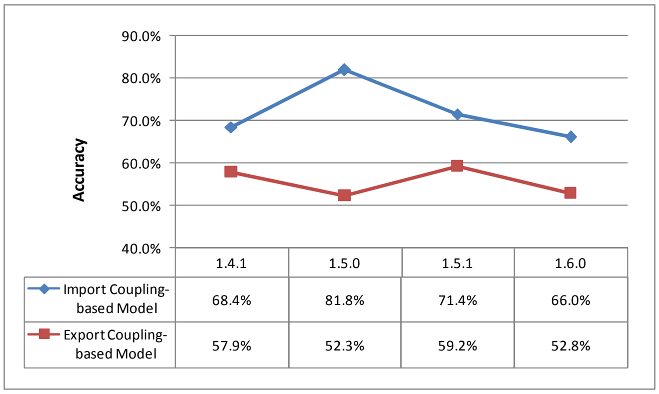


Figure 6. Accuracy of import coupling-based model and export coupling-based model for LabPlot system

3.7.3 Models based on Each Metric in C&K Suite

The C&K suite of metrics consists of six metrics [6]: CBO (coupling between object classes), RFC (response set for class), LCOM (lack of cohesion in methods), WMC (weighted methods per class), DIT (depth of inheritance tree), NOC (number of children). Two of them are coupling metrics which are CBO and RFC. The following analysis aimed to compare the prediction performance of each of these two metrics in identifying change-prone classes against each of the other four metrics in the C&K suite.

For Stellarium system, six prediction models were built; each based on one of the six metrics in the C&K suite. For LabPlot system, only four models were built since there is no inheritance in LabPlot system and thus DIT and NOC metrics were excluded. Figure 7 and Figure 8 illustrate the accuracy curve of these models on Stellarium and LabPlot systems respectively.

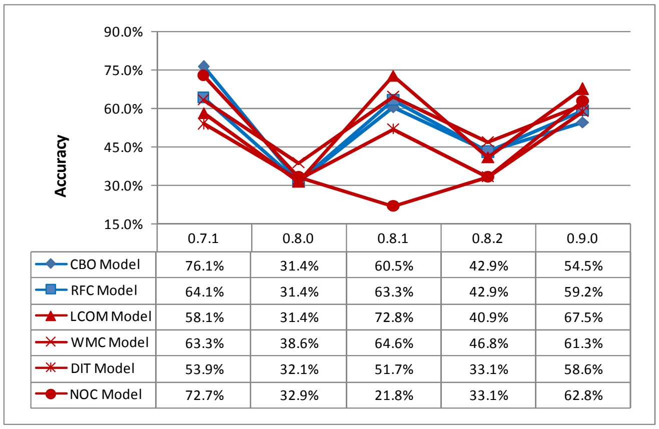
We now compare the accuracy of the coupling metrics (CBO and RFC) against the other four metrics (LCOM, WMC, DIT, and NOC) in the C&K suite. Out of the five releases of Stellarium system, the LCOM model was the best in two releases, the WMC model was best in two other releases, and the CBO model was the best in one release. The average accuracy achieved by each of the CBO, RFC, LCOM, WMC, DIT and NOC models was 53.1%, 52.2%, 54.2%, 54.9%, 45.9% and 44.6% respectively. It can be observed that the average accuracy achieved by each of the CBO, RFC, LCOM, WMC models is competitive (less than 3% differences). However, the performance of the DIT and NOC models is noticeably lower than the other metrics. This suggests that inheritance-based metrics are not good indicators of change-prone classes. In LabPlot system, the average accuracy achieved by each of the CBO, RFC, LCOM and WMC models was 57.4%, 72.6%, 67.5% and 61.6% respectively. The RFC model outperformed all other three models in all release except the last release where it was outperformed by the WMC model. The CBO model, however, has the lowest accuracy on average. These results suggest the rejection of hypothesis 4.

3.8 Confounding Effect of Class Size

We also investigated the potential confounding effect of class size on the validity of the investigated coupling metrics with respect to their relationship with change-proneness. This helps to determine if the relationship between these metrics and change-proneness of classes is real regardless of the class size. We followed the same approach suggested by El Emam et al. [7] to determine whether there is a confounding effect of class size. The idea is to include a size metric (LOC in this study) as another independent variable, in addition to a coupling metric, in a prediction model for change-proneness. If there is a statistically significant difference in the results with and without the size metric, then this indicates a confounding effect of size on the validity of that coupling metric.

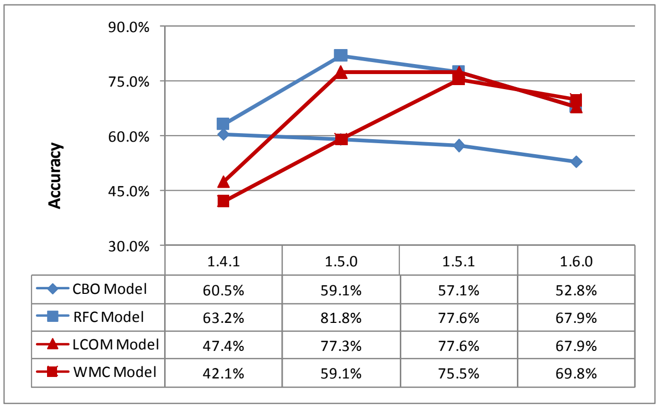
In order to examine the confounding effect of size on each coupling metric Ci, two prediction models for class change-proneness were built: (i) one based on the coupling metric Ci only and (ii) one based on the Ci and LOC. Wilcoxon nonparametric test was then performed, at 95% confidence level, to evaluate the significance difference between the results obtained from each model. Table 6 reports the Z statistic values and p-values for the Wilcoxon test results. If the p-value is less than 0.05, then there is a confounding effect of size on the validity of the corresponding coupling metrics.

It can be observed that, in LabPlot system, there is no confounding effect of class size on the validity of all of the coupling metrics. However, in Stellarium system, the associations between some coupling metrics (i.e. CBO, CBO1, RFC, ICP, IH-ICP, OCAEC, and OCMEC) and change-proneness of classes disappear after controlling for class confounder.



	0.7.1	0.8.0	0.8.1	0.8.2	0.9.0
CBO Model	76.1%	31.4%	60.5%	42.9%	54.5%
RFC Model	64.1%	31.4%	63.3%	42.9%	59.2%
LCOM Model	58.1%	31.4%	72.8%	40.9%	67.5%
WMC Model	63.3%	38.6%	64.6%	46.8%	61.3%
DIT Model	53.9%	32.1%	51.7%	33.1%	58.6%
NOC Model	72.7%	32.9%	21.8%	33.1%	62.8%

Figure 7. Accuracy of models based on each metric in C&K suite for Stellarium system



	1.4.1	1.5.0	1.5.1	1.6.0
CBO Model	60.5%	59.1%	57.1%	52.8%
RFC Model	63.2%	81.8%	77.6%	67.9%
LCOM Model	47.4%	77.3%	77.6%	67.9%
WMC Model	42.1%	59.1%	75.5%	69.8%

Figure 8. Accuracy of models based on each metric in C&K suite for LabPlot system

3.9 Limitations

The results of this study were obtained by analyzing two C++ open source systems. Although these systems are normal open source systems and representing different size and application domains, more studies should be conducted to further support the results and to accumulate knowledge.

This study focused on static coupling metrics. It did not explore the relationship between dynamic coupling metrics and change-processes of classes. This study was also focused on change-proneness of classes, i.e., whether or not a class was changed from one software release to the next release. The capability of coupling metrics in estimating change size and density was not evaluated.

In addition, this study was a regression and correlation study. Association between most of the investigated coupling metrics and change-proneness of classes was observed but causality of the association cannot be claimed. In other words, claims cannot be made as altering a class to reduce its coupling would necessarily decrease its likelihood of being changed.

Table 6. Wilcoxon test for confounding effect of class size

Metric	Stellarium		LabPlot	
	Z	p-value	Z	p-value
CBO	3.29	<0.05	0.67	0.50
CBO1	2.58	<0.05	1.80	0.07
RFC	3.51	<0.05	0.53	0.59
RFC1	1.62	0.11	0.53	0.59
MPC	1.61	0.11	0	1
DAC	1.22	0.22	0	1
DAC1	1.60	0.11	0	1
ICP	3.00	<0.05	0	1
IH-ICP	5.17	<0.05	---	---
NIH-ICP	1.87	0.06	0	1
OCAIC	0.39	0.69	0	1
OCAEC	4.95	<0.05	1.23	0.22
77OCMIC	1.30	0.19	0	1
OCMEC	4.17	<0.05	0.67	0.50

4 Concluding Remarks

In this study, empirical evaluation of coupling metrics was performed to explore their capability to identify change-prone classes in evolving object-oriented software systems. Coupling metrics were found to be statistically correlated with change-proneness of classes. Different logistic regression models were constructed for predicting change-prone classes from one software release to the next. The results indicate that a prediction model based on coupling metrics is generally more accurate than a model based on cohesion metrics. Furthermore, a prediction model based on import coupling metrics is more accurate than a model based on export coupling metrics. We also found that the coupling metrics in the C&K suite are not necessarily more accurate than other metrics in the suite in identifying change-prone classes in evolving object-oriented software. Moreover, there is no confounding effect of class size in the validity of some of the investigated coupling metrics.

There are several directions for future work. One direction is to conduct more studies that include software systems written in different programming languages (Java, C#, etc.), and also proprietary software systems and compare the results. Another direction is to empirically evaluate dynamic coupling metrics and compare them against static coupling metrics. In addition, it would be interesting to evaluate the capability of coupling metrics in estimating change size and density, and to explore the relationship between coupling metrics and other software quality attributes. Finally, a comparative study of different computational intelligence models for identifying change-prone classes could be also conducted.

5 References

[1] L. Briand, J. Daly, and J. Wust, "A Unified Framework for Coupling Measurement in Object-Oriented Systems," *IEEE Transactions on Software Engineering*, vol. 25, no. 1, pp. 91-121, 1999.

[2] L. Briand, J. Daly, and J. Wüst, "A Unified Framework for Cohesion Measurement in Object-Oriented Systems," *Empirical Software Engineering*, vol. 3, no. 1, pp. 65-117, 1998.

[3] L. Briand, P. Devanbu, and W. Melo, "An Investigation into Coupling Measures for C++," in *19th Int'l Conf. Software Eng., ICSE'97*, pp. 412-421, 1997.

[4] L. Briand, J. Wüst, J. Daly, and V. Porter, "Exploring the Relationships between Design Measures and Software Quality in Object-Oriented Systems," *Journal of Systems and Software,* vol. 51, no. 3, pp. 245-273, 2000.

[5] M. Bruntink and A. Deursen, "An empirical study into class testability," *Journal of Systems and Software,* vol. 79, pp. 1219-1232, 2006.

[6] S. Chidamber and C. Kemerer, "A Metrics Suite for Object Oriented Design," *IEEE Transactions on Software Engineering,* vol. 20, no. 6, pp. 476-493, 1994.

[7] K. El-Emam, S. Benlarbi, N. Goel, and S. Rai, "The Confounding Effect of Class Size on the Validity of Object-Oriented Metrics," *IEEE Transactions on Software Engineering,* vol. 27, pp. 630-650, 2001.

[8] K. El-Emam, W. Melo, and J. Machado, "The Prediction of Faulty Classes Using Object-Oriented Design Metrics," *Journal of Systems and Software,* pp. 63-75, 2001.

[9] M. Elish and M. Al-Khiaty, "A suite of metrics for quantifying historical changes to predict future change-prone classes in object-oriented software," *Journal of Software: Evolution and Process,* vol. 25, no. 5, pp. 407-437, 2013.

[10] M. Elish and D. Rine, "Investigation of Metrics for Object-Oriented Design Logical Stability," in *7th IEEE European Conference on Software Maintenance and Reengineering,* pp. 193-200, 2003.

[11] R. Ferenc, A. Besze´des, M. Tarkiainen, and T. Gyimo´thy, "Columbus—reverse engineering tool and schema for C++," in *IEEE International Conference on Software Maintenance*, pp. 172–181, 2002.

[12] G. Gui and P. Scott, "Ranking reusability of software components using coupling metrics," *Journal of Systems and Software,* vol. 80, pp. 1450-1459, 2007.

[13] T. Gyimothy, R. Ferenc, and I. Siket, "Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction," *IEEE Transactions on Software Engineering,* vol. 31, no. 10, pp. 897-910, Oct. 2005 2005.

[14] A. Koru and H. Liu, "Identifying and characterizing change-prone classes in two large-scale open-source products," *Journal of Systems and Software,* vol. 80, pp. 63-73, 2007.

[15] W. Li and S. Henry, "Object-Oriented Metrics that Predict Maintainability," *Journal of Systems and Software,* vol. 23, no. 2, pp. 111-122, 1993.

[16] H. Olague, L. Etzkorn, S. Gholston, and S. Quattlebaum, "Empirical Validation of Three Software Metrics Suites to Predict Fault-Proneness of Object-Oriented Classes Developed Using Highly Iterative or Agile Software Development Processes," *IEEE Transactions on Software Engineering,* vol. 33, no. 6, pp. 402-419, 2007.

[17] N. Tsantalis, A. Chatzigeorgiou, and G. Stephanides, "Predicting the Probability of Change in Object-Oriented Systems," *IEEE Transactions on Software Engineering,* vol. 31, no. 7, pp. 601-614, 2005.

[18] F. Wilkie and B. Kitchenham, "An Investigation of Coupling, reuse and Maintenance in a Commercial C++ Application," *Information and Software Technology,* vol. 43, pp. 801-812, 2001.

Effective Representation of Object-oriented Program: *the key to change impact analysis*

Isong Bassey
Department of Computer Science
North-West University
Mafikeng, South Africa
24073008@nwu.ac.za

Obeten Ekabua
Department Of Computer Science
North-West University
Mafikeng, South Africa
obeten.ekabua@nwu.ac.za

Abstract— *Today, object-oriented (OO) technology has gained worldwide popularity in software development and several OO software applications are currently in use. It is imperative that these systems are effectively and efficiently maintained. However, OO software components have different complex dependencies which often make it difficult to anticipate and identify ripple-effects of a change when faced with change proposal. In the perspective of Software Engineering Education at the undergraduate level, software maintenance stands a software development phase where much has not been done to induct students into the act. The existing software change impact analysis (CIA) approaches seems complex for students at the undergraduate level. Thus, these students have to be taught how to maintain OO software system since they are the future software developers and any representation of OOS that is effective would aid program comprehension and facilitate CIA process at this level of study. In this paper, we have proposed an approach called OOComDN for representing OO software using complex networks. The goal is to facilitate CIA and assist learners to comprehend OO program for onward maintenance. OOComDN provides a good representation of the system characteristics and is practicable for impact analysis of OOS systems as well as the quantification of the structural complexity of the software. We evaluated the approach and the results obtained were significant.*

Keywords: *Impact Analysis, Software Change, Complex Network, Students*

I. INTRODUCTION

Change is an indispensable property of software. Software during development or its life-time have to undergo changes in order to continue to remain useful and meets its operational requirements. Drivers of software change on existing systems include defects fixing, new features introduction to meet customers changing requirements, environmental adaptation or internal quality enhancement [1]. However, all these changes have possible risk in them due to unanticipated side effects elsewhere in the system. This is exacerbated especially, when the program structure or dependencies is neglected.

In such cases, impact analysis is the technique that is used as leverage. It tries to identify or estimate the consequences of the proposed change impact from the analysis of software product [2]. Software change impact analysis (CIA) is used to quantify the potential consequences of a given change or evaluates what needs to be changed to realize a change in the software with respect to time and effort [2]. In this case, CIA is serves the purposes of assisting engineers to take appropriate actions with respect to change decision, schedule plans, cost, resource and efforts estimates and so on [1][2]. In the literature today, several CIA approaches exist: *static* [3][4][5], *dynamic* [6][7][8] or *hybrid approach*es [9]. However, the current or existing CIA techniques are not explicit enough and seem complex for "novice" like undergraduate students in terms of understanding and usage during maintenance task. One reason for this is that, the teaching and learning of software engineering at the undergraduate level only places much emphasis on the development aspect and the maintenance aspect are not taken seriously. Therefore, these students have to be taught early how to maintain software systems, object-oriented software (OOS) in particular which is becoming a *de facto* in software development today. This is important because they are the future software developers and some of these students will be maintaining OOS when they start work in the industry.

Based on the above issue, in this paper we developed an approach that will assist beginners to perform CIA effectively in order to be successful in the maintenance of OOS. In this case, we proposed the use of complex networks to build an intermediate presentation (IR) of the entire OOS in order to make explicit, its implicit structures and dependencies. The approach is geared towards enhancing static CIA approach. Thus, the research question we want to answer in this paper is:

> *How can we represent OOS such that it is effective, aid program comprehension and facilitate CIA at the undergraduate level?*

The answer to the above question is provided in this study. We believe that if OOS especial small or medium scale systems at the undergraduate level is effectively represented, it would create a huge impact for students in learning how to perform CIA successfully while preserving the quality of the software with less cost in terms of time and effort.

The rest of this paper is organized as follows: Section II gives the background information, III discusses the intermediate representation of OOS, and IV discusses the OOComDN, V is the empirical evaluation of the IR, VI is the study discussion while VII is the paper conclusion.

II. STUDY BACKGROUND

Change to a software system is inevitable and software engineers would be making changes in the dark if they don't understand *what, how* and *where* of the changes they are to perform. CIA is an important technique that will assist them to determine the consequences of such software changes. As an important property of software, change is necessary in software either in requirements, design or source codes. In addition, existing approaches of CIA centered on static and dynamic approaches or hybrid approach [11]. In the perspective of static CIA approach, the static representation of the software are used to reveal its structural dependencies from the source code while in the dynamic approach, analysis on the source code is based on program event traces and execution [10].

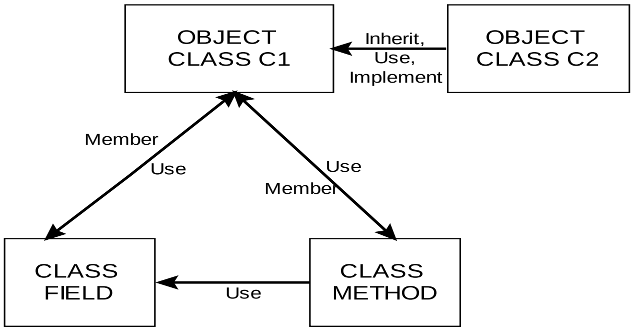


Figure 1. OOS component dependencies

In OOS source code point of view, these entities are known as software components which are fields, methods/functions, classes and packages. In particular, fields, methods and classes are usually the components that are used at granular level for analysis. When changes are considered on a component, the task of CIA technique is to find the initial change component and other components that will be truly affected by the change. However, OOS are embedded with complex dependencies that often make it difficult to identify the ripple-effects of changes [10]. The drivers of this complexity are the features such as *encapsulation, inheritance, polymorphism and dynamic binding* that distinguishes it from structured-oriented paradigm [10]. (See Fig. 1) In this case, a change in one component may affect others in a manner not anticipated.

In addition, the teaching and learning of software engineering at the undergraduate level have long been centered on software development, in particular coding. This is evident in several students' course projects both published and unpublished. This shows that much has not been done on the area of software maintenance which is a crucial area in software engineering. What needs to be known is that software engineering students at the undergraduate level are the future of the software development and many will be involved in the maintenance of software systems, OOS in particular, which is becoming the mainstream in today's software development. Thus, they have to be taught how to maintain a system alongside its development at the undergraduate level in order to be better equipped with the necessary core competencies and technical skills expected of every software engineer when they graduate. In addition, several OOS CIA approaches exist today [8][9][10]11], and due to their complex nature, they seems not suitable in terms of use at the undergraduate level. Thus, a simple approach that will assist the students at their level to understand OOS and carry out maintenance successfully is indispensable. That is, any approach that will effectively represent OOS to expose its characteristics would definitely facilitate CIA task and aid comprehension of the entire OO program at the undergraduate study level. Hence, in order to facilitate CIA and support beginners to carry out maintenance effectively, we propose the use of an IR of OOS using the complex networks. Details of the proposed approach are given in subsequent sections.

III. PROPOSED INTERMEDIATE REPRESENTATION OF OBJECT-ORIENTED SOFTWARE

This section discusses the proposed IR of OOS that will assist engineers in facilitating program understanding and CIA. The approach is the extension of the work by [11] and [12]. In this case, we used the idea of complex networks to model OOS system's structure.

A. Complex Networks in Software Systems

Complex networks in recent decades have gained increasing momentum and software system is not an exception due to its topological structure [12][13]. In this state of affairs, a software system can be modeled as complex networks where software components are represented as nodes and their interactions as edges. This is made possible because the design structure of OOS can better be explained by its structural properties in terms of components and the relationships among them. In that case, the components are the fields, methods, classes and packages, while their interactions are the different dependencies that exist between these components. Thus, the structure of OOS can effectively be represented using complex network idea of graph theory.

The importance of this IR is that today, software systems especially OOS has increased in complexity and size with structure becoming more complicated such that a change or fault in one component often requires changes/faults to several other parts in a way not anticipated. In addition, the complex structure makes it difficult to quantify the overall quality of the final software products. Therefore, analyzing OOS system's structure using complex network will help the maintainer to achieve the following objectives:

1) To visualize the software components and their complex dependencies. This will help the maintainer to have an understanding of which components will be impacted by a change when a change request is considered on a component.

2) To quantitatively analyze the quality of the entire OOS structure. That is, measuring the degree of component in terms of coupling and fault propagation from one component to another either directly or indirectly. In this case, analyzing software structure quantitatively would help software maintainer to know before hand, the quality of the system and the risk of fault propagation from one component to the other. This is necessary to take mitigating actions where necessary, in order to reduce the risk of software failure after implementing the actual change.

IV. OO Component Dependency Networks

The IR proposed in this paper is called the OOComDN. This is used to represent components and their relationships in OOS system. Consequently, OOS components are nodes and the interaction between every pair of the components is a directed "*weighted*" edge with an edge type indicating the probability that a change or fault in one component may propagate to the other. OOComDN will be considered in two perspectives: change and fault diffusion networks.

A. Change Diffusion Networks

In change diffusion network (CDN), we represent OOS system using a "*weighted*" direction graph, G where components are the vertices and the dependencies among the components are the edges taking both the semantics and syntactic structure into consideration. CDN is used to represent the software components and their relationships for onward maintenance task, perhaps, CIA. It explicitly represents the structure of the OOS source code and assists the software maintainer in quantifying which components will be truly affected by a change. In other words, the representation is basically used to discover the evolution mechanism of the software system.

In this study, we identified four types of dependencies, $\mathbf{D^{Type}}$ that exist in OOS: *inheritance (H)*, *usage (U)*, *invocation (V)*, and *membership (M)* [11]. These dependencies are non-numeric weight assigned on the edges of the OOComDN-1 and constitutes the links by which a change or fault propagates from one component to other once a change is consider on a specific component. Based on the CDN and the $\mathbf{D^{Type}}$ the following definitions of OOComDN are considered: *OOComDN-1* and *OOComDN-2*.

Definition 1: *[OOComDN -1]*
Given OOS, program, P let G = $<(\mathbf{N},\mathbf{D^E}),\ \mathbf{D^{Type}}>$ *represent OOComDN given by:*

$$\text{OOComDN-1} = <(\mathbf{N},\ \mathbf{D^E}),\ \mathbf{D^{Type}}>$$

Where $N = NP^k + N^C + N^M + N^F$ are the nodes and $D^E = N\times N\times \mathbf{D^{Type}}$ represents the set of various edges with dependencies types, $\mathbf{D^{Type}}$. We referred to $\mathbf{D^{Type}}$ as the weight of the graph and NP^k, N^C, N^M and N^F represent the set of packages, classes, member methods and fields respectively. Each component is represented by only one node and the weighted-directed edge between two nodes indicates that a component is in, or uses a member of or invokes or inherits other components.

B. Typical Illustration

A typical illustration of the OOComDN is captured in Fig. 3 using the program, P written in Java as shown in Fig. 2. The various shapes used to represent each component in the OOComDN-1 are also captured in Fig. 3.

```
package p1;
public class A {

public A(){};
private int d;

public void M1()
{ d=2; }

public int M2(int x)
{ M1();
x= d + 10;
return x; }}

public class B
extends A {
public B() {};
private int a;

public void M3()
{ a=5; }

public int M4(int b)
{ M3();
int c = a+b+10;
return c; }}
```

```
package p2;
import p1.*;

public class C {
    public C(){};
    private p1.B k;

public void M5()
{    k.M4();   }}

class D extends C {
public D() {};

private String q;

public void M6()
{ q="Boy!";
B j ;  j.M4();
A p; p.M1(); }}
```

Figure 2. Java program

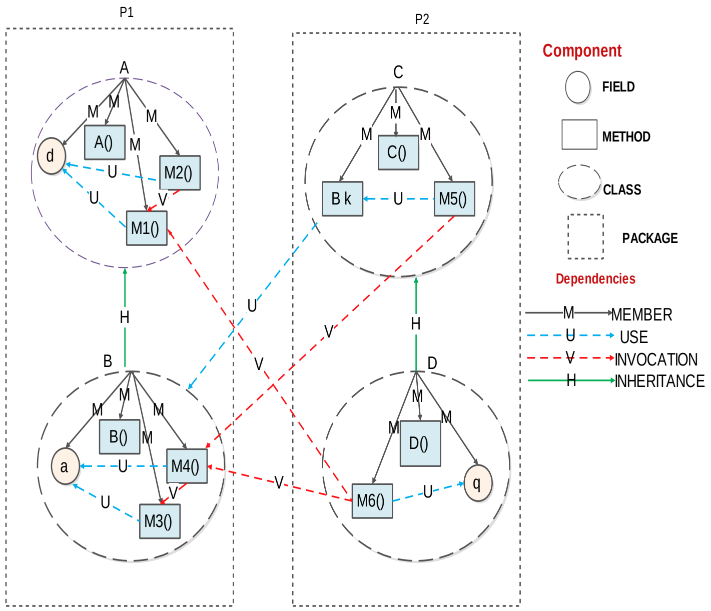


Figure 3. OOComDN of the java program in Figure 2

In Figure 3 we represent OOS system captured in Fig. 2. On the OOComDN-1 A, B, C and D are the classes in P. In this case, if a component says D uses or inherits or invokes a class say A, there will be an edge emanating from the node D to node A. Furthermore, the multiplicities of these dependencies are taken into account based on the *type of change* to be performed on a given node in the OOComDN-1. The weight of each directed edge will indicate the probability that a change in A may impact D.

C. *Degree of OOComDN-1*

After the construction of the OOS as OOComDN-1, we can compute its degree, Z. Z of a node in an OOComDN is the number of dependencies a component has against other components connected to it or it is connected to. Two types of Z exist: *in-degree* and the *out-degree*. The computation of the Z is used to identify the nature of coupling of each component in the program as well as the structural complexity of the software at the class level. (see Fig. 4) It gives an insight into how components are related to one another in terms of coupling and what need to be done to accomplish a change when a change is consider on one component. Degree computation is done at the class level and in order to compute it, we have to prune OOComDN-1 to include only classes and their dependency types as shown in Fig. 4.

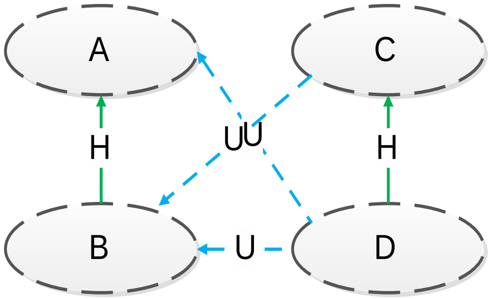


Figure 4. Class level OOComDN-1

Definition 2: [*Degree of OOComDN-1***]**

Given, OOComDN, < **(N, D^E), D^{Type}***>, with an adjacency matrix* $\boldsymbol{A_{ij}}$*, the degree of a vertex,* Z_i*, we defined the out-degree of an OOS component as the number of edges or connections originating from that component. It is given by* $|Z^{out}(n_i)|$ *which is the sum of the* i^{th} *column of the* $\boldsymbol{A_{ji}}$*.*

$$Z_i^{out} = \sum_j A_{ji} \text{ ………………………………………………..1}$$

While in-degree of an OOS component, n_i *is the total number of edges or connections onto that node and it is given by* $|Z^{in}(n_i)|$ *which is the sum of the* i^{th} *row of the* $\boldsymbol{A_{ij}}$*.*

$$Z_i^{in} = \sum_j A_{ij} \text{ ………………………………………………...2}$$

$\boldsymbol{Z^{tot}}(n_i)$ *is the total number of directed edges into and out of the node,* $\boldsymbol{n_i}$ **ЄN***. It is simply the sum of* Z_i^{in} *and* Z_i^{out}*.*

$$Z_i^{tot} = Z_i^{in} + Z_i^{out} \text{ ……………………………….3}$$

In other words, $Z^{in}(n_i)$ indicates the number of classes that has dependency on class n_j ЄN and $\mathbf{Z}^{out}(n_i)$ the number of classes on which class n_i ЄN depends on. The in-degree and out-degree for the program shown in Fig. 2 is captured in Table I.

TABLE I. IN-DEGREE AND OUT-DEGREE IN OOCOMDN-I OF FIGURE 4

Node, n_i	Z_i^{in}	Z_i^{out}	Z_i^{tot}
A	(B,A) = 1, (D,A) = 1	-	2
B	(C,B) = 1, (D,B) = 1	(B,A) = 1	3
C	(D,C) = 1	(C,B) = 1	2
D	(D,A) = 1, (D,B) = 1 (D,C) = 1	-	3

As can be seen in Table I, class A has one in-degree for the ordered paired (B,A) and (D,A) and no out-degree. This clearly shows the nature of coupling in A which will assist a maintainer to know the complexity of the class at hand before making a change. In addition, the complex relationships among a very large number of software components in OOS would often result to the structural complexity of software system. Thus, the degree of a class, Z being similar to CK's CBO metric, in a software network actually shows the degree to which each class depends on other classes. In the context of this study, Z is used to measure software degree of coupling in a small or medium sized system.

D. *Fault Diffusion Network*

Fault diffusion network (FDN) is similar to the one proposed by [12] and is represented just as CDN. The only difference is that the semantics of the relationship is neglected and every relationship has the same importance. In the context of this paper, FDN is used to characterize the risks a component poses on others due to the direct or indirect dependency existing between them. The rationale is that, though it is believed that a fault in one component will propagate to other components that depend on it, it is not always true with respect to OOS systems. The intuition is that, OOS class is composed of several fields and methods and a class is considered faulty if it has at least one fault emanating from either itself or its members. In this case, members of another class that depends on such faulty class do not all connect to the faulty member directly or indirectly. Hence, the propagation of fault from one component to another is based on probability. The definition below is important:

Definition 3: *[OOComDN-2]*
In FDN the nodes represent the classes and a class is represented by only one node in the entire OOComDN-2. Interactions between classes are represented by directed weighted edges.

Thus, OOComDN-2 can be described as:

$$\text{OOComDN-2} = <N_C, D_C, P_b>$$

Where $\mathbf{N_C}$ is the set of classes, $\mathbf{D_C}$ is the set of edges linking one class to another and $\mathbf{P_b}$ is the probability that a fault in a class will propagate to another.

The interaction is based on the principle that, if members in class, say A use class members of B, an edge will originate from the node of the member in class **A** to the node in **B**, and vice versa. For simplicity, we consider the existence of dependency and ignore the $\mathbf{D^{Type}}$ as well as the multiplicity of dependencies irrespective of how many times **A** depend on **B** and so on. Furthermore, the weight of each $\mathbf{D_C}$ represents the probability that a fault in class **B** will impact or spread to class **A**. We captured this in Fig. 5.

Definition 4: ***[Fault Propagation Probability]***
*Let **P** be an OO program having class **i** and class **j**, where class **j** depends on class **i**. We therefore, define the probability of fault propagating from class **i** to class **j** as $\boldsymbol{P_b}$ **(i,j)**. In this paper, we defined it as follows:*

$$\mathbf{P_b}(\mathbf{j}, \mathbf{i}) = \frac{|\mathbf{CM(i,j)}|}{|\mathbf{MT_j}|} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 4$$

Where **CM(i,j)** is the set of members in class *j* whose faults will propagate to the members in class *i*, which they are directly or indirectly linked to thereby rendering class faulty. While $\mathbf{MT_j}$ is the total number of class members present in class *j*.

$$\text{CM(D,A)} = \{M1()\} \text{ and } \mathbf{MT_A} = \{d, A(), M1(), M2()\}$$

$$\text{CM(D,B)} = \{M4()\} \text{ and } \mathbf{MT_B} = \{a, B(), M3(), M4()\}$$

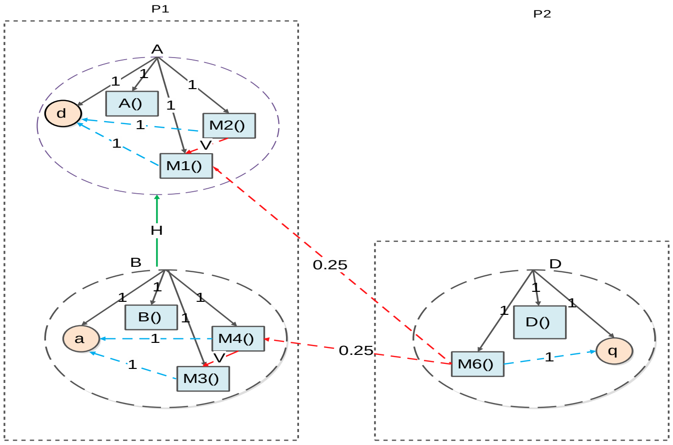


Figure 5. Class fault propagation probability

In Fig. 5 we captured the fault propagation probability in a class. The edges of all members in a class are denoted by 1which indicates the probability of one member of the class being faulty due to the dependency it has with a faulty member is 1. For inter-class dependency, the case is not always true. This is the rationale for this approach. For instance, as shown in Fig. 5, it is clear that class **D** depends on class **A** and **B** as follows:

$$\text{(D.M6(),A)} = \{\text{M1()}\} = \text{D.M6()} \rightarrow \text{A.M1()}$$

$$\text{(D.M6(),B)} = \{\text{M4()}\} = \text{D.M6()} \rightarrow \text{B.M4()}$$

Therefore,

$$\mathbf{P_b}(\mathbf{D}, \mathbf{A}) = \frac{|\mathbf{M1}()|}{|\{\boldsymbol{d}, \boldsymbol{M1}(), \boldsymbol{M2}(), \boldsymbol{A}()\}|} = \frac{\mathbf{1}}{\mathbf{4}} = \mathbf{0.25}, \text{ and}$$

$$\mathbf{P_b}(\mathbf{D}, \mathbf{B}) = \frac{|\mathbf{M4}()|}{|\{\boldsymbol{a}, \boldsymbol{M3}(), \boldsymbol{M4}(), \boldsymbol{B}()\}|} = \frac{\mathbf{1}}{\mathbf{4}} = \mathbf{0.25}$$

The above computation is based on eqn(4) where P_b(D, A) = P_b (D, B) = 0.25, 25%. This denotes that, since M6() in class D depends on class A and B, the probability that a fault in class A or B will have impact on class D is 25%. In this case the higher the probability, the higher the risk of fault propagation. For dependency of inheritance type, probability can't be quantified because members in the classes are not connected directly. The approach discussed in this paper can be used for quantitative measurement of the structural quality of the software as well as its risk assessments. This is important to allow the maintainer know which components has higher risk probability of propagating faults to its neighbors during the course of maintenance such that mitigating actions could be taken on time. In the same vein, a smaller risk value signifies that a fault in the measured component poses no serious impact on the other components and modification can be performed hitch-free.

V. Empirical Evaluation

A. Study Subject, Setting and Maintenace Tasks

To assess the effectiveness and significance of the approach, IR of OOS proposed in this paper, we performed a controlled experiment using small-size systems developed by students in their projects. The subjects were only undergraduate Computer Science students of our department and the study was in fulfillment of the Software Engineering curriculum with a focus on software maintenance techniques. The subjects in their third year of study were divided into nine groups (A, B, C, D, E, F, G, H and I) of five students each and each student had comparable levels of education and experience in software development and java programming in particular. For each team selection, strict measures were taken to blend the team with the required skills needed. In order to be effective in carrying out maintenance, subjects had a week of theoretical knowledge on software maintenance, the basic knowledge needed for CIA using IR of OO program and others. The goal of the controlled experiment was to demonstrate whether a good and effective representation of OO program can increase the understandability of the maintainer to perform modification tasks successfully. In this case, to be able to maintain and change a system efficiently and correctly, the maintainer has to have an in-depth understanding of the systems' structure (source code). By *efficiency*, we mean the minimum time taken to carry out the change while *correctness* is the intended functionality and less side-effects of the change.

The characteristics of the system collected from the subjects include Team A, D, F, H, and I system's had five class each while team B, C, E, and G six classes each. The maintenance task was to perform modification task on other team's system. In this case, there were four maintenance tasks the subjects performed during the course of the experiment: MTask$_1$ - *one class change*, MTask$_2$ - *one class change*, MTask$_3$ - *two methods change*, and MTask$_4$ - *one field change*. The changes were based on the different change types applicable for OO program. The overview of the experiment design is captured in Fig. 6.

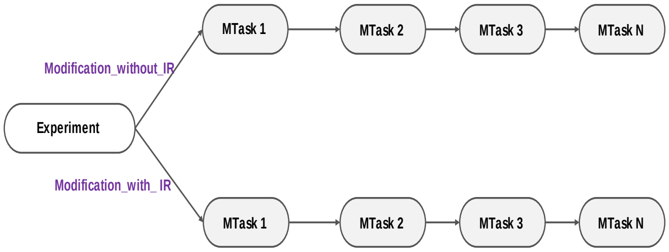


Figure 6. Experimental design overview

B. *Experimental Variables*

During the course of the experiment, the variables that were of importance at each phase of the maintenance task are the change duration, program correctness, the number of errors the change introduced and the task phase. For change duration (CD), we took the starting and finishing time of the modification task. Also for the program correctness (PC), each team was graded between 0-100% based the outcome of the tasks and the correct program execution while for the number of errors (NoE), we computed NoE introduce by the modification task after the changes were made via recompiling the program. In this case, NoE were computed based on the number of lines affected as indicated on the development IDE used. These were all performed by the supervisor and the team members. Lastly, for the TaskPhase, two variables were important: *modification without IR* or *modification with IR* (MTask1- MTask4). (See Fig.6)

Due to the programming skills of the subjects, we first assessed the each team's program for actual amount and complexity of classes that would be impacted by each change and the approximate time required to carry out the tasks. This was necessary in order to quantify the degree of difficulty of the change tasks. However, the results we obtained from the experiment put forward that this approach was adequately appropriate in this regard.

C. *Hypothesis*

We tested hypotheses in the experiment to assess the significance of the IR to CIA during the maintenance task. Thus, the null hypotheses of the experiment were as follows:

Impact of TaskPhase on Change Duration (CD):
$H0_1$: The time taken to perform maintenance task is equal for modification without IR and modification with IR.

Impact of TaskPhase on Program_Correctness (PC):
$H0_2$: The correctness of the program after maintenance task is the same for both modification without IR and modification with IR.

Impact of TaskPhase on Number of Error Introduced:
$H0_3$: The number of error introduced in a changed program is equal for modification without IR and modification with IR.

For the effect on duration (CD), the test was to evaluate if using IR constitutes a time wastage or not on the part of the maintainer while the effect on correctness (PC) would be to evaluate if using IR during maintenance contributes to program understanding or not. In this case, if correctness is equal for both, then it is not useful for CIA. However, if the program correctness is more for *modification with IR* than *modification without IR*, then it is useful for CIA and aids comprehension of the program as well. Furthermore, for NoE, the task would be to test if the number of errors introduced after modification is equal in both case or not. If it is lower with the TaskPhase, *modification with IR*, then it is useful, otherwise not useful for CIA. The statistical techniques use is the dependent T-test.

D. *Results*

The main results based on the task phases: *modification without IR* and *modification with IR* for MTask1 – Mtask4 are visualized in Fig.7 and Fig.8 respectively. The change duration, % program correctness and a count of error are shown on the Y-axis, while the project group is shown on the X-axis. As can be seen, there are some clear indications that TaskPhase affect CD, PC and NoE in the two phases. For instance, a small amount of time was utilized to implement a change on a program when IR was used in phase II than in phase I. Accordingly, the program correctness was better when IR was used in the maintenance task and the same result is applicable to NoE introduced in both phase. However, for practical importance, it is essential to see if these differences are significant by testing the above specified hypotheses.

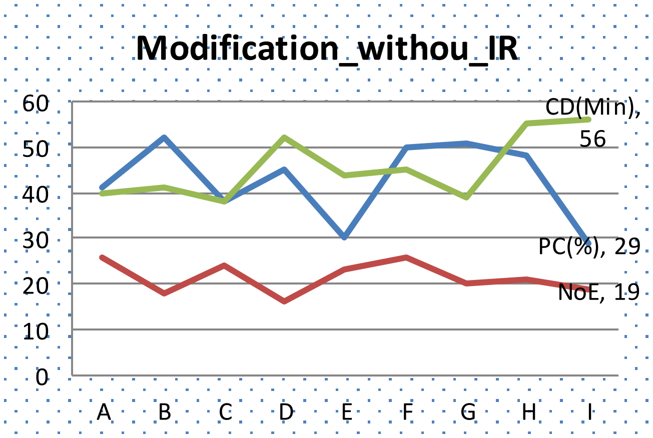


Figure 7: Effect of TaskPhase on CD, PC and NoE

Copyright © 2014 CSREA Press, ISBN: 1-60132-286-0; Printed in the United States of America

In this case, we employed the paired-sample T-test to test the hypothesis. The results of the hypotheses regarding CD, PC and NoE for the maintenance tasks ($MTask_1$-$MTask_4$) in both Task phases are captured in Table I. The results indicate that TaskPhase does have a significant effect on the program correctness, change duration and number of errors introduced. The hypothesis was tested at a significance of $p \leq 0.05$.

TABLE II. DEPENDENT T-TEST RESULTS FOR CD, PC AND NOE

Paired variable	T	DF	P-value Sig.
CD - CDII	-8.541	8	0.000
NoE - NoEII	10.509	8	0.000
PC - PCII	5.646	8	0.000

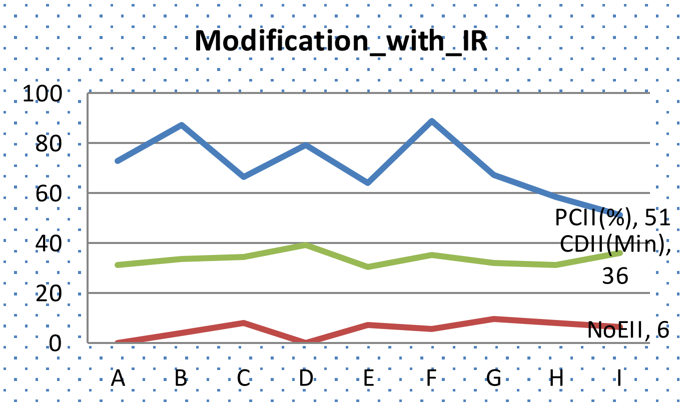


Figure 8: The effects of TaskPhase on CDII, PCII and NoEII

The summary of the results of the hypothesis tests is as follows: (1) For the impact of TaskPhase on CD, we rejected $H0_1$ since p-value ≈ $0.00 \leq 0.05$. (2) For the impact of NoE introduced, we rejected $H0_2$ since p-value ≈ $0.00 \leq 0.05$, and lastly, (3) For the impact of TaskPhase on PC, we rejected $H0_3$ since p-value ≈ $0.00 \leq 0.05$. In conclusion, at the $\alpha = 0.05$ level of significance, there exists enough evidence that there is a huge difference in mean CD, PC and NoE of both phases of the of maintenance tasks (*modification without IR* and *modification with IR*). These results thus, prove that the IR of OOS is effective and useful in facilitating CIA.

VI. DISCUSSION

The results of the experiment seem very interesting in terms of duration, program correctness and the number errors introduce after change were implemented for phase II. As captured in Fig. 7 and Fig. 8 respectively, we can see that time taken by the subjects to perform the maintenance task in phase II (36 min maximum) were significantly smaller than the modification duration of phase I (56 min maximum). Accordingly, the correctness of the maintenance task (correct solutions) was significantly higher for phase II (56% minimum) than for the phase I (51% minimum). Moreover, the number of errors introduced after the changes were made was significantly lower for phase II (6 maximum) when the *modification with IR* was used as opposed to *modification without IR* (19 minimum).

The results suggest the effectiveness of the IR for CIA. In this case, using IR of OOS will actually reduce the time needed to make changes, the correctness of the solution and the number of errors that will be introduced after the change. However, the interpretation of these results requires care. This is because, though we took good time to blend each team with skillful and experienced subjects, the experiment actually did not took care of such experiences and skills in term of the team. In this case, the level of skill and experience of each team differs and could affect the maintenance task in terms of efficiency and comprehension. Factor that could also affects the results are the system's structural properties such as coupling, cohesion and inheritance. Naturally, a good design involves having low coupling and high cohesion in a system for maintenance to effective. Unfortunately, the reverse of these design properties (high coupling and low cohesion) is known to have negative effect on change propagation across systems. Consequently, much time could be spent by each team on comprehending and performing changes correctly. In addition, while some errors still remained in most of the team's program after changes were made could be as a result of either undiscovered indirect impacts resulting from the system's structural properties or the programming experience of the subjects.

VII. CONCLUSION

In this paper, we have proposed an effective approach to represent OOS such that it can aid program comprehension and onward software maintenance. The OOComDN constructed is quite simple, easy and do not analyze deeply into method body. All the dependencies are clearly revealed. Unlike other dependency graphs, OOComDN is not complex and the components involved are countable. In this case, OOComDN can be used to teach beginners such as undergraduate to understand the structure of OOS and perform CIA effectively during maintenance. In addition, it can be used to quantify the structural complexity of the system especially for small or medium-size systems without using OO design metrics. To ascertain the significance of the IR, we performed empirical evaluation of the approach and the results obtained were significant. In general, the representation is effective and practicable for impact analysis of OOS systems.
However, the limitation of the study is that, we used small size systems to evaluate the IR. In addition, the participants involved were students and are not as skillful as professionals. We believe these will affects the results reported here. However, we took strict measures to ensure quality in the experiments and the results presented are valid.

REFERENCES

[1] Bohner, S. A. and Amold, R. S., "Software Change Impact Analysis," IEEE Computer Society Tutorial, IEEE Computer Society Press,1996

[2] Bohner, S. A., "A Graph Traceability Approach to Software Change Impact Analysis," Ph.D. Dissertation George Mason University, Fairfax, VA, 1995

[3] X. Sun, B. Li, C. Tao, W. Wen, and S. Zhang. "Change Impact Analysis Based on a Taxonomy of Change Types" 2010 IEEE Proceedings of 34th Annual Computer Software and Applications Conference (COMPSAC 2010), 2010. pp.373-82

[4] L. Badri, M. Badri, and S. D. Yves. Supporting predictive change impact analysis: a control call graph based technique. In Proceedings of Asia-Pacific Software Engineering Conference, 2005

[5] S. Zhang, Z. Gu, Y. Lin, and J. J. Zhao. Change impact analysis for AspectJ programs. In Proceedings of International Conference on Software Maintenance, pages 87 – 96, 2008

[6] Law, J., Rothermel, G., "Whole program path-based dynamic impact analysis", The Intl Conf. on Software Engineering, 2003.

[7] J. Law and G. Rothermel. Incremental dynamic impact analysis for evolving software systems. In Proceedings of International Symposium on Software Reliability Engineering, 2003

[8] T. Apiwattanapong, A. Orso, and M. J. Harrold. Efficient and precise dynamic impact analysis using execute after sequences. In Proceedings of International Conference on Software Engineering, pages 432 – 441, 2005.

[9] M. Oliveira et al: "The Hybrid Technique for Object-Oriented Software Change Impact, Analysis" Proceedings of the 14th European Conference on Software Maintenance and Reengineering (CSMR 2010), IEEE Press, 2010, pp.252-255

[10] Lee, M. et al.: "Algorithmic analysis of the impacts of changes to object-oriented software" 34th International Conference on Technology of Object Oriented Languages and Systems. pp. 61-70, (August 2000)

[11] Sun,X., Li, B., Tao, C. Wen, W. and Zhang, S.: "Change Impact Analysis Based on a Taxonomy of Change Types" 2010 IEEE Proceedings of 34th Annual Computer Software and Applications Conference (COMPSAC 2010), pp.373-82, 2010

[12] Pan, W.F., Li B, Ma Y.T. et al: Measuring structural quality of object-oriented software via bug propagation analysis on weighted software networks. Journal of Computer Science and Technology, 25(6): 1202–1213 Nov. 2010. DOI 10.1007/s11390-010-1095-2

[13] Liu, J., Lu, J., He, K. and Li, B.: Characterizing the structural quality of general Complex software networks. International Journal of Bifurcation and Chaos, Vol. 18, No. 2 (2008) 605–613

Web Based Automated Workflow System for Employees’ Welfare and Loan Scheme Using Lightweight Methodology

Okonigene Robert E[1]., John-Otumu M. A[2]., John Samuel N[3]., Ojieabu Clement E[4].

[1,4]Department of Electrical and Electronics Engineering, Ambrose Alli University, Ekpoma, Edo State, Nigeria.
[2]Department of Banking & Finance Ambrose Alli University, Ekpoma, Edo State, Nigeria.
[3]Department of Electrical and Information Engineering, Covenant University, Ota, Ogun State, Nigeria

Abstract - *This paper addresses the daily challenges encountered by employees as it relates to their voluntary contributory welfare scheme in most established institutions, establishments or organizations in Nigeria. We critically examined the challenges of joining as member, amount to save, dividends sharing, assets and liabilities, and also the loan scheme using Ambrose Alli University, Ekpoma, Edo State, Nigeria as our study center. A Web Based Automated Workflow System for Employees Welfare and Loan Scheme was developed, which captured the employees’ monthly contributory savings, and other parameters through an interface. Employees’ can interact with the system from the comfort of their homes and from anywhere in the world via Internet connectivity. The application is web based and enables functions like online application for loan facilities, checking of total contributions, tracking of loan application status and other services rendered by the respective welfare scheme. The Web Based Automated Workflow System was developed based on light-weight methodology. Hypertext Markup Language (HTML), Hypertext Preprocessor (PHP), Javascript, Dreamweaver and MySQL were used to realize the interface and Web Based solutions for the Automated Workflow System for Employees Welfare and Loan Scheme. However, the security built into the scheme was not discussed in this paper.*

Keywords: Web-Based, Automated Workflow, Service Oriented Architecture, Employees Welfare

1 Introduction

In some Nigerian Universities, employees voluntarily join any of the different workers union welfare and loan schemes or associations available based on their job classification on assumption of duty. In Ambrose Alli University, Ekpoma, specifically, there are two unions, that is, the Academic Staff Union of Universities (ASUU) which has the Teaching or Academic staff as members and the Non-Academic Staff Union (NASU) which has the Junior Non-teaching staff as members. The university also has two Associations and a cooperative society. The Senior Staff Association of Nigerian Universities (SSANU) has the Senior Non-Teaching staff as members and the National Association of Academic Technologist (NAAT). The Ambrose Alli University (AAU) Multipurpose cooperative society has as its members all fulltime staff of the University who followed the due process to register as a member. Equally, non staff of the University can also be registered under separate conditions. These bodies have their various welfare and loan schemes, in which members saves certain amount of money monthly in any of the welfare scheme, from their monthly salaries. Each welfare scheme, through elected representatives, may use part of the money saved by members to do business of different kinds in order to realize some level of profit which is then shared as dividend to its registered members. Often, employees of the University are attracted to their union or association welfare and loan scheme or other union’s welfare and loan scheme or cooperative society for financial assistance or loan facilities due to the union’s or cooperative society reduced interest rates and short queue on loan applications. This is preferable rather than going to banks for such loan facilities with higher interest rates to solve their immediate needs like payment of house rent in order to avoid quarrels with landlords or caretakers; payment of children’s school fees; poor dietary intake that might result to malnourishment; inaccessibility to adequate medical care; inability to meet with social clubs financial obligations, and so on. Employees’ and their respective welfare and loan schemes and cooperative society are faced with different challenges on the day-to-day operations of the scheme; this is due to the manual workflow system of operations. Some of the major problems are as follows:

i. The delay in the end-to-end processing of loan / commodity application forms submitted.
ii. The errors sometimes made due to manual computations.
iii. Improper records keeping due to human nature.
iv. The inability for an employee who is a member of

welfare scheme to check the amount he or she has saved with the scheme at real time and at will.

v. The inability for members to apply for loan or other facilities from the comfort of their homes and track the status of their application at any time and from anywhere in the world without physically going the welfare scheme office.

However, the major businesses done by these bodies are granting of loan facilities at moderate interest rate to its members and stand as surety in procuring of commodity items for their members. The daily operations of the various welfare and loan schemes are governed by set of business processes, in which there is interaction between humans and manual information system (paper based). Conventionally, these processes have been supported by the exchange of information recorded on paper. This paper work enables the sharing, computation and archival of information as work is transferred from one desk to another until the process is fully executed. These tasks are executed when the relevant office receives a request containing the relevant information to be treated in a paper trail during office hours. For example, an employee requesting for loan facility to take care of his or her pressing needs from his / her union's welfare and loan scheme will have to apply officially by filling a form. The application forms ideally involve team working collaboratively for documentation, recommendations, and approval of the applicant intent during office hours no matter the urgency of the applicant's challenges. Therefore, this mode of processing takes a longer time to complete the given task. Thus, the paper-based process is somewhat slow due to the time it takes to move information from one desk to the other. Therefore, we decided to develop a Web Based Automated Workflow System for Employees' Welfare and Loan Scheme Using Lightweight Methodology. Welfare is a corporate attitude or commitment reflected in the expressed care for employees at all levels, underpinning their work and the environment in which it is performed [1]. Employee welfare is a comprehensive term including various services benefits and facilitates offered to employees by employer. Employee welfare includes providing staff and workers' canteens, providing savings schemes; pension funds and leave grants, making loans on hardship cases; providing assistance to staff transferred to another area and providing fringe benefits [2]. Job satisfaction is generally recognized as a multifaceted construct that includes employee feelings about a variety of both intrinsic and extrinsic job elements. Welfare schemes are a means to improve the productivity and efficiency of the employees. Employee benefits are the elements of remuneration given in addition to various forms of cash pay [3]. The benefits contribute to a competitive total remuneration package that both attracts and retains high quality employees. The cost of employee benefits has been rising in developing world [4]. The various types of employee benefits includes pension schemes, personal security, financial assistance, personal needs, subsidized meals, clothing allowance, mobile phone credit, company car and petrol allowance among others [5]. Employee benefits are provided with the understanding there is a return to the organization in terms of improved employee commitment and productivity [6]. The implementation of employee benefits requires significant amount of financial, physical and human resources [7]. The government can intervene with a policy to obligate employers to provide certain benefits to employees [8]. Workflow concept has evolved from the notion of process in manufacturing and in the office [9]. Such processes have existed since industrialization, and are product of a search to increase efficiency by concentrating on the routine aspects of work activities. They typically separate work activities into well-defined tasks, roles, and procedures which regulate most of the work in manufacturing and the office. Initially, processes were carried out entirely by humans who manipulated physical objects. With the introduction of information technology, processes in the work place are fast becoming automated by information systems, that is, computer programs performing tasks and enforcing rules which were previously implemented manually. Processes in an organization are categorized into material processes, information processes, and business processes [9]. The scope of a material process is to assemble physical components and deliver physical products. That is, material processes relate human tasks that are rooted in the physical world. Such tasks include, moving, storing, transforming, measuring, and assembling physical objects. Information processes relate to automated tasks (that is, tasks performed by programs) and partially automated tasks (that is, tasks performed by humans interacting with computers) that create, process, manage, and provide information. Typically an information process is rooted in an organization's structure and/or the existing environment of information systems. Database, transaction processing, and distributed systems technologies provide the basic infrastructure for supporting information processes. Business processes are market-centered descriptions of an organization's activities, implemented as information processes and/or material processes. That is, a business process is engineered to fulfill a business contract or satisfy a specific customer's need. Thus, the notion of a business process is conceptually at a higher level than the notion of information or material process. Business process reengineering involves explicit reconsideration and redesign of the business process. It is performed before information systems and computers are used for automating these processes. Information process reengineering is a complementary activity of business process reengineering. It involves determining how to use legacy and new information systems and computers to automate the reengineered business processes. The two activities can be performed iteratively to provide mutual feedback. While business process redesign can explicitly address the issues of customer satisfaction, the information process reengineering can address the issues of information system efficiency and cost, and take advantage of advancements in technology.

2 Using the Light Weight Methodologies

The method employed in this research work included, data collection, Engineering software development suites, light weight methodology, design and web development tools/Internet programming language. We used the sources below to collect data about the various Welfare and Loan Schemes, including the Cooperative Society available in the University.

(i) Archival Records
(ii) Observations
(iii) Interviews of stake holders.

Light weight development methodologies embrace practices that allow programmers to build solutions more quickly and efficiently, with better responsiveness to changes in business requirements [10]. Light weight methodology mainly focuses on development based on shirt life cycles. Some popular light weight development methodologies are Agile Process Model, Extreme Programming (XP), Prototype Model, Scrum, Rapid Application Development (RAD) Model. In this research work we applied the Rapid Application Development (RAD) Model.

2.1 Rapid Application Development (RAD) Model

Rapid Application Development (RAD) is an incremental software development process model that emphasizes a very short development cycle and encourages constant feedback from customers throughout the software development life-cycle. The main objective of Rapid Application Development is to avoid extensive pre-planning, generally allowing software to be written much faster and making it easier to change environments. Figure 1 is a typical RAD Protype Model.

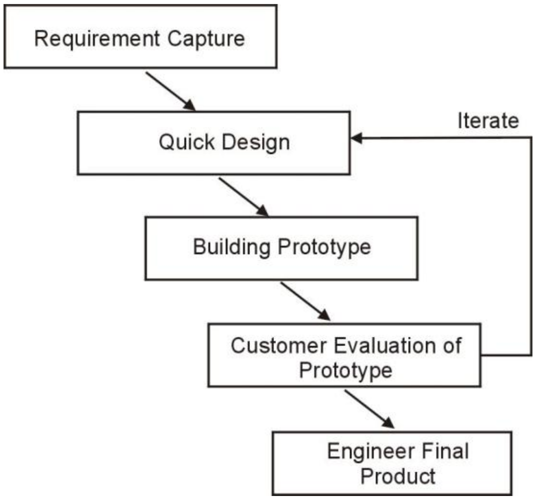


Figure 1. RAD Prototype Model adopted from [11]

The following are the advantages of RAD Model

1. Time to deliver is less
2. Quick development results in saving of time as well as cost.
3. Productivity with fewer people in short time.
4. Progress can be measured.

2.2 Software Architecture Design

Software architecture intuitively denotes the high level structures of a software system. It can be defined as the set of structures needed to reason about the software system, which comprise the software elements, the relations between them, and the properties of both elements and relations [12].

Software architecture also denotes the set of practices used to select, define or design software. Figure 2 depicts the software architecture of our developed system.

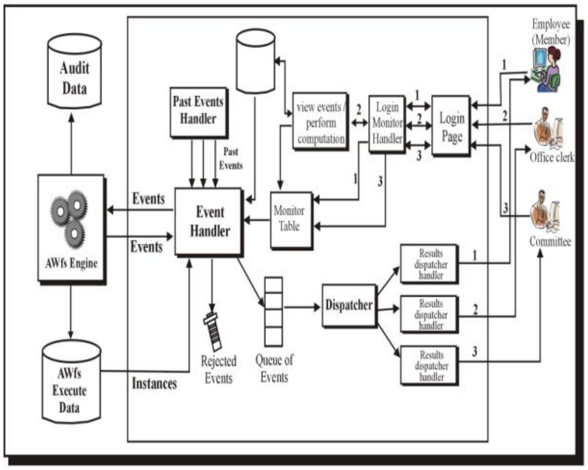


Figure 2. Software architecture of the developed system

3 Test results: Users Interaction with the proposed system

Step 1: The Office Clerk establish a connection to the Internet through an Internet Service Provider (ISP), open up a web browser and enter the web site address at the uniform resource locator (URL) to make a request for the Site’s home page. The client device can be a Laptop, Smart phone, Desktop computer, PDA, or Tablet.

Step 2: The Office Clerk login through the home page.

Step 3: At this stage, the Office Clerk can now view his desktop profile or control panel, in order to perform his full function like capture old and new members, edit member records, post and view monthly contributions, view and process various forms of loan application forms, post loan repayment, generate different types of reports, etc on the developed system.

3.1 Registered Member

Step 1: The registered member establishes a connection to the Internet through an Internet Service Provider (ISP), open up a web browser and enter the web site address at the uniform resource locator (URL) and then make intended request from the Site's home page.

Step 2: The Member then login following displayed instructions.

Step 3: The Member then proceed to view his/her desktop profile, in order to perform his/her functions like changing of password, view operation procedures, view contribution details, apply for loan, track loan status, etc on the developed system.

3.2 Use Case Diagram

A Use Case diagram is a representation of a user's interactions with the system. A Use Case diagram can portray the different types of users of a system and the various ways that they interact with the system. The Use Case diagram, shown in Figure 3, depict the Employee user and Office clerk or Office Administrator interaction with the system

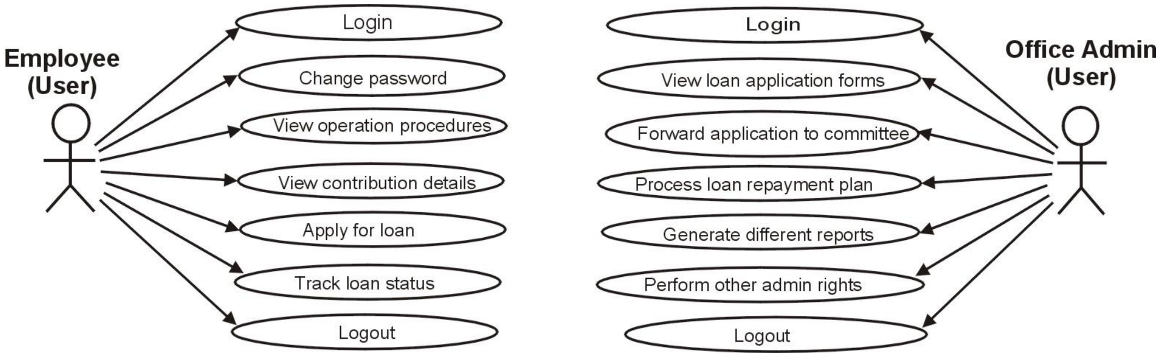

Employee (User)
Login
Change password
View operation procedures
View contribution details
Apply for loan
Track loan status
Logout
Office Admin (User)
Login
View loan application forms
Forward application to committee
Process loan repayment plan
Generate different reports
Perform other admin rights
Logout

Figure 3 Use case diagram showing Employee user and Office Admin or Clerk

3.3 Automated Workflow Process between users and the developed system for Loan Application and Approval

Step 1: The Employee user(s) logs on to the developed web application, views his or her total savings with the welfare scheme online, applies for loan facility and then submit the application online. In this step, the internal process moves in this order: Login Monitor handler→ Monitor table → Event handler → Automated Workflow System Engine (AWfs Engine). From the AWfs Engine, the event is committed to the database, while the system also puts the event to queue for the next action.

Step 2: The Office Clerk/Admin login in order the view the processes or events on queue waiting for attention. The events are stacked and programmed using the order of First Come, First Serve (FCFS) Algorithm. He pulls the first event from the queue and treats; after treating he submits the process. Submitting involves the event to move to the next phase i.e. the committee members. The process flow is in this order in the software architecture: Awfs engine → Event handler → Queue of Events → Dispatcher → Results Dispatcher Handler

Step 3: The Committee user(s) at this stage, will also view the events as they come in queue. They also process each event by approving or denying the events.

4 Conclusion

We have achieved our objective in developing the software architectural design using light weight methodology to build up a web based automated workflow system for employees welfare and loan scheme in Ambrose Alli University, Ekpoma, Nigeria to solve their local problems. The light weight software process model or methodology we used was Rapid Application Development (RAD) because of its peculiar advantages. The developed system is also client/server based. The web based automated workflow system for employees' welfare and loan scheme using light weight methodology provided new software architecture with single solution platform for any welfare and loan scheme administrators within the University system. Test results revel that the automated workflow system carefully eradicated the delay in the end-to-end processing of loan and commodity application forms submitted. The automated workflow system also eradicated the errors made due to manual computations. The automated workflow system was able to store, retrieve, and secure records more effectively and efficiently. The employees who are members of the welfare scheme were able to check their savings with the scheme in real time. Employees members of the welfare scheme were also able to apply for loan or other facilities from the comfort of their homes and track the status of their application online any time and from anywhere in the world without physically going the welfare scheme office. The proposed system was able to eradicate most of the challenges faced by employees and their employees' welfare and loan scheme.

5 References

[1]. Coventry, W. F. and Barker, J. K. (1988). Management. International Edition: Heinemann Professional Publishing.

[2]. Cowling, A. and Mailer, C. (1992). Managing Human Resources. 2nd Edition. London: Edward Arnold.

[3]. Armstrong, M. (2010). Handbook of Human Resource Management Practice, Keagan Page, London.

[4]. Mortocchio, J. J. (2001). Strategic Compensation, Prentice Hall, USA.

[5]. Chung , H. (2006). Human Resource Management Theory and Practice, 3rd Edition, Palgrave MacMillan, London UK.

[6]. Nzuve S. N. (2010). Management of Human Resources, Basic Modern Management Consultants, Nairobi, Kenya.

[7]. (Armstrong 2010). Handbook of Human Resource Management Practice, Keagan Page, London.

[8]. Medina-Mora, R.; Winograd, T.; and Flores, R. (1993). "ActionWorkflow as the Enterprise Integration Technology," Bulletin of the Technical Committee on Data Engineering, IEEE Computer Society, Vol. 16, No.2.

[9]. Medina-Mora, R.; Winograd, T.; and Flores, R. (1993). "ActionWorkflow as the Enterprise Integration Technology," Bulletin of the Technical Committee on Data Engineering, IEEE Computer Society, Vol. 16, No.2.

[10]. Khan, A. I., Qurashi, R. J. and Khan, U. A. (2011). A Comprehensive Study of Commonly Practiced Heavy and Light Weight Software Mothodologies. Internation Journal of Computer Science Issues, Vol. 8, Issu 4, No. 2

[11]. Sommerville, I. (2004). Software Engineering, UK: Addison Wesley.

[12]. Paul, C; Bachmann, F; Bass, L; Garlan, D; Ivers, J; Little, R; Merson, P; Nord, R; Stafford, J (2010). *Documenting Software Architectures: Views and Beyond, Second Edition*. Boston: Addison Wesley. ISBN 0-321-55268-7.

Towards a Formal Framework for Hybrid Analysis of Composite Web Services

May Haydar
Computer Science Department /DIRO
Fahad Bin Sultan University /Université de Montréal
Tabuk, Saudi Arabia /Montreal, QC
mhaidar@fbsu.edu.sa

Hesham Hallal
Electrical Engineering Department
Fahad Bin Sultan University
Tabuk, KSA
hhallal@fbsu.edu.sa

Abstract— **In this work, we propose to develop an integrated formal framework where both static and dynamic analysis techniques complement each other in enhancing the verification process of an existing web services based application. The proposed framework consists of the following main components. The first component is a Library of Property Patterns which we intend to build on existing work [2, 14] and compile a library and a classification of web services properties (patterns and antipatterns [13]). These would include BPEL4WS and WISCI requirements in the form of property patterns which can be instantiated in different contexts and for different purposes like verifying correctness, security, and performance related issues. The property library will be based on an easy to use template that depicts mainly the type, formal model, and example of a property. The second component is the development of Static Analysis Techniques that include direct code inspection, abstraction based techniques, and model based techniques. The third component is the development of dynamic analysis techniques that include extracting behavioral models of applications from observed executions and verifying them (mainly using model checking) against behavioral properties. These properties specify defects that cannot be detected using static analysis techniques. A set of tools (prototype) are to be implemented to realize the proposed approaches for static analysis, modeling, and dynamic verification of the applications under test.**

Static Analysis, Dynamic Analysis, Property Patterns, Web Services

I. INTRODUCTION

Businesses are increasingly adopting service orientation to shape the architecture of their enterprise solutions and to increase the efficiency of their software applications. At the foundation of this ever more popular paradigm, web services are heavily used to enhance decentralization and cross platform and language portability. The power of services resides mainly in the high degree of dynamism and flexibility they exhibit throughout their lifecycle: publication, discovery, and binding are all dynamic activities that make a service an evolving entity capable of adapting to continuously changing and new requirements. In addition, compositions of services, which can also be dynamic, have added to the power of services in building larger enterprise solutions for heterogeneous businesses. However, the fast paced growth of service implementation and deployment in various contexts has resulted in a growing gap between the development and verification of service based applications. On one hand, static analysis techniques [1, 13] remain insufficient to detect behavioral flaws and defects that are exhibited only when services, especially composite ones, are executed. In particular, such techniques face two major problems: difficulty of generating executable models that can be used in the analysis, and limited coverage of defects that are exhibited only during runtime, e.g., concurrency incurred problems. On the other hand, dynamic and runtime techniques, which depend mainly on monitoring, can only claim to detect errors and flaws in the observable behavior of a service, or a dynamic composition of services. In the meantime, formal methods have become a viable option to automate the verification process and increase its efficiency in modeling, testing, and error detection. Formal verification techniques are currently used in several domains including communications systems, software and program analysis [13], and web based applications [2, 14].

II. Formal Framework

We develop an integrated formal framework as illustrated in Figure 1, where both static and dynamic analysis techniques complement each other in enhancing the verification process of an existing web services based application. The proposed framework consists of the following main components.

A. *Library of Property Patterns*

Patterns have long been used in the development of software applications, and service oriented architectures as well, since they introduce clever and insightful ways to solve common problems. Along with patterns, which are intended to facilitate the design and development processes, the term antipattern is defined. An antipattern is simply a solution to a problem that does not work correctly.

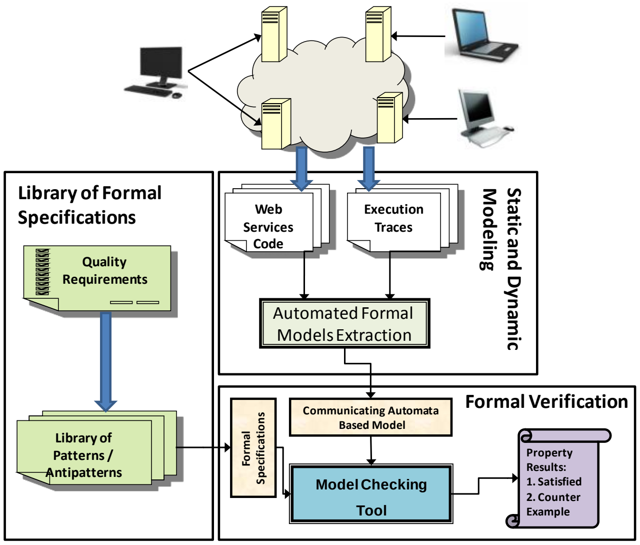

Library of Formal Specifications
Quality Requirements
Library of Patterns / Antipatterns
Web Services Code
Execution Traces
Static and Dynamic Modeling
Automated Formal Models Extraction
Formal Verification
Communicating Automata Based Model
Formal Specifications
Model Checking Tool
Property Results:
1. Satisfied
2. Counter Example

Figure 1. Formal Framework for Service Composition Analysis

Following the definition, efforts exist to document antipatterns in catalogs (similar to design patterns) so that they can be avoided. In the proposed framework, we intend to build on existing work [7,8,13,16,17] and compile a library and a classification of web services properties (patterns and antipatterns). The classification of properties will be hierarchical: static/dynamic, correctness/functional, style/performance, etc. Such classifications should help developers identify the antipatterns to better avoid them, and testers detect them in the application using the appropriate techniques. On the other hand, documented properties, which would include BPEL4WS and WISCI requirements in the form of property patterns, can be instantiated in different contexts and for different purposes like verifying correctness, security, and performance related issues. The property library will be based on an easy to use template that depicts mainly the type, formal model, and example of a property.

For example, in our previous work [16,17], we have defined a pattern template and identified 119 patterns and property specification for the verification Web applications (WAs). Figure 2 is an example of such patterns. Each pattern is specified in LTL which makes it easy to use in model checking.

B. *Static Analysis Techniques*

In general, static techniques for software, mainly based on analysis of (compiled) code (existing specifications or textual descriptions), are independent of specific input data sets or individual execution paths. They are usually classified into:

1. Direct code inspection, where suspicious code segments are directly identified in the code (through linear scanning for example).
2. Abstraction based techniques, where code representations (class diagrams, call graphs, etc.) are used to match the exhibition of certain predefined patterns (or antipatterns).
3. Model based techniques, where an executable model (often formal) is extracted from the code and verified against predefined properties using techniques like model checking or theorem proving.

ID	FGS6
Pattern description	Banking information is entered no more than once before submitting form
Category	Functional – General – Security and Authentication
Page Attributes	*Banking_info*: Boolean identifying the presence of fields for banking information *Submit*: identification of page where form submit action exists
LTL Mapping	PrecedenceGlobally (($\neg(\mathit{banking\_info})\,\mathrm{W}\,(\mathit{banking\_info}\,\mathrm{W}\,(\mathrm{G}\neg(\mathit{banking\_info})))), \mathit{submit})$
Comments	
Source	Newly introduced

Figure 2. Web Specification Pattern

In the case of web services based applications; such techniques would be applied to the available documents containing the descriptions of individual and composite services. However, some complex faults cannot be detected with these approaches or only at a high cost (like deadlocks). Another disadvantage is false warnings (mainly false positives) that can be produced as results. This justifies the need for the third component, a set of dynamic analysis techniques.

C. *Dynamic Analysis Techniques*

These techniques have emerged as complementary to static analysis techniques, especially when concurrency and large architectural structures of applications make the latter inefficient and rather incomplete. Dynamic analysis techniques do not necessarily rely on existing specifications or textual descriptions of the applications under test. Instead, they are applied to executable behavioral models that are derived from the application's observed executions (traces or logfiles). This solution is particularly efficient in the case of web services based applications, often characterized by their readiness to compose web services, especially dynamically. Although many standards have been introduced to address the problem of web service composition, including BPEL4WS (Business Process Execution Language for Web Services) and WSCI (Web Service Choreography Interface), they address mainly the description and execution of workflow specifications for web service compositions. Yet, they are not sufficient to support automated verification techniques based on static analysis. The proposed techniques include extracting behavioral models of applications from observed executions and verifying them (mainly using model checking) against behavioral properties specifying defects that cannot be detected using static analysis techniques. The known techniques in the field include:

1. offline (postmortem) techniques, where recorded executions of an application are stored and later used in modeling and verifying the application under test.
2. Online (runtime) techniques, where an application under test is analyzed as the executions are generated.

In our previous work [14,16], we designed a framework for formal modeling and verification of web applications. We intercepted http requests/responses that depict the behavior of web applications and extracted communicating automata models translated into Promela. We verified properties of WAs using Spin model checker. Results were promising and properties of concurrent behavior were verified, which could not be verified using other methods. Concurrent behavior of WAs represented behavior of WAs with multiple displays (windows/frames).

We use a similar approach for model extractions from behavioral executions of composite Web services. Collected traces are analyzed and abstracted as communicating automata models. Each automata depicts the behavior of one service where requests are modeled as events and responses as states. Events will be distinguished as local and common. Common events (rendezvous) represent service communications with each other.

III. IMPLEMENTATION

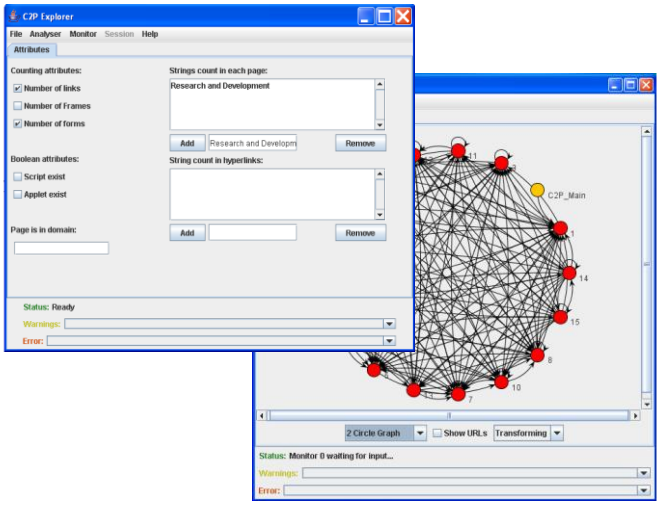
The implementation of the proposed framework includes the following main tasks:

1. Surveying the literature and common practices of various developers of web services based applications to compile a set of most frequently encountered properties (patterns and antipatterns).
2. Formulation of properties in specification languages that can be used in both static and dynamic analysis techniques.
3. Identifying proper static analysis techniques for each class of properties and evaluating their efficiency and robustness. In particular, this task includes identifying the proper abstractions, along with methods to extract them, to be used in detecting corresponding antipatterns in the code.
4. Record execution traces from the applications under test. This task includes studying the instrumentation based and interception based techniques.
5. Extracting models from monitored executions. This includes extracting models from completed traces and incremental models in the case of runtime analysis that can be used in known model checking tools.
6. Integrating the compiled library and developed tools in a user friendly toolset which masks the details of the underlying analysis techniques form the users and makes the dissemination of the produced framework easier.

The proposed framework is implemented using Spin model checker [18]. The automata models are represented using Promela language and the patterns/antipatterns are represented in LTL. We use the Java Eclipse environment for the toolset implementation. The complete toolset will include integrated components as follows:

a. A library of compiled patterns/antipatterns translated in LTL.
b. A monitoring tool that intercepts Web services communications and logs the intercepted behavior.
c. An analysis module that analyzes and abstracts the intercepted communication (online or offline) and extracts a communicating automata model represented in XML.
d. A model translator that translates the XML extracted model into Promela, the modeling language of Spin.
e. A verification module that uses the Spin Model checker to verify patterns (from the library) on the extracted Web services automata model.
f. A graphical user interface that coordinates the user command on the toolset.

Figure 3 illustrates our initial toolset prototype for the dynamic modeling of Web services.



Counting attributes:
☑ Number of links
☐ Number of Frames
☑ Number of forms

Strings count in each page: Research and Development

Research and Developm

Boolean attributes:
☐ Script exist
☐ Applet exist

String count in hyperlinks:

Page is in domain:

Status: Ready

Warnings:

Error:

2 Circle Graph ☐ Show URLs Transforming

Status: Monitor 0 waiting for input...

Warnings:

Error:

Figure 3. Prototype Tool for Web Services Monitoring and Modeling

IV. RELATED WORK

Run time verification of software applications has grown as a major field covering major activities related to the development of software. At the same time, webbed, and web service-based, applications have gained a lot of attention in many research activities both in academia and in the industry given the role such applications have in the shaping of today’s economy based on e-commerce and e-services.

Our related work that is closely connected to this new proposed work is published in [8,14,16,18]. We have implemented an integrated formal framework for run-time verification of web applications. Results were interesting and we were able to verify properties that could not be verified using other approaches.

Recently, a large body of research has been produced with a focus on formal modeling of web services based applications in order to induce automation in the analysis of the developed applications against some predefined properties specified from the description and requirements texts. Derived models are often generated from textual descriptions of applications (BPEL, BPEL4WS, and WSCI), and can be used mainly to check static properties that relate to the structure and content of the application, usually described as a composition of services. Examples of such research include the work of Foster et al. [1,2], which models BPEL descriptions as Finite State Process models, which can be verified against properties that are mainly derived from design specifications written in UML notations like the Message Sequence Chart (MSC) or activity diagrams. Properties sought for verification include mostly semantic failures and difficulties in providing necessary compensation handling sequences that are tough to detect directly in common workflow languages like BPEL. Other attempts have been described in the literature as well including the work of Breugel and Koshkina [3, 4] who introduce the BPE-calculus to capture control flow in BPEL descriptions and programs. The service descriptions in the proposed language allow for checking against properties like dead path elimination and control cycles. The verification, mainly formal model checking, is performed in the toolset Concurrency Workbench (CWB). However, as discussed in Section 1, proposed verification approaches based mainly on the static analysis of an existing source code, where different types of models like EFA, Promela, and communicating FSMs [11, 12] are used, have their limitations and impracticalities. Consequently, more efforts are being spent on performing run-time verification of web service applications based on monitoring and model extraction. Also, [5] address the run-time monitoring of functional characteristics of composed Web services, as well as for individual services [6].

V. CONCLUSIONS

In this paper, we proposed an integrated formal framework for the analysis and verification of Web services composition. We propose a hybrid of both static and dynamic analysis techniques which complement each other. We also intend to develop a library of patterns and antipatterns of interesting specifications of web services. These specifications will be automatically translatable to a formal specification language namely LTL.

Based on our previous experience and the initial results obtained in the use of our formal approach for run-time verification, we believe that results of this proposed work are very promising.

REFERENCES

[1] Foster, H. (2008). Tool Support for Safety Analysis of Service Composition and Deployment Models. Proceedings of *the 2008 IEEE International Conference on Web Services*, pp. 716-723. IEEE Computer Society.

[2] Foster, H., Uchitel, S., Magee, J., & Kramer, J. (2003). Model-based Verification of Web Service Compositions. Proc. of 18th IEEE International Conference on Automated Software Engineering, pp. 152-161. Montreal, Canada.

[3] Koshkina, M., & van Breugel, F. (2004). Modeling and verifying web service orchestration by means of the concurrency workbench. SIGSOFT Software Engineering Notes, 29(5):1-10. ACM.

[4] Van Breugel, F., & Koshkina, M. (2005). Dead-Path-Elimination in BPEL4WS. Proceedings of the 5th International Conference on Application of Concurrency to System Design, pp. 192-201. IEEE Computer Society.

[5] Kallel, S., Char_, A., Dinkelaker, T., Mezini, M., Jmaiel, M.: Specifying and Monitoring Temporal Properties in Web services Compositions. Proceedings of the 7th IEEE European Conference on Web Services (ECOWS). (2009).

[6] Simmonds, J., Gan, Y., Chechik, M., Nejati, S., O'Farrell, B., Litani, E., Waterhouse, J.: Runtime Monitoring of Web Service Conversations. IEEE Transactions on Services Computing. 99, 223-244 (2009).

[7] May Haydar, Sergiy Boroday, Alexandre Petrenko, and Houari Sahraoui . "Properties and Scopes in Web Model Checking". In Proc. of 20th IEEE/ACM International Conference on Automated Software Engineering (ASE 05). Long Beach, California, USA, November 2005.

[8] May Haydar, Sergiy Boroday, Alexandre Petrenko, and Houari Sahraoui. "Propositional Scopes in Lenear Temporal Logic". In Proc. of 5th International Conference on New Technologies of Distributed Systems (NOTERE 05). Gatineau, Quebec, Canada, August 2005.

[9] Dwyer M, Avrunin GS, Corbett JC. Patterns in Property Specifications for Finite-state Verification. 21st Int. Conference on Software Engineering, May, 1999.

[10] X. Fu et al, Analysis of interacting BPEL web services. 13th Int. World Wide Web Conference, 2004.

[11] Nakajima, S. (2006, May). Model-Checking Behavioral Specification of BPEL Applications. Proceedings of the

International Workshop on Web Languages and Formal Methods, 2(151):89-105.ENTCS.

[12] Fu, X., Bultan, T., & Su, J. (2004). Analysis of interacting BPEL Web Services. Proceedings of the 13th International World Wide Web Conference, pp. 621-630. ACM Press.

[13] H. H. Hallal, E. Alikacem, W. P. Tunney, S. Boroday, A. Petrenko,(2004) "Antipattern-Based Detection of Deficiencies in Java Multithreaded Software," qsic, pp.258-267, Quality Software, Fourth International Conference on (QSIC'04).

[14] Haydar, M., Petrenko, A. and Sahraoui, H. (2004) "Formal Verification of Web Applications Modeled by Communicating Automata" In Proceedings of 24th IFIP WG 6.1 IFIP International Conference on Formal Techniques for Networked and Distributed Systems (FORTE 2004), pp. 115-132. Madrid, Spain. [LNCS, vol. 3235]

[15] Boroday, S., Petrenko, A., Sing, J. and Hallal, H. (2005) "Dynamic Analysis of Java Applications for MultiThreaded Antipatterns" In Proceedings of the Third International Workshops on Dynamics Analysis (WODA 2005). St-Louis, MI, USA.

[16] May Haydar. *A Formal Framework for Run-Time Verification of Web Applications: An Approach Supported by Scope Extended Linear Temporal Logic*. VDM Verlag, Germany, September 2009. ISBN: 978-3-639-18943-8.

[17] May Haydar, Houari Sahraoui, and Alexandre Petrenko. "Specification Patterns for Formal Web Verification". In Proc. of 8th International Conference on Web Engineering (ICWE 08). Yorktown Heights, New York, USA, July 2008.

[18] Gerarld Holzmann. *The SPIN Model Checker: Primer and Reference Manual*. ISBN-10: 0321228626. Addison-Wesley, Sptember 2003.

Application of a Fuzzy Inference System to Measure Maintainability of Object-Oriented Software

Nasib Singh Gill and Meenakshi Sharma
Department of Computer Science & Applications
Maharshi Dayanand University, Rohtak-124001
Haryana, India

Abstract - *In the software development life cycle, maintenance is the most costly activity because it requires more effort compared to other activities. To reduce the cost of software maintenance, it is essential to predict software maintainability during the early phases of software development. As a consequence of early estimation, further corrective and preventive actions can be performed more efficiently to improve the maintainability of the software. Predicting maintainability of software using fuzzy logic is gaining more attention among researchers due to its ability to deal with uncertain, imprecise and incomplete data. This paper develops a fuzzy logic-based model for predicting the maintainability of a class. The given model is based on the Mamdani's fuzzy inference system. Chidamber and Kemerer metrics are used as inputs to the model and the maintainability is computed as output. Maintainability can be used as an indicator of the quality of software at design time.*

Keywords: Maintainability, Metric, Fuzzy Inference Engine

1. Introduction

"Software maintenance in software engineering is the modification of a software product after delivery to correct and to improve performance or other attributes" [1]. Software maintenance is the most important activity in the life of a software product. The total cost of the maintenance phase is much higher than the development cost of the software. It can consume 40% to 90% of the cost of the entire life cycle [2,3]. There are four types of software maintenance: Corrective, Adaptive, Preventive and Perfective. Software maintainability is the ease with which a software system can be understood, modified and adapted [4]. It is an important software quality attribute. Software maintenance is a post-development activity, but it is highly influenced by the way the software was developed. The object-oriented software development approach has the ability to produce highly maintainable software. The maintainability of a software system can significantly impact software costs. For effective management of software cost, software maintainability must be evaluated. Many metrics have been proposed by various researchers to evaluate the maintainability of software. Among these metrics, the Chidamber and Kemerer (CK) metrics suite [5] is the most widely used metric suite for assessing the maintainability of object-oriented software. Software maintainability is a difficult factor to quantify [6]. In object-oriented software, the maintainability of a class can be assessed in terms of its complexity using design-oriented metrics. However predicting the maintainability of a class using crisp logic is not appropriate in the presence of fuzzy input factors. Therefore, the fuzzy logic-based approach of measuring maintainability is being used by many researchers.

Fuzzy logic is a generalization of classical boolean logic. It provides a mechanism for representing linguistic constructs such as "low," "medium," "high," "very high" etc. The conventional binary set theory describes crisp data, which is either fully true or false. Fuzzy logic operates on the concept of degree of membership. The degree of membership is continuous on an interval [0,1]. A fuzzy set captures vagueness using membership functions by assigning a degree of membership to each element of the set. The fuzzy logic also uses a fuzzy inference engine to manipulate imprecise, uncertain and conflicting data. The fuzzy inference engine maps the input fuzzy set to an output fuzzy set using a fuzzy rule base.

This paper evaluates the maintainability of object-oriented software using Mamdani's Fuzzy Inference System (MFIS). Six CK metrics are used as inputs to the model. The value of each metric is fuzzified into one of three values: Low, Medium and High. The trapezoidal membership function is used for fuzzification. A total of 729 fuzzy rules were developed, which are used by the fuzzy inference engine. The maintainability computed by MFIS is correlated with the composite complexity, which is computed as a weighted sum of the CK metrics. The weight that will be multiplied to each CK metric is derived from the corresponding importance factor given by Jubair et al. [7].

The rest of the paper is divided into three sections. Section 2 describes the MFIS along with inputs, outputs, membership functions and rules. Section 3 applies the given model to 15 classes and correlates the composite complexity of each class with its maintainability. Finally, the conclusions of paper and the future directions are given in section 4.

2. 2. Model Adopted

3.

This paper uses the MFIS proposed in 1975 by Ebrahim Mamdani [8]. The main components of the MFIS are a fuzzy rule base, dictionary and reasoning mechanism. A fuzzy rule base is a collection of fuzzy rules obtained from experts. Fuzzy rules are linguistic statements, which describe how the fuzzy inference system should make a decision regarding classifying an input or controlling an output [9]. The general format of a fuzzy rule is if (input-1 is membership function-1) and/or (input-2 is membership function-2) and/or …… (input-n is membership function-n) then (output is output membership-function). The dictionary defines all the membership functions used in the fuzzy rules. The fuzzy reasoning mechanism performs the inference procedure. The input to the FIS can be either fuzzy or crisp. If the input is a crisp value, it can be converted to a fuzzy value using a fuzzification process. Generally the output produced by MFIS is fuzzy; however if crisp output is required it can be converted into crisp form using a defuzzification process. This model uses the following steps to compute the output [10].

1. Design a set of fuzzy rules.
2. Convert the input into fuzzy form.
3. Combine the fuzzified inputs according to the fuzzy rules for establishing rule strength.
4. Determine the consequent of the rules for establishing rule strength and the output membership function.
5. Combine all consequents to get an output distribution.
6. Defuzzify the output distribution.

2.1 Inputs and Output of Model

Software maintainability is an important attribute of software quality. It can not be measured directly. It can be measured indirectly using measures of design structures such as class diagram. For object-oriented software, Chidamber and Kemerer proposed a metrics suite consisting of six metrics that can be used to measure the complexity, reusability, coupling and cohesion of a class [11]. These metrics are further correlated with the maintainability of a class. This paper uses the CK metrics suite as input for the adopted model. The CK metrics suite includes the following six metrics.

2.1.1 Weighted Methods Per Class (WMC)

WMC is the sum of the complexities of all the methods of a class. It is one the predictors of class maintainability. A higher value of WMC indicates a lower maintainability of the class.

2.1.2 Depth of Inheritance Tree (DIT)

The DIT of a class is its maximum depth from the root node in the inheritance hierarchy. An empirical study by Daly et al. [12] indicates that a class with DIT 3 is easier to maintain compared to a class with no inheritance.

2.1.3 Number of Children (NOC)

The NOC of a class is the total number of immediate sub-classes of that class. A class with higher NOC requires more testing efforts.

2.1.4 Coupling Between Object Classes (CBO)

The CBO of a class is the total number of other classes with which it is coupled. A higher value of CBO indicates lower maintainability of a class due to higher sensitivity to changes in the other classes.

2.1.5 Response For a Class (RFC)

The RFC of a class is the total number of methods that can be executed in response to a message received by an object of that class. It is the sum of the methods defined in the class and the methods that are directly invoked by methods of the class. A higher value of RFC indicates a higher complexity of the class, therefore less maintainability.

2.1.6 Lack of Cohesion in Methods (LCOM)

The LCOM of a class is the difference between the number of pairs of methods that have no common attribute and the number of pairs of methods that have common attributes. It measures the dissimilarity of methods in a class on the basis of attributes used by methods of the class. A positive value of LCOM for the class indicates that the class is less cohesive. For good quality design a positive value of LCOM is not desirable. Low cohesion is an indicator of higher complexity.

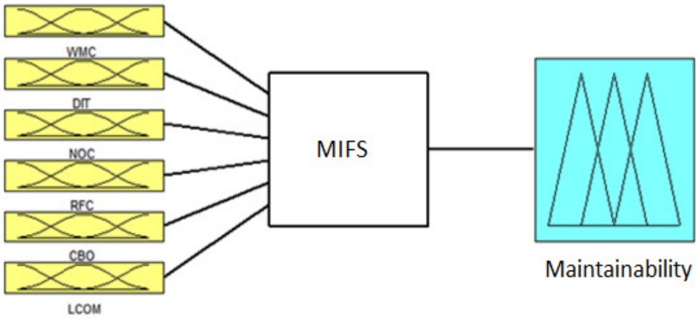


Figure 1: Mamdani's Fuzzy Inference System

The output of the model is the maintainability of the class, which measures how easily we can understand and modify the class. The lower values of input complexity parameters indicate better maintainability of the class.

2.2 Membership Functions for Inputs and Output

All inputs and outputs are represented using three fuzzy values - Low, Medium and High. The trapezoidal membership function is used for representing fuzzy values. The trapezoidal function depends upon four scalar parameters - a, b, c and d.

$$f(x;a,b,c,d)=\left\{\begin{matrix} 0, & x\le a \\ \dfrac{x-a}{b-a}, & a\le x\le b \\ 1, & b\le x\le c \\ \dfrac{d-x}{d-c}, & c\le x\le d \\ 0, & d\le x \end{matrix}\right\}$$

Different threshold values of CK metrics are given by various researchers [13,14,15,16]. This paper designs three membership functions (LOW, MEDIUM and HIGH) for each input and output. Details of all inputs (Input1 to Input6) and output (Output1) along with their graphics representations are as follows. MF and trapmf stand for membership function and trapezoidal membership function respectively. Each membership function is defined in terms of four parameters.

```
[Input1]
Name='WMC'
Range=[0 30]
NumMFs=3
MF1='LOW':'trapmf',[0 0 5 10]
MF2='MEDIUM':'trapmf',[5 10 15 20]
MF3='HIGH':'trapmf',[15 20 30 30]
```

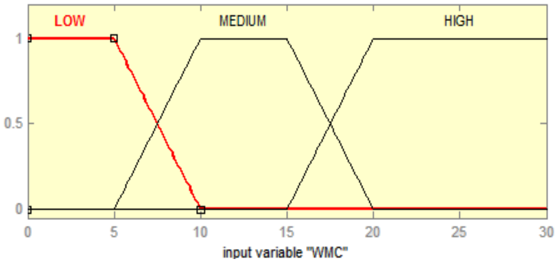


Figure 2: Membership functions of WMC

```
[Input2]
Name='DIT'
Range=[0 10]
NumMFs=3
MF1='LOW':'trapmf',[0 0 1 3]
MF2='MEDIUM':'trapmf',[1 3 4 6]
MF3='HIGH':'trapmf',[4 6 10 10]
```

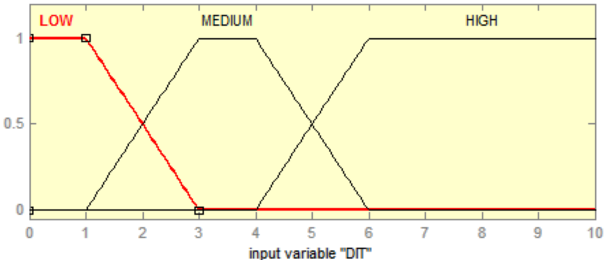


Figure 3: Membership functions of DIT

```
[Input3]
Name='NOC'
Range=[0 12]
NumMFs=3
MF1='LOW':'trapmf',[0 0 2 4]
MF2='MEDIUM':'trapmf',[2 4 6 8]
MF3='HIGH':'trapmf',[6 8 12 12]
```

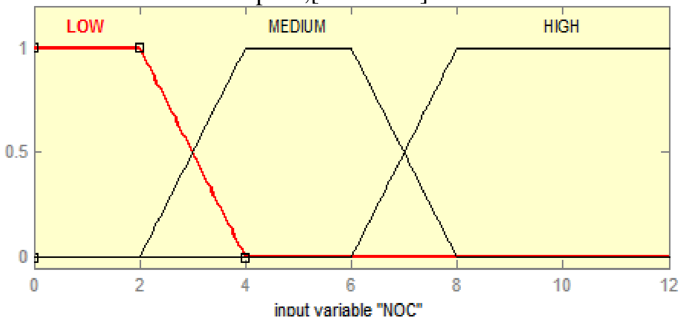


Figure 4: Membership functions of NOC

```
[Input4]
Name='RFC'
Range=[0 60]
NumMFs=3
MF1='LOW':'trapmf',[0 0 10 20]
MF2='MEDIUM':'trapmf',[10 20 30 40]
MF3='HIGH':'trapmf',[30 40 60 60]
```

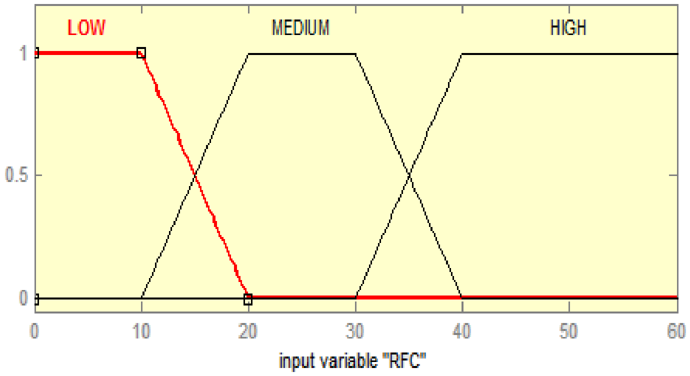


Figure 5: Membership functions of RFC

```
[Input5]
Name='CBO'
Range=[0 10]
NumMFs=3
MF1='LOW':'trapmf',[0 0 2 4]
MF2='MEDIUM':'trapmf',[2 4 5 7]
MF3='HIGH':'trapmf',[5 7 10 10]
```

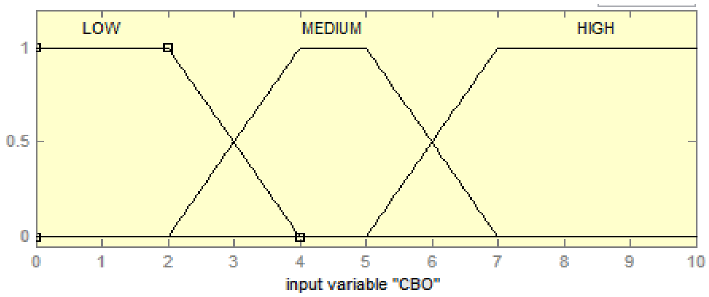


Figure 6: Membership functions of CBO

```
[Input6]
    Name='LCOM'
    Range=[0 10]
    NumMFs=3
    MF1='LOW':'trapmf',[0 0 1 3]
    MF2='MEDIUM':'trapmf',[1 3 4 6]
    MF3='HIGH':'trapmf',[4 6 10 10]
```

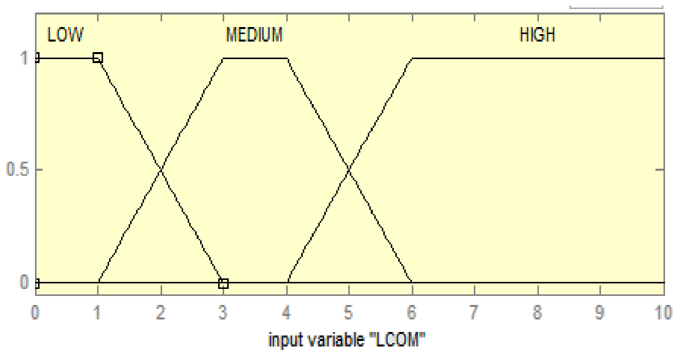


Figure 7: Membership functions of LCOM

```
[Output1]
Name='Maintainability'
Range=[0 1]
NumMFs=3
MF1='LOW':'trapmf',[0 0 0.2 0.4]
```

MF2='MEDIUM':'trapmf',[0.2 0.4 0.6 0.8]
MF3='HIGH':'trapmf',[0.6 0.8 1 1]

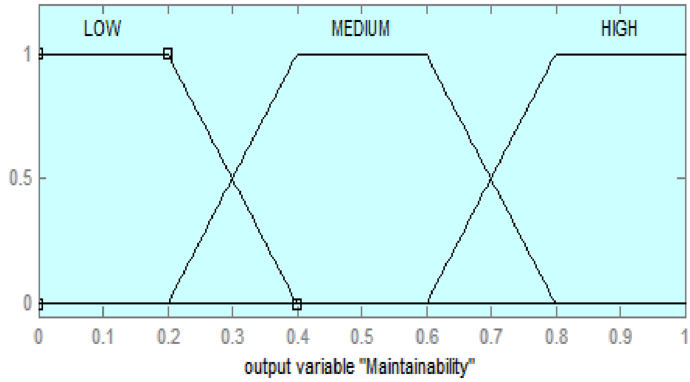
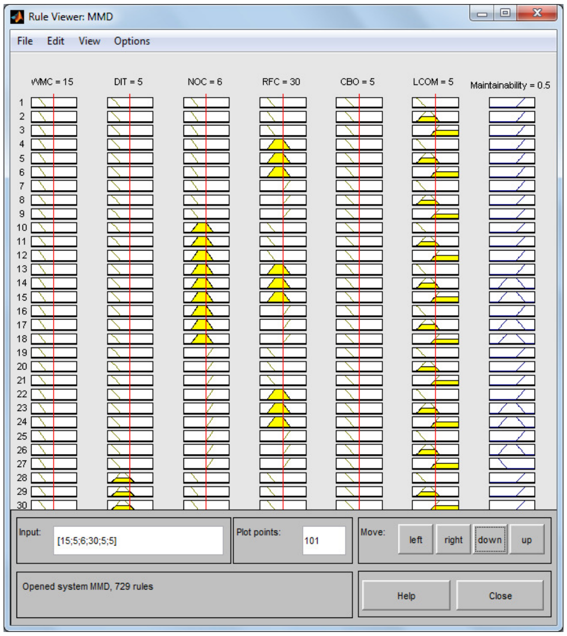


Figure 8: Membership functions of Maintainability

2.3 Rule Base

In general with n inputs and m membership functions, m^n rules can be generated. In this model, six inputs are used and for each input three membership functions are used; as a result 729 rules are designed. The format of a rule is as follows.

If (WMC is LOW/MEDIUM/HIGH) and (DIT is LOW/MEDIUM/HIGH) and (NOC is LOW/MEDIUM/HIGH) and (RFC is LOW/MEDIUM/HIGH) and (CBO is LOW/MEDIUM/HIGH) and (LCOM is LOW/MEDIUM/HIGH) then (Maintainability is LOW/MEDIUM/HIGH). For example If (WMC is LOW) and (DIT is LOW) and (NOC is LOW) and (RFC is LOW) and (CBO is LOW) and (LCOM is LOW) then (Maintainability is HIGH). The following snapshot displays a graphical representation of the rules.



Input: [15;5;6;30;5;5]

Plot points: 101

Opened system MMD, 729 rules

Figure 9: Graphical Representation of Rules

3. Measuring Class Complexity and Maintainability

The given model was applied to 15 classes and the results are shown in Table 1.

Table 1: Maintainability of classes

Class	WMC	DIT	NOC	RFC	CBO	LCOM	Composite Complexity	Maintainability (Mamdani Model)
C1	17	0	0	9	0	5	5.37	0.83
C2	21	3	0	32	5	5	12.86	0.44
C3	28	7	0	36	0	0	13.53	0.34
C4	6	3	1	12	3	4	5.48	0.63
C5	6	0	0	7	1	0	2.74	0.84
C6	25	7	0	46	0	5	16.13	0.17
C7	8	3	1	12	1	0	4.72	0.72
C8	9	5	0	8	1	2	4.47	0.63
C9	12	0	1	18	1	0	6.25	0.84
C10	7	0	0	5	1	0	2.43	0.83
C11	12	0	0	6	0	0	3.18	0.85
C12	26	4	0	30	0	0	11.4	0.5
C13	3	0	0	9	3	0	3.21	0.83
C14	27	4	2	50	6	1	17.84	0.38
C15	15	2	0	25	6	5	10.43	0.5

Jubair et al. [7] suggested an importance factor for each CK metric, which is represented by LOW(1), MEDIUM(2) and HIGH(3). The importance factor reflects the importance of the metric in determining the software quality. WMC, DIT, NOC, CBO, RFC and LCOM are assigned importance factors 2, 2, 1, 3, 3 and 2 respectively [7]. On the basis of the normalized values (0.08 for Low, 0.15 for Medium and 0.23 for High) of the importance factors, a weight is assigned to each CK metric. WMC, DIT, NOC, CBO, RFC and LCOM are assigned the weights 0.15, 0.15, 0.08, 0.23, 0.23 and 0.15 respectively. Using these weights, the composite complexity is computed. The coefficient of correlation between the composite complexity and the maintainability of the class is -0.92, which indicates that a higher value of class complexity lowers the maintainability of class.

4. Conclusion and Future Directions

This paper presents the application of MFIS for computing the maintainability of a class in object-oriented software. The results produced by the MFIS satisfy the relationship between complexity and maintainability i.e. higher complexity leads to lower maintainability. However, the given model was applied to small academic-level classes and the same study can be replicated with larger industrial projects.

5. References

[1] ISO/IEC 14764:2006 Software Engineering — Software Life Cycle Processes — Maintenance. http://www.iso.org/iso/catalogue_detail.htm?csnumber=39064

[2] Yunsik Ahn, Jungseok Suh, Seungryeol Kim and Hyunsoo Kim 2003. The Software Maintenance Project Effort Estimation Model Based on Function Points. Journal of Software Maintenance and Evolution: Research and Practice, 15:71–85.

[3] How to Save on Software Maintenance Costs. Omnext white paper, March 2010. http://www.omnext.net/downloads/Whitepaper_Omnext.pdf

[4] Pressman, R, Software Engineering. A Practitioner‘s Approach, Fourth Edition, McGraw Hill, ISBN: 0-07-709411-5, 1997.

[5] S.R. Chidamber, C.F. Kemerer, A Metrics Suite for Object Oriented Design. IEEE Transactions on Software Engineering, Vol. 20, No.6, pp. 476-492, 1994.

[6] Dimitris Stavrinoudis, Michalis Xenos and Dimitris Christodoulakis 1999 “Relation Between Software Metrics And Maintainability. Proceedings of the FESMA99 International Conference, Federation of European Software Measurement Associations, Amsterdam, The Netherlands, pp. 465-476.

[7] Jubair Al-Ja’afer and Khair Eddin Sabri, 2004. Chidamber-Kemerer (CK) and Lorenze-Kidd (LK) Metrics to Assess Java Programs. International Workshop on Software System (IWSS’04) Turkey.

[8] Mamdani, E.H. and S. Assilian 1975. An experiment in Linguistic Synthesis with a Fuzzy Logic Controller. International Journal of Man-Machine Studies, Vol. 7, No. 1, pp. 1-13.

[9] http://www.cs.princeton.edu/courses/archive/fall07/cos436/HIDDEN/Knapp/fuzzy004.htm

[10] Jyh-Shing Roger Jang, Chuen-Tsai Sun and Eiji Mizutani, Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence, PHI Learning Private Limited.

[11] Sandeep Srivastava, 2012. Software Metrics and Maintainability Relationship with CK Matrix. International Journal of Innovations in Engineering and Technology (IJIET) Vol. 1 Issue 2.

[12] Daly J., Brooks A., Miller J., Roper M. and Wood M. 1996. An Empirical Study Evaluating Depth of Inheritance on Maintainability of Object-Oriented Software. Empirical Software Engineering”, Vol. 1, No. 2, pp. 109-132.

[13] Zhou Yuming and Hareton Leung, 2006. Empirical Analysis of Object-Oriented Design Metrics for Predicting High and Low Severity Faults. IEEE Trans. Softw. Eng.,32(10):771-789.

[14] Edith Linda P and E. Chandra, 2010. Class Break Point Determination Using CK Metrics Thresholds. Global Journal of Computer Science and Technology, 73 Vol. 10 Issue 14 (Ver. 1.0).

[15] Mago Jagmohan and Kaur Parwinder, 2012. Analysis of Quality of the Design of the Object Oriented Software using Fuzzy Logic, iRAFIT, p 21- 25.

[16] Camrazo Cruz Ana Erika,2008. Chidamber & Kemerer suite of Metrics. Thesis submitted to Japan Advanced Institute of Science and Technology.

SESSION

SOFTWARE ENGINEERING AND MANAGEMENT + DIVERSITY ISSUES

Chair(s)

TBA

Females in Software Engineering Teams: A Social Sensitivity Perspective

Sourabh Khosla[1], Lisa L. Bender[2], Gursimran S. Walia[3], Kendall Nygard[4]
North Dakota State University[134], University of Houston – Clear Lake[2]
Department of Computer Science
Fargo, ND 58108[134]; Houston, Texas, USA[2]
{sourabh.khosla[1], gursimran.walia[3], Kendall.Nygard[4]} @ ndsu.edu; bender@uhcl.edu[2]

***Abstract*— Software Development is a complex and time-demanding task which requires a group of individuals working effectively as a team for long durations of times. Achieving effectiveness in productivity and quality of work is a challenging task that needs investment and commitment. A recent study conducted at MIT established that teams with greater *social sensitivity* (SS) tend to perform better when completing a variety of specific collaborative tasks. SS is an empathic ability to correctly understand feelings and perspective of others, and is clearly measureable. However their study was based only on set of generic tasks carried for a short period of time which required only hours to finish. Software development projects require teams to work collaboratively for much longer durations and more complicated tasks. Our goal is to determine if previous research that validated that adding more women to the team improves the team's social intelligence, which was not focused on students or professionals in scientific or technical fields, is germane for people in computing disciplines. This paper reports the results that investigate the presence of females in the teams, and how the "female factor" effects the average team social sensitivity, team performance activities and the satisfaction achieved during the tenure of team projects.**

I. Introduction

Software development is a complex activity that requires a group of individuals working effectively as a team [1]. Since the success rate of software development projects is low (32% of all projects succeeding), it is important to understand the factors of software development teams that can have a significant influence on their performance [29]. In the domain of Software Development, projects are complex and teamwork comes into play, as it is not suffice to be done by one person alone. Software development projects are not only difficult because of the complexity of the technologies involved, but also due to complexity of social interactions between the project team members.

Team projects play a central role in the education of engineers. The objective of any good curriculum design is to prepare the graduates for their envisaged professional careers by providing them with appropriate education [27]. In software engineering, software developers develop and maintain software of such a complexity that these tasks cannot be handled at individual level [27]. Various researchers assert that the ability to use soft skills to navigate interpersonal relationships and negotiate social interaction is very crucial to team success [30, 31]. With the current academic syllabus and course of study, many students graduate with the technical, hard skills (i.e., the ability to perform a certain type of task or activity) however they lack the essential soft skills (e.g., their interpersonal abilities to interact effectively with the team members and customers). An employer survey conducted by a staffing company Adecco, turns up similar results; "*44% of respondents cited soft skills, such as communication, critical thinking, and collaboration, as areas with biggest gap*" [3].

A recent survey conducted by the Workforce Solutions Group at St. Louis Community College found that more than 60 percent of employers said applicants lack "communication and interpersonal skills" [28]. Recent studies indicate that employers consistently rate these skills as deficient in their incoming hires. Skills such as communication, teamwork, leadership and adaptability are commonly ranked as deficient. The National Association of Colleges and Employers surveyed more than 200 employers about their top 10 priorities in new hires. Overwhelmingly, they wanted candidates who are team players, problem solvers and who can plan, organize and prioritize their work; Technical and computer-related know-how placed much further down the list [28]. Research conducted by Begel, et al. also identified that recently hired software developers often struggle to adequately communicate when they were in need of assistance or struggling with a problem [15]. Scott, et al. indicated that the ability to work as part of a cross-disciplinary team was necessary in industry [16]. A recent study by Radermacher et al. identified various areas of personal skills and professional ethics such as communication, teamwork, ethics etc. that the graduating students lack when beginning their job in industry but are expected to have by employers [17]. Begel also presented results which signify the importance of communication and collaboration techniques that recent graduates lack [18].

Therefore, there is a need to measure and evaluate the impact of soft skills in order to make sure that the teams are performing at the higher level. Researchers like Carnegie Mellon's Anita Williams Woolley and MIT's Thomas Malone have been successful so far in figuring out three main factors that can have most impact on the team collaboration and collective intelligence [5]. Their study measured Social Sensitivity factor using "Reading the Mind in the Eyes" test which was created and validated by Baron-Cohen et al. [11]. This test gauges the accuracy of individuals in judging someone's emotional state by looking at their eyes.

A subject is presented with a series of 36 photographs of the eye-area of actors. For each photograph, the subjects are asked to choose which one of four adjectives best describes how the person in the photograph is feeling. This study conducted by Wolley et al., established three main factors that have the most effect on team performance and collective intelligence viz. 1) ***Social Sensitivity*** which is basically the ability to correctly understand the feelings, 2) ***Turn-taking behavior*** is giving everyone the chance to speak up during a conversation and 3) ***proportion of females in the group***, which is number of females in a group [5]. The results from the previous studies show that the teams whose members had higher levels of Social Sensitivity score were collectively more intelligent. This study also states that the groups with higher number of females tend to perform better.

The above study was based on generic tasks such as solving visual puzzles, brainstorming, making collective moral judgments, and negotiating over limited resources. Our study is aimed at finding how the group behavior and team performance are affected by Social Sensitivity and particularly presence of female team member. We had previously evaluated the impact of Social Sensitivity (SS) in the context of student teams enrolled at North Dakota State University (NDSU) on a semester long technical project [9]. The results from the study showed that the average SS of teams had a positive and significant correlation with the team performance. That is, the higher the average SS of team (calculated by averaging the individual SS scores of team members), the better the team performed. Furthermore, the individual SS scores were also correlated with the individual performance of the subjects. Thus, there is evidence that the SS of individuals and team members impact the team performance.

This research goes back to the original study findings by Woolley et al., which had showed that the "*proportion of females*" can have a significant impact on the team performance [5]. Also, a study by Snodgrass, Sara E. [4] showed that the females are known to be more socially sensitive than men. Yet again, these studies were not conducted with software professionals or in the context of software engineering domain. Therefore, this research attempts to analyze if including more women in a group would significantly impact the performance of the software engineering teams in the context of students enrolled in the computer science classes at NDSU.

Similar to the research plan of validating SS findings in the context of software engineering team projects [9], this research analyzes the impact of the number of females on the team performance in the context of SS studies at NDSU. To perform this analysis, this research utilizes the SS data from a large number of Computer Science (CS), Software Engineering (SE), and Management of Information Science (MIS) students enrolled in the computer science department at NDSU. The students in teams worked on the semester long projects and their performance was recorded. This paper evaluates the impact of the number of females on their team's average SS values and the team performance by varying the number of females from 0 to 4 (i.e., no female to all females in a team of 4 individuals). For all team sizes (with 0 to 4 females), virtual teams were formed and their SS values along with team performance was compared. The qualitative evaluation of the peer reviews were also performed to provide insights into the results and help researchers better understand the results.

II. Background

This section outlines the motivation for evaluating the impact of SS and the number of females on the performance of software development teams and describes relevant background work to help provide context for the research presented in the remaining sections of this document.

A. Motivation

Successful teamwork relies upon synergism existing between all team members creating an environment where they are all willing to contribute and participate in order to promote and nurture a positive, effective team environment. Team members must be flexible enough to adapt to cooperative working environments where goals are achieved through collaboration and social interdependence rather than individualized, competitive goals (Luca & Tarricone, 2001) Also, Cohen and Ledford examining more than eighty self-managing teams at an American telecommunications company, found that self-managing teams had significantly better job performance and higher employee job satisfaction than tradition working groups or departments [19].

Therefore, it is important to understand the factors that can impact the performance of teams. It is an obvious fact that, "*Together, everyone accomplishes more*" (Michael Lembach, 2005). When it comes to teamwork, most people will consider teamwork in terms of being part of a baseball, basketball, or football team. In contrast, a team is "*really just a group of people who use their skills, experience, and knowledge to work toward a common goal*" (Beverly K. Bachel, 2007) [20]. Although sacrificing individuality for the advancement of a team's interest or goals is difficult for some, teamwork is "*truly greater than the sum of its parts*" (Paul F. Levy, 2005) [20]. Therefore, working with a group, and thinking as a team can have greater advantages [20]. As a team, "*you see different points of view and learn new ways of solving problems*" (Beverly K. Bachel, 2007) [20].

Researchers have identified that there are two main measures of team effectiveness: task performance and team member effectiveness (e.g. satisfaction, participation, and willingness to work together) [22, 23]. Much research has been presented on the subjects of team composition and factors effecting team effectiveness as well, but no single attribute stands out as a key to greater performance. Intriguing questions were raised by a recent study, stating that team success has less to do with smartness of individual team members but more on team dynamics, comprising how well team communicates and collaborates [5]. Our previous study found that Social Sensitivity [9], was the largest contributing factor to a team's collective intelligence and was the central predictor of the team effectiveness and performance [5].

B. Related Work on Role of SS in Teamwork

Social Sensitivity (SS) is the ability to correctly understand the feelings and viewpoints of people [25], which in layman terms is often known as "social" or "soft" skills. Social sensitivity also includes knowledge of social norms, roles and scripts. Possessing emotional and social skills is also associated with higher quality social relationships and more supportive social support systems [26]. Social skills that are key components of social intelligence include the following: the ability to express oneself in social interactions, the ability to "read" and understand different social situations, knowledge of social roles, norms, and scripts, interpersonal problem-solving skills, and social role-playing skills [26].

Our research is motivated by two studies viz. Woolley et al. [5]; a study on social sensitivity that established a correlation between social sensitivity and effective teamwork and by our previous investigation [9]; on role of social sensitivity and classroom team projects. The following paragraphs discuss the major findings from these two studies (by Woolley et al., at MIT and at NDSU) and how it motivated the research presented in this paper.

In the first study, Wolley et al. [5] established a correlation between SS and effective teamwork. During this study, the researchers first randomly assigned 699 individuals to groups of two to five. They then employed social psychologist Joseph E. McGrath's team-task taxonomy to measure how the groups performed in a series of exercises involving brainstorming, physical coordination and even moral reasoning. They found that neither the intelligence of the smartest member nor the average intelligence of the group played much role in the team performance. SS score was measured using "Reading the mind in the eyes" test, in which a subject is presented with a series of 36 photographs of the eye-area of actors. For each photograph, the subjects are asked to choose which of four adjectives best-describes how the person in the photograph is feeling. This study presented three main interesting findings which directly relate to team performance and team dynamics: 1) *Social Sensitivity* 2) *Turn-taking* 3) *proportion of females*; with all three of these factors directly relating to increase in team performance.

The above study was based on generic tasks such as solving visual puzzles, brainstorming, making collective moral judgments, and negotiating over limited resources which ranged from few hours to few days of time-frame. Therefore, it was important to evaluate whether the SS results would hold true in the context of longer projects with software professionals. We had earlier performed a study on the group of 76 students enrolled at NDSU [9]. The results showed that SS is correlated with the team performance and member satisfaction even when applied on semester long project.

Our current study is aimed at finding the relationship between having more females in a team and the team performance activities, as well as satisfaction gained during the tenure of projects. The main motivation of this research comes from the study conducted by Wolley et al., at MIT which stated that adding more women; makes team smarter [5]. We wanted to analyze validity of this claim in computing discipline.

III. EXPERIMENT DESIGN

This study was designed to analyze the relationship between the social sensitivity of student teams in context of the number of female members within each team and the quality of work in computer science team projects. It entails analyzing the SS and team performance data collected during previous studies, which would allow us to evaluate and validate the impact of SS on the quality of student projects. To understand the impact of the number of females, we created all possible combinations of teams (of size 4) with the number of females ranging from 0 (i.e., a team of all males) to 4 (i.e., team with all females) and everything in between.

The study was performed on one hundred and fifty seven subjects (76 in one study {Males: 59, Females: 17} and 81 in its replication in a different year {Males: 76, Females: 5}) enrolled in the Social Implications of Computing course at NDSU. These studies used a randomized experimental design in which participants were tested to determine their SS scores and were then randomly assigned to virtual teams of four participants each. The students worked within their assigned team and completed a semester long project that dealt with an ethical topic in Information technology. The students produced different deliverables throughout the semester and their performance was recorded on each deliverable in order to compute the overall score on the group project.

To evaluate the impact of varying number of females in a team of 4 individuals (varying from no female to all females), we created five virtual team groups viz. MMMM, MMMF, MMFF, MFFF, FFFF. For each of these five virtual groups, all possible combinations were created using all possible combinations. We call these virtual groups, because we combined their SS scores to compute their average SS scores and the members did not actually worked together. We just combined their individual performance and their individual SS data for the purpose of evaluation.

A post Study Survey was evaluated to perform the qualitative analysis of satisfaction and feedback; evaluation was primarily constructed on the terms of satisfaction achieved based on gender and to evaluate how male and female members evaluate their Team-mates.

A. Research Questions and Hypotheses

RQ 1: How the proportion of females is correlated with the performance of student teams on large semester-long projects?

Related hypothesis: Adding more women significantly improves the team performance

Recent studies suggest that adding more women to a team can make them collectively smarter as women are generally found to be better (than men) at reading and responding to other peoples' emotions [5]. But, these researches were conducted via very generic tasks and that too for really short period of duration so it cannot really be confirmed that the adding women would really help in increasing team performance in software industry.

RQ 2: Do females report greater job satisfaction than males even in the same work environment?

Related hypothesis: Female attitude towards jobs are more favorable than males

There have been various studies representing the notion that females report equal or greater job satisfaction than men [32]. Again, these studies were not performed in the context of software development, and it is hoped that the answer to this question will improve our understanding of team composition in software industry and computer science classroom.

B. *Independent and Dependent Variables*

The experiment manipulated the following independent variable:

a) ***Social Sensitivity Score***: Each participant completed the ―Reading the Mind in the Eyes [11] test in order to determine their individual social sensitivity score.

The following dependent variable was measured:

a) ***Average Team Social Sensitivity***: Team performance and the Team Average Social Sensitivity would be measured via forming Virtual Teams and averaging the individual SS of members that make up each team.

C. *Experiment Procedure*

This study used a randomized experimental design in which participants were tested to determine their SS scores and then randomly were assigned into a team. Fig. 1 illustrates the study design overview. After conducting the SS test, the team projects were started and after the completion of the study a Post-study Satisfaction survey was conducted.

Step 1: Pre-Study Social Sensitivity test - SS (social sensitivity) survey was conducted using *Reading the Mind in the Eyes* [11] test, so that we can utilize the SS scores. A glossary that contained definition and a sample sentence was provided to make sure, that the subjects had a clear understanding of the adjectives used in the test. The students were advised to read through the glossary thoroughly and refer to, if need be during the survey.

Step 2: Team formation - Virtual Teams are formed with each group have four subjects; an automated script was used to form the Virtual Teams from the set of enrolled subjects forming teams into groups viz. MMMM, MMMF, MMFF, MFFF, FFFF.

a. MMMM: All four members are male in the team.
b. MMMF: One of the members is female, and the rest three are males.
c. MMFF: Half of the members are male and other half are female.
d. MFFF: One of the members is male, and the rest three are females.
e. FFFF: All four members are female in the team.

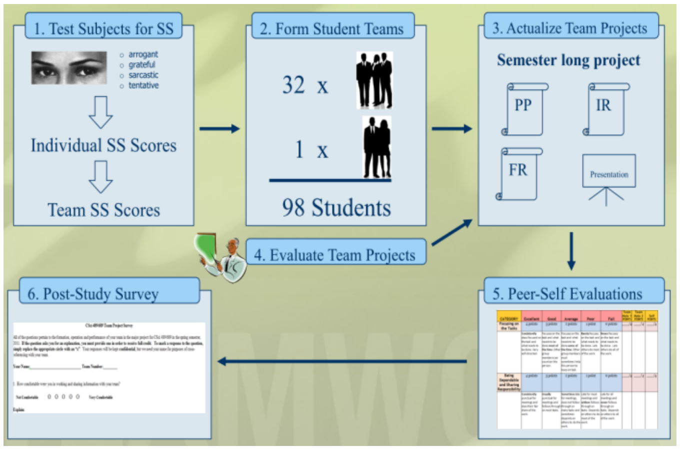


Fig. 1: Study Design Overview

While forming Virtual Teams, extra efforts were put in order to maintain the consistency along the lines that no individual is counted twice for the same Team formation. In simple terms, for example Person A cannot be in one generation of the formation (MMMM) and also again in the other generation of the same formation (MMMM). Though same person could be in different formations viz. Person A could be in (MMMM) and (MMFF), to better evaluate the effect of his/her presence in different possible formations.

The reason behind this kind of Virtual group formation was to understand and evaluate the effect of having female team members in a group; and how the presence of female participants effect the Average Team Social Sensitivity which would eventually effect the Team Performance based on many recent studies [9] [4].

Step 3: Project Actualization - With progress of the semester, the projects are distributed and it marks the start of the project phase. The project includes producing a project proposal, an interim report, a final report, and a final presentation. The proposal required them to articulate ethical questions that they planned to investigate, justify the question's importance, identify major stakeholders and ethical values, specify their research methods, and plan the project. Half way through the semester, each team submitted an interim progress report that described the project goal, objectives, and scope, employed research methods, used evidence to support ethical viewpoints, and evaluated potential stakeholder actions. Near the semester end, each team gave an oral presentation on their project and submitted a final written report.

Step 4: Team performance evaluation - As the Team Performance directly relates to the Average Team Social Sensitivity [4] [9], the team performance is evaluated based on the Team Average Social Sensitivity. As for now, this research would aim at evaluating the Average Team Social Sensitivity and also the impact of female proportion in the teams.

Step 5: Peer and Self Evaluation - After each deliverable, the subjects completed an evaluation of each of their team members as well as themselves. The following ten candidate characteristics of an effective team member were included: *focusing on the tasks*, *being dependable*, *responsibility sharing*, *listening*, *questioning*, *discussing*, *research and information sharing*, *individual performance*, *brainstorming*, and *group teamwork*. Subjects rated each of the ten attributes on a 5-point Likert scale and provided comments. These results were captured to help researchers better understand the results.

Step 6: Post-study survey - A nineteen-question survey was administered to the students at the end of the semester. The post-study survey collected data regarding the self-perceived effectiveness of each team, including whether members felt valued; if the team cooperated, communicated, and interacted well; if effective feedback occurred among team members; if conflict existed and how it was resolved; and what the quality was of the team work environment overall.

D. Data Collection

From Social Sensitivity test, Social Sensitivity scores were collected via Reading the Mind in the Eyes [11] which were used to evaluate the SS values as per female participation.

Teams were divided into 5 groups by varying the number of females. To maintain the consistency we added the constraint of that no individual is counted twice for the same team formation. Table I shows the number of possible combinations (virtual teams) belonging to each group. For example, there were 56 teams made of all males, so that no male was part of two teams. Similarly, 68 virtual teams included at least one female and so on.

Team's average social sensitivity scores for each of the 193 teams formed by varying number of females are shown in Fig. 2. The social sensitivity (SS) scores range from a minimum of 9 to a maximum of 32, with most teams scoring in the range of 19 to 25. It can be seen in Fig. 2, that the SS score of teams is centered around the value (mean = 22.07), a normal distribution. The horizontal axis signifies the average scores for each team and the vertical axis signifies the frequency of SS scores for each team.

TABLE I NUMBER OF VIRTUAL TEAMS FOR EACH TEAM GROUP

Team Group	*Number of teams*
MMMM *– All Males*	56
MMMF *– 3 males, 1 female*	68
MMFF *– 2 males, 2 females*	32
MFFF *– 1 male, 3 females*	21
FFFF*- All Females*	16

A summary of the Social Sensitivity data for all the team groups is shown in Table II.

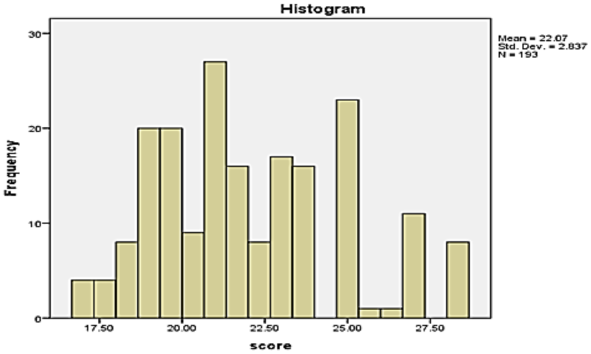


Fig. 2: Social Sensitivity Scores

As mentioned earlier, we investigated the peer evaluation data to analyze if there exists any relation between the SS score and the satisfaction achieved and also the dynamics of female factor in the different team groups. As there are many different kinds of team activities which can impact the performance and dynamics of the group we carefully analyzed the ten question survey to understand the effects of the presence of females and subjects with high SS score. The survey questions were designed mainly on the tasks of brainstorming, sharing responsibility and other team activities described below.

TABLE II SOCIAL SENSITIVITY SUMMARY DATA FOR EACG GROUP

Team Group	*Average Team SS Score*	*Variance(σ^2)*	*Range*
MMMM	23.17	11.28	9.7 (28.5-18.7)
MMMF	22.70	5.41	8.2 (27-18.7)
MMFF	20.81	5.55	7.2 (24-16.7)
MFFF	20.31	1.79	4.9 (23.6-18.7)
FFFF	20.31	4.29	4.5 (22.5-18.5)

a. **Focusing on the Tasks**: How well does the team member stay focused on the task and does what needs to be done?
b. **Being Dependable**: How good is the team member at being punctual for meetings?
c. **Sharing Responsibility**: How good is the team member at doing their fair share of the work?
d. **Listening**: How good is the team member at listening respectfully to all members of the team during discussions and at considering others opinions?
e. **Questioning**: How well does the team member respectfully pose questions to all the team members?
f. **Discussing**: How well does the team member respectfully interact and discuss issues with all team members?
g. **Research and Information-sharing**: How well does the team member gather research, share useful ideas and defend/ rethink ideas relating to the group's project goals?
h. **Individual Performance**: What is quality of the team member's work?
i. **Brainstorming**: How often does the team member originate, seek and develop ideas and solutions collaboratively with others?
j. **Group Teamwork**: How good is the team member at consistently collaborating, cooperating and compromising as necessary to achieve goals?

This peer evaluation questionnaire survey provided the data regarding team satisfaction and team cohesion, which would be discussed in the results sections.

IV. RESEARCH RESULTS

This section provides analysis of the quantitative and qualitative data that includes average team social sensitivity scores and the feedback (peer evaluation) respectively. Because each team (virtual) consisted of four subjects and the SS data was for each individual, the individual SS scores were combined into one team score based on the Team Formation strategy. The SS score of each team was calculated by averaging the individual team member's SS scores.

Fig. 3 shows the average social sensitivity of virtual teams within each group. The horizontal axis in Fig. 3 depicts the no. of virtual teams belonging to team types viz. MMMM, MMMF, MMFF, MFFF, FFFF and the vertical axis show the average Social Sensitivity score for all the teams of particular team type. The result in Fig. 3 shows that the teams with higher number of females did not have higher SS score in comparison to the teams with fewer or no females.

To further test our hypothesis 1, One-way ANOVA test was performed to test whether the mean of SS score differs between the five team types. Via unpaired *t test,* we also analyzed the

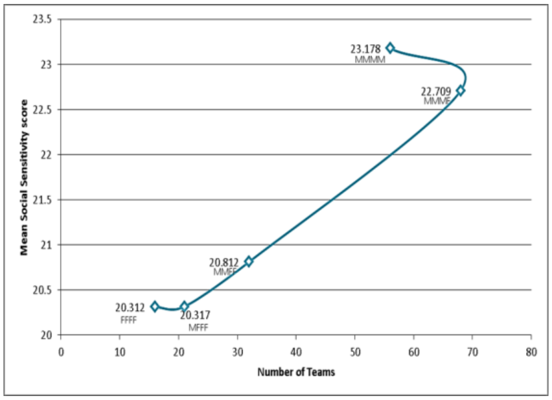


Fig. 3: Average Social Sensitivity Score Comparison

statistical significance for the difference in the team group's social sensitivity score, which is shown in the Table III below.

Based on the results shown in Table III, it is evident that adding a female to all male group did not significantly improved the SS of the group (MMMM vs. MMMF had a non-significant value of p = 0.36). On the contrary, a team of all males (MMMM) were significantly more socially sensitive (on average) when compared with the teams with 2 females (MMFF), three females (MFFF), and all females (FFFF). The p-values were less than 0.05 for these three comparisons.

TABLE III PAIR WISE COMPARISON BETWEEN SS OF TEAMS

Team Groups	Statistical Significance
MMMM – MMMF	p-value = 0.3617(Not statistically significant)
MMMM – MMFF	p-value = 0.0007(statistically significant)
MMMM – MFFF	p-value = 0.0007(statistically significant)
MMMM – FFFF	p-value = 0.0019(statistically significant)
MMMF – MMFF	p-value = 0.0003(statistically significant)
MMMF - MFFF	p-value = 0.0001(statistically significant)
MMMF - FFFF	p-value = 0.0003(statistically significant)
MMFF – MFFF	p-value = 0.3867(Not statistically significant)
MMFF - FFFF	p-value = 0.4751(Not statistically significant)
MFFF – FFFF	p-value = 0.9935(Not statistically significant)

For the next part of the study, we analyzed the 10 question peer evaluation survey conducted at the completion of the project to understand how does the team activities relate to people with high social score and presence of females. We analyzed whether the team cohesion and peer evaluation was dependent on the gender.

Post study peer evaluation survey was conducted using 10 questions based on team process activities which highly impacted the team effectiveness: Brainstorming, Dependability, Discussing, Task focus, Listening, Performance, Questioning, information sharing, responsibility and teamwork. Survey questionnaire was handed out to every student in the class to evaluate his team members. We used 5 point Likert scale (0-4) to evaluate the survey responses, and the average scores for both the genders are shown in the table below (Table 4).

Table IV shows the results of the 10 question survey conducted in the class after the team projects were completed. Average scores received by both the genders, in each category is shown, where 28 responses were received for females, while 88 responses were received for males. The results show that average female scores for all of the project process activities was similar in comparison to males; and there was no significant difference. Though it does not directly align with Woolley et al., at MIT [5] findings that women tend be much more collaborative and increases the collective intelligence; it is highly possible that Woolley's claims are not valid for women in computing disciplines.

For evaluating the notion of females being more satisfied at work in comparison to males, we stressed on the list of 'buzzwords' that showed extreme emotions such as 'exceptional', 'distracted', 'expressed' and used these buzzwords in evaluating the satisfaction level achieved based on gender differences.

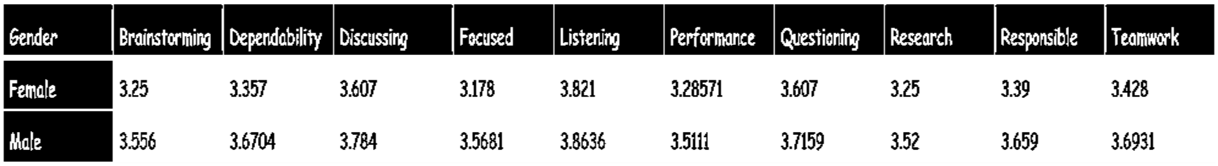
While Evaluating Post Study Survey, the results showed various instances where Male subjects seemed to be unsatisfied with the team members such as "Person X doesn't actively participate in group discussions" and "Person X's research for the first few assignments wasn't very thorough" while female subject were generally extremely satisfied with other Team member performance and efforts.

V. DISCUSSION OF RESULTS

Our fundamental finding is that the proportion of females is not highly correlated with the performance of student teams, on large semester-long projects. Average social sensitivity score of teams with high number of female members was significantly lower in comparison to teams with fewer numbers of females. Our initial hypothesis, "*Adding more women can significantly improve the team performance*" does not hold true. The presence of females did not have any effect on the Team performance activities such as Brainstorming, Research and Information Sharing, Teamwork to name a few. We averaged scores for each performance activity based on the gender, and found that females on average had a similar score in comparison to males on all of the team performance activities.

One other important finding in our study is that it supports the recent study by PayScale, Inc. which shows that women tend to be more satisfied with their jobs as compared to men [12][13]. While analyzing the Post Study and Peer evaluation survey's open ended questions, it as seen that females tended to be more satisfied with the team performance and team dynamics. This finding asserts our hypothesis that "*Female attitude towards jobs are more favorable than males*" holds true even in the fields of Computing. Our key finding also supports new global research [33] from Accenture, which has found that a greater number of women (40%) are satisfied with their current job

TABLE IV PEER EVALUATION STUDY RESULTS



Gender	Brainstorming	Dependability	Discussing	Focused	Listening	Performance	Questioning	Research	Responsible	Teamwork
Female	3.25	3.357	3.607	3.178	3.821	3.28571	3.607	3.25	3.39	3.428
Male	3.556	3.6704	3.784	3.5681	3.8636	3.5111	3.7159	3.52	3.659	3.6931

Copyright © 2014 CSREA Press, ISBN: 1-60132-286-0; Printed in the United States of America

and are not looking for new job opportunities as compared to men (28%). Research across various nations across Europe, also assert the fact that females show a significantly higher level of job satisfaction [34].

VI. Conclusion And Future Work

Our initial belief that the effect of including female members would be exhaustive and wide-spread, not only in terms of achieving higher team performance but also would lead to higher team satisfaction and constructive growth with less interpersonal challenges, proved to be a negative hypothesis in terms of increase in average social sensitivity. Also, the results showed that female presence did not have any subtle contribution in the team process activities essential for a project to succeed. Through the qualitative analysis of data, it is suggested that females tend to have a more favorable attitude towards job and report more satisfaction than males even in the same work environment. Research study published by Accenture [33], also has found that a greater number of women (40%) are satisfied with their current job and are not looking for new job opportunities as compared to men (28%). Authors such as Randy Hodson from Indiana University at Bloomington have also suggested that women's attitudes toward their jobs are often more favorable than men's [32]

References

[1] Sarmasik, G. ; Demirors, O. . The role of teamwork in software development: Microsoft case study. EUROMICRO 97. New Frontiers of Information Technology.

[2] M.Anderson. Add Women, Get Smarter: What's the Deal with Social Sensitivity. Retrieved August 25th , 2011 from: http://www.theglasshammer.com/news/2011/08/25/add-women-get-smarter-whats-the-deal-with-social-sensitivity/

[3] M.White. The Real Reason New College Grads Can't Get Hired. Retrieved Nov 10th,2013 from: http://business.time.com/2013/11/10/the-real-reason-new-college-grads-cant-get-hired/?iid=biz-article-mostpop1

[4] Snodgrass, Sara E., Women's intuition: The effect of subordinate role on interpersonal sensitivity. Journal of Personality and Social Psychology, Vol 49(1), Jul 1985, 146-155.

[5] Anita Williams Woolley, Christopher F. Chabris, Alex Pentland, Nada Hashmi, Thomas W. Malone. Evidence for a Collective Intelligence Factor in the Performance of Human Groups

[6] J. Hsu. Good Decision- Making Groups Need More Women, Not Geniuses. Retrieved September 30th, 2010 from: http://www.livescience.com/10755-good-decision-making-groups-women-geniuses.html

[7] M.Garber. MIT management professor Tom Malone on collective intelligence and the "genetic" structure of groups. Retrieved May 4th, 2011 from: http://www.niemanlab.org/2011/05/mit-management-professor-tom-malone-on-collective-intelligence-and-the-genetic-structure-of-groups/

[8] Thomas W. Malone, Robert Laubacher, Chrysanthos Dellarocas. MITSloan Management Review: The Collective Intelligence Genome, Spring 2012, Vol.51 No.3

[9] Lisa Bender, Gursimran Walia, Krishna Kambhampaty, Kendall E. Nygard, Travis E. Nygard. Social sensitivity and classroom team projects: an empirical investigation. SIGCSE '12 Proceedings of the 43rd ACM technical symposium on Computer Science Education

[10] http://news.nationalgeographic.com/news/2010/09/100930-collective-intelligence-groups-teams-women-sensitive-health-science/

[11] S. Baron-Cohen, S. Wheelwright, J. Hill, Y. Raste, I. Plumb, J. Child Psychol. Psychiatry 42, 241 (2001).

[12] Daily Mail UK. Women happier at work than men. Retrieved from: http://www.dailymail.co.uk/news/article-7356/Women-happier-work-men.html

[13] C. Rampell. Women May Earn Less, but They Find Their Work More Meaningful. Retrieved February 16th, 2012 from: http://economix.blogs.nytimes.com/2012/02/16/women-may-earn-less-but-they-find-their-work-more-meaningful/

[14] Glenn, Taylor, and Weaver 1977; Penley and Hawkins 1980; Quinn, Staines, and McCullough 1974

[15] A. Begel and B. Simon. Novice software developers, all over again. In Proceedings of the Fourth international Workshop on Computing Education Research, ICER '08, pages 3 - 14,New York, NY, USA, 2008. ACM.

[16] G. Scott and D. N. Wilson. Tracking and profiling successful it graduates: An exploratory study. In Proceedings of the 13th Australasian Conference on Information Systems, ACIS '02 , pages 1185 - 1195, 2002 .

[17] A.Radermacher and G.Walia. Gaps Between Industry Expectations and the Abilities of Graduates: Systematic Literature Review Findings. SIGCSE'13, Colorado, USA.

[18] A. Begel and B. Simon. Struggles of New College Graduates in their First Software Development Job. SIGCSE'08, Orgeon, USA, 2008. ACM.

[19] D.Gallie, Y. Zhou, A. Felstead, and F.Green. Teamwork, Productive Potential and Employee Welfare. SKOPE Research Paper No.84 May 2009.

[20] (2008, 03). Teamwork. StudyMode.com. Retrieved 03, 2008, from http://www.studymode.com/essays/Teamwork-137596.html

[21] Teamwork. Anti Essays. Retrieved March 22, 2014, from the World Wide Web: http://www.antiessays.com/free-essays/101425.html

[22] Cohen, S.G. and Bailey, D.E. 1997. What Makes Teams Work: Group Effectiveness Research from the Shop Floor to the Executive Suite. Journal of Management. 23, 3, 239-290.

[23] Hackman, J.R. 1983. A Normative Model of Work Team Effectiveness, Technical Report #2, Research Program on Group Effectiveness, Yale School of Organization and Management, November, 1983.

[24] P. Tarricone and J. Luca. Successful teamwork: A case study. HERDSA 2002, from http://www.deakin.edu.au/itl/assets/resources/pd/tl-modules/teaching-approach/group-assignments/case-studies/case-study-edith-cowan-university.pdf

[25] Greenspan, S. 1981. Defining childhood social competence. Advances in Special Education. 3, 1-39.

[26] R. Riggio and R.Reichard. The emotional and social intelligences of effective leadership – An emotional and social skill approach. Kravis Leadership Institue, Claremount McKenna College, California, USA.

[27] M.Bielikova and P. Navrat. Experiences with Designing a Team Project Module for Teaching Teamwork to Students.

[28] E. Plannin. The Surprising Reason College Grads Can't Get a Job. Jan 29th, 2014 Retrieved from: http://www.thefiscaltimes.com/Articles/2014/01/29/Surprising-Reason-College-Grads-Can-t-Get-Job#sthash.qoG7wvSb.dpuf

[29] The Standish Group International. CHAOS Summary 2009 Report. Retrieved from: http://emphasysbrokeroffice.com/files/2013/04/Standish-Group-CHAOS-Summary-2009.pdf

[30] C.Chan, J. Jiang and G. Klein 2008. Team Task Skills as a Faciltator for Application and Development Skills. IEEE Transactions On Engineering Management, Aug 2008.

[31] M. Ikonen and J. Kurhila 2009. Discovering High-Impact Success Factors in Capstone Software Projects. SIGITE '09.

[32] R. Hodson 1989. Gender Differences in Job Satisfaction: Why Aren't Women More Dissatisfied? The Sociological Quarterly, Volume 30. 1989

[33] The Path Forward (Accenture Inc.), 2012. Retrieved from: http://www.accenture.com/SiteCollectionDocuments/PDF/Accenture-IWD-Research-Deck-2012-FINAL.pdf

[34] L. Kaiser 2005. Gender-Job Satisfaction Differences across Europe: An Indicator for Labor Market Modernization. Institute for the Study of Labor, December 2005.

[35] S. Cohen, S. Wheelwright, J. Hill, Y. Raste, I. Plumb 2001. The "Reading the Mind in the Eyes" Test Revised Version: A Study with Normal Adults, and Adults with Asperger Syndrome or High-functioning Autism. J. Child Psychol. Psychiat. Vol. 42, No. 2, pp. 241±251, 2001 Cambridge University Press (2001 Association for Child Psychology and Psychiatry)

Relational Metrics Model for Software Configuration Management

Charles Donald Carson, Jr.
Hassan Pournaghshband
Department of Computer Science and Software Engineering
Southern Polytechnic State University

Abstract - Changes during a software products life cycle are inevitable. These changes can correct or enhance software functionality and/or reduce costs associated with the software product. However, changes can also introduce added risks and unknowns during a software products life cycle and can result in unpredictable software behavior. Software configuration management (SCM) is a part of configuration management (CM) which supports a complex framework for monitoring, managing, and controlling changes to a software products configuration during its life cycle. In this paper, we introduce the development of a relational metrics model as a possible means for better managing and controlling software configuration change during a software products life cycle from the SCM activities, software engineering measures, and any available CM metrics. This relational approach to managing changes within a software product life cycle will result in a more effective means of validating change configurations by envisioning change from a unified quantifiable view instead of by individual change artifacts and components within the SCM framework.

1. Introduction

Software Configuration Management (SCM) is a software change management framework for managing any configuration changes during the development of a software product. A software configuration can include existing, functional and physical attributes of a software system as well as combinations of software systems [1.] "Software Configuration Management is an umbrella activity that is applied throughout the software process. Because software configuration change can occur at any time, SCM activities are developed to (1) identify change, (2) control change, (3) ensures that change is being properly implemented, and (4) report changes to others who may have an interest" [2.] Changes to a software product's configuration can typically result in revised software characteristics, and by derivation, affect the metrics that report on the software product itself. Though changes to a software product are intended to enhance its performance, and/or reduce costs, they can also introduce added risks. Software systems can comprise of multitudes of individual change artifacts and components. Furthermore, each of these change artifacts and components can involve individual dependencies and constraints. Documentation of changes in these systems can be inconsistent, incomplete, and may or may not adequately cover the individual changes in each of the change artifacts and components as a whole. It is difficult to control the software configuration change process solely by visualizing individual change artifacts and components with the expectation of accurately and consistently controlling any resulting new software configuration behavior. In this paper, we propose a relational approach to help in managing software configuration changes throughout the life cycle of a software product.

Also we will investigate the development of relations between entities and attributes derived from the various activities within the SCM framework along with the available software engineering measures and associated CM metrics.

2. Our Approach

In developing our relational metrics model, the primary functional elements of SCM, that is, configuration identification, configuration change control, configuration status accounting, and configuration audit were analyzed. Possible entities and attributes from activities within these functional elements of SCM are then derived, as well as resulting software engineering product, process, and project measures.

Copyright © 2014 CSREA Press, ISBN: 1-60132-286-0; Printed in the United States of America

Primary elements of SCM activities and any resulting software engineering measures were used in establishing the data requirements of the relational metrics model as listed below in Table 1;

Table 1

SCM Activity/Software Engineering Measures	Description
Configuration identification	Identification of software configuration items.
Configuration control activity	Change process, control, and release process.
Configuration status accounting	Reports based on any configuration management data.
Configuration accounting audit	Recording and reporting the status of change requests/components in the software product.
Product measure metrics	Indirect and or direct measurements of software related activities.
Process measures metrics	Deliverables involving artifacts and or documents resulting from process activities.
Project measures metrics	Production of items used by processes in order to produce their outputs.

Through the use of an Entity Relation Diagram (ERD), we represent our established data requirements as relationships between the individual configuration item entities undergoing the actual configuration change, as well as the SCM process activities and any resulting software metrics that are expressed as attributes of these [3.] Subsequently, the ERD will then be converted into the relational metrics model itself.

In Figure 1, we have illustrated placing configuration items (CI's) that are typical in the development of a software product, such as software component, related requirements, and associated test cases under SCM control. During particular points of the CI's life cycle, the CI's undergoing any configuration change is represented as versions as reflected through the various entities illustrated in the diagram. The attributes of these entities are the actual representations of the activities and resulting software engineering metrics within the functional elements of the SCM process. In the example entity, Configuration Item Version, changes to a CI are represented as change attributes; the individual change activities within the functional elements of SCM and any artifacts from these activities. The resulting metrics are represented as attributes of that entity. An actual abstraction of the SCM process; its activities and metrics produced within each of the functional elements is essentially represented throughout the ER diagram.

We have organized any metrics produced from the various activities within the SCM process into the respective areas of the software engineering measurements. From this type of organization, a better understanding can be realized regarding the measurements produced from the various activities within the functional activities of the SCM process and how these impact the software product and the SCM process itself.

Typical metrics within configuration management can be correlated with each of the software engineering measures based on the product, process, and project measurement types and what these might represent. Correlation of CM metrics and these software engineering measures are discussed in [1.] From the analysis of the individual activities derived from functional elements of SCM process and the correlation of the CM metrics with its respective software engineering measures, we have established an understanding of the SCM process as a whole.

Through correlating the entities and their respective attributes we have indicated that the conceptual model provides a sufficient representation of the overall activities within the functional elements of the SCM process.

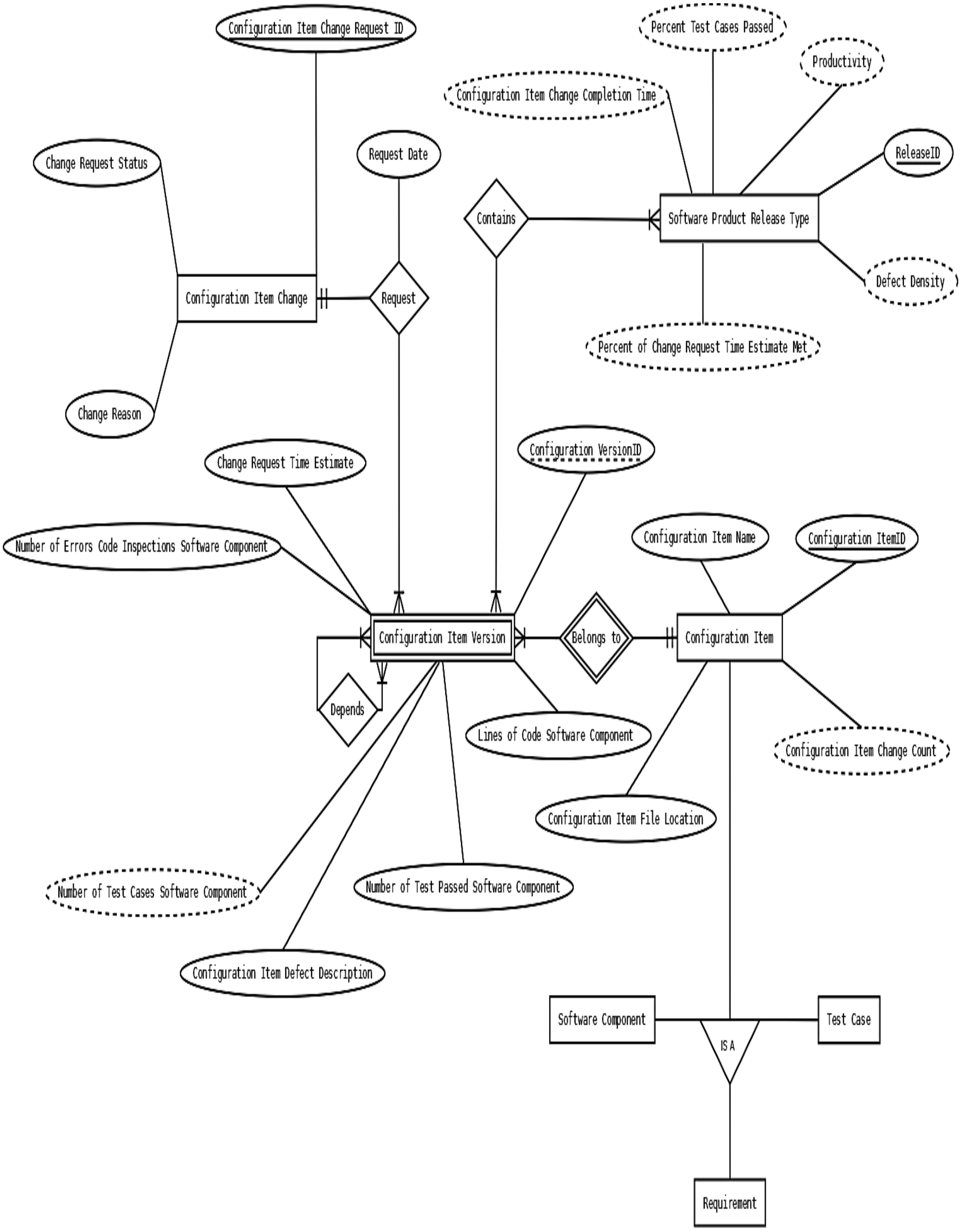

Configuration Item Change Request ID
Percent Test Cases Passed
Productivity
Configuration Item Change Completion Time
Change Request Status
Request Date
ReleaseID
Contains
Software Product Release Type
Configuration Item Change
Request
Defect Density
Percent of Change Request Time Estimate Met
Change Reason
Configuration VersionID
Change Request Time Estimate
Configuration Item Name
Configuration ItemID
Number of Errors Code Inspections Software Component
Configuration Item Version
Belongs to
Configuration Item
Depends
Lines of Code Software Component
Configuration Item Change Count
Configuration Item File Location
Number of Test Cases Software Component
Number of Test Passed Software Component
Configuration Item Defect Description
Software Component
IS A
Test Case
Requirement

Figure 1 - ER Diagram of Relational Metrics Model

Furthermore, the activities within the functional elements of the SCM process provides a means of identifying what CI's in the various software configuration baselines are undergoing changes at various points in time. The conceptual relation metrics model through the use of relations provides an overall view of the SCM process by envisioning change from a unified quantifiable view instead of by individual change artifacts and components within the SCM framework [1.] Figure 2 shows the relation metrics data model resulting from the conversion of each entity and its respective attributes represented in the ER diagram into individual relations.

3. Traditional SCM Approach and the Relational Metrics Model

We have found that the traditional approach to SCM consists mainly of individually managing the individual change artifacts and components within a software products lifecycle. However, this presents a level of difficulty, possible inconstancies, and is susceptible to being error prone. Furthermore, it is difficult to control the software configuration change process solely by visualizing individual change artifacts and components with the expectation of accurately and consistently controlling any resulting new software configuration behavior. Though various SCM tools exist, these tools can tend to focus only on the SCM aspect and does not fully integrate any software engineering measures which can be crucial in providing any important SCM activity metrics.

Software engineering metrics which consists of product, process, and project measures can be derived from any of the various SCM activities. These measures can help in quantifying change within the SCM activity framework and help drive any necessary improvements in the SCM process as well of improvements to individual artifacts of the software project and the software project itself.

Lastly, in comparing the proposed Relational Metrics Model to traditional SCM approaches, we believe there exits an advantage to managing change configuration from a unified view perspective through interrelations of the SCM activities and the software engineering metrics rather than by individual change artifacts and components as presented by traditional SCM approaches and tools. Although change activities within traditional SCM approaches track changes and any artifacts affected, however questions arise in quantifying the various areas of change in order to identify, correlate, and prioritize changes. Through an abstracted view of software products change activities and related metrics, software project managers for example, can anticipate and project issues and establish a "preemptive" approach rather than a "reactive" approach which is a prevailing factor in many software projects presently.

4. Conclusions

In this study, we proposed the development of a relational model as a possible means for managing and controlling software configuration change during a software products life-cycle from the SCM activities and the available software engineering metrics. Current approaches to SCM focus primarily on individually managing the individual change artifacts and components within a software products lifecycle which is susceptible to errors as well as not providing an abstracted view of a software products change activities. We demonstrated that our relational approach to managing changes within a software product lifecycle will result in a more effective means of validating change configurations by visualizing change from a unified quantifiable view rather than by individual change artifacts and components within the SCM framework.

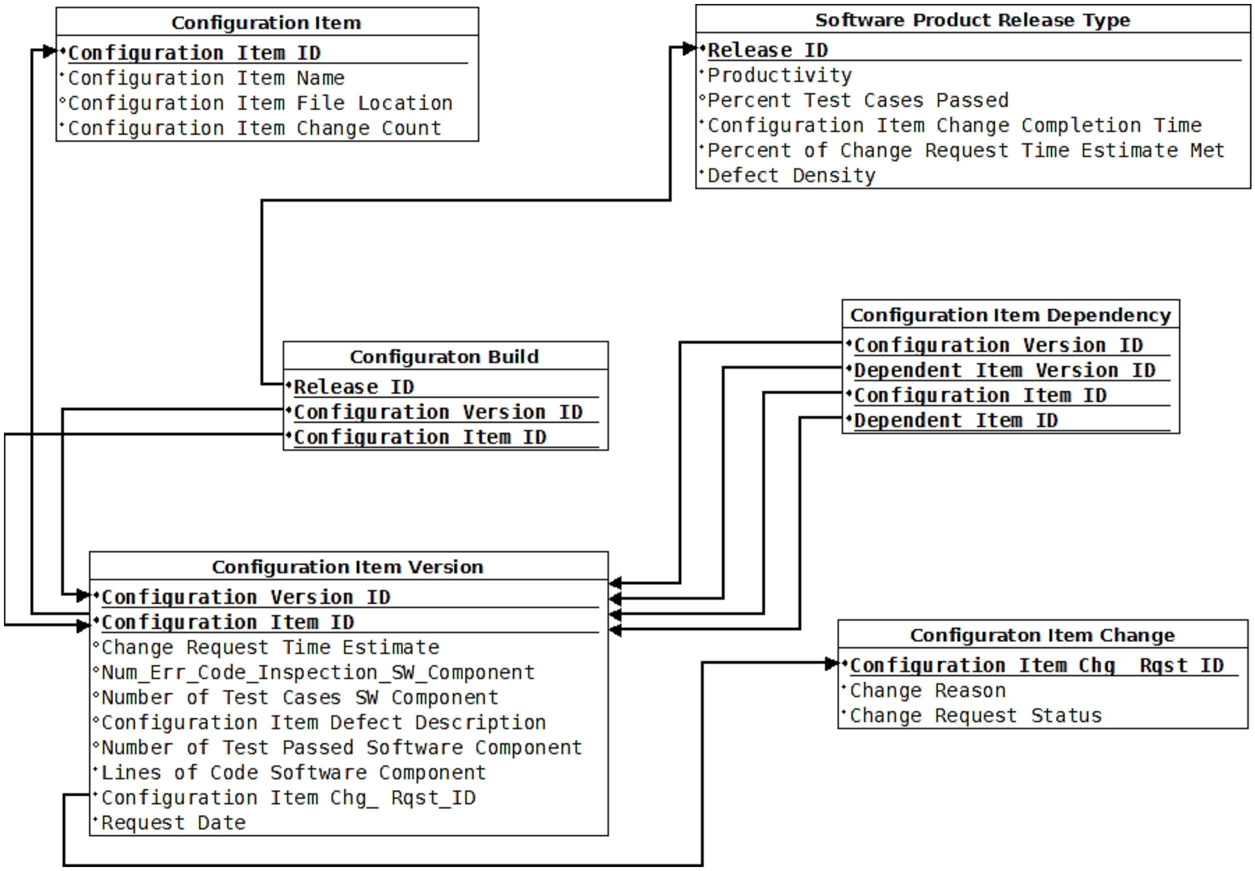

Configuration Item
Configuration Item ID
Configuration Item Name
Configuration Item File Location
Configuration Item Change Count
Software Product Release Type
Release ID
Productivity
Percent Test Cases Passed
Configuration Item Change Completion Time
Percent of Change Request Time Estimate Met
Defect Density
Configuraton Item Dependency
Configuration Version ID
Dependent Item Version ID
Configuration Item ID
Dependent Item ID
Configuraton Build
Release ID
Configuration Version ID
Configuration Item ID
Configuration Item Version
Configuration Version ID
Configuration Item ID
Change Request Time Estimate
Num_Err_Code_Inspection_SW_Component
Number of Test Cases SW Component
Configuration Item Defect Description
Number of Test Passed Software Component
Lines of Code Software Component
Configuration Item Chg_ Rqst_ID
Request Date
Configuraton Item Change
Configuration Item Chg Rqst ID
Change Reason
Change Request Status

Figure 2- Relation Metrics Data Model

References

[1].J.Keyes, “Software Configuration Management,” Auerbach Publications, 2004.

[2]. R.Pressman, “Software Engineering-A Practitioner's Approach,” Seventh Edition, McGraw-Hill, 2010.

[3]. Gornik, D. (2003). Enity Relationship Modeling With UML. Retrieved from www.ibm.com/developerworks/rational/library/content/03July/2500/27 5/2785_uml.pdf

[4].Software Metrics (http://www.cs.ucl.ac.uk/staff/A.Finkelstein/advmsc/11.pdf)

[5]. Harrington, J. (2002). Relational Database Design, San Diego, California: Morgan Kaufmann Publishers

[6]. IEEE Standards: "IEEE 828-2012: STANDARD FOR CONFIGURATION MANAGEMENT IN SYSTEMS ANS SOFTWARE ENGINEERING."

Modeling and Self-Configuring SaaS Application

Nadir K.Salih, Tianyi Zang
School of Computer Science and Engineering, Harbin Institute of Technology, Harbin, Heilongjiang, China

Abstract - *The main objectives of SaaS application are to make the management and control of software easier and take the management strain away from consumers. However, it also leads to software services available globally and this has been realized in our paper by designing a new model for SaaS application. The three levels we have classified in our model easy adapted to workflow and services. From the application layers meat-model description we discovered a new algorithm for the self-configuration of SaaS application. We used a feature model to define the variation of our model's management levels. The Xml file obtained from the feature model gave interactive communication between three levels and our new self-configuration algorithm. That increased the performance by selecting from the web a suitable configuration for every level. We have explained all the processes by an online booking example. Finally we present a conclusion and future work.*

Keywords: SaaS application, Modeling, meta-model, Self-configuration, Feature model

1 Introduction

Modeling SaaS application is very important field and building a SaaS by leveraging existing technology is a challenging issue and needs brand new software technology [1]. It is useful for both business and educational purposes, such as businesses can be easily adopted in several domains, like healthcare, education and OA (Office Automation) for this to be modeled, the SaaS application [2] [3] demands new requirements. In this paper we have drawn a new model [4] [5] of SaaS application. We have summarized our contributions as follows:

- Built new model for SaaS application.
- By meta-model defined four layers to compose the system and showed the associations and dependencies of the layer elements.
- Demonstrated the relationship between the three levels in our model by a workflow as a business process layer
- We observed the necessity of sharing the workflow (can share other things, e.g. software components, SLA/QoS, etc) in each level and how it can improve efficiency and better control customer service.
- We have classified services of SaaS application according to three levels. Some services are done by the user; others are by the tenant and some by the provider.
- Increased the quality of system by showing it has different levels of services which can serve by order of importance. The service of the provider it is more important than the service of the tenant and tenant services are more important than user services.
- Self-configuration of the algorithm to dynamically configure SaaS components.
- Commonality and variability are indicators for components costs.

We organized this paper by beginning with the design of the new general architecture for SaaS application in section 2. Depending on the model driven development we derive SaaS meta-model layers in section 3. That classifies the SaaS application management in three levels. To demonstrate this new opinion we take online booking SaaS application as running example in section 4. In section 5 we have described the service architecture for SaaS application. We realized self-configuration of the model by a new algorithm in section 6. Section 7 described the related work. Finally, we present the conclusion and point to future work.

2 Architecture of SaaS Application

System modeling is a very important issue in software engineering, because it has great importance in system development. Thus, we have defined our architecture of SaaS application, and described our model by using the meta-model concept to show we could easy achieve management by the new model. Application architecture specifies that technologies are to implement one or more information systems in terms of data, process, and interface, and that these components interact across a network [6]. Architecture is a transferable abstraction of a system [7]. As we study from recent researches architecture development of SaaS is a large part of the application. Our novelty here is to create a conceptual model for SaaS application as depicted in figure 1.

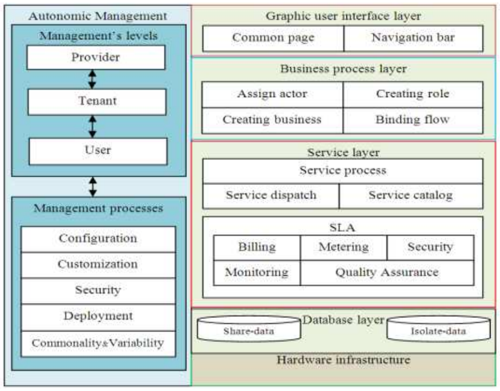


Fig 1 Architecture of SaaS Application

Copyright © 2014 CSREA Press, ISBN: 1-60132-286-0; Printed in the United States of America

This architecture includes main three parts:

- Application layers contain four layers beginning from the graphic user interface (GI) that uses the web page and navigation bar to communicate to the user and SaaS application. The second layer is the business process (BP) to show the workflow for business by defining some roles and actors activities. In the service layer service (S) process is determined by the dispatch manner and catalog. In addition the service level agreement has been defined for some services like billing, monitoring, QA, metering, and security. The final layer is the database (DB) layer which shares the common data and isolates variable data.
- The hardware infrastructure includes all hardware resources working in SaaS application servers, storages, network, etc. The allocation and placement algorithm is used to optimize these resources.
- The autonomic management manages all management in SaaS application and will be self-managed in the three levels of provider, tenant and user. This will be applied in the application layers to manage the processes (configuration, customization, security, validation, commonality, and variability).

Adapting the same application in the case of multiple users to somewhat different and specific needs of a certain user is important therefore creating a new architecture suitable for development is needed. The new concept in our proposed architecture of SaaS application is the base in three levels adapting to develop SaaS by adjusting to the tenant's instant functions from the provider level. We have looked to adaptive to different instances for all users from the tenant level. Also, adjusting user requirements from the user level are controlled.

The goals of modeling are:

- Develop architecture for SaaS approach based on three levels to realize organization and user requirements.
- Configuration and adaptation of SaaS applications must be performed.
- Customized adaptation for every level to ease management of SaaS application.

3 Meta-Model of SaaS Application

Looking at the proposed model for SaaS application three management levels have been classified that are depicted in figure 2. According to the kind of service SaaS system can determine the level of management. The reasonability of this classification is a variation [8] of the application layers from level to level.

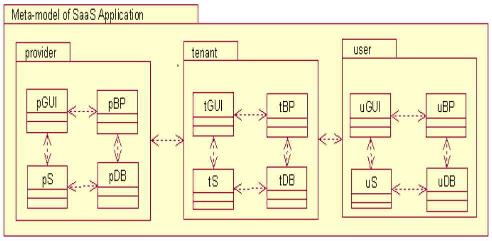


Fig 2 Three Management Levels for SaaS Application

All application layers can be variables in the provider level for different tenant requirements. Likewise, in the tenant level all application layers are changeable for different user requirements. However in the user level we observed it is the same as in GUI, BP, and S, but DB it different from user to user. The general meta-model [9] of these layers are depicted in figure 3 below.

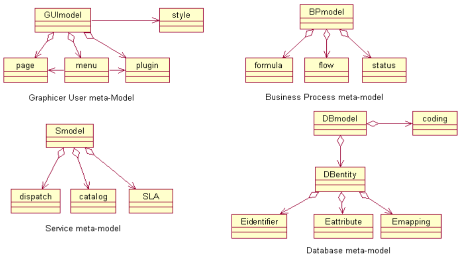


Fig 3 Meta-model of SaaS Layers

4 Demonstrate SaaS Meta-Model

As we mention our SaaS model has three levels of management including the provider, tenant and user. Every level has different managements for the application layers, which are defined in the upper meta-model of SaaS layers. We can take an on-line hotel booking example to demonstrate this model as seen in figure 4. The provider is a highly configurable service that travel agencies can use for booking hotels on behalf of their customers. For that we can say the provider is the administrator for all travel agencies. The travel agencies look like tenants and customers are users that want to book a travel service.

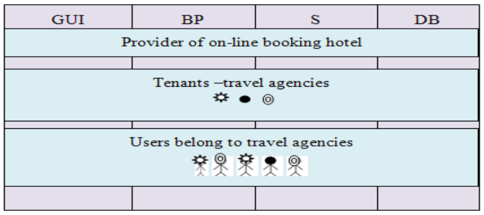

GUI
BP
S
DB
Provider of on-line booking hotel
Tenants –travel agencies
Users belong to travel agencies

Fig 4 SaaS Application of on-line Booking

Depending on the meta-model layers we describe our own model of SaaS application. At first, the provider is the administrator of all the travel agencies to management the activities that appeared through the layers:

- The Graphic User Interface(GUI) has different style for the GUI layer which has various types like standard menu and tree menu to display page or plug ins as requirement the from tenant.
- In the Business Process (BP) the provider puts the business logic in a formula. It can be a variable from agency to agency. And the workflow defined by a sequence, branch and return according to the agencies requirements. In addition the status is used to show that the software is open or closed in each different status.
- Services (S), can be different for the service dispatcher and service catalog between agencies. For example if the agency categorizes the services as and VIP they will be

dispatched differently and then cataloged which means the sending, indexing services from provider to tenant are not the same. However the variation in the service level agreement for many services billing, metric, and security can be according to agency requirements.

- Data Base (DB), in this layer we should define as any agency by a unique identifier. And attributes of the data will normally be different from one agency to another. However, the mappings that describe the relationship between different entities vary in the data.

The second level is the tenant that corresponds to the travel agency, and in our example to the management of all users' activities inside the layers:

- The Graphic User Interface (GUI) is the job of the travel agency to show a suitable style interface for the users as a classification for the user as a normal or VIP user.
- In the Business Process (BP), the business logic can be different from user to user so that travel agency can use a different formula according to the type of user. The workflow can be a variable in this system like low season is different from high season booking. However it defines the status of the system as open or close in various cases.
- In the Service layer the travel agency sending and indexing the services depends on the type of user, and the classification of the service for different costs. This will then be applied according to the service level agreement between the agency and customers.
- In the Data Base the travel agency defines any user by an identifier because it is unique for every user, and attributes data can be different from user to user. In addition, the mapping that describes the relationship between entities will vary.

The third level is user that can communicate with on-line travel agency for hotel booking. In this level the management for SaaS application layers is defined as:

- The variation in graphic user Interface is defined by the tenant or travel agency and it needs management if the user uses a different machine such as the Windows client program running in a PC with the resolution of 1680×1050, a smartphone application with the resolution of 640×480, and a tablet application with the resolution of 1024×768. Moreover, in the business process the logic and workflow is the same put forward by the travel agency. However, in the service layer introductions from travel agency are according to user requirements. While the data base layer needs management because it is different from user to user in the identifier, attributes and mapping relationships for different entities.

In the relationship between the three levels we can consider the workflow in our example of online booking with the hotel as the provider level and is managed at the tenant level or by the travel agencies. Figure 5 shows the process of booking when the request reaches the travel agencies. Then, they can begin to display the information that is filled out by the customer as they have an office to check this data and submit it to be accepted or rejected. This sequence is the same in the two travel agencies, but the second travel agency has a difference in workflow due to the manager check. Here the provider can share a customizable workflow for multiple travel agencies using the assembled workflow.

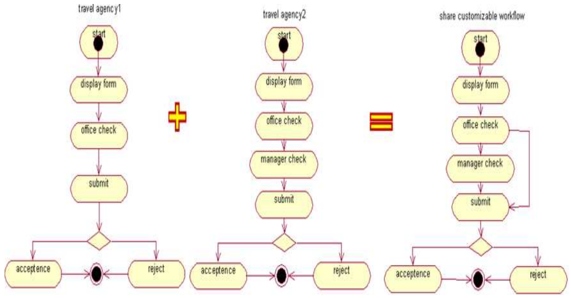


Fig 5 Share Customizable Workflow on the Provider Level

Here the relationship between the provider level and tenant level is a sharing customizable workflow. This can be managed and controlled by many travel agencies in a process by which it shares the same sequences. Our model realized the benefit for a business process by minimizing the many workflow processes in sharing a one workflow process. The relationship between the tenant level and user level can define by the booking process from customer to travel agency. For example, customers in one travel agency web begin by browsing and searching for bookings and payments to finish the transaction. Another travel agency after searching and booking lets the customers to make another search to see new options for booking as depicted in figure 6.

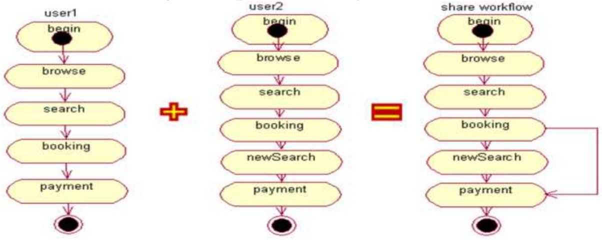


Fig 6 Share Workflow on the Tenant Level

Share workflow can eliminate many unnecessary steps that lead to increased efficiency more than other strategies [10]. It also improved consistency and control result for better customer service. From this relationship between the tenant level and user level we can obviously see our model easily manages and adapts to SaaS application.

5 Service Architecture in SaaS Application

To adapt and manage SaaS application we should understand the service architecture represented in our model. Then the feasibility will be clear of our novelty in classifying our model in the three levels of management. The online booking hotel running example will illustrate this principle. Though the web service SaaS application provides different services as defined in our model as a variable from level to level. From figure 7 we classified our concrete service in $C_{Si}= \{s_{pi},s_{pi}\ldots s_{pn},\ s_{ti},s_{ti}\ldots s_{tn},\ s_{ui},s_{ui}\ldots s_{un}\}\ ,1\leq\ i\ \leq m,\ s_p,\ s_t,\ s_u$ are provider service, tenant service, and user service, respectively. These concrete services obtain the same abstract services as from a functional view, which can be defined by the application layer in formal methods.

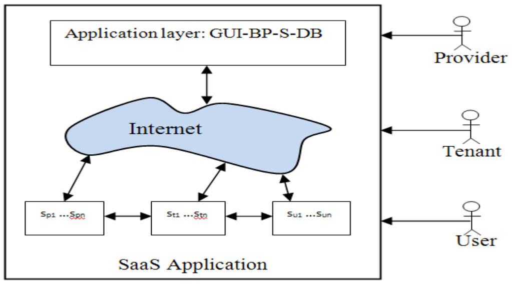


Fig 7 Service Architecture in SaaS Application

To explain this we return to our example of the online hotel booking and see the workflow of this system as appeared in the activity diagram below in figure 8.

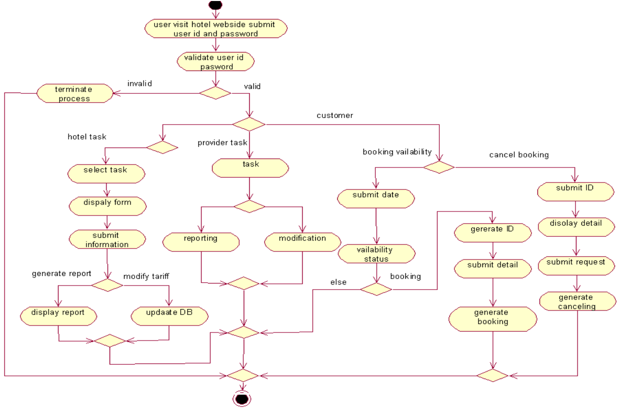


Fig 8 Activity Diagram of Online Hotel Booking SaaS System

Looking at this workflow diagram we start executing as soon as user visits the hotel administrator web side and submits a user ID and password. The SaaS application validates this data and if it is invalid it automatically terminates the process. Here, because the data is valid, it will go on to make the diction from our three levels of management. This user is the customer of the SaaS application and then will provide two services as the different workflow booking are available. The user is the tenant or administrator of the hotel and looks like the travel agencies. This level has services managed by the system display in the form of a customer, submits the information, modifies the tariff, and generates the required report for the customer. The user can be the last level of our model management and is the provider that administrates for all travel agencies here the SaaS application which will provide services to management agencies like reporting for all events and modifying (delete, update, add) the travel agencies data. The SaaS online booking hotel system associates the following abstract services:

- *Available booking*, which lets the customers book a hotel.
- *Canceling booking* prevents customer from booking the hotel.
- *Modify tariff*, the price may change in low season of booking.
- *Generate customer report*; display some reports to the customer.
- *Modify travel agency data*, update, delete and add travel agency data by the provider.
- *Reporting travel agency* displays payment, resources and all management activities of the travel agencies from the provider.

We define two abstract services from each level as follows:

as_{u1} = *Available booking*
as_{u2} = *Canceling booking*
as_{t3} = *Modify tariff*
as_{t4} = *Generate customer report*
as_{p5} = *Modify travel agency data*
as_{p6} = *Reporting travel agency*

6 Self-Configuration SaaS Application

By Self-Configuration Algorithm SAAS application (SCAS) and the model driven development approach can be used to implement self-management for SaaS application. The autonomic diagnosis, failures and performance reconfiguration that is required for repair can occur in every layer. The constraint model to check the data conformance has been used in the meta-model to specify constraints. We defined the monitoring model to the instrumentation for collecting data about system behavior. It is very important to reference the architectural entities in reconfiguration or what should happen in any given condition, for what is suitable in the meta-model for monitoring requirements for the environmental and constraint model. We defined the meta-model for the runtime model to reduce the managing complexity during runtime. The prediction method is used to select a suitable configuration [11].To realize self-configuration for SaaS application we should monitor the requirements and environmental conditions. The model systems can be revised and used to generate new codes automatically. The model and meta-model can be control by some constraints. We have used the feature model to define the variation for all meta-model layers in three levels. Simply, we take the provider configuration from the provider to the tenant as an example seen in figure 9. This feature model shows the variations of the configurations for every layer according to constraints that defines the features.

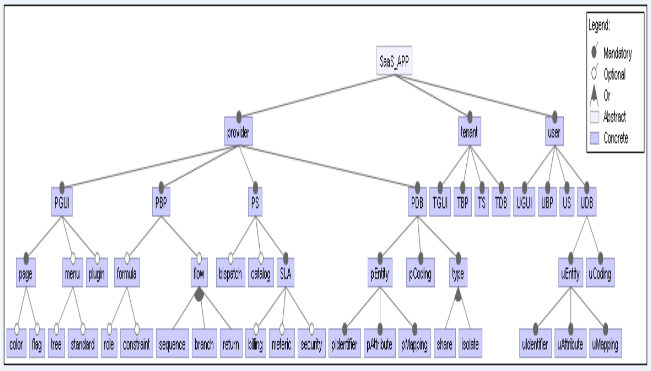


Fig 9 Feature Model for SaaS Meta-model Layers

6.1 Logic of SCAS Algorithm

To simplify understand this variation we represent this feature model in a hyper-arc,e, with a multiplicity value, where mv = [min…max], whose tail (feature) is selected, no less than the min and no more than the max features of the hyper-arc's head (child features) should also be presented in the configuration.

$H = (V, E)$, where $V = \{v_1,v_2,v_3, ... ,v_n\}$ is the finite set of vertices (or nodes)

$E = \{E_1,E_2,E_3, ... ,E_m\} / E_i \subseteq V \quad i = 1,...,m$ is the set of hyper-edges.

$Ei = (T(Ei),H(Ei)) = T(Ei) \subseteq V \wedge H(Ei) \subseteq V$

E_i is a directed hyper-edge , where T(E_i) is the set of tail nodes and $H(E_i)$ is the set of head nodes of E_i .

When $|H(e)| = 1$ (children's cardinality set is one):

- If $min = 1 = max$, the feature is mandatory, and should be present if the parent, or it is a required constraint and then the child should also be present [1..1].
- If $min = 0$, $max = 1$, the feature is optional [0..1]

When $|H(e)|>1$(children's cardinality set is more than 1):

- if $min = 1 = max$, it is a XOR alternative feature group, and only one child should be present at most if the parent is present.
- if $min = 0$, $max = 1$, it is an optional feature group, and the child features can be present or not as long as its parent is present, or it is a mutex constraint and at most one of the child's features can be present
- if $1 \leq min \leq max \leq |H(e)|$, it is a OR feature group, and no more than the max and no less than the min child features can be present if the parent feature is present. Figure 10 shows the hyper-arc diagram for the feature model.

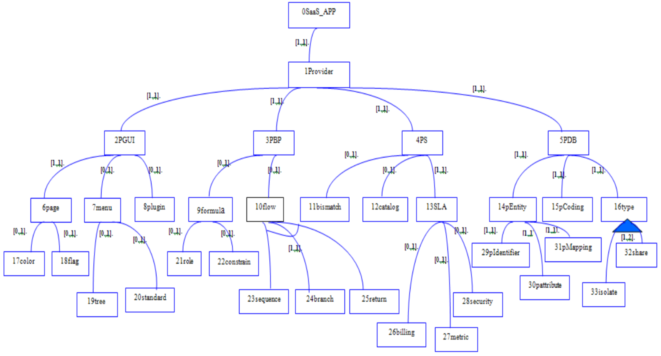


Fig 10 Hyper-arc Diagram for the Feature Model

Formula of variation and commonality:

We can represent the variability and commonality of each layer of SaaS application by formula 1 and 2, respectively.

$$variability = \frac{k}{2^{n}-1} \ldots\ldots\ldots\ldots\ldots (1)$$

k: is number of products

n: is number of all features

Variability increase the number of tenants and cost

$$commonality = \frac{sharenode}{k} \ldots\ldots\ldots (2)$$

Sharednode: number of appeared nodes in all products

Node in provider diagram as input:

{0(0.SaaS_APP); 1(1.Provider); 2(2. PGUI); 3(3.pBP); 4(4.PS);5(5.PDB);6(6.page);7(7.menu);8(8.plugin);9(9.formula);10(10.folow);11(11.dismatch) ;12(12.catalog);13(13.SLA);14(14.pEntity);15(15.pCoding);16(16.type); 17(17.color);18(18.flag);19(19.tree);20(20.standard);21(21.role);22(22.constraint);23(23.sequence);24(24.branch);25(25.return);26(26.billing);27(27.metric);28(28.security);29(29.pIdentifier);30(30.pAttribute);31(31.pMapping);32(32.share); 33(33.isolate) };

6.2 Provider Level Hyper-arcs

We described the vertices and edges for all models. Also we showed the relationship between the vertices and determined all groups belonging to any vertices in table 1. Hyperarcs: From[mult]H{To}: as input

Table 1 Inputs of Vertexes and Relationship

Node	Relation	Childs Group	Node	Relation	Childs Group
0	[1,1]	{1}	5	[1,1]	{14 }
1	[1,1]	{2 }	5	[1,1]	{15}
1	[1,1]	{3 }	5	[1,1]	{16 }
1	[1,1]	{4 }	6	[0,1]	{17,18 }
1	[1,1]	{5 }	7	[0,1]	{19,20}
2	[1,1]	{6 }	9	[0,1]	{21,22 }
2	[0,1]	{7 }	10	[1,1]	{23,24,25}
2	[0,1]	{8 }	13	[1,1]	{26}
3	[0,1]	{9}	13	[1,1]	{27}
3	[0,1]	{10}	13	[1,1]	{28}
4	[0,1]	{11,12 }	14	[1,1]	{29,30,31}
4	[1,1]	{13 }	16	[1,1]	{32,33}

From our description of the configuration for the provider level we have observed that there are many variations. Those will help SaaS application to give multiple choices and provide many tenants. We should input the data as the system can make self-configuration by table 2.

Table 2 SCAS Algorithm of SaaS Application Layers

Algorithm Name: SCAS

```
    Inputs: n : nodes of all model, Relation: Relationship
    between   nodes, Group :all item belong to any
    nodes. L: application layers(GUI, BP, S, DB)
   Outputs: All configurations for layers
 For each $l \in L$
 // children's cardinality set is more than one
   While $H(e) > 1$ do
 // alternative constraint.
       if $min = 1$ and $max=1$ Then only one node
 // optional or mutex constraint
       if $min = 0$ and $max= 1$  Then in configuration
                  can select or not
 //   OR constraint
       if $1\leq min\leq max\leq|H(e)|$ Then will select all or a
                  part of nodes
         end if
         end if
         end if
 // children's cardinality is one
    While $H(e) = 1$ do
 // mandatory or require constraint
 if $min$ = 1 and $max$= 1 then must select in configuration
// optional constraint
        if $min$ = 0 and $max$= 1 then may select or not
        end if
        end if
        configure($l$)
// return number of SaaS Layer configurations
  $k$ = count configuration ($l$)
// return number of all nodes in full configuration
      $n$=count node ($l$)
// calculate variability of layers components
                $V = \frac{k}{2^{n}-1}$
// calculate commonality of layers components
   sharenode=count sharenode(configure(l)).
                $C = \frac{sharenode}{k}$
      end while
      end while
            if $H(e)$ = 0
            invalid configuration
            end if
 end for each
```

In our running example the system needs to reconfigure because the application exchanges from time to time and from travel agency to travel agency and from user to user. As an example, in high season booking will need different configurations to realize all travel agency requirements, which depends on the variation of customer requirements. In this time the travel agency needs to offer various options for booking like different rates. For that we can monitor the variability of every layer as we mentioned above and make decisions to best configure and realized the agency requirements. In addition we can monitor the commonality of any component in every layer to show the degree of sharing of this component in different configurations.

The algorithm can dynamically configure every layer to show all options that show the variation of the configurations from tenant to tenant. However, to calculate the variability and commonality they will be an indicator to monitor the system configuration.

7 Related Work

The direction of the work is for the meta-model and modeling for the evolution of SaaS application. In [12] defined the criteria for designing the process model and realized commonality and variability of modeling to maximize the reusability. Researchers in [13] analyzed tenancy history metadata from the graphic user interface (GUI), workflow, service, and data layer for dynamically adjusting template objects. In [14] provided an on-demand service-oriented model driven architecture to develop an enterprise mashup prototype as a practical case study. Authors in [15] regarded PIM can be used to generate different PSMs using transformation tools to minimize the time, cost and efforts in developing cloud SaaS and enhance the return on investment. They identified technical issues and proposed their effective solution spaces in [16]. In [17] proposed a QoS model and MCDM (Multi Criteria Decision Making) system for SaaS ERP. They empirically examined main drivers and inhibiting factors of SaaS-adoption for different application types in [18]. In [19] studied forecasts effects expected when the SaaS model will be fully applied to the library network. And they presented functional requirements and an operation model of SaaS-based library management systems. In [20] extensible business component model named xBC is proposed for describing both the structural and behavioral properties of generic SaaS applications to minimized the amount of sources needed to be reexamined by a transformation when the source is changed. All development and evolution done by meta-model, but it is not mention how to enable model-driven development and tool support for the integration of self-management functionality into SaaS application.

8 Conclusions

This research is a foundation to build a new model for SaaS application. By meta-modeling it defined four layers to composite system and showed the associations and dependencies of the layer elements. We have demonstrated the relationship between three levels in our model by a workflow model as a business process layer. We observed the necessity of sharing the workflow in every level which can improve efficiency and better control service to the consumer. In our new model we could classify services of SaaS application according to three levels. We have increased the quality of the system by showing it has different level services and can serve by important ordering. From meta-model layers we have conducted the variation of the element layers and can obtain different configuration than other methods [21]. In addition we have described the self-configuration algorithm to dynamically configure SaaS components.

In future work we will see the effect of our new model to QoS in the SaaS application.

9 Acknowledgement

This work has been developed with the support under the project with number: 2012AA02A604, 863 Program key projects in China: The Technology and the System Development for Smart Acquirement of Personal Healthcare Information. And so the Key Project of NSF in China: Methodology of Value-oriented Software Services: Theory, Method and Application with number: 61033005

10 References

[1] L. Cui ,, T. Zhang , G. Xu , D. Yuan. A Scheduling algorithm for Multi-Tenants Instance-Intensive Workflows. Applied Mathematics & Information Sciences An International Journal, 2013,pp.99-105.

[2] W. Huang, X. Wei, Y. Zhao, Z. Wang, Y. Xiao. A Multi-tenant Software as a Service Model for Large Organization. International Conference on Cloud and Service Computing, IEEE,2013, pp.112-119.

[3] J. Lewandowski, A. O. Salako, A. Garcia-Perez. SaaS Enterprise Resource Planning Systems: Challenges of their

adoption in SMEs. International Conference on e-Business Engineering, IEEE, 2013, pp.56-61.

[4] Y. Demchenko, C. Ngo, C. de Laat. Intercloud Architecture Framework for Heterogeneous Cloud based Infrastructure Services Provisioning On-Demand. International Conference on Advanced Information Networking and Applications Workshops. IEEE, 2013, pp.777-784.

[5] C. Tan, K. Liu, L. Sun, C. Spence. An Evaluation Framework for Migrating Application to the Cloud: Software as a Service. Springer-Verlag Berlin Heidelberg, 2013, pp.967-972.

[6] H. Yuan, X. Liu, C. Guo. A Design of Two-tier SaaS Architecture Based on Group-tenant. International Conference on Computer Science and Network Technology. IEEE, 2012, pp.340-344.

[7] Ralph Hatch. SaaS Architecture, Adoption and Monetization of SaaS Projects. The Art of Service Pty Ltd,2008.

[8] W. Na, Z. Shidong, K. Lanju, Z. Yongqing. Dynamic Adaptive Model of the Multi-Tenant Data Replication Based on Queuing Theory. International Conference on Computer Science and Network Technology,IEEE, 2012,ppt.1755-1759.

[9] Thomas Stahl, Markus Volter. Model-Driven Software Development Technology, Engineering, Management. John Wiley Sons & Ltd. 2006.

[10] T, Zhang, Y. Shi, M. Xu, L. Cui. A Service Provisioning Strategy Based on SPEA2 for SaaS Applications in Cloud. International Conference on Cloud and Green Computing, IEEE, 2012, pp.290-295.

[11] L. Jiang, J. Cao, P. Li, Q. Zhu. AMixed Multi-tenancy Data Model and Its Migration Approach for the SaaS Application. Asia-Pacific Services Computing Conference, IEEE, 2012, pp.295-300.

[12] Hyun Jung La, Soo Dong Kim. A Systematic Process for Developing High Quality SaaS Cloud Services. Springer, 2009.

[13] R. Xiaojun, . Z. Yongqing , K. Lanju. SaaS Template Evolution Model Based on Tenancy History. Third International Conference on Intelligent System Design and Engineering Applications.IEEE,2012.

[14] X. Zhang, K. He, J. Wang, J. Liu1, C. Wang1, H. Lu.On-Demand Service-Oriented MDA Approach for SaaS and Enterprise Mashup Application Development. International Conference on Cloud Computing and Service Computing.IEEE,2012.

[15] R. Sharma, M. Sood Cloud SaaS: Models and Transformation. Springer-Verlag Berlin Heidelberg ,2011, pp. 305–314.

[16] H. J. La, S.W. Choi, . S. D. Kim. Technical Challenges and Solution Space for Developing SaaS and Mash-up Cloud Services. International Conference on e-Business Engineering.IEEE,2009.

[17] J. J. H. Park , H.Y. Jeong. The QoS-based MCDM system for SaaS ERP applications with Social Network. Super computer journal,springer,2012, pp. 614–632.

[18] A. Benlian, T. Hess, P. Buxmann. Drivers of SaaS-Adoption – An Empirical Study of Different Application Types. Business & Information Systems Engineering Journal,2009,pp.357-367.

[19] J. Cho. Study on a SaaS-based library management system for the Korean library network. Emerald Group Publishing Limited. Vol. 29 No. 3, 201,pp. 379-393.

[20] K. Ma, B. Yang, A. Abraham. A Template-based Model Transformation Approach for Deriving Multi-Tenant SaaS Applications. Acta Polytechnica Hungarica. Vol. 9, No. 2, 2012.

[21] J. Li, S. Zhang, Z. Liu2, L. Kong. A Data Rights Control Model for a SaaS Application Delivery Platform. Springer-Verlag Berlin Heidelberg, 2012. pp. 139–146.

Appendix – XML file of feature model

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<featureModel chosenLayoutAlgorithm="1">
<struct> <and abstract="true" mandatory="true"
name="SaaS_APP"><and mandatory="true" name="provider"><and
mandatory="true" name="PGUI"> <and mandatory="true"
name="page"><feature name="color"/><feature name="flag"/>
</and> <and name="menu"><feature name="tree"/><feature
name="standard"/></and> <feature name="plugin"/></and> <and
mandatory="true" name="PBP"> <and name="formula">
    <feature name="role"/> <feature name="constraint"/>
    </and><or name="flow"> <feature mandatory="true"
name="sequence"/><feature mandatory="true" name="branch"/>
    <feature mandatory="true" name="return"/>
     </or></and><and mandatory="true" name="PS">
     <feature name="bispatch"/>
     <feature name="catalog"/> <and mandatory="true"
name="SLA"> <feature name="billing"/>
   <feature name="meteric"/> <feature name="security"/>
</and></and><and mandatory="true" name="PDB">
     <and mandatory="true" name="pEntity">
<feature mandatory="true" name="pIdentifier"/>
      <feature mandatory="true" name="pAttribute"/>
      <feature mandatory="true" name="pMapping"/>
       </and> <feature mandatory="true" name="pCoding"/>
        <or mandatory="true" name="type">
        <feature mandatory="true" name="share"/>
        <feature mandatory="true" name="isolate"/>
         </or></and></and><and mandatory="true"
name="tenant"> <feature mandatory="true" name="TGUI"/>
     <feature mandatory="true" name="TBP"/>
      <feature mandatory="true" name="TS"/>
       <feature mandatory="true" name="TDB"/>
        </and><and mandatory="true" name="user">
      <feature mandatory="true" name="UGUI"/>
       <feature mandatory="true" name="UBP"/>
        <feature mandatory="true" name="US"/>
         <and mandatory="true" name="UDB">
              <and mandatory="true" name="uEntity">
              <feature mandatory="true" name="uIdentifier"/>
              <feature mandatory="true" name="uAttribute"/>
              <feature mandatory="true" name="uMapping"/>
              </and>
               <feature mandatory="true" name="uCoding"/>
               </and> </and></and></struct>
<constraints/><comments/>
<featureOrder userDefined="false"/></featureModel>
```

SESSION

SOFTWARE ARCHITECTURE AND DESIGN PATTERNS + PROCESS MINING + AUTONOMIC COMPUTING

Chair(s)

TBA

A place for everything and everything in its place

Carlo Montangero Laura Semini
Dipartimento di Informatica, Università di Pisa

***Abstract*—The paper presents a top down transformational approach to derive lower level multi-view designs from the Component and Connector (C&C) architectural view. The transformation goes through 3 steps. First a transformation pattern inspired on the Distributed Proxy pattern introduces proxies for all ports in a design C&C view. The second step adds a module and a deployment design view. The last step has to do with quality requirements (in the form of profiles) that are annotated on some elements of the design C&C view and systematically propagated to other elements in the other design views. The process alternates steps in which the views are enriched with containers for model elements, which are inserted in following steps. Hence, the title of the paper.**

Keywords: Software architecture and design. Patterns.

I. Introduction

Software architecture (SA) and detailed design are two key elements in software development. SA is "a multidimensional reality, with several intertwined facets, and some facets – or views – of interest to only a few parties" [4]. A *view* is a projection of the SA according to a given criterion. A view considers only some concerns, e.g. it considers the structuring of the system in terms of components, or it considers some relationships between subsystems: The *module view* highlights the code structure, the *components and connectors (C&C) view* a snapshot of the system in execution in terms of components and connectors, the *allocation view* the deployment of the system on the hardware.

The problem we face here is that of SA and design documentation. The usual practice is to focus on the C&C view, since it is in this perspective that the architect addresses system decomposition: it is natural to reason in terms of the behavior of sub-systems and of their interaction at run-time. In particular, it is easier than reasoning in terms of the structure of the code, and in most cases the architect has no time to invest to complete the other views. As a consequence, the module and allocation views at the detailed design level may easily drift away from the architect's intent and create a mismatch between the architecture and the design.

We intend to show that this problem can be overcome to a large extent, since the skeleton of the three views at the *design* level can be generated from the C&C view systematically, to obtain a good code structure, which mirrors the architectural C&C decisions and facilitates the achievement of the objectives of portability, modifiability, and separation of concerns that a good design must have.

The novelty of the approach is the combination of low level design tactics and patterns with high level C&C structures.

More precisely, we introduce a transformational process that results in detailed multi-view design of the system at hand. Key characteristics of the resulting design are that it supplies the structure of the code and that it keeps the implementation of the communication and of other non functional, e.g. security, requirements clearly separated from the implementation of the component functionalities. So, adaptations to a different communication/security context and changes in the functional requirements can be dealt with independently without any interference. Besides, it is particularly important, in distributed applications, to have a well structured model of the target run time architecture, to guide in the complex task of configuring and initialize the system, taking into account all the related requirements. The last step in our process achieves this goal too.

The process alternates steps in which the models are enriched with containers of model elements, which are inserted in other steps. Hence, the title of the paper.

In Section II we revisit the views on a SA, according to [4]. In Section III we define the transformation process that leads to the platform independent lower level multi-view designs from the C&C architectural view. The specification of quality requirements, and their propagation to all the generated views, are dealt with in Section IV: as an example, we consider secure communication between components.

II. Architectural view types

To fix the terminology and the notation we use (all the diagrams are in UML2), we recall the three main view types on SA, according to [4].

Components and connectors views describe the architecture in terms of execution units (components), and their interactions in terms of connectors (Figure 1, upper right). A component can be an object, a process, a collection of objects, a client, a server, a data store, etc. A connector represents communication paths, protocols, access to shared memory,etc. Components and connectors have an associated specification that defines functional requirements and qualities they have to satisfy. They are related through ports, representing component interfaces.

Module views describe the structure of the software in terms of implementation units and the relationships among them (Figure 1, left). An implementation unit may be, e.g., a class, an interface, a Java package, a software layer. The relationships define dependencies, like *use* and *call*, the *decomposition* of a module into sub-modules (illustrated with containment in the figure), and generalization/implementation relations. In this view, our focus is in documenting the *use* relation: a module *uses* another one if the correctness of the first depends on the availability of a correct implementation of the second. This relationship makes explicit the dependencies among the modules supporting the incremental development and deployment of useful subsets of the system under development.

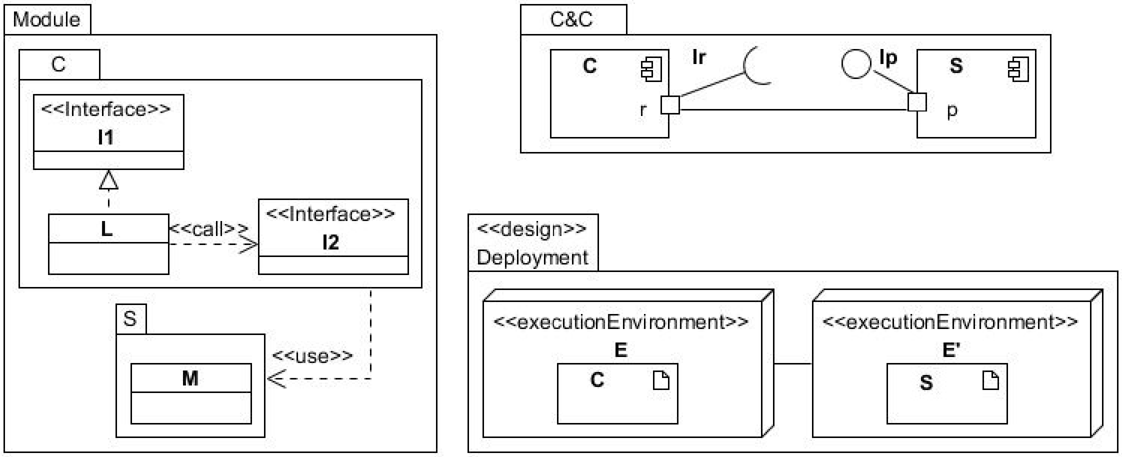


Fig. 1: Architectural Views.

Allocation views describe the relationships between software and other structures, such as hardware or organizational charts. The most common view describes of the *deployment* of the executable artifacts on the environment where they will run (Figure 1, lower right).

III. THE DESIGN VIEWS

In this section we deal with the refinements that lead to the platform independent design views, namely introjection and design view generation, represented by the left and center dependencies in Figure 2.[1] The next section will take care of some platform dependent characteristics (the third dependency in Figure 2).

The process starts at an abstract level from an architectural model, where the C&C view is complete, in terms of black boxes with an associated specification, and the other views are empty.

The first step, that we call *introjection*, refines the C&C view (Figure 2, left), decomposing each component into parts, each dealing with a different responsibility, in the C&C view at the ≪design≫ level (Figure 2, center-left). This refined view is detailed enough to entail the gross structure of the code, in the Module view, and the implementation relations between the modules and the parts in the C&C view. Indeed, the second refinement, *projection*, i) creates the ≪design≫ Module view, making (separate) space for the code of components and connectors; ii) puts the related skeleton parts in their place; and ii) creates the ≪design≫ Deployment view, for the allocation of the system at hand in its execution environment (Figure 2, center-right).

The focus on the SA allows enriching the model driven approach to software development with advantages also in dealing effectively with non functional requirements and design decisions regarding the run time platform, like the choice of the implementation language for a component, the libraries to be used in its implementation, the middle-ware supporting the communication between components, the security characteristics of a connection, the execution environment for a component.

Once the architect has introduced the relevant constraints on the appropriate elements in the ≪design≫ C&C view, the last refinement step, *constraints propagation*, propagates them to the Deployment view, to be exploited by the configuration engineer, and to the Module view, to inform the developers.

Overall, the refinements transform the architectural C&C view into a complete design model, where the different views are pairwise related to insure the consistency of the design, as shown in Figure 2 (right). The intended meaning of the stereotypes in the figure is the following: The execution environments in the deployment view execute the artifacts that *manifest* the components in the C&C view, i.e., the artifacts behavior is the one specified for the C&C components (and related connectors) they depend on. On the other side, the executables are *built* from the code modules in the Module view they depend on, which in turn *implement* the components specified in the C&C view, that is, the code modules satisfy both the functional requirements expressed in the C&C view and the requirements on the execution environment propagated to the other views.

A. Introjection: The Butterfly

The generation of the structure of the ≪design≫ views is made possible by the systematical application of the Distributed Proxy (DP) pattern [10], which in turn uses the *proxy* pattern [6] to decouple the communications among distributed objects from the object specific functionalities. The DP pattern addresses the problem of designing a distributed application, where complexity arises from the need to deal with the specificities of the underlying communication mechanisms, protocols and platforms. More complexity is also due to the fact that the communication mechanism can change in subsequent versions or in coexistent different configurations of the application.

In the ≪design≫ C&C view, the DP pattern is applied to introduce a *remote proxy* for each port of the component

[1] All the dependencies are refinements: the right side model depends on the left one, since it adds details to it.

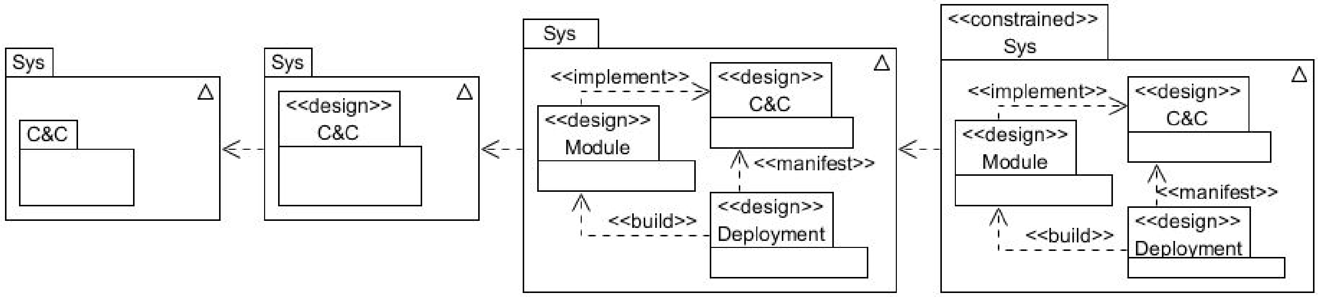


Fig. 2: The refinement steps.

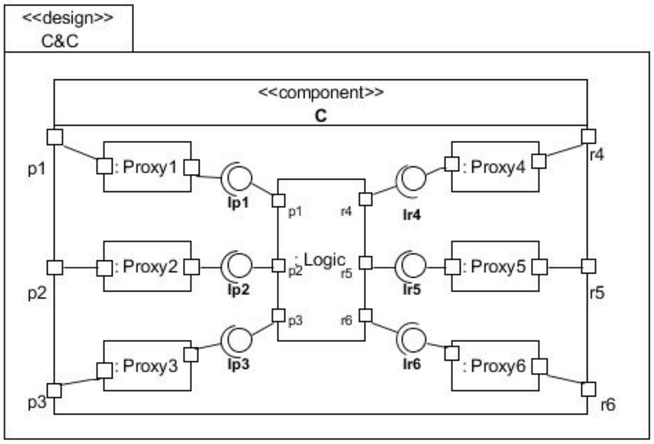


Fig. 3: The butterfly.

and delegating it to deal with all the aspects of the communications through the interface typing the port. The remote proxy represents the server in the client address space (and viceversa), and transfers data to/from the external world using the appropriate middle-ware: they take care of establishing the connections and transforming the data to/from the raw bytes that transit in the communication channels. An additional part, dubbed *Logic* as the middle layer in the Three-Tiers pattern, is introduced to take the responsibility of the functionality of the component, and it is not further decomposed by the pattern.

The resulting structure separates the concerns as required by any good design practice. Moreover, the proxies *internalize* the communications, since they act, with respect to the *Logic*, as local data sources and sinks. By taking care of all the details of the interactions with the low level communication mechanisms, they permit the design and implementation of stand alone components to be deployed independently, as required by the component based software development paradigm (CBSD).

The resulting model, after the introjection step, is the ≪design≫ C&C view: Figure 3 shows what may happen to a component with six ports, three with provided interfaces Ip1-Ip3 and three with required interfaces Ir1-Ir3. Despite the introjection, we keep the delegating ports and the connectors (not shown here), since the connectors are the natural place to record the platform dependent requirements on the middleware, as we will see in the next Section.

We refer to the systematic application of the DP pattern in a component as the *butterfly* design pattern, because of the shape shown in Figure 3, where the "wings" take care of the communications and the "body" of the functionality.[2]

This step of design, which distributes the responsibilities among the parts (the logic and the proxies) and puts them in place in the component's butterfly structure, that is, introjects the interfaces of the components, is obtained by Algorithm 1.

B. Projection: Model and Deployment views

In order to generate the ≪design≫ Module view, we proceed in several steps: first, we create the space for the Logic and Proxies in the code. More precisely, we introduce a ≪component≫ package for each component, to place the implementation of the Logic, and a ≪connector≫ package for each connector. The latter includes two more packages,

Algorithm 1 Birth of the butterfly

```
for every component C in view C&C
  introduce in <<design>> view C&C
    a <<structuredComponent>> C with
      a part with type Logic with
        for every port P of component C
           a port P
  for every <<provided>> interface Ip at port P
                         of component C
    copy in <<structuredComponent>> C
       interface Ip and port P
    introduce in <<structuredComponent>> C
      a realize relation
                    from port P of the Logic
                    to interface Ip and
      a part P with type P_Proxy and
      a require relation from part P
                  to interface Ip
  for every <<required>> interface Ir at port P
                        of component C
    copy in <<structuredComponent>> C
     interface Ir and
    introduce in <<structuredComponent>> C
      a require relation from port P
                   to interface Ir and
      a part R with type R_Proxy with
      a realize relation from part R
                            to interface Ir
  between every part P and port P
              of <<structuredComponent>> C
     introduce a delegate connection
```

[2] The nice symmetry of this suggestive image may often be lost, if the communications are not as symmetric as suggested here.

one per each role of the connector, to place the code for pairs of communicating proxies. Each of these packages is related to the package representing the component the role is attached to. The relation is a ≪use≫ one, since a component can work correctly only if there is a correct implementation of the end-point of each channel it exploits to communicate with its partners. The code that is found in the role packages, once the development is over, may be anything in between a simple delegation to a standard middle-ware and a complete implementation of (one side) of a custom communication protocol. This step is defined by Algorithm 2. Next, we put in place the classes related to the component functionality: we fill each ≪component≫ package with the Logic and its interfaces, reflecting the butterfly in the code: Algorithm 3. Finally, Algorithm 4 puts in place the proxies in the ≪connector≫ packages, together with the appropriate *realize* and *require* relations.

The generation of the Deployment view is similar. The algorithm in Table 5 makes use of the function *image*, with the following meaning. The generation of the Module view establishes a natural correspondence between the elements in the ≪design≫ C&C view and those in the ≪design≫ Module view: given an element *e* of the ≪design≫ C&C view, image(*e*) the generated element in the ≪design≫ Module view. For instance, the image of a component is the homonym ≪component≫ package, etc. In the few cases in which an element gives rise to more then one element, they can be sorted out by their type. A similar argument applies to the relation between the ≪design≫ C&C view and the Deployment view, once generated.

The Deployment view is generated by reflecting each component in an executable artifact deployed in its own execution environment, and each connector in a communication path between the environments of the connected components. Note that the artifact need not be the only one actually used to deploy the component: often it will be so, but the designer is free to use this artifact just to configure the deployment of a (complex) component made up of several pieces. An explicit ≪manifest≫ dependency is also introduced between related executables and components. Note also that the designer is not constrained in the allocation of the execution environments in the available/necessary machines, so that he can take in due consideration also the available/necessary physical connections where to group the required communication paths between the components.

An example. Consider two components *C* and *S* as in Figure 1, middle. Ports *C.r* and *S.p* are connected via the interfaces *Ir* and *Ip*, respectively. The results of the generating the algorithms are in Figures 4, 5, and 6.

Algorithm 2 Place for component and connector roles

```
for every component C in view C&C introduce
    a <<component>> package C in view Module
for every connector between port R, P
          of component C, S in view C&C
  introduce in view Module
    a <<connector>> package C-R-S-P  with
       a <<role>> package R and
       a <<role>> package P
    a <<use>> dependency between C and R and
    a <<use>> dependency between S and P
```

Algorithm 3 Components in place

```
for every <<structuredComponent>> C in <<design>>
                                        view C&C
  introduce in <<component>> package C in
                           <<design>> view Module
    a class Logic
  for every interface I in
                        <<structuredComponent>> C
    copy in <<component>> package C
      interface I
  for every require relation from port P of
                      component C to interface Ir
    introduce in <<component>> package C
      a require relation from class Logic to
                                interface Ir
  for every provide relation from port P of
                  component C to interface Ip
    introduce in <<component>> package C
      a realize relation from class Logic to
                                interface Ip
```

Algorithm 4 Proxies in place

```
for every component C in view C&C
  for every port R with <<required>> interface Ir
    let Conn be the <<connector>> package whose
                          name includes CR and
        Ro be the <<role>> package in Conn
                                depending from C
    introduce
      class R_Proxy in package Ro and
      a realize relation from R_Proxy to
                                interface Ir
  for every port P with <<provided>> interface Ip
    let Conn be the <<connector>> package whose
                        name includes CP and
        Ro be the <<role>> package in Conn
                                depending from C
    introduce
      class P_Proxy in package Ro and
      a require relation from R_Proxy to
                                interface Ir
```

Algorithm 5 Place for artifacts and artifacts in place

```
for every component C in <<design>> view C&C
  introduce in view Deployment
    an <<executionEnvironment>> E with
      an executable artifact C with
         a <<manifest>> dependency to component C
         and
         a <<build>> dependency to image(C) in
                                view Module
for every connector in view C&C
       from port R of C to port P of S
  introduce in view Deployment
    a communication path
       from image(C) to image(S) with
         role names R and P
```

C. More on ports and roles

Here we discuss how connectors can be actually implemented on top of an actual middle-ware, to clarify the nature of the involved objects. We use Java RMI, where the required and provided interface across a connector must coincide, and extend `java.rmi.remote`. So, the `rmic` compiler can generate the classes of the objects that take care of the inter-component communications, namely the *Stub* for the client and the *Skeleton* for the server. These objects are precisely

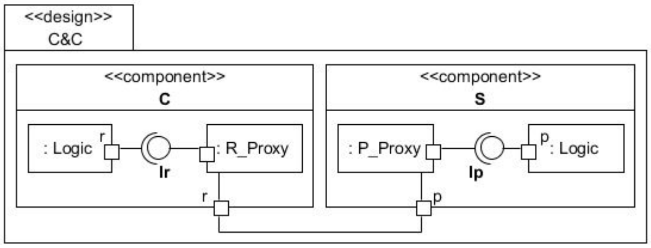


Fig. 4: The introjected scenario.

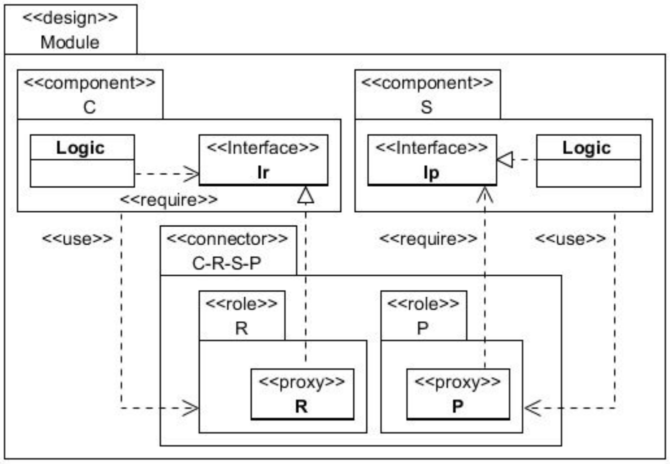


Fig. 5: The design module view of the simple scenario.

the one playing the Proxy roles in the butterflies of the pair of components connected via the common interface, and are created via operations `rebind` and `lookup`, respectively.

Figure 7 shows the behavior of two communicating components, in the context of the class diagram of Figure 8. The remote object, on the server side, has type Logic: when the component is activated, it invokes the Build object of the P role, which in turn invokes the *rebind* operation in such a way that i) the ≪skeleton≫ Proxy (a subtype of the server Proxy) object is created; ii) the remote object is registered in the Naming service, at a well known location; iii) the ≪stub≫ Proxy (a subtype of the client Proxy) code is generated and uploaded in the Naming service, ready to be downloaded in the client.

On the other side, the activation of the Client involves the Build object of the R role, which performs a lookup operation on the Naming service, to obtain a ≪stub≫ Proxy object, which is passed to the client Logic, so that it can access the remote component.

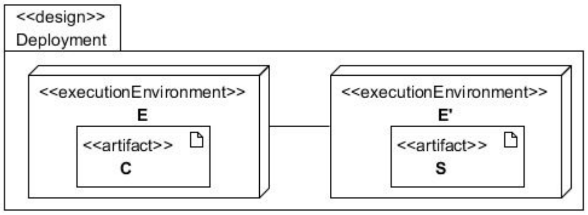


Fig. 6: The design deployment view of the simple scenario.

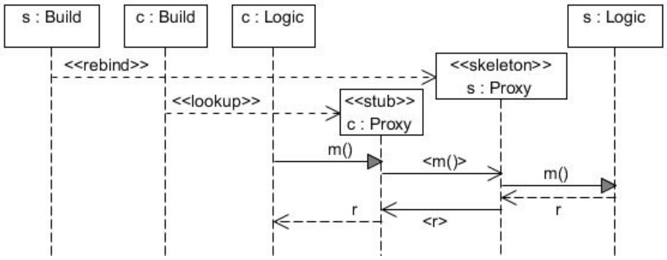


Fig. 7: Proxies in RMI.

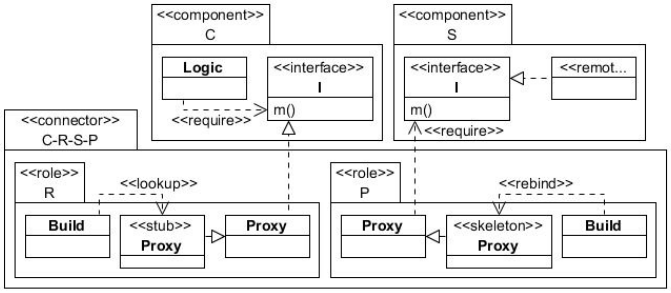


Fig. 8: ≪design≫ Module view for RMI.

The Build classes in the connector roles can be viewed as concrete factories of the component ports, depending on the chosen middle-ware.

Using the butterfly pattern with RMI, the Proxy classes of the connector need no code: they simply stand for the types of the stub and skeleton objects, that are anonymous types in the framework. With another middle-ware, it may be necessary to flesh the proxy with code for themselves, may be by identifying them with classes of the framework itself.

IV. Quality requirements

We now consider how to support recording some kinds of non functional requirements that arise from the design, in such a way that they can be propagated automatically to the views where they have to be obeyed. We consider run-time requirements on the distribution of the components and the execution environments and requirements that affect the development, like which programming language to use to develop a given component. In our approach, these requirements are quite naturally introduced in the ≪design≫ C&C view and propagated to the Module view (those related to development time) and to the Deployment view (those related to run time), where the pertinent engineer can take them in consideration.

UML provides a natural and flexible way to express the requirements, via its notion of *Constraint*, since no specific logic language is prescribed and almost any modeling element can be decorated with constraints.

We consider only a few kind of constraints, and a simple encoding, since our purpose here is only to show how few simple rules can effectively propagate the constraints. Consider first the constraints on the implementation language: a single

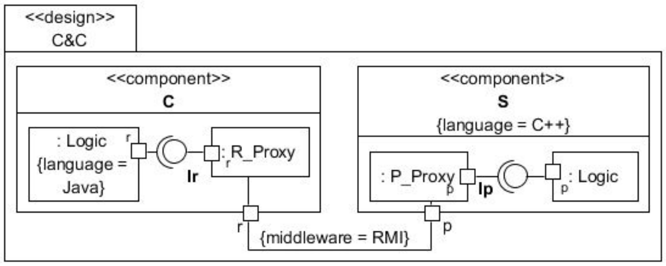


Fig. 9: Requirements in the ≪design≫ C&C view

one can be used for a whole component, or one for the logic and some queer others for the proxies: attaching the constraint to the affected element in the model naturally conveys its scope, as shown by the constraints {language = ...} in the ≪design≫ C&C view in Figure 9: Java is to be used only for the Logic of C, the choice for R_Proxy is delayed, and the whole of S has to be coded in C++.

The next example considers the middle-ware to be used for a given communication between two components: the natural place where to put a constraint is the connector expressing the need for such a communication. For instance, in Figure 9, RMI is required for the communication between C and S.

Before considering how to propagate these constraints to the Module and Deployment views, we need a few remarks.

First, it is reasonable to assume that the constraints are classified with their impact, i.e., if they are relevant at development time, run time, or both. For instance, in general a requirement on the language must be taken into account during the development, but it may have an impact also on the run time environment, e.g., when a specific virtual machine is needed. Similarly, the specification of the middle-ware used for a connector has an impact at development time, since the appropriate libraries must be used, but often also at run time, namely, whenever a well known Naming service has to be available and initialized, for the connection to take place.

Besides, we assume that a propagation function propertyInducedBy(c) is defined for the constraints, so that they are adapted to the target view of the propagation according their classification.

Using also the image function introduced for Algorithm 5, the schema of the propagation algorithm of the constraints is very simple and essentially the same for both views, but for the influence of the target view on the form of the propagated constraint:

Algorithm 6 Constraint propagation

```
for every element E in view C&C
                affected by constraint C
   affect image(E) in view V with
          propertyInducedBy(C, V)
```

As an example of constraints propagation from the C&C view, consider Figure 9, where the Logic in C has to be implemented in Java: image(Logic) in view Module is ≪component≫ package C, since the rest of the code pertains to the proxies, which are placed in the ≪connector≫ packages. The induced property for ≪component≫ package C is the use of Java, so the constraint can be copied in the Module view, as shown in Figure 10.[3] On the other side, the use of Java for the Logic in C affects also image(C) in view Deployment, since the artifact built from C needs a JVM to be executed: the induced property in this view is that the executing environment is such a machine, as shown in Figure 10.

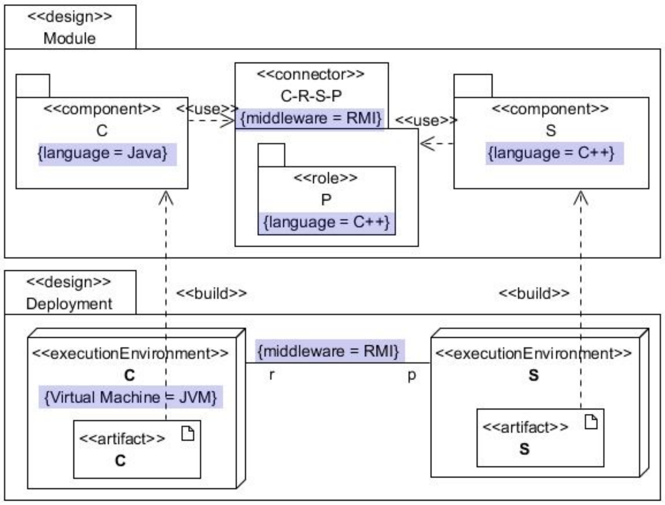


Fig. 10: Propagated requirements

The connector constrained to use RMI in the C&C view has as images the C-R-S-P ≪connector≫ package in the Module view, and the connection between roles r and p in the Deployment view, respectively: they both receive the same constraint, meaning, also respectively, that the two ≪role≫ packages in the former view must use an RMI implementation, and that the connected ≪executionEnvironment≫ C and S in the latter view must support such a protocol.

An example (cont'ed). The results of building the Deployment views and propagating the constraints from the ≪design≫ C&C view to the other two are shown in Figure 10.

A. A different concern: Security

Let us assume that the communication between ports *r* and *p* in Figure 1, middle, needs encryption, to secure sensible data. Security requirements may be conveniently expressed using one of the several UML *profiles* for security, like those in [8], [9]. In our case, the minimal annotation may be a stereotype on the connector, e.g. ≪secure≫ on the connector between ports *r* and *p*. The architect can decide to implement this requirement using the secure version of RMI based on SSL. Then, the new constraint on the connector is propagated as discussed in the previous section. However, the balance of the design constraints may lead to a different solution, to exploit the fact that the Butterfly allows inserting an *intermediary* between a Proxy and the Logic. For instance, this can be an *adapter* [6] used to conform different interfaces:

[3] We remark that only the relevant elements are shown in this diagram.

the interface of a “proxy-from-the-shelf” is likely different from that of the logic. However, an intermediary can fulfill other non-functional requirements, as securing communication by symmetric encryption. In this case the intermediary will encrypt/decrypt the message, and let the proxies deal only with pure communication. To do that, the designer may apply the “symmetric encryption” pattern [7], with the intermediary acting as the *sender* or *receiver* of the pattern, according to its role in the component.

To accommodate such an enriched structure, two packages must be created in the Module view, to place the code for the encryptor and the decryptor, respectively. Like it happens for connectors, a ≪use≫ dependency is added, from the < <component≫ package to the new ones, and encryptor and decryptor are placed in a common container package, which can group all the shared auxiliary code needed to implement the security features.

V. Discussion

One of the advantages of the model-based development process is that a change in a part (e.g., a view) of the design can be propagated to the related parts, as advocated, e.g. in [1]. We can imagine to extend our work to accommodate, in addition to design generation, also propagation of subsequent changes. An approach for change propagation among different Architecture Description Languages has been proposed in [5], with a focus on the C&C view, and state machines.

The Module view generated by our refinements reflects the structure of C&C view in the package structure, making it easy to keep trace in each piece of code of the component/connector/part it is implementing, and bridge the common gap found between architectural concepts and code, as advocated in [2].

Like any pattern, Butterfly does not create or invent a solution, it just organizes, unifies, and documents some good practices. The proposed solution decouples distributed object communication from object specific functionalities. For the communication aspects, it generalizes the *proxy* [6] and the distributed proxy [10] patterns: not only the remote communication concern is taken into account, but also other aspects, like security, can be accommodated. Moreover, in the process we propose, three views are considered and related which each other.

The problem of dealing with several different concerns as independently as possible, which is the core theme of Section IV, is also the main concern of the Aspect Oriented approach to software development. A thorough survey of this line of research, at the different levels (requirements analysis, architectural and detailed design, besides the original programming language level) can be found in [3], [11]. Several of the surveyed techniques use UML and consider, as we do, multiple views at the architectural and design views: the main difference is that they aim at taking into account the details of the composition of the aspect as early as the architectural level, while we focus on propagating the requirements for the different concerns from the ≪design≫ C&C view to the Module and Deployment views at the same level. We expect that the details of aspect composition can be dealt with effectively at this level with any of the known AO approaches, though more work is needed to validate such a claim.

We presented a systematic design derivation process, which starts from an architectural C&C model and creates a platform independent design: the components are given an internal structure in the ≪design≫ C&C view, and the Module and Deployment views are created. The architect then specifies non functional constraints on the ≪design≫ C&C view, and the process propagates them to the other views. Actually, it is seldom the case that development starts from just a set of requirements: usually there are also other constraints, depending on the nature of the problem, the development environment and the general context. For instance, the run-time support may be partially determined, e.g., because the physical structure of the domain entails some degree of distribution, or the implementation choices must follow the organization of the software factory with respect to the use of languages, libraries, etc. The extension of the process to deal also with non functional requirements identified at the beginning of the project, not only those identified during design is left to future work: we envisage that the required techniques will also cater for maintaining the view consistency in front of changes to the architecture/design.

Acknowledgements. The work was partly supported by the Italian MIUR PRIN project “Security Horizons”.

References

[1] B. Brown. Model-based system engineering: Revolution or evolution? In *IBM Rational White Papers*, 2011.

[2] M. Broy and R. Reussner. Architectural concepts in programming languages. *IEEE Computer*, 43(10):88–91, 2010.

[3] R. Chitchyan, A. Rashid, Pete Sawyer, A. Garcia, J. Bakker, M. Pinto Alarcon, B. Tekinerdogan, S. Clarke, and A. Jackson. Survey of aspect-oriented analysis and design. Technical report, AOSD Europe - EU Network of Eccellence, 2005.

[4] P. Clements, D. Garlan, L. Bass, J. Stafford, R. Nord, J. Ivers, and R. Little. *Documenting Software Architectures: Views and Beyond, 2nd Edition*. Pearson Education, 2010.

[5] R. Eramo, I. Malavolta, H. Muccini, P. Pelliccione, and A. Pierantonio. A model-driven approach to automate the propagation of changes among architecture description languages. *Software and System Modeling*, 11(1):29–53, 2012.

[6] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns : Elements of Reusable Object-Oriented Software*. Addison Wesley, 1995.

[7] K. Hashizume and E.B. Fernandez. Symmetric Encryption and XML Encryption Patterns. In *PLoP’09: Proceedings of the 16th Conference on Pattern Languages of Programs*. ACM, 2009.

[8] J. Jürjens. *Secure Systems Development with UML*. Springer–Verlag, 2005.

[9] T. Lodderstedt, D.A. Basin, and J. Doser. Secureuml: A uml-based modeling language for model-driven security. In *Proceedings of the 5th International Conference on The Unified Modeling Language*, pages 426–441, London, 2002. Springer-Verlag.

[10] A. Rito Silva, F. Assis Rosa, T. Gonçalves, and M. Antunes. Distributed proxy: A design pattern for the incremental development of distributed applications. In *2nd Int. Workshop on Engineering Distributed Objects (2000)*, volume 1999 of *LNCS*, pages 165–181. Springer, 2001.

[11] B. Tekinerdogan, A. Garcia, C. Sant’Anna, E. Figueiredo, M. Pinto, and L. Fuentes. Approach for modeling aspects in architectural views. Technical Report d77, AOSD-Europe - EU Network of Eccellence, 2007.

Architecture and Design of Corrosion Prediction Software Multicorp

Arkopaul Sarkar
Institute of Corrosion and Multiphase Technologies
Ohio University
Athens, Ohio
sarkara1@ohio.edu

Dušan Šormaz
Institute of Corrosion and Multiphase Technologies
Ohio University
Athens, Ohio
sormaz@ohio.edu

Abstract: *Multicorp is a corrosion prediction application based on a simulation engine called CorrSim developed in FORTRAN. Multicorp application is able to take information on various chemical and environmental conditions from user through a user interface. An underlying model called Multicorp Model is responsible for managing, calculating, transferring data to CorrSim engine and also reporting corrosion rate and other information. Multicorp application also supports persistent storage and retrieval of various corrosion prediction models as a form of XML files. In this paper, the architecture of Multicorp application, including its data model, data storage and retrieval strategies as well as flow of data within models and engine and general use case demo.*

Index Terms—Software, Design Pattern, Case Study, XML, Corrosion

I. Introduction

Institute of Corrosion and Multiphase Technology (ICMT), part of Ohio University, has been conducting research on internal corrosion of oil wells and pipelines for more than two decades. An industrial consortium joined by worlds twelve leading oil and chemical companies not only supports the entire research and facilities but also work closely with researchers to investigate new ways to deal with corrosion in multiple areas in refinery, rig and transportation of crude oil.

Multicorp, developed by Institute of Corrosion and Multiphase technology researchers and developers, models different electrochemical mathematical equations which can predict underlying corrosion faithfully by taking information on physic-chemical environment. Multicorp software is able to predict corrosion rate for various environment through numerical simulation of the chemical reactions over time and helps user to analyze the root cause of the corrosion with an insightful analysis dashboard.

Although the core of the simulation engine, called CorrSim, is developed in FORTRAN, Multicorp model built in Visual Basic .NET package is solely responsible for managing modeling information which is basically different parameters of physico-chemical environment. Section II describes a brief background of Multicorp development. In section III, the architecture of the model and its instantiation strategy is described in detail. Multicorp implements its own file system to store modeling data persistently. In section IV a detailed discussion on storage and retrieval strategy of Multicorp is presented. In section V, overall flow of the data among models and user interface connection to CorrSim engine and build strategy of Multicorp application is discussed. A walkthrough of the user interface of Multicorp along with analysis of different performance metrics of Multicorp is presented in section VI. Before all of these, a short history of Multicorp is presented in the next section.

II. Background of Multicorp

Corrosion prediction in oil and gas pipelines is a critical aspect of modern oil and gas exploitation [1]. Models to predict corrosion can be classified into two groups:

Empirical models are based on experimental measurement of corrosion rates, and regression model (or multiregression models) to fit the experimental data. Those models tend to be rather simple (deWard [2]) and they capture limited ranges of independent variables (for which experiments were performed). Some of those models have been implemented and marketed as corrosion prediction standards (Norsok [3]) and software [4].

Mechanistic models, which attempt to capture the electro-chemical processes that govern corrosion formation, including mass transport, diffusion, gas and liquid flow, and electrolyte formations. Those models are of different levels of complexity. Freecorp [5] uses a simplified electrochemical model of steady state corrosion formation to predict corrosion rates. Solution to corrosion formation has been presented in the form of a system of partial differential equations describing both transitional and steady state behavior [5]. Such model is usually solved by some finite difference method in the form of dynamic simulation

Multicorp software is based on the work by Nesic and his collaborators at ICMT in modeling multiphase flow corrosion mechanisms. Solution to partial differential equations is obtained by in-house solver, which is now converted into CorrSim solver implemented in Fortran 90. Initial versions of Multicorp (MULTICRP V3 and MULTICORP V4) were implemented in VB6 as preprocessor and User-interface language. The work described in this paper relates to redesign of those earlier systems into efficient corrosion prediction tool using the modern software development technologies (object oriented modeling, XML, and MVC paradigm).

III. MULTICORP MODEL

Multicorp Model defines different types of physical and chemical environments through different classes from a hierarchical class model. The class model is created by taking advantage of inheritance property of any object oriented modeling where different sub-models inherit attributes and properties from their parent model. The entire Multicorp Model is depicted in the UML class diagram in figure 1. Brief descriptions of important models in the Multicorp are given below.

A. AbstractModel

The root of the model is an abstract class called *AbstractModel* which owns all general attributes and properties of every model in Multicorp. Two most important attributes of *AbstractModel* are two collections which store parameters and parameter groups for individual models. All other attributes can be classified into three different categories.

Identifier attributes – Identifier attributes store model specific identification data. *ModelName* stores the textual name of the model, *ModelID* stores the runtime instance number of the model, *ModelType* stores the type of model and *ParentModel* holds the textual name of the parent model.

UI attributes – UI attributes help model to connect to the GUI element, manages different user interaction in the model. Few examples of these type of attributes are *DisplayName* (textual name of the model shown on the UI), *mInstruction* (text to store instruction for a particular model), *ModelState* (state of the model at a certain time of the corrosion modeling process) and few integer flags to identify different states of the model during user interaction and execution.

Listener attributes – Every model registers to different listeners in Multicorp to either respond to any user interaction on GUI or any change in other models. Through listeners models work in a coherent and synchronized manner in Multicorp. All listeners are stored in a collection called *ModelListeners*.

AbstractModel also has the set of most generic properties. Most of these properties are either targeted to managing different attributes of the model notably parameters and parameter groups. There are a set of properties which are designated to read and write into XML files meant for persistent storage of data. Another set of properties perform all types of calculation in the model based on the values of parameters. The calculation properties are mainly overridden in sub models and different chemical equations are implemented which perform calculation to produce results specific to a particular model.

B. CompositionModel

Composition model is inherited from ChemistryModel which is in turn inherited from *Abstractmodel*. Chemistry model is responsible for storing contents of different chemical components of aqueous, gas and hydrocarbons in the system. Numerous chemical calculation is performed in this model to calculate the percentage of different ions in the system, concentration of gas and other chemicals and saturation level of pH and other hydrocarbons.

C. FlowModel

Flow model is responsible for capturing different metrics for defining the condition of various gas and liquid flow in the pipeline. *FlowModel* itself is an abstract class inherited from *AbstractModel*. *FlowModel* is further classified by two abstract classes such as, *AbstractGasFlow* for gas flow and *AbstractLiquidFlow* for liquid flow. Different flow parameters such as, viscosity, velocity, surface tension, percentage of mixture and superficial velocity is used to calculate not only the flow rate of the oil, water, or mixture in the pipeline but also flow pattern, stress on the pipeline, slug deposition rate and water wetting durations.

D. CondensationModel

CondensationModel is a special type of corrosion model which captures environmental parameters which affects the corrosion of the top of the pipeline. This class is inherited from *AbstractCondensationModel* which in turn is inherited from *AbstractModel*. *CondensationModel* captures pipe quality, outside environment variable for either land, ocean or air and insulation data. This model can calculate water and hydrocarbon condensation rate at top of the pipeline.

E. PipeLineModel

PipeLineModel is inherited from *ProtoPipeline* which is an abstract class extended from *AbstractModel*. *PipelineModel* lets user to define pipe topography. Pipelines can be layed in 100m sections with chosen inclination and declination. Along with the topography, every pipeline section also stores individual properties such as, diameter, roughness, thickness of wall, conductivity and one or more insulation layer and their properties.

F. CorrosionModel

CorrosionModel is the most important model of Multicorp because this model manages the simulation. As *Corrsim* is packaged in a Fortran DLL file, this model also manages native function calls to *Corrsim* dll. Multicorp application supports three types of simulations, single point, parametric and line.

Corrosion at a single point is simulated over time through *PointModel*, which is inherited from Abstract Model. Parametric simulation simulates corrosion over time by varying different parameters. Batch Model is mainly responsible for managing this type of simulation. In the end, *LineModel* manages line run, which simulates corrosion along the length of pipeline. Both *BatchModel* and *LineModel* class are inherited from *MultiplexModel* class which is inherited from *AbstractModel*.

G. Parameter Model

Multicorp Models stores data in different types of parameters. Parameters are tightly linked to user interface elements. Every types of parameter are inherited from *AbstractParameter* class (see *Figure 2*).

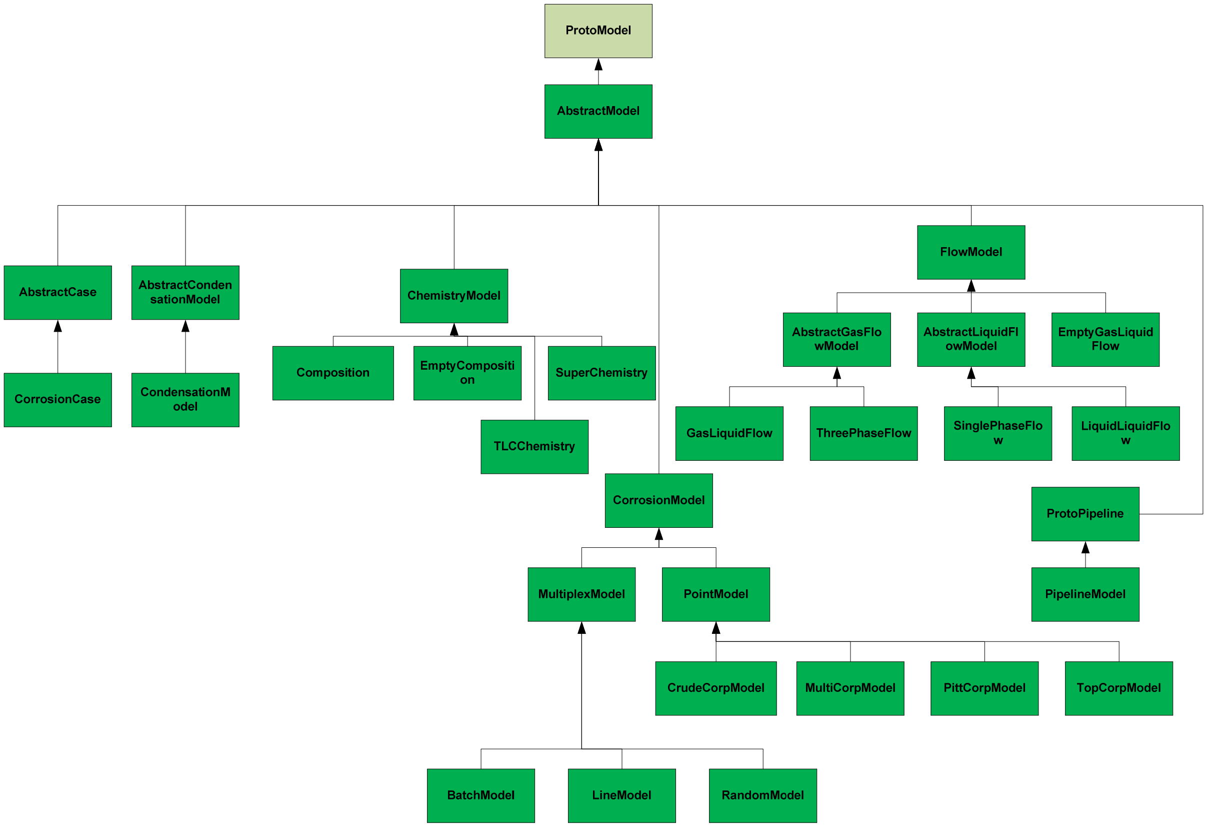


Figure 1: Class hierarchy diagram of Multicorp Model

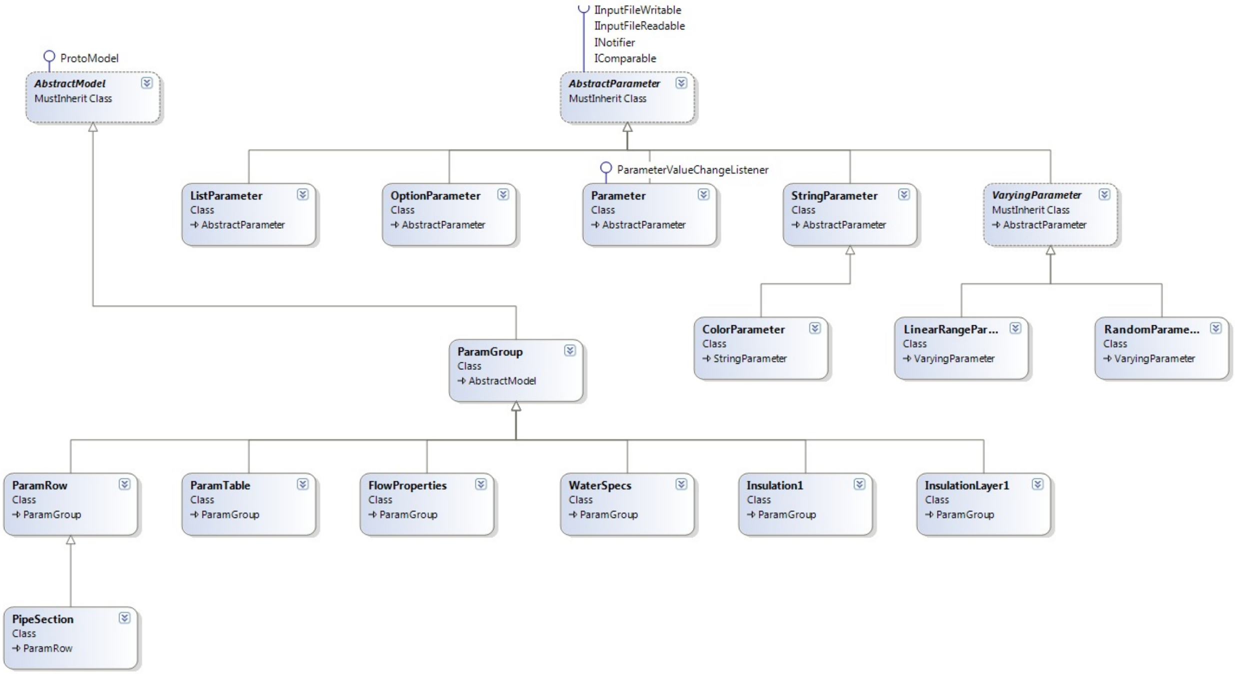


Figure 2: Multicorp Parameter model

Parameter class is the most used type of parameter which stores values of different types of chemical and physical species and samples. It is worth to mention that Multicorp supports multiple languages and unit conversion. Every parameter thus also has list of units it supports along with conversion factors. This enables user to change units of individual parameter separately on the fly. *Parameter* also stores two types of limits such as, hard limit and soft limit. For the reason that corrosion chemistry is always valid in a certain range of these parameters, user can be warned easily with the help of these limits, in case any value is entered which violates these limits.

Every model stores several parameters from same family under a parameter group. Parameter group class *ParamGroup* is also inherited from *AbstractModel* giving Multicorp model tremendous flexibility to store parameters either directly under a model or under a parameter group. Two special types of parameter groups are *ParamTable* and *ParamRow*, both inherited from *ParamGroup*. These two special Parameter groups are responsible for storing table data. Same principle is applied to store *PipeLine* topology data in *PipeLineModel* where each pipe section data is stored in individual *ParamRow*.

H. Multicorp Model association and dependency

Multicorp system instantiate different models to build a complete corrosion profile, based on the options chosen. For example, a corrosion model investigating bottom of the line corrosion for only water flow and simulating corrosion at single point will consist of instances of *CompositionModel*, *SinglePhaseFlowModel* and *PointModel*. *CorrosionCase*, inherited from *AbstractModel*, is also a model but it works as composite. Following classical composition pattern, where multiple similar objects are stored in a composite, *CorrosionCase* stores multiple instances of various model classes. As every model including *CorrosionCase* is inherited from same *AbstractClass*, any generic operation requested to CorrosionCase may also requested to all other objects in *CorrosionCase*. This is mainly useful for least common denominator type of operations such as, data load, calculation and save, where this composition model provides tremendous ease in development and object management. However, it is to be noted that in traditional composition model, Composite associates with its components individually by 'has a' relationship. In *CorrosionCase*, all models are stored in a collection of type *AbstractModel*. These collection stores instances of specific subtypes at runtime but *CorrosionCase* can perform any least common denominator type operation on any model stored in the collection without knowing which subtype it is calling the operation on.

I. Multicorp factories and prototypes

Multicorp implements abstract factories to return concrete instances of models casted into abstract parent class of the particular model. One factory is created for each of the main models such as; Composition, Flow, Condensation, Pipeline, Simulation, and *CorrosionCase* (see Figure 3). Every factory class is singleton and has a static function createModel, inherited from *AbstractFactory* which returns the instance of the model. Thus, user needs to know only the type of *AbstractFactory* to access *createModel* method of any other factory. In this way, complex conditional statements are abstracted in the factory class and concrete model classes are never exposed.

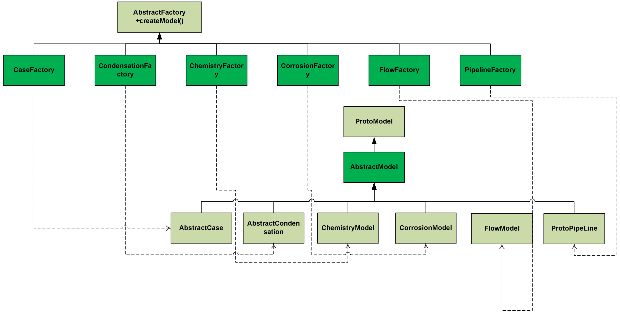
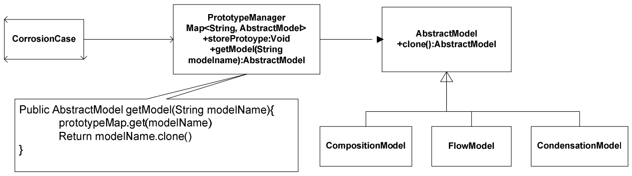


Figure 3: Abstract factories to instantiate Multicorp Model

Moreover, abstract factories don't create instance of any model directly. Instantiation process of any model is complex and heavy of memory as it includes I/O operations and XML query (explained in section IV). Therefore an ingenious object oriented design pattern, called prototype, is implemented to avoid repetitive execution of performance-heavy operations. Prototype is a classical creational pattern often used along with Abstract Factory. Prototype design pattern leverages on reusability of objects and saves memory and execution time [6]. *AbstractModel* class being the base class of all other model implements a pure virtual clone which is implemented by sub-models (See Figure 4). *ProtoManager* is a singleton class, which implements a collection to store instances of every model, which are called prototypes. When any model is requested by the corresponding factory, *ProtoManager* performs a deep clone on the prototype instance saved in the collection and returns it. Deep clone copies every attribute from the prototype to the clone but doesn't store any reference between them. Thus changes on the cloned instance doesn't impact prototype instances.



```
Public AbstractModel getModel(String modelName){
        prototypeMap.get(modelName)
        Return modelName.clone()
}
```

Figure 4: Prototype pattern in Multicorp

IV. DATA MANAGEMENT

Multicorp system models corrosion by making composite of different models. This is explained in Section III.H. As a desktop application Multicorp saves model data in an XML formatted file. One big reason for choosing XML as a formatting style is that XML is a well-formed document as well as extremely flexible to support any user defined schema [7]. In addition to DTD is a common practice to ensure the validity of the document, Multicorp needs set of parameters for each model with default values, units and limits besides just the structural integrity of the document. Therefore Multicorp uses a default XML document which stores all default values, descriptions, UI tags, units and limits for every parameter inside a well-planned XML node hierarchy which defines the composite structure of models. When Multicorp is asked to save the modelling data, it creates another XML document of type .mcinput, which follows the same node hierarchy as in default XML document and but stores only current values for individual parameters. As many non-changeable attributes of the parameters are always found from default XML document, .mcinput file doesn't store that information, resulting in much smaller file size.

Along with .mcinput Multicorp also saves simulation data and some temporary files along its process. They are listed below.

.mcorp - This is an archive file which contains other Multicorp files , which are

- .mcinput - Stores model data
- .input - FORTRAN generated pre simulation data
- .output – FORTRAN generated simulation results
- .mccase – Multicorp generated case file to execute in CorrSim project.

.tmpinput – A binary file of type .dat which contains object serialized data. This is used to save state of CorrosionCase temporarily.

The entire strategy of handling different files is explained as a flow chart in *Figure 5*.

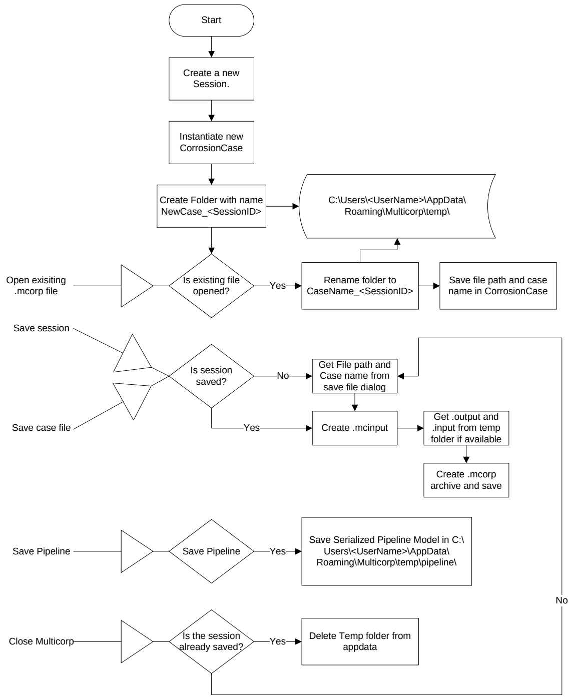

Start
Create a new Session.
Instantiate new CorrosionCase
Create Folder with name NewCase_<SessionID>
C:\Users\<UserName>\AppData\Roaming\Multicorp\temp\
Open exisiting .mcorp file
Is existing file opened?
Yes
Rename folder to CaseName_<SessionID>
Save file path and case name in CorrosionCase
Save session
Is session saved?
No
Get File path and Case name from save file dialog
Save case file
Yes
Create .mcinput
Get .output and .input from temp folder if available
Create .mcorp archive and save
Save Pipeline
Save Pipeline
Yes
Save Serialized Pipeline Model in C:\Users\<UserName>\AppData\Roaming\Multicorp\temp\pipeline\
No
Close Multicorp
Is the session already saved?
Yes
Delete Temp folder from appdata

Figure 5: Multicorp file management strategy

Multicorp system is going through rapid evolution. In near future Multicorp system is going to be available through service APIs in near future. Moreover, work is also underway for transforming Multicorp into multi-tier web based product. Multicorp data architecture supports multiple types if data sources and storage system besides XML. To facilitate this, *AbstractModel* implements variable of type i. *DataHandler* is an interface which contains pure virtual methods such as, saveModel and loadModel. This interface is extended by concrete data handlers such as, XMLHandler or *DatabaseHandler* as displayed in *Figure 6*. Concrete handlers implement complex data management operations specific to the data targets. However, models don't need to know those specifics of data management and can simply call *saveModel* or *loadModel* method on the handler variable to load and save data.

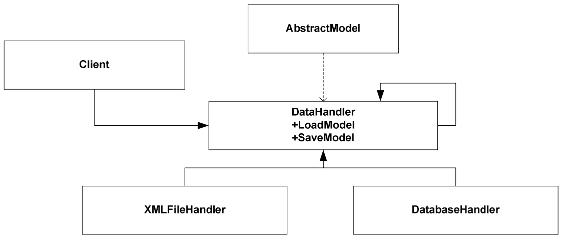


Figure 6: Abstracting data handling in Multicorp

V. System Architecture

In this section different design aspects of the overall system are discussed.

A. User Interface

Multicorp desktop application is built in conventional model-view-controller architecture. Multicorp model, discussed in section III, manages the data. Different view element depends on various controller classes, either custom or member of .net framework. The smooth interaction between GUI elements and data models is established by implementation of observer pattern, in which event raised by any GUI element is broadcasted to all observers registered to the event sender. For example if user changes a value of a parameter in GUI, every model listening to the event, will trigger the corresponding action (See Figure 7).

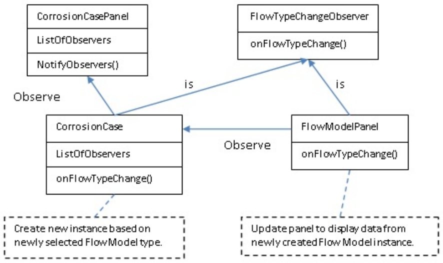


Figure 7: Observer pattern in Multicorp

This particular pattern lets the execution control transition from view to model seamlessly over a loose coupling between listener and sender [6]. The user interface of Multicorp is built closely following Microsoft Office's ribbon style [8].The interface is separated in four resizable areas, such as, ribbon area (top), process area (left), data area (middle) and trace area (bottom) (See Figure 8). Ribbon area contains buttons, which triggers generic actions on Multicorp models. Every time a new tab is selected buttons are changed depending on the tab content. Process area shows the steps to create a corrosion model and also displays the status of every model. Data area displays parameters in groups, charts, tables and various other output elements. In the end trace area displays vital messages including warning and errors at the time of corrosion modeling.

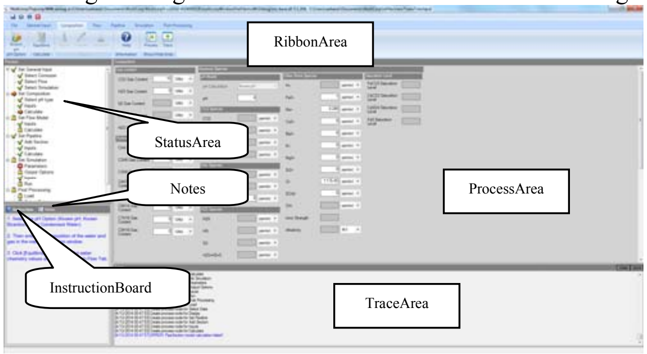


Figure 8: Multicorp user interface

In Figure 9 the loading sequence of Multicorp user interface components is explained in a sequence diagram. In the sequence diagram, it is shown that MulticorpWindowApp is the starting point of execution, which in turn instantiates and run MulticorpForm in a separate UI thread. *MulticorpForm* extends .NET library class for UI container called *Form*. *MulticorpForm* instantiates and build different panels, namely, RibbonArea, MiddleArea and StatusArea inside it sequentially. MiddleArea is the container for different other panels; such as, ProcessArea, TracePanel, and InstructionBoard.

B. Dynamic Modeling

One of the unique features of Multicorp is to create additional models on the fly addition to the existing models. Every model of Multicorp, when instantiated to be part of a corrosion case, they are stored in a hash table as key value pair, where key is the name of the model. As explained in section III.H, when user request an action performed on a specific model, corresponding model is retrieved from the hash table and subsequent operation is performed on the model. This is possible because every generic function is owned by *AbstractModel*. The most important operation to be performed on any model is calculation of corrosion and various data. Models are capable of calculating itself too. This calculation is done based on the input parameters. Any new model created on the fly needs to know its parameters. This can be achieved by defining parameters and storing them in the parameter hash table of the new model. Next the model need to know implement a calculate function. The calculate method is not a part of *AbstractModel* but an interface called *ICalculate*. When a new model is created, a concrete implementation of *ICalculate* is supplied to the model. As the new model is extended from *AbstractModel* and *AbstractModel* has a dependency on the *ICalculate*, when user requests calculate operation on the model, the calculate method from the concrete calculation class is called. Moreover, every model is capable of storing multiple instances of subclass of *ICalculate*. At any point of time, only one instance is set active, however, it gives tremendous flexibility to users because alternative calculation logic can be implemented and compared without rewriting same block of code.

C. Multicorp build strategy

Multicorp is distributed in four different versions. They are Standard(CC-JIP), Water-Wetting(WW-JIP), Topcorp(TLC-JIP) and Topcorp-Water-Wetting(TLC-WW-JIP). Standard version provides basic features with other version including extra features on top of it. To facilitate this segregation, Multicorp is built in different functionally segregated modules (see Figure 10). Every module is compiled into separate DLL file. This type of modularized development helps Multicorp to package only required DLLs for a particular version and distribute as installable.

Corrsim is a separate FORTRAN development and has its own modular structure.

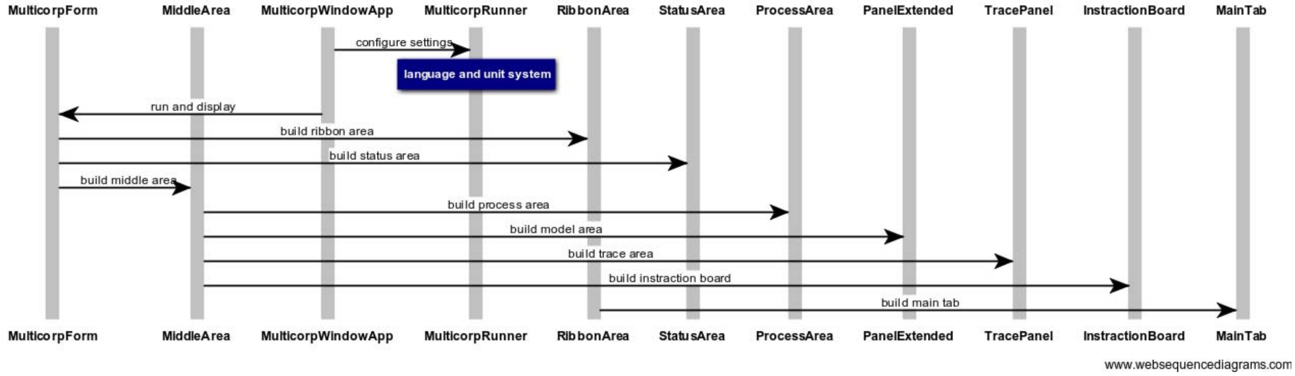


Figure 9: Sequence diagram of Multicorp GUI building process

After that compilation dependency, configured in Multicorp solution, links *Corrsim* DLL to Multicorp DLLs as reference. In that way, Multicorp function can make direct call to *Corrsim* functions.

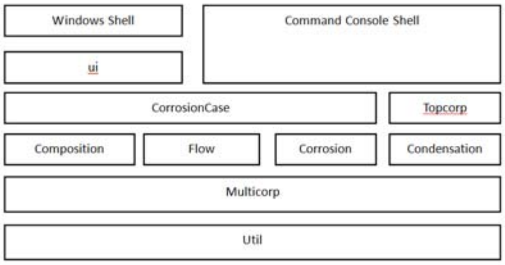
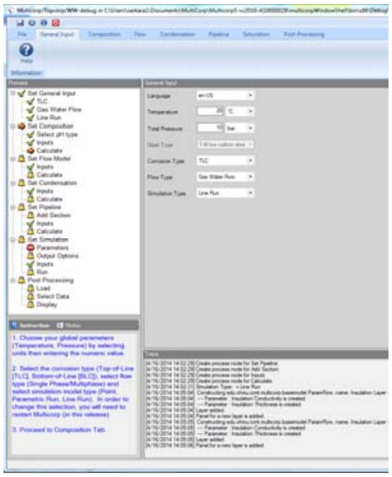


Figure 10: Package dependencies in Multicorp



Language	en-US
Temperature	25 °C
Total Pressure	10 bar
Steel Type	[illegible]
Corrosion Type	TLC
Flow Type	Gas Water Flow
Simulation Type	Line Run

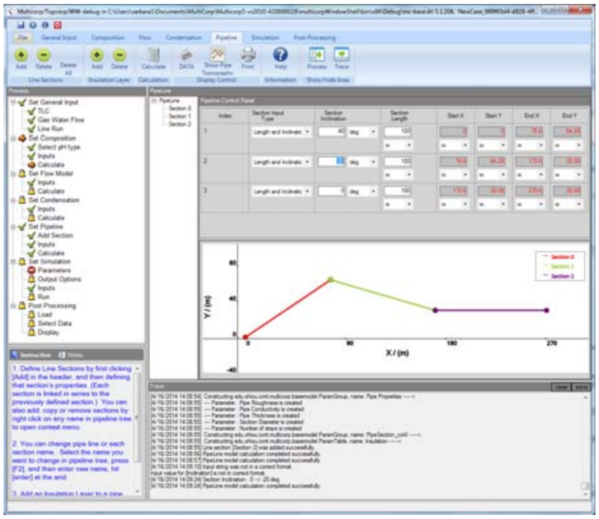
1. Choose your global parameters (Temperature, Pressure) by selecting units then entering the numeric value.

2. Select the corrosion type (Top-of-Line [TLC], Bottom-of-Line [BLC]), select flow type (Single Phase/Multiphase) and select simulation model type (Point, Parametric Run, Line Run). In order to change this selection, you will need to restart Multicorp (in this release).

3. Proceed to Composition Tab.

```
[4/16/2014 14:02:29] Create process node for Set Pipeline
[4/16/2014 14:02:29] Create process node for Add Section
[4/16/2014 14:02:29] Create process node for Inputs
[4/16/2014 14:02:29] Create process node for Calculate
[4/16/2014 14:02:31] Simulation Type : > Line Run
[4/16/2014 14:05:04] Constructing edu.ohiou.icmt.multicorp.basemodel.ParamRow, name: Insulation Layer
[4/16/2014 14:05:04] --- Parameter : Insulation Conductivity is created
[4/16/2014 14:05:04] --- Parameter : Insulation Thickness is created
[4/16/2014 14:05:04] Layer added
[4/16/2014 14:05:04] Panel for a new layer is added.
[4/16/2014 14:05:05] Constructing edu.ohiou.icmt.multicorp.basemodel.ParamRow, name: Insulation Layer
[4/16/2014 14:05:05] --- Parameter : Insulation Conductivity is created
[4/16/2014 14:05:05] --- Parameter : Insulation Thickness is created
[4/16/2014 14:05:05] Layer added
[4/16/2014 14:05:06] Panel for a new layer is added.
```

Figure 11: Multicorp screen for configuring various models



Index	Section Input Type	Section Inclination		Section Length	Start X	Start Y	End X	End Y
1	Length and Inclinatio	40	deg	100	0	0	76.6	64.28
2	Length and Inclinatio	[illegible]	deg	100	76.6	64.28	170.6	30.08
3	Length and Inclinatio	0	deg	100	170.6	30.08	270.6	30.08

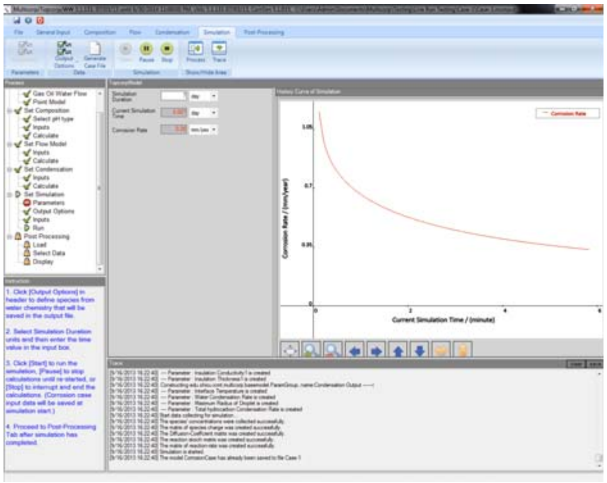
1. Define Line Sections by first clicking [Add] in the header, and then defining that section's properties. (Each section is linked in series to the previously defined section.) You can also add, copy or remove sections by right click on any name in pipeline tree to open context menu.

2. You can change pipe line or each section name. Select the name you want to change in pipeline tree, press [F2], and then enter new name, hit [enter] at the end.

3. Add an Insulation Layer to a new

```
[4/16/2014 14:08:54] Constructing edu.ohiou.icmt.multicorp.basemodel.ParamGroup, name: Pipe Properties ----<
[4/16/2014 14:08:55] --- Parameter : Pipe Roughness is created
[4/16/2014 14:08:55] --- Parameter : Pipe Conductivity is created
[4/16/2014 14:08:55] --- Parameter : Pipe Thickness is created
[4/16/2014 14:08:55] --- Parameter : Section Diameter is created
[4/16/2014 14:08:55] --- Parameter : Number of steps is created
[4/16/2014 14:08:55] Constructing edu.ohiou.icmt.multicorp.basemodel.ParamGroup, name: PipeSection_col4 ----<
[4/16/2014 14:08:55] Constructing edu.ohiou.icmt.multicorp.basemodel.ParamTable, name: Insulation ----<
[4/16/2014 14:08:55] Line section [Section 2] was added successfully.
[4/16/2014 14:08:56] PipeLine model calculation completed successfully.
[4/16/2014 14:08:57] PipeLine model calculation completed successfully.
[4/16/2014 14:09:10] Input string was not in a correct format.
Input value for [Inclination] is not in correct format.
[4/16/2014 14:09:24] Section Inclination : 0 -> -20 deg
[4/16/2014 14:09:24] PipeLine model calculation completed successfully.
```

Figure 12 Multicorp screen for defining the pipeline topology



Simulation Duration		day
Current Simulation Time	0.00	day
Corrosion Rate	0.38	mm/year

1. Click [Output Options] in header to define species from water chemistry that will be saved in the output file.

2. Select Simulation Duration units and then enter the time value in the input box.

3. Click [Start] to run the simulation, [Pause] to stop calculations until re-started, or [Stop] to interrupt and end the calculations. (Corrosion case input data will be saved at simulation start.)

4. Proceed to Post-Processing Tab after simulation has completed.

```
[9/16/2013 16:22:40] --- Parameter : Insulation Conductivity1 is created
[9/16/2013 16:22:40] --- Parameter : Insulation Thickness1 is created
[9/16/2013 16:22:40] Constructing edu.ohiou.icmt.multicorp.basemodel.ParamGroup, name:Condensation Output ----<
[9/16/2013 16:22:40] --- Parameter : Interface Temperature is created
[9/16/2013 16:22:40] --- Parameter : Water Condensation Rate is created
[9/16/2013 16:22:40] --- Parameter : Maximum Radius of Droplet is created
[9/16/2013 16:22:40] --- Parameter : Total hydrocarbon Condensation Rate is created
[9/16/2013 16:22:40] Start data collecting for simulation...
[9/16/2013 16:22:40] The species' concentrations were collected successfully.
[9/16/2013 16:22:40] The matrix of species charge was created successfully.
[9/16/2013 16:22:40] The Diffusion-Coefficient matrix was created successfully.
[9/16/2013 16:22:40] The reaction stoich matrix was created successfully.
[9/16/2013 16:22:40] The matrix of reaction-rate was created successfully.
[9/16/2013 16:22:40] Simulation is started.
[9/16/2013 16:22:40] The model CorrosionCase has already been saved to file Case 1
```

Figure 13 Multicorp screen showing dynamic simulation

VI. Demo and Performance Analysis

Multicorp has been tested with two groups of corrosion engineers: faculty and graduate students at ICMT (15-20 engineers), and ICMT industrial partners (25-30 engineers) in order to finalize it GUI to correspond to the ways that corrosion engineers would use to predict corrosion rates. Resulting user interfaces are shown in few figures. Figure 11 shows the general input screen where user configures various models that need to be calculated, by selecting select corrosion, flow and simulation type. All other tabs are dynamically generated once three selections are made. It is to be noted that the process tree shows the all steps required before simulation may be performed. Currently available tab and steps possible in that tab are indicated with a red arrow and all other tabs which are not available are marked as locked. The screen in Figure 12 show how user defines line topology in pipeline tab which only appears when line run is selected as simulation type. An interactive pipeline modeler helps user to define pipeline over an intended topology. The screen in Figure 13 shows the simulation tab while user is running a simulation and can monitor its progress in a dynamic plot.

Multicorp is also tested for its computationally efficiency and it has shown order of 10x speed up over previous version. In addition it is continuously been validated against experimental results obtained in ICMT labs and from its industrial partners.

References

[1] S. Nešić, “Key issues related to modelling of internal corrosion of oil and gas pipelines – A review,” Corros. Sci., vol. 49, no. 12, pp. 4308–4338, Dec. 2007.

[2] C. de Waard, U. Lotz, and D. E. Milliams, "Predictive Model for CO 2 Corrosion Engineering in Wet Natural Gas Pipelines," Corrosion, vol. 47, no. 12, pp. 976–985, Dec. 1991.

[3] NORSOK, "M-506 CO2 corrosion rate calculation model." NORSOK, p. Rev. 2, 2005.

[4] Wood Group Intetech, "Electronic Corrosion Engineer." 2013.

[5] S. Nešić, H. Li, J. Huang, and D. Sormaz, "An open source mechanistic model for CO2/H2S corrosion of carbon steel," in NACE International Corrosion Conference & Expo, 2009.

[6] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design patterns: elements of reusable object-oriented software. Pearson Education, 1994.

[7] S. Holzner, Inside XML. Indianapolis, Indiana: New Riders Publishing, 2001.

[8] D. You, "Re-engineering of the Legacy Software Systems by using Object-Oriented Technologies," Ohio University, 2013.

A multi-step process mining approach based on the markov transition matrix

A. Hong Li[1], B. Kunman Li[2], and C. Lin Liu[3], and D. Shaowen Yao[4]
[1]School of Information Science and Engineering, Yunnan University, Kunming, Yunnan, China
[2] School of Computing and Engineering, University of Huddersfield, Queensgate, Huddersfield
[3] School of information, Yunnan Normal University, Kunming, Yunnan, China
[4] National Pilot School of software, Yunnan University, Kunming, Yunnan, China

Abstract - *According to the situation that many workflow instances may deviate from the predefined model, this paper proposed a new process mining approach based on analyzing the workflow log to realize the workflow process reconstruction. First, build the markov transition matrix based on the workflow log, then design a multi-step process mining algorithm to mine the structural relationships between the activities, finally, this paper verifies the feasibility and applicability of the approach through a simulation example.*

Keywords: process mining; markov transition matrix; process reconstruction

1 Introduction

With the development of information technology, more and more enterprises use workflow management systems (Workflow Management System, WFMS) to realize business process automated execution both inside and outside the enterprise. WFMS is driven by a predefined workflow model. Once the workflow model is been established, it usually won't change in a very long period of time. However, due to the changing environment, the workflow instances which are the actual execution processes of the workflow often change. For example, during the process execution, companies will confer certain privileges to add executives or skip some of the nodes, even reset the given process transfer conditions, over time, the actual implementation of workflow instances will deviate from the predefined workflow model. If this change has not been detected in the long time, and the workflow instance is still driven by the old predefined model, then the efficiency of enterprises must have been affected. Based on this consideration, in order to improve the accuracy of workflow execution, modelers must be regularly informed of the actual implementation of the workflow, reconstruction or recommend better workflow model which reflects the actual needs of enterprises. In real life, each execution of the workflow instances is recorded in the WFMS log database in the form of workflow logs, which provide a solid data foundation for analyzing the changes in real execution workflow instance. In recent years, there exists a new process modeling idea - process mining, which extracts the structural description from the execution process collection [1]. Process mining is a new application of data mining, which is reproducing the real process of the business through analyzing workflow execution logs [2].

2 Process mining algorithms

The idea of process mining first appeared in the field of software engineering, was proposed by Joan-than E.Cook in 1995 [3]. In 1998, Agrawal first applied process mining technology to enterprise business process modeling [1]. Based on the mining process, workflow model reconstruction can be divided into single-step reconstruction and multi-step reconstruction [4]. Single-step reconstruction method is directly mined the workflow model based on the activities dependencies which explicit exists in the log. The single-step reconstruction algorithms include directed graph -based mining methods and WF-net based α algorithms proposed by Aalst. α algorithm is one of the classic algorithms of process mining, the mining process is simple, and the computation time is short, but the log noise handling capacity is insufficient, which is not suitable for mining complex structures workflow model such as non-free choice, loop, hidden activities and duplication activities [5]. A lot of research has been done on the improving of α algorithm, such as α* algorithm and β algorithm [6-7]. Multi-step reconstruction method adds the workflow log preprocessing or workflow model evaluation process, which makes the mining with higher accuracy, but the algorithm execution time is longer. Multi-step reconstruction methods include region-based mining method, clustering based mining method, genetic algorithm based mining method, frequency/dependency based mining method, multi-model mining and incremental mining methods [4, 8-15]. These algorithms all required of the integrity of the workflow model, which means that the mined workflow model needs to meet all the log, so the workflow model mined by these algorithms always with lower accuracy.

Process mining has attracted wide attention in academic domain, but there still lack of the algorithms with strong effectiveness for a wide range, good robustness and high efficiency. Some of the algorithms can only identify simple business process structure from the log due to the constraints of formal representation capacity, and some of the mining algorithms have higher requirements of the log, which is difficult to handle the incomplete information or the noise. Consider of this situation, we designed a multi-step process mining method that first analyzes the workflow log transition probabilities between activities to build the Markov transition matrix, then mines the process logic relationship by defining a set of rules of logical relational mining, finally, we design the process mining algorithm to establish the actual structure relationship between the activities in order to reconstruct the workflow. At last, we use an elevator company's standard contract process for example to verify the feasibility and applicability of the method.

3 Design of multi-step process mining method

In the enterprise business execution process, the workflow model can be seen as a stochastic process from one activity to another activity state which is a kind of Markov process or Markov chain. Because of the limited activity states, and the process activity to be performed in time “t+1” only related with the activity state in time “t”, so it is a finite-state Markov chain. This paper represents this process in $\{\mathrm{x(t)}, \mathrm{t} \in \mathrm{T}\}$, x(t) is the process state in time “t”, p_{ij} is the one step transfer probability from x_i to x_j, and p_{ij} has no relationship with the time “t”.

$$p_{ij} = \mathrm{p}\{\mathrm{x(t+1)} = x_j | \mathrm{x(t)} = x_j\} \; (x_i, x_j \in X) \qquad (1)$$

The matrix consists of all the transfer probability is called the transition probability matrix.

This paper puts forward a multi-step process mining algorithm based on the Markov transition matrix, and the algorithm process is shown in Fig.1.

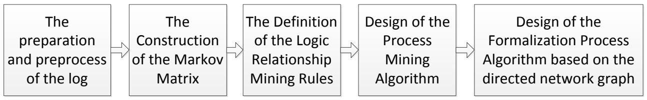

The preparation and preprocess of the log
The Construction of the Markov Matrix
The Definition of the Logic Relationship Mining Rules
Design of the Process Mining Algorithm
Design of the Formalization Process Algorithm based on the directed network graph

Fig.1. The main steps of the algorithm

3.1 Preparation and preprocess of the log

Workflow log usually records the actual implementation process of the workflow model, and the log usually made up with the workflow instance name ‘Case_id’, activity name ‘Activity’, performer ‘Performer’ (can be specific people, or the application program) , and execution time ‘Time’, etc. Among these, ‘Case_id’ used to identify execute times of one workflow, such as Case_1, Case_2, ...; ‘Activity’ used to identify a specific activity of the workflow process, like x0, x1, ...; ‘Performer’ and ‘Time’ are used to represent the specific actors and execution time of the activity.

Because of the input errors, there may be records missing, duplicate records and other reasons which may cause the noise or workflow logs incomplete. For the log with noise, it can do the noise filtering by setting the frequency threshold ‘θ’. For incomplete logs, it can be filtered using the following two methods: a) list the sets of the end events of the log, if an instance’s end event does not belong to the set, then the instance logs are incomplete and should be removed; b) in the log, if a task is only has the start event without a corresponding end event, or only has the end event without a corresponding start event, then the instance is also incomplete and should be removed.

3.2 Construction of the markov matrix

The main steps are as follows:

Step 1: Compute the workflow instance numbers based on the log, which is means the executions times of workflow, make it ‘n’.

Step 2: Analyze the workflow instance, statistic all of the possible activity, i.e. X ={x_0 x_1…x_k}.

Step 3: Compute the transition possibility between the activity, make is ‘p_{ij}= m_{ij}/n’, m_{ij} means the one step transition times from activity x_i to x_j in the n workflow instances.

Step 4: Establish the workflow transition matrix P.

3.3 Definition of the logic relationship mining rules

Workflow process model usually consists of sequential structure, And-split/join, OR-split/join, loop structure (self-loops and multi-step cycle), which is composed of a variety of structural and process logic, includes sequence relationship, causal relationship, selection relationship, synchronous relationship and circular relationship, respectively use symbol ">", "→", "#", "//", "◇" to represent. This paper aims to derive the logical relationship in the workflow process through the analysis of the transition matrix P. X_S represents the start of the process, X_E represents the end of the process, X represents the sets of process nodes, x_i represents the activity node in the process, p_{ij} represents the transition possibility between process nodes x_i and x_j.

The logic relationship mining rules are as follows:

(a) *Rule1: Identify the start of the process X_S,*
$\exists \mathrm{x_j} \in \mathrm{X}, \text{and } \forall \mathrm{x_i} \in \mathrm{X}, \text{if } \mathrm{p_{ij}} = 0, \text{then } \mathrm{X_S} = \mathrm{x_j}$

(b) Rule2: Identify the end of the process X_E
$\exists \mathrm{x_i} \in \mathrm{X}, \text{and } \forall \mathrm{x_j} \in \mathrm{X}, \text{if } \mathrm{p_{ij}} = 0, \text{then } \mathrm{X_E} = \mathrm{x_i}$

(c) *Rule3: Mining rules of the sequence relationship,*
$\forall \mathrm{x_i}, \mathrm{x_j} \in \mathrm{X}, \text{if } \mathrm{p_{ij}} \neq 0, \text{then } \mathrm{x_i} > \mathrm{x_j}$

(d) Rule4: Mining rules of the causal relationship
$\forall \mathrm{x_i}, \mathrm{x_j} \in \mathrm{X}, \text{if } \mathrm{p_{ij}} = 1, \text{and } \mathrm{x_i} \neq \mathrm{x_j} \text{then } \mathrm{x_i} \rightarrow \mathrm{x_j}$

(e) *Rule5: Mining rules of the synchronous relationship (And-Split, And-join),* $\forall \mathrm{x_i}, \mathrm{x_{\eta_k}} \in \mathrm{X}$,
Rule 5.1:

if $p_{i\eta_1} = p_{i\eta_2} = \ldots = p_{i\eta_k} = \frac{1}{k} \neq 0$, and $x_i \neq x_{\eta_k}$

then$\{x_{\eta_1} // x_{\eta_2} //..// x_{\eta_k}, x_i > x_{\eta_k}\}$,$x_i$ is the And-Split activity node.

Rule 5.2:

if $p_{\eta_1 i} = p_{\eta_2 i} = \ldots = p_{\eta_k i} = \frac{1}{k} \neq 0$, and $x_i \neq x_{\eta_k}$,

then$\{x_{\eta_1} // x_{\eta_2} //\ldots// x_{\eta_k}, x_{\eta_k} > x_i\}$, x_i is the And-Join activity node.

(f) *Rule6: Mining rules of the selection relationship (OR-Split, OR-join)*

$\exists x_i, x_{\eta_k} \in X,\ p_{i\eta_k} > 0$
if $\sum_k p_{ij} = 1$,
then $\{x_{\eta_1} \# x_{\eta_2} \#..\# x_{\eta_k}, x_i > x_{\eta_k}\}$, x_i is the OR-Split activity node.
$\exists x_i, x_{\eta_k} \in X, s_i \in S$, and $x_{\eta_1} \# x_{\eta_2} \#..\# x_{\eta_k}, x_i > x_{\eta_k}$

```
for (i≦k, i=0, i++)
{
    if $\exists s_i \in S$ and $x_i > x_{\eta_k}$
    then $s_i$ is the execution condition of the OR-split
    $x_i > x_{\eta_k}$
}
```

$\exists x_i, x_{\eta_k} \in X,\ p_{\eta_k i} > 0$,
if $\sum_k p_{\eta_k i} = 1$,
then $\{x_{\eta_1} \# x_{\eta_2} \# \ldots \# x_{\eta_k},\ x_{\eta_k} > x_i\}$, x_i is the OR-Join activity node.

(g) Rule7: Mining rules of the self-circulation relationship
$\exists x_i \in X, x_{\eta_k} \in X - \{x_i\}$,
if $p_{ii} = p\ (p \neq 1)$, then $\{\Diamond x_i;\ x_i > x_{\eta_k}\}$

(h) Rule8: Mining rules of the multi-step cycle relationship

Rule8.1: Mining the multi-step cycle activity node
$\forall x_i, x_{\eta_k} \in X, x_i \neq X_S, x_i \neq X_E$ and $x_j \neq X_E, X_{re} \neq \emptyset$

if $p_{ij} = 0, p_{ji} > 0\ and\ \exists k (k \leq n-1)$,
$then\ p_{ij}^{(k)} > 0\ (p_{ij}^{(k)} \in p^k)$
then {there is multi-step cycle between x_i and x_j, and $x_j > x_i$; $X_{re} = X_{re} + \{x_i, x_j\}$; }
Refunction()

The main steps of function *Refunction()* are as follows:

```
Do {
for ($\eta \leq n+1, \eta = 0, \eta++$)
{if $p_{i\eta}^{(k-1)} > 0\ and\ p_{ij} > 0$
 then $\{X_{re} = X_{re} + \{x_\eta\};\ x_\eta > x_j\}\}$
$k = k-1; j = \eta$
}
```

While ($k \geq 2$) the activities in X_{re} has cycle relationship, i.e. $x_j \Diamond x_i \Diamond x_{\eta_1} \Diamond \ldots \Diamond x_{\eta_{k-3}}$

Rule8.2: Identify the start X_{rS} and end X_{rE} of the multi-step cycle

$if\ \exists x_{\eta_k} \in X_{re}, x_i \in X - X_{re}\ and\ p_{i\eta_k} > 0$
$then\ \{X_{rS} = x_{\eta_k};\ x_i > X_{rS}\}$
$if\ \exists x_{\eta_k} \in X_{re}, x_i \in X - X_{re}\ and\ p_{\eta_k i} > 0$
$then\ \{X_{rE} = x_{\eta_k};\ X_{rE} > x_i\}$

3.4 Design of the process mining algorithm

Based on the workflow process logical relationship mining rules Rule1~Rule8, we design the workflow mining algorithm Process (P, X). The algorithm can reconstruct the structural relationship between all the activities, and establish the 'W', 'W_{and}' and 'W_{select}' set.

Input: the Markov matric P, the activities set X, $X = \{x_0, x_1 \ldots x_n\}$

Output: W is the set of the activities which have causal relationship, sequence relationship and cycle relationship; $W = \{(x_i, x_j) | x_i \rightarrow x_j$ or $x_i > x_j$ or $x_i \Diamond x_j\}$; W_{and} is the set of the activities which have synchronous relationship, $W_{and} = \{(x_{\eta_1}, x_{\eta_2}, \ldots) | x_{\eta_1} // x_{\eta_2} // \ldots\}$; W_{select} is the set of the activities which have selection relationship, $W_{select} = \{(x_{\eta_1}, x_{\eta_2}, \ldots) | x_{\eta_1} \# x_{\eta_2} \# \ldots\}$;

The main steps of the mining algorithm $Process\ (P, X)$ are as follows:

```
    Initialization:
        $W = \emptyset, W_{and} = \emptyset, W_{select} = \emptyset$
    Step 1:
for ($i \leq n+1, i = 0, i++$)
    {
        $\forall x_i \in I$,
        $If\ Rule1, then\ X_S = x_i$;
        $If\ Rule2, then\ X_E = x_i$;
    }
    Step 2:
for ($i \leq n+1, i = 0, i++$)
for ($j \leq n+1, j = 0, j++$)
    {
        $if\ Rule3, then\ W = W + \{(x_i, x_j) | x_i > x_j\}$;
        $if\ Rule4, then\ W = W + \{(x_i, x_j) | x_i \rightarrow x_j\}$;
    }
    Step 3:
for ($i \leq n+1, i = 0, i++$)
    {
        $if\ \exists x_{\eta_1}, x_{\eta_2}, \ldots x_{\eta_k} \in X,\ and\ Rule\ 5.1$
        $then\ \{W_{and} = W_{and} + \{(x_{\eta_1}, x_{\eta_2}, \ldots x_{\eta_k}) | x_{\eta_1} // x_{\eta_2} // \ldots x_{\eta_k}\},\ W = W + \{(x_i, x_{\eta_k}) | x_i > x_{\eta_k}\}\ \}$;
        if $\exists x_{\eta_1}, x_{\eta_2}, \dots x_{\eta_k} \in X$, and Rule 5.2
        then {$W_{and} = W_{and} + \{(x_{\eta_1}, x_{\eta_2}, \dots x_{\eta_k}) | x_{\eta_1} // x_{\eta_2} // \dots x_{\eta_k}\}$, $W = W + \{(x_{\eta_k}, x_i) | x_{\eta_k} > x_i\}$};
        if $\exists x_{\eta_1}, x_{\eta_2}, \dots x_{\eta_k} \in X$, and Rule 6
        then
        {$W_{select} = W_{select} + \{(x_{\eta_1}, x_{\eta_2}, \dots x_{\eta_k}) | x_{\eta_1} \# x_{\eta_2} \# \dots \# x_{\eta_k}\}$,
        $W = W + \{(x_{\eta_k}, x_i) | x_{\eta_k} > x_i\}$};
        if Rule 7
        then { $W = W + \{(x_i, x_i) | \diamond x_i\} + \{(x_i, x_{\eta_k}) | x_i > x_{\eta_k}\}$ }
      }
        Step 4:
  for ($i \leq n + 1, i = 0, i++$)
  for ($j \leq n + 1, j = 0, j++$)
    {
        if Rule 8.1
        then
        $W = W + \{(x_i, x_{rS}) | x_i > X_{rS}\} + \{(x_{rE}, x_j) | x_{rE} > x_j\} + \{(x_{\eta_{k-1}}, x_{\eta_k}) | x_{\eta_{k-1}} \diamond x_{\eta_k}\}$
    }
```

4 Design of the formalization process algorithm

The previous mining algorithm results only can be described in a certain formal description which limits the application scope of the model. Based on this situation, this paper separates the process mining algorithm and process formalization algorithm, which can express the process mining result in many different formal process models according to the different mining needs. This paper put forward a process formalization algorithm *DirectedNet(X, W') based on* directed network graph which describe the mining result in directed network graph.

Input: The process activities set X, $\mathrm{X} = \{\mathrm{x}_0, \mathrm{x}_1 \dots \mathrm{x}_\mathrm{n}\}$

$$\mathrm{W}' = \mathrm{W} + \mathrm{W}_{\mathrm{and}} + \mathrm{W}_{\mathrm{select}} \tag{2}$$

Output: The directed network graph model of the process

```
  $DirectedNet\ (X, W')$:
    Do
    t=0
    { for ($j \leq n + 1, j = 0, j++$)
    {if $(X_S, X_j) \in W'$, and $X_S \neq X_j$
     then {direct $X_S$ and $X_j$ use arrow,
                    $X_S \rightarrow X_j$ or $X_S >* X_j$}
    t=t+1;
    $W' = W' - \{(X_S, X_j)\}$; $I_j = I_j + \{x_j\}$}
    if $(X_S, X_j) \in W'$, and $X_S = X_j$
     then {direct $X_S$ and $X_j$ use double arrow,
    there is self
    − circulation relationship between $X_S$ and $X_j$ }
    t=t+1;
    $W' = W' - \{(X_S, X_j)\}$; $I_j = I_j + \{x_j\}$
    }}
    if $t = 1$ then $X_S = X_j$,
    there is only one node $X_j$ connect with $X_S$
    if $t = 1$ then { Do $x_j = I_j$; $X_S = x_j$
    $DirectedNet\ (X, W')$
    $I_j = I_j - \{x_j\}$
    While $I_j \neq \emptyset$
    }
    While $X_S = X_E$
```

The process of the algorithm $\mathrm{DirectedNet}\,(\mathrm{X}, \mathrm{W}')$ is a cyclic process, which use self-circulation method to connect the active nodes in W'.

The algorithm $DirectedNet\,(X, W')$ first direct the start node X_S and the other nodes which has logical relationship with X_S, and then based on the known logical relationship between activities, connect all the activity nodes use a single or double arrow until complete the whole direct network graph.

5 Simulation

To verify the feasibility of the method, this paper chooses a WFMS process ('Pro') log from one enterprise WFMS as the simulation example (see Table 1).

The log has 52 instances in total, Table. 1 only lists the instance of Case 1~ Case 4.

TABLE.1 THE WORKFLOW LOG OF 'PRO'

Ca_id	Ac	Ca_id	Ac	Ca_id	Ac	Ca_id	Ac
Case_1	A	Case_2	D	Case_2	E	Case_2	J
Case_1	B	Case_3	D	Case_1	H	Case_4	E
Case_2	A	Case_1	F	Case_1	J	Case_3	G
Case_3	A	Case_1	F	Case_3	E	Case_4	F
Case_1	C	Case_4	B	Case_2	F	Case_3	I
Case_3	B	Case_5	A	Case_3	F	Case_4	G
Case_1	D	Case_2	C	Case_4	D	Case_4	H
Case_2	B	Case_3	C	Case_2	G	Case_3	J
Case_4	A	Case_4	C	Case_2	I	Case_4	J
Case_1	E	Case_1	G	Case_3	F	…	

The steps of the mining process are as follows:

1) Build the Markov transition matrix of 'Pro'

a) the process of ‘Pro’ has 52 instances in all, i.e. n=52;

b) the process of ‘Pro’ has 10 activities in total (from A to J), $X = \{A, B, \ \dots J\}$

c) compute the one-step transition possibility between the activities, $p_{ij} = {}^{m_{ij}}/_{52}$, e.g., $p_{AB} = {}^{m_{ij}}/_{52} = {}^{52}/_{52} = 1$;

d) Establish the Markov transition matrix P

$$P = \begin{matrix} 0 & 1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.4 & 0.6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.3 & 0.7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

2) Execute the process mining algorithm Process (P,X), we can get the logical relationship sets W, W_{and}, and W_{select}.

$$X_S = A; \ X_E = J$$
$$W = \{(A,B)(B,C)(B,D)(C,E)(D,E)(E,F)(F,F)$$
$$(F,G)(G,H)(H,I)(I,J)(H,J)\}$$
$$and\colon \ A > B, E \rightarrow F, F \rightarrow G; B > C, B > D, C > E, D >$$
$$E, G > H, G > I, H > I, I > J; \ \Diamond F$$
$$W_{and} = \{(C,D)\}, i.e. \ \ C//D$$
$$W_{select} = \{(H,I)\}, i.e. \ \ H\#I$$

3) According to the relationship sets (W, W_{and} and W_{select}) and the activity set X, the directed network graph model of ‘Pro’ is shown in Fig.2.

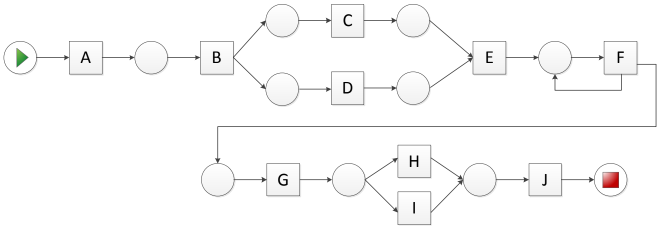


Fig.2. The direct network graph model of ‘Pro’

6 Conclusion

Since the changing environment internal and external of the enterprise, the workflow instances may deviate from the predefined models, so modelers need to reconstruct the model according to the actual situation. To overcome the limitation of the existing process mining algorithms, this article designed a multi-step process mining method based on Markov transition matrix, by analyzing the workflow log, automatically deduced the actual structure of the relationship between activities, thus implement the workflow reconstruction. Finally, a practical simulation case study shows that the method is feasible and applicable. Follow-up studies will focus on how to automatically derive sub-processes, as well as how to automatically mining rename tasks in order to improve and enhance the mining ability of the method.

7 References

[1] Agrawal R，Gunopulos D，Leymann F. “Mining process models from workflow logs[C],” Proceedings of the 1998 6th International Conference on Extending Database Technology，EDBT’98. Valencia，Spain：[s.n.]，1998，1377：469-483.

[2] Cook J E，Wolf A L. “Discovering models of software process from event-based data[J],” ACM Transactions on Software Engineering and Methodology，1998，7（3）：215-249.

[3] Cook J E. “Process discovering and validation through event data analysis,” Technical Report CU-CS-817-96[R].University of Colorado，Boulder，Colorado，1996.

[4] Zhao Weidong, Fan Li. “Situation and development on workflow mining research [J],”. Computer Integrated Manufacturing Systems, 2008,14 (12) :2289-2296.

[5] Aalst W，Weijters A，Maruster L. “Workflow mining：discovering process models from event logs[J],” Knowledge and Data Engineering，2004，12：369-378.

[6] De Medeiros A K A，van Dongen B F，van Der Aalst W M P. “Process mining：extending the α-algorithm to mine short loops [EB/OL] （.2007-10-11）,”

[7] http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.5.7518.

[8] Wen L，Wang J，Van Der Aalst W M P. “A novel approach for process mining based on event types[J],” Journal of Intelligent Information Systems，2009，32(2)：163-190.

[9] Van der Aalst W M P，Dustdar S. “Process mining put into context[J],” IEEE Internet Computing，2012，16(1)：82-86.

[10] Van der Aalst W M P. “Decomposing process mining problems using passages[C],” 33rd International Conference on Application and Theory of Petri Nets and Concurrency，PETRI NETS 2012.Hamburg，Germany：[s.n.]，2012，7347：72-91.

[11] van der Aalst W M P，Kindler E. “Process mining：a two-step approach to balance between underfitting and overfitting[J],” Software and Systems Modeling，2010，9（1）：87-111.

[12] Greco G，Guzzo A. “Discovering expressive process models by clustering log traces[J],” IEEE Transactions on Knowledge and Data Engineering，2006，18（8）：1010-1027.

[13] Demedeiros A K A，Guzzo A. “Process mining based on clustering：a quest for precision[C],” 5th International Conference on Business Process Management, BPM 2007.Brisbane, Australia：[s.n.]，2007，4928：7-18.

[14] Herbst J，Karagiannis D. “Workflow mining with InWoLvE[J],” Computers in Industry，2004，53（3）：245-264.

[15] Vaidya J，Guo Qi. “The role mining problem：finding a minimal descriptive set of roles[C],” Proceedings of the 12th ACM Symposium on Access Control Models and Technologies. New York：ACM，2007：175-184.

[16] Liang, Gao Jianmin, Chen Fumin, etc. “Research on workflow mining based quality management improvement [J],” Computer Integrated Manufacturing Systems, 2006,12 (4) :603-608

Self-optimization in Autonomic Computing Systems based on the Methodology of Bees Swarm Intelligence

Anselmo Leonardo O. Nhane[1], **Mark A. J. Song**[2]
Computer Science Department, Pontificial Catholic University of Minas Gerais,Belo Horizonte, MG, Brasil
[1] Computer Science Department and Information System, National Statistical Institute, Mozambique
[2] Computer Science Department, Centro Universitario UNA, Belo Horizonte, Brasil

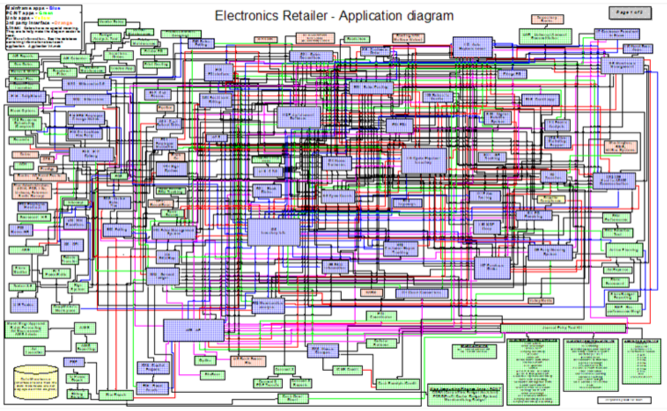
Abstract—*This paper proposes a mechanism for self-optimization in autonomic computing systems, inspired in the functioning of the bees swarm in the process of searching for food sources, exchanging information about the located sources and in the allocation of bees to such sources. This analogy is applicable to autonomic computing systems, on which we also seek to continually optimize the system operation by monitoring and adjusting its parameters and evaluating the fitness of proposed solutions. The goal is try to meet 100 Percent of the Demand while minimizing Costs by the system. The methodology follows the component of autonomic control loop. The article includes a case study focused on the statistical information dissemination system of Mozambique, which uses resources from a private datacenter.*

Keywords: Autonomic computing, Autonomic computing systems, Self-Management, Self-optimization, The Bees autonomic management

1. Introduction

The increasing complexity in computer systems, Figure 1, applications and services, allied with the growing in the demand for integrated services, are factors that limit traditional mechanisms for software system managenent with human intervention.

Fig. 1: Example of the IBM Complex System



Mechanisms such as definition of protection policies, elicitation of the system, resource management, troubleshooting, configuration and implementation definition that meets optimization aspects, require a thorough knowledge of the systems. However, the human capacity to intervene and meet all these systems' needs is a limiting factor (organization challenges), which motivates the scientific community to propose systems with minimal human intervention. That is, the paradigm that aims at reducing administrative overhead by providing self-managing applications.

In poor and developing countries like Mozambique, which has one of the fastest-growing rate of Gross Domestic Product in Sub-Saharian Africa (about 7.3 percent per year), infrastructure is also in constant expansion, accompanied by the increase in the number of users and services provided using software systems, which implies an increased complexity. But these countries also have low availability of trained human resources to deal with the growth of the software systems and infrastructures and also they do not have the budget to ensure the continuous expansion in number of systems. In this context of computing, introducing autonomic computing can make a difference to this group of countries. First, the rate of computer literacy in the whole countries can be minimized, once autonomic computing can take care of its functioning with minimal human intervention.

In 2007, Mozambique had less than one percent of the population with internet access [5], but in 2010, the number of users rose to approximately 2.7 percent and to 4.3 percent in 2012 [7]. This increase is resulting in complex systems in a short period of time to meet the growing demand for current services and products of the Mozambique's statistical system.

In this paper we propose a self-optimizing mechanism, in autonomic computing system, drawing inspiration from the operation of the bees swarm on its process of looking for food and sharing information in the dance space. In this meter we have confidence that the working process in the hive can be applied to the operation of an autonomic system, which constantly seeks to optimize the system functioning through the monitoring, control and analysis of parameters to identify changes that require configuration changes or allocation of more resources in a situation of efficiency loss.

The article includes a case study, aimed to show how developing countries can optimize their resource usage, by monitoring demand and making decisions to adjust the level of needed resources from the datacenter, at the dissemination unit of National Statistical System in Mozambique, with budget restrictions.

2. Autonomic Computing

The concept of Autonomic Computing (AC) was introduced by Paul Horn [9]. This concept is inspired by the Human Nervous System in order to find new ways of implementing software systems that are able to answer some challenges posed by the increasing complexity of current IT systems. The human nervous system is the most complex system and it is an example of autonomic behavior in nature. Such system is the master controller of the functioning of the human body, which monitors the changes that occur inside and outside the body. Then, responds appropriately in order to maintain the balance of the functioning of other organs [8]. Complex software systems can be comparable with the human nervous system which controls the most part of body works, eliminating from the human conscious the activities of managing all the actions of the body. So, complex systems must possess autonomic properties that allow them to control part of their operation without human intervention. This paradigm means, the change of the patterns of the systems managed by humans (traditional), into the new era of technology standards managed by the technology itself [19]. Such systems are called autonomic. Several proposals have emerged in the last decade, challenged by the complexity of IT systems. In 2003, for example, Defense Advanced Research Projects Agency (DARPA) proposed self-regenerative systems for military purposes, that can react to unintentional errors or attacks. An example is the Situation Awareness System, from the DARPA, where they intended to create a communication device for location of the soldiers on the battlefield, that can detects and collect data on the presence of the enemy tank, and independently reporting on the location of these to all soldiers. Even in situations of extreme hardship, it can aid in minimizing interference from the enemy [1].

2.1 Autonomic computing system properties

An autonomic system has the following properties: ability to self-configure, self-heal, self-protect and self-optimize. Self-configuration is a property on which the system must ensure, automatically, dynamic adaptation to the changes that occur in their environment. Self-healing is the property on which the system must discover/identify, diagnose and react to the disturbance that can cause the system to malfunction. Self-protection is another property inherent in autonomic systems. This property is in the system to anticipate, detect, identify and protect itself from attacks from any source. Finally, we have the self-optimizing property, through which the system must monitor and adjust its own resources. For this, it is necessary that hardware and software systems maximize the efficient use of resources to achieve the end-user requirements with minimal human intervention. However, several aspects of software engineering, such as life cycle elements, software processes, elicitation of system requirements, self-healing implementation, self-protection implementation, self-configuration and the implementation of self-optimization are still major challenges to be overcome. Thus, the research focus on self-optimization, based on the use of the bees swarm method [10] to optimize the system parameters that govern the system operations in the allocation of resources.

3. The Bees Algorithm

In the natural process of exploring environment, the bees are divided into groups (onlookers, foragers, employed). The Foragers seek those regions with good food (nectar and pollen) sources. After returning to the hive they perform movements known as Waggle Dance, to communicate to the others bees the profitability of the sources found, the distance and amount of these food sources.

Then, the employed bees and onlookers travel to the selected food sources and for the fields with higher abundance of food, more resources are allocated (greater presence of employed bees) [16], [3], [4]. The process repeats and the bees are recruited more and more, in accordance with the food demand.

Karaboga, in his study of the Swarm Intelligence, proposed the bees swarm algorithm for parameter optimization [10]. Successively, the algorithm versions were being updated with substantial improvements [11].The latest version offers a parallel approach [15], [13], where the artificial agent used is divided into several independent subpopulations.

The distribution of food sources positions are randomly produced to the allowable limit of the parameters. From the Karaboga algorithms illustrated in 1, we have:

$$x_{ij} = x_j^{min} + rand(0,1)(x_j^{man} - x_j^{min}) \qquad (1)$$

Where $i = 1, 2, ...SN, j = 1..D$. SN is the number of food source and D the number of optimization parameters. The employed bees operating the sources calculate food sources and share information with onlookers bees, according to the formula in 2:

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{ik}) \qquad (2)$$

Where j is a random integer in the range $[1, D]$ and $k \in 1, 2, .., SN$ is a randomly chosen index different from i. $\phi_{ij} \in [-1, 1]$ uniformly distributed. After the v_i is specified with values that are within the limits of the parameters, a fitness value for minimization problems can be associated with the value v_i by the formula: $fitness_i$=$1/(1 + f_i)$ if

$f_i >= 0$ and $fitness_i = 1 + abs(f_i)$ if $f_i < 0$, where f_i is value of the objective function that represents the cost value of the solution v_i.

The greedy selection is used between v_i and x_i for each mutant solution, based on minimal cost of each v_i generated. After all employed bees complete their search for food, they share their information about the amount of nectar and the position of the source with bees onlookers in the dance area. The onlookers bees select food sources according to their profitability, and then they look for new sources of food for a probabilistic choice given by 3:

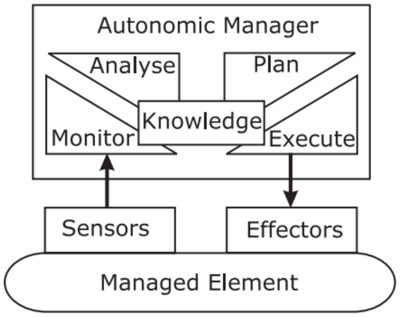
$$p[i] = \frac{fitness[i]}{\sum\limits_{i=0}^{SN}(fitness[i])} \tag{3}$$

Finally, if a food source is not improved, then the bees leave the food source and call upon an onlooker to discover a new source in the search area.

4. The Proposed Approach: Self-Optimization in Autonomic Computing

The self-optimization process is based on the MAPE-K (Monitor, Analysis, Plan, Execute and Knowledge) control loop proposed in [12], Figure 2.

Fig. 2: The IBM Autonomic Control Loop



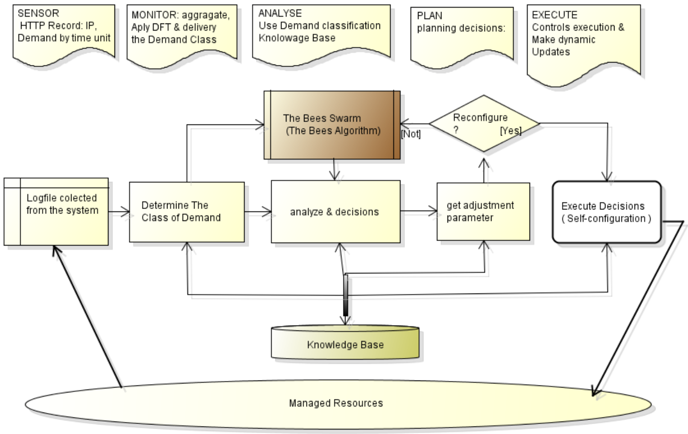
In this section we introduce *The Bees Autonomic Manager* (BAM) to generalize the idea for self-optimization processes.

BAM introduces two features in its self-optimization process. First, is the use of clustering method to harmonize monitored data from the system, in order to get more accurate decisions parameters, that are less influenced by outliers, when we consider the use of statistical mean as an estimator. Second, we proposed the use of the bees algorithms in the decisions making with optimal selection of parameters, and subsequently we use polices to map the states of the system based on the demand level which we rank in three levels.

We consider that self-configuration studies are available. That is, we will not consider in our self-optimization proposal specifications related to configurations to change the system, as we consider that is related to the self-configuration properties. We are not showing how to retrieve sensor data form the managed element, as the system must include features to report statistics of its functioning. The control loop for autonomic computing deals with high level polices that administrators can specify. In this case, we do not show how to implement the policies, once again there are many options using XML to deal with policy specification.

The general overview of the BAM can be seen Figure 3. The figure focuses on monitoring, analyzing and decision executions. The study will consider the two process: Monitoring and Decisions among the changes in the users demand.

Fig. 3: The BAM



The sensors,[12] collect data from the managed resource, via a log file. These data are grouped according to the objective criteria that guide the process. Example: collected data from the dissemination service in Statistic office, of the user demand include: time, IP, information concerning data sets, database, number of data cells extracted, information size of extracted data. The data is set to SQL database.

The **monitor** processes the data collected by the sensor and communicates the level of demand indicators. The processed data are used for the recognition of demanded levels, according to the two following steps: (1) extraction of processing characteristics, from the information passed in the bees dance area, and (2) the classification of the processing level. The extraction of (1) is obtained by applying the Discrete Fourier Transform (DFT), as illustrated at formula 4.

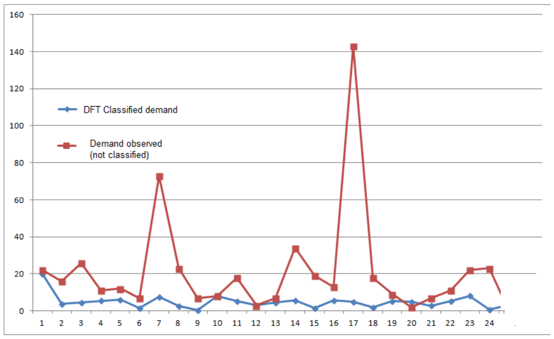
$$X[k] = \sum_{n=0}^{N-1} x[n]\epsilon^{(\frac{-2j\pi n)}{N}} \tag{4}$$

The Discrete Fourier Transform (DFT) is a function of the user demand frequency grouping of extracted sample data, at periodic intervals (which can be defined by the users).

The recognition of demand patterns (high, medium, low) is done by classifying the extracted demand intervals. This information is used to map the needs for resources. The clustering is done from the analysis functions x(n) of the observed demand data, and X(k) is the transformed form of x(n) according to the formula in 4.

The example in Figure 4, includes data collected by the sensor according to the methodology of bees, which corresponds to the demand registered in the system environment. The DFT line of graphic shows clustered data,reported in the area of dance(ruled by the methodology of bees) with less amplitude, which helps in finding accurate values to the decision step.

Fig. 4: DFT and Demand frequency: Service usage for a day constant number of resources



The DFT properties are inherent to the needs of the study, allowing vertical translation of the observed frequencies in order to perform analysis on the situations at low (A), middle (B) and high (C) demands while achieving better precision in demand levels, as Figure 5.

The Monitor uses DFT with the aim of grouping similarity levels by proximity. That is, determining values with the same distribution characteristics to converge within a given range for a certain demand class. This enables the elimination of the influence of extreme values if using the average metric. The adapt case of autonomic monitor, ilustrate the 5 the actions of monitor activity.

The step of **decision analysis** correlates reported event data to identify the current status of the managed element and then suggest actions to be taken in case of performance loss or cost degradation. Then, the BAM use the Bees algorithms and determines which resources are going to be used, correlating the quantities of resources used, and the level of demand identified that optimizes the cost of operation.

This phase uses classified demands from the autonomic monitor, to determine which level of resource is necessary to minimize the cost. In Figure 6, we represent the general overview of the use case for the bees autonomic manager. The bees autonomic manager loop considers that the system interacts with users. And the external component - autonomic manager - is added. The diagrams representation are sourced from Markus in [14], where the study proposes the *Adapt Case Modeling Language* that can be applied for specification of the autonomic computing systems.

Fig. 5: Adapt Case BAM: Autonomic monitor

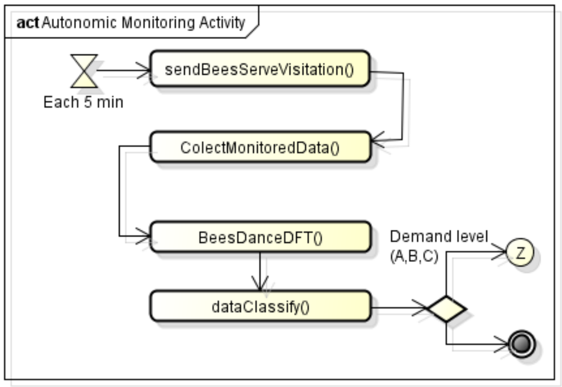
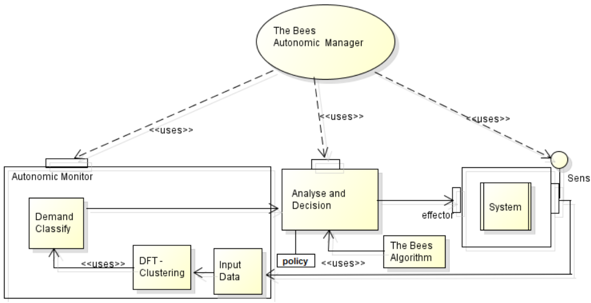


Fig. 6: The Bees autonomic manager



The analysis and decisions also use knowledge to enable decisions in the situations which are planned beforehand, and not in case of dictation of high or low demand. The system in the diagram uses the datacenter(the source for resources) to obtain resources to supply the demands specified.

The resource fitness function is obtained from the cost of each food source: each type of resource has its cost. Each food source includes all types of resources. $uC^i[j]$ - cost of j resource in the the i^{th} food source.

Each mutant solution $s[i]$ is evaluated using the following formula:

$$fsuC[i] = \sum_{j=0}^{NP-1} s[i] * uC^i[j] \qquad (5)$$

The formula to determine the fitness related to the cost of each class of demand is the following:

$$fs(i) = 1 - \frac{fsuC[i]}{Bc} \qquad (6)$$

Where Bc - is the Budget of demand class, that represents the Total value that enables administrators to specify resource limitations for the system. That is, there is the cost value of each set of resource from the food source related to the demand. In each situation of demand we consider a particular budget of the demand class. This information will help the autonomic bees manage the different cost degradation for each class (A - lower, B - medium-size , C - high) of demand. Based on the fitness value of each solution:

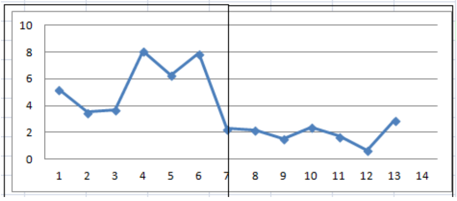
$$fitness[i] = \begin{cases} \frac{1}{1+fs(i)} & \text{if } fs(i) >= 0, \\ 1 + abs(fs(i)) & \text{if } fs(i) < 0 \ . \end{cases} \qquad (7)$$

Management Levels in the BAM: The bees make decisions ir order to achieve global parameter optimization. But the proposal results does not completely guarantees the optimal allocation of resources. Two steps are considered for the decisions. In the first level, we consider the cost of current resource of proposed solution and the total cost of the demand class. The second level indicates at Lower level, resource based on efficiency indicator (like speedup, response time) optimization in order to balance the needed resource, from the bees output proposed parameters.

The **knowledge base** provides data in addition to information about past decisions. An example of data provided is the time of day, with historical information demand levels. This information adds significant value for predicting the situation of high levels of access to the service.

The example, in Figure 7, illustrates the behavior change of service requests to the systems in the first seven hours, where there was a drop from 5.8(avarage value) to nearly 2.0(avarage value) of demand. This indicates that demand has fallen about three times the value of the last seven hours. Implying the change of track B to the range of low demand A. That is, the resource in use are in high cost, so it is necessary to reduce the resource to better attend the current situation.

Fig. 7: Example of changing demand from (B Class) at the time 7 to (A Class).



The mean and standard deviation, maximum and minimum values that were observed are key indicators to consider in this range. So the new parameters are defined based on this information. In the example considered, we use the average to select the demand class.

In the *decision planning* step we evaluate the current demand and observed demand, as the criteria for optimizing based on the specifications of the administrator in the form of policy or from analysis and decision, providing new parameters for the system.

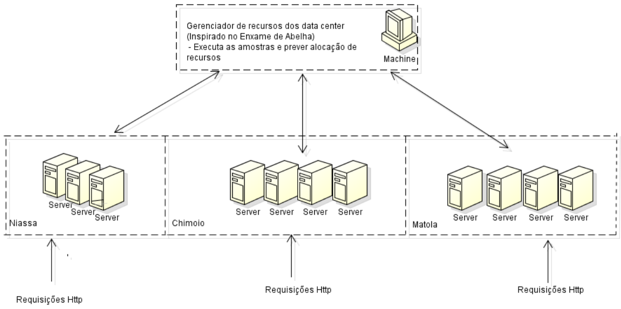
The process of *executing decisions* occurs when the subset of parameters can improve the current standard of processing resource consumption. The process receives data on the approved update to the next time interval. Then it updates the information on the use of resources in the knowledge base. This step allows the system reconfiguration to attend the suggested levels of demand at low adequate cost of resource usage.

5. The Case Study: I

In the case study we apply the optimization to the dissemination system of National Statistical Institute of Mozambique, based on a private datacenter.

Background: We consider that there is a computer network that can support the configuration in order to attend the needed of services levels. There are technologies available for self-configuring to attend if the optimization decisions if achieved. These assumptions help to visualize the proposed methodology for the few steps of self-optimization based on the cost in the datacenter for the statistic office.

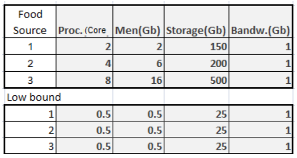
Fig. 8: Source of resources: Datacenter



The Private Datacenter of Mozambican statistical service, Figure 8, represents the source of resources for the managed element: the output database of the dissemination unit.

The infrastructure includes a Web-server with non relational database, Pxweb, with $.Px\ files$(PC-AXIS software family format), and it also includes statistical software applications for dissemination. These data set have homogeneous sizes (less than 1Mb). In accordance with the data collected from the Logs of the web-server, 99 percent of queries are characterized by extracting data of smaller size. The main feature is to build the px-matrix and send it to the user. The user can then either manipulate or view the table. To manipulate tables, there are application tools based on javascript which run in the user side. All the processes are affected by processor, memory, disk, and bandwidth. That is, the optimization process have to consider the four parameters for each food source(datacenter node with processing capacity) in the datacenter.

Fig. 9: Initialization: Up bound and low bound

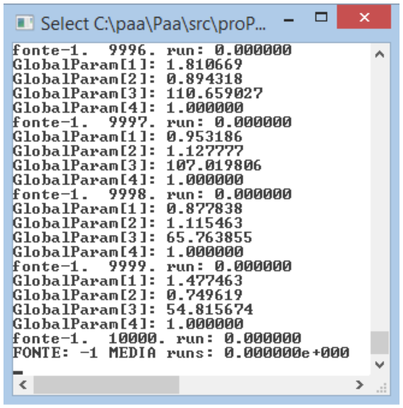


Food Source	Proc. (Core	Men(Gb)	Storage(Gb)	Bandw.(Gb)
1	2	2	150	1
2	4	6	200	1
3	8	16	500	1
Low bound				
1	0.5	0.5	25	1
2	0.5	0.5	25	1
3	0.5	0.5	25	1

Control parameters: Number of bees (employed bees + onlookers), NP=6; Max cycle 10; execution time 10000; Number of optimization parameters, D =4. Number of food sources, which represents data center sites, SN=3; Mutation rate $\phi = 0.05$; the resource bounded by Upper and Lower resources, Fig. 9. Representing the availability of resource that can bee allocated to the system from the datacenter. The cost is considered 1 USD per resource unit, per hour. We consider the budget for A is the minimal.

The output of executions is a set of minimal parameters that represent the A demand situation, that where selected from fitness in formula (5) with an estimated cost 10 USD.

Fig. 10: Output with the A ranked demand



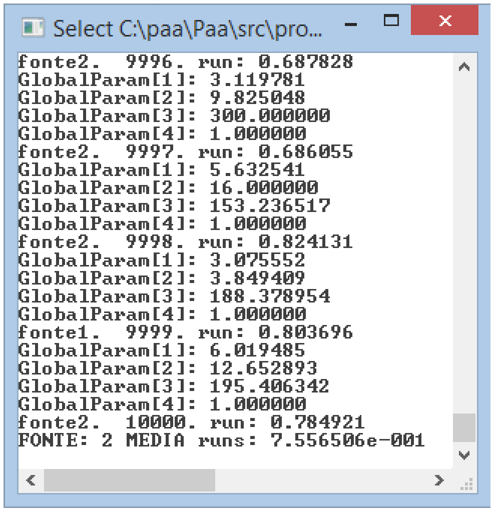
```
fonte-1.  9996. run: 0.000000
GlobalParam[1]: 1.810669
GlobalParam[2]: 0.894318
GlobalParam[3]: 110.659027
GlobalParam[4]: 1.000000
fonte-1.  9997. run: 0.000000
GlobalParam[1]: 0.953186
GlobalParam[2]: 1.127777
GlobalParam[3]: 107.019806
GlobalParam[4]: 1.000000
fonte-1.  9998. run: 0.000000
GlobalParam[1]: 0.877838
GlobalParam[2]: 1.115463
GlobalParam[3]: 65.763855
GlobalParam[4]: 1.000000
fonte-1.  9999. run: 0.000000
GlobalParam[1]: 1.477463
GlobalParam[2]: 0.749619
GlobalParam[3]: 54.815674
GlobalParam[4]: 1.000000
fonte-1.  10000. run: 0.000000
FONTE: -1 MEDIA runs: 0.000000e+000
_
```

When the system reports from *Monitor classify* A, no selection is necessary, Figure 10, since it will be working in the lowest demand. It is time for reducing resource, by using low bound resources level. That happens in Time 7, then the system changes from Class B to A.

The output from picture in figure shows when the autonomic manager sets the resource for the budget necessary to supply level B demand levels.

The source 2 is ranked as on of the best for the amount of budget, and there also source 1 available. If the fitness is positive, then the allocation can be authorized.

Fig. 11: Output with the B ranked demand



```
fonte2.  9996. run: 0.687828
GlobalParam[1]: 3.119781
GlobalParam[2]: 9.825048
GlobalParam[3]: 300.000000
GlobalParam[4]: 1.000000
fonte2.  9997. run: 0.686055
GlobalParam[1]: 5.632541
GlobalParam[2]: 16.000000
GlobalParam[3]: 153.236517
GlobalParam[4]: 1.000000
fonte2.  9998. run: 0.824131
GlobalParam[1]: 3.075552
GlobalParam[2]: 3.849409
GlobalParam[3]: 188.378954
GlobalParam[4]: 1.000000
fonte1.  9999. run: 0.803696
GlobalParam[1]: 6.019485
GlobalParam[2]: 12.652893
GlobalParam[3]: 195.406342
GlobalParam[4]: 1.000000
fonte2.  10000. run: 0.784921
FONTE: 2 MEDIA runs: 7.556506e-001
```

In each interval of time the reward will be difference from the constant utilization of the system resource. In non autonomic systems, mines are used to the demand the cost in use for each interval.

6. Review of Related Works

The question of how to achieve self-optimization in autonomic computing has dominated research in the autonomic computing field, during the first decade. The self-optimization property has received the most contributions since IBM formally introduced the autonomic computing paradigms. The proposals on self-optimization are based on different technique like the Control theory, Utility functions, and Queue theory.

Walsh proposes the use of model-free reinforcement learning based on the use of SARSA [17], [18]. This approach estimates future learning rewards, Q, for each state-action pair. Its learning rule is given by:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t)+\alpha[r_{t+1}+\gamma Q(s_{t+1},a_{t+1})-Q(s_t,a_t)] \tag{8}$$

Where Q is the estimated reward (expected); s is a state, a is an action, r is the reward. E t and $t+1$, are used to indicate states, present and future and action respectively. $Alpha$ is the learning rate and $gamma$ is the discount factor. The SARSA name is given due to the quintuple $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$ that the algorithm uses.

Van (2010) quoted by [6], in his study on "Managing performance and energy consumption in cloud infrastructure ", uses an approach based on utility function to optimize performance and energy consumption of the cloud system, by determining how many resources could be allocated, and how the applications should be put in the infrastructure to optimize energy consumption.

Copyright © 2014 CSREA Press, ISBN: 1-60132-286-0; Printed in the United States of America

The study proposes one framework on two levels. The first is the application level, where for each application, there is a performance model, measured by the utility function and use expressed in response time. At the second level of infrastructure, there is a global asset manager. The first is responsible for determining the amount of resources in terms of VMs for each application, while the second , by provisioning VMs resulting from the first manager, is responsible for optimizing the allocation of all VMs, as well as to minimize the number of physical machines needed and power consumption .

And the control is done sequentially and periodically, if and only if, there is a difference between current number of VMs and the resulting number of provisioning manager. As a result of the implementation of the provisioning of VMs manager is a set of VMs to be requested or released. Manager allocation of VMs produces a set of VMs with purpose or to create or to destroy or to migrate , as well as to turn off or turn on the power consumption [20]

7. Conclusion

The proposal and ongoing project contribute in the study of autonomic computing, exploring alternatives for achieving self-optimization for the systems. This is done by applying the BAM. This analyse uses a wide cycle for convergence to obtain good results, but they do not interfere in the system once the decision are made in each interval of at least 5 minutes or more depending on system type or number of parameters, and the algorithm convergence is achieved in less than a minute. The example considers the management of resources to the dissemination service, allocating and removing resources provided from the datacenter. This approach helps to maintain the good level of resource usage at low cost based predicted demand. During the observation periods, we show situations where the system was using resources to fulfill the demand for those periods. From Figure 7, the system uses the total resources available for the system during all 13 time units. With the BAM we change from 7 to 13 the amount of resource, form B to A. This result in improvements on the use of resources (eg. low power consumption, less computer or VM allocated). In general, this causes great money savings. This really fulfills the the poor countries, since in these countries there are less literacy rates in computer fields, as well as smaller budgets to maintain the systems. So we agree that using Autonomic computing systems, it will be possible to improve management of total cost for system owners. And so on, minimizing governments challenges in the decision level of the organization in how to introduce the IT systems in remote locations.

Future works: The BAM coordination is the next step to deal with. We agreed that two major situation are important for it: Global parameter optimization coordination - That indicates the best source of resource in term of adequate cost for the demand level, to allow multi-application management.

We intend to develop a framework that will implement the bees autonomic manager and test it in real-time systems for Population Census. We consider that this can help with data collection in distributed environments, by adding and removing resources that can be sourced from external enterprises based on cloud computing in order to minimize cost of the activity.

Acknowledgment

The authors would like to thank the National Statistical Institute(Mozambique) and FAPEMIG for the financial support.

References

[1] L. Badger, “ Self-Regenerative Systems Program Abstract”. *Defense Advanced Research Projects AgencyDARPA*, 2004.

[2] B. Basturk, D. Karaboga, “An Artificial Bee Colony (Abc) Algorithm For Numeric Function Optimization”. In *Swarm Intelligence Symposium*, IEEE Computer Society, 2006. P. 459–471.

[3] E. Bonabeau, M. Dorigo, G. Theraulaz, “Swarm Intelligence: From Natural To Artficial Systems”, The Oxford University Press, New York, 1999.

[4] S. Camazine, “Self-Organization In Biological Systems”, Princeton, Princeton University Press, 2003.

[5] INE Mozambique(2007), The INE output database of the Census 2007, Available: http:http://www.ine.gov.mz/censo2007.

[6] F. G. Junior, “Multi Autonomic Management For Optimizing Energy Consumption In Cloud Infrastructures”, Universitè De Nante, 2013.

[7] M. Group(2013),The Worldstats website, Internet World Stats: Internet Usage And Population, Available: http:http://www.Internetworldstats.com.

[8] S. Hariri, “The Autonomic Computing Paradigm”, in *Proc.Cluster Computing*, Springer Science, 2006.

[9] P. Horn, Autonomic Computing: The IBM Perspective on the State Of Information Technology. Available: http://researchweb.watson.ibm.Com/autonomic, 2001.

[10] D. Karaboga, “An Idea Based On Honey Bee Swarm For Numerical Optimization”, Erciyes University, Turkey, 2005.

[11] D. Karaboga, B. Basturk, “A Powerful And Efficient Algorithm For Numerical Function Optimization: Artificial Bee Colony (Abc) Algorithm. Journal Of Global Optimization, vol 39, IEEE Computer Society, P. 459–471, 2007.

[12] J. O. Kephart,D.M. Chess, “The Vision Of Autonomic Computing. Ibm Thomas J. Watson Res. Center”, Hawthorne,USA,IEEE, Vol 36, 2003.

[13] H. Narasimhan, “Parallel Artificial Bee Colony Algorithm”, IEEE, 2009.

[14] M. Luckey, G. Engels, “ High-Quality Specification Of Self-Adaptive Software Systems: IEEE SEAMS 2013”, San Francisco, Ca, Usa. 2013.

[15] H. Narasimhan, “Parallel Artficial Bee Colony (Pabc) Algorithm”, IEEE, 2009.

[16] D. Pham, A. Ghanbarzadeh, “The Bees Algorithm. A Novel Tool For Complex Optimisation Problems”,Manufacturing Engineering Centre, Cardiff University, Uk, 2005.

[17] W. E. Walsh, J. O. Kephart, “Utility Functions In Autonomic Systems. In: Autonomic Computing, 2005”, P. 342–343, 2005.

[18] G. Tesauro, “Reinforcement Learning In Autonomic Computing: A Manifesto And Case Studies”. IEEE, 2007.

[19] S. Rangswamy, “Techniques And Theories Of Self-Otimizationin Autonomic Systems”, International Journal Of Advaced Engineering Technology., 2011.

[20] H. N. Van, F. D. Tan And J. M. Menaud, “Performance And Power Management For Cloud Infrastructures”. *IEEE International Conference on Cloud Computing*, Usa, P.329–336, 2010.

Software Architecture Evolution in the Open World through Genetic Algorithms

Myriam Torres and Germán H. Alférez
Facultad de Ingeniería y Tecnología
Universidad de Montemorelos
Apartado 16-5 Montemorelos N.L. Mexico

Abstract - *In the open world, it is impossible to know at design time all the possible context events that can arise (e.g. sudden security attacks, decreased performance, or service unavailability). Moreover, it is unthinkable to fix these situations manually because of the rapid responses required in modern systems (e.g. in banking or trading). In this paper, we propose an approach to support the dynamic evolution of software architectures in the open world by means of genetic algorithms and models at runtime. Dynamic evolution actions try to preserve the expected quality attributes of the software architecture when facing unknown context events. Models at runtime are used to reason about the quality attributes to be preserved and the tactical functionality to preserve these requirements. Genetic algorithms inject the tactical functionality into the running software architecture. Our approach is supported by a running prototype.*

Keywords: Autonomic Computing, Genetic Algorithms, Dynamic Evolution, Models at Runtime, Open World.

1 Introduction

In recent years, there has been a trend to self-adapt software architectures in order to face arising context events (e.g. events in the computer infrastructure, such as security attacks). Instead of carrying out manual adjustments, which can be slow and error-prone, self-adaptation facilitates how the system can respond to changing contexts.

Most of the solutions to achieve self-adaptive software have focused on the closed-world assumption. When developing software for the closed-world, the set of all possible adaptations is known at design time. However, in the open world, it is impossible to know beforehand (i.e., at design time) all the possible context events that can arise. This is specially truth in ubiquitous and pervasive computing settings, for example in software based on cloud computing or in wearables. Therefore we need approaches to develop systems that self-adjust in the open world.

Our contribution is an approach to support the evolution of software architectures in the open world by means of Genetic Algorithms (GAs) and models at runtime. On one hand, a GA is a search heuristic that mimics the process of natural selection. GAs are especially useful for optimization, machine learning, and business applications. On the other hand, models at runtime are used during execution to reason about the quality attributes that need to be preserved at runtime and how these attributes can be preserved.

When an unexpected event occurs in the open world, GAs are used to find out the most appropriate software architecture. First, a set of possible software architectures is auto-generated. Then, these architectures are mated and a set of abstract tactics is used to mutate them. We propose the concept of *tactics* in order to extend the software architecture with new functionalities that can be used to keep the Quality of Service (QoS) level. Then, the utility function, or fitness level, of each software architecture is evaluated. The most suitable solutions are used to create new populations.

We use GAs in our approach because they are useful to search for the most appropriate solution in the large and uncertain open world. Specifically, in the open world there is a large space of possible context events (foreseen and unforeseen) and a large number of software architecture configurations that can be used to face these events. Also, the proven efficiency of GAs offers an attractive option to get a resulting software architecture in a short amount of time. Last but not least, since GAs are supported by an evolutionary approach, they offer a logical way to guide the dynamic evolution of software architectures. In this work, we see the concept of *evolution* as the gradual and continuous growth of the software architecture. *Dynamic evolution* does not imply just punctual adaptations to punctual context events but a gradual structural or architectural growth into a better state.

The remainder of this paper is organized as follows. Section II presents the background. Section III presents a motivating scenario. Section IV describes our approach for the dynamic evolution of software architectures. Section V presents the tool support. Section VI describes related work and Section VII presents conclusions and future work.

2 Underpinnings of Our Approach

Nowadays, an important trend in software engineering is to support the automation of decision-making at runtime with Autonomic Computing [1]. Instead of doing manual reconfigurations of software architectures to adapt to changes in the context, it is desirable that the software architecture self-adapts during execution. In order to achieve this goal, we propose an approach based on the following concepts:

1. ***Context:*** Applications should be aware of their contexts and automatically adapt to their changing contexts in order to provide adequate services to users. Specifically, “context is any information that can be used to characterize the situation of an entity” [2]. A system is context-aware if it can extract, interpret and use context information and adapt its functionality to the current context of use.

2. ***Dynamic adaptation vs. Dynamic evolution:*** The self-adaptive-software community is concerned with the increasing complexity of the context [3]. This complexity is caused by systems that are moving from the closed world to the open world (e.g. ubiquitous and pervasive computing).

Under the closed-world assumption, the possible context events are fully known at design time. These events will eventually trigger the dynamic adaptation of the software architecture. Nevertheless, it is difficult to foresee all the possible situations arising in uncertain contexts where the system runs. Therefore, the software architecture should react to continuous and unanticipated changes in complex and uncertain contexts for a better functioning.

We define *dynamic evolution* as the process of moving the software to a new version, which cannot be supported by predefined dynamic adaptations, in order to manage unknown context events at runtime [4]. We refer to *unknown context events* as those arising situations in the context that have not been foreseen at design time. *Uncertainty* is caused by how the software architecture should deal with these unknown context events.

3. ***Autonomic Computing:*** It is a branch of software engineering concerned with creating software systems capable of self-management [1]. The term *autonomic* is derived from human biology. For example, the autonomic nervous system monitors heartbeat without any conscious effort. In a similar way, self-managing autonomic capabilities try to resolve problems with minimal human intervention.

4. ***Models at Runtime:*** Models at runtime can be defined as “causally connected self-representations of the associated system that emphasize the structure, behavior, or goals of the system from a problem-space perspective” [5]. In response to changes in the context, the system itself can query these models to determine the necessary modifications in the underlying architecture.

5. ***Evolutionary Algorithms:*** Evolutionary Algorithms (EA) are a branch of artificial intelligence for solving search and optimization problems. EA includes search methods that allows to treat optimization problems where the objective is to find a set of parameters for a function adaptation. From this set, one of the most popular models are GAs [6].

GAs work with a set (population) of candidate solutions (individuals) with the best capability (adaptation) [7]. The evolution usually starts from a population of randomly generated individuals and happens in generations. The population changes as an iterative process (generations) where individuals that have better capabilities are likely to survive and move to the next generation and participate of the genetic operators. In each generation, the fitness of every individual in the population is evaluated. Multiple individuals are selected from the current population (based on their fitness), and modified to form a new population. New individuals are created by means of using genetic operators, namely selection, crossover, and mutation.

3 Motivating Scenario

In order to illustrate the need for dealing with unexpected context events in the open world, we introduce a critical system that supports on-line product shopping at EUROTECH, a multinational retailer of technology products. In Figure 1, the UML is used to design the software architecture in terms of software components.

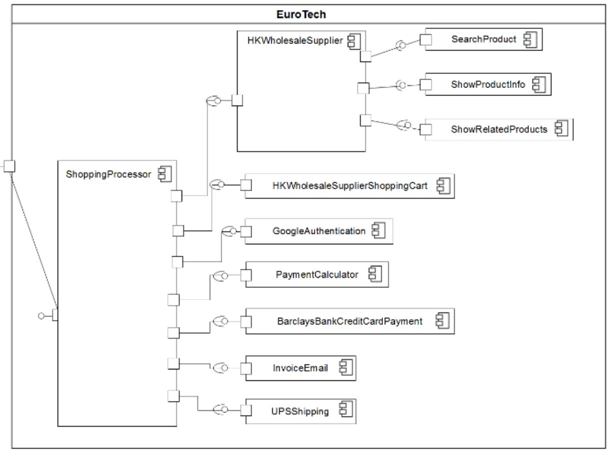


Figure 1. EUROTECH software architecture

The operation for product searching is provided by the SEARCHPRODUCT component, which is part of the HKWHOLESALESUPPLIER component. The product information is sent to the customer by the SHOWPRODUCTINFO component and the information for other related products is listed by the SHOWRELATEDPRODUCTS component. Customers can add products to the shopping cart through the HKWHOLESALESUPPLIERSHOPPINGCART component. When the customer is ready to checkout, he or she is authenticated by the GOOGLEAUTHENTICATION component. The PAYMENT CALCULATOR component calculates the total amount to be paid. The payment is done through the BARCLAYSBANKCREDITCARDPAYMENT component. Finally, the in-house EMAILINVOICE component sends an e-mail to the customer with the invoice and the UPSSHIPPING component is invoked to deliver the product.

In order to support the dynamic adaptation of the system to keep this process available 24/7, systems engineers have programmed a set of predefined adaptation actions for specific context events. For instance, if the BARCLAYSBANKCREDITCARDPAYMENT component is unavailable, then other components can be invoked instead.

Nevertheless, implementing predefined adaptation actions has the following drawback: If there are not predefined adaptation actions for a particular context situation, then no adaptation is carried out. This situation helps us to identify the following *challenges* for context-aware systems in the open world: 1) context-aware systems should be able to count on corrective actions that trigger the dynamic evolution of the system to preserve the expected requirements when facing unknown context events; and 2) the dynamic evolution of the software architecture should be carried out by auto-generated evolution actions in order to avoid human intervention. The following section describes our approach to face these challenges.

4 Dynamic Evolution of Software Architectures

Manual dynamic adaptation of software is unfeasible in complex computational scenarios that require prompt response and high availability. Moreover, critical software, such as software that supports power grids, cannot be stopped in order to inject new functionality or make changes. Therefore, our approach is focused on supporting autonomic adjustments of the software architecture. To this end, our solution is based on IBM's reference model for autonomic control loops [1] (which is sometimes called the MAPE-K loop). By following the principles of the MAPE-K loop, it is possible to inject autonomic behavior into the system.

Figure 2 describes the pieces of our approach in terms of the MAPE-K loop. The CONTEXT MONITOR retrieves information from the context by means of sensors (e.g. one sensor measures performance and another one observes security levels). In turn, the EVOLUTION PLANNER analyzes the collected context information and looks for quality attributes that can be negatively affected by unknown context events (i.e., unforeseen at design time). Also, the EVOLUTION PLANNER plans dynamic evolutions by looking for surviving tactics to preserve the software architecture despite unknown context events. Afterwards, the GA EVOLVER executes the dynamic evolution of the software architecture by means of GAs.

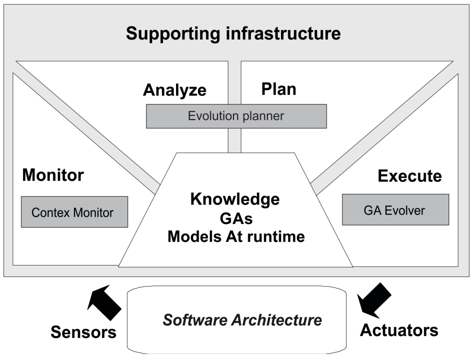


Figure 2. Our approach for dynamic evolution

The steps that are carried out by our approach to face unknown context events in the open world are as follows:

1. **Observe the Context:** The objective of this step is to get information from the context. This information will be used later to trigger dynamic adjustments on the software architecture. To this end, we propose a CONTEXT MONITOR that processes context information that is collected by sensors and updates an ontology accordingly. The CONTEXT MONITOR captures the basic metrics of significant quality attributes from the context. The CONTEXT MONITOR and the underlying ontology are described in our previous work [8, 9].

2. **Look for Quality Attributes that Can Be Affected.** The objective of this step is to look for the *quality attributes* that can be negatively affected by unknown context events. Quality attributes are the basis of software architectures [10]. Therefore, quality attributes, such as accuracy, security, reliability, availability, and performance must be preserved at runtime despite arising unknown context events.

In order to find the quality attributes that can be negatively affected by unknown context events, we propose the EVOLUTION PLANNER, which uses forward chaining. Forward chaining evaluates arising context facts against rule premises, which are defined at design time and kept in a knowledge base. In our previous work, we define how forward chaining can be used to realize if a particular fact (an unknown context event) can trigger an evolution [4]. For example, it is possible to find out that the HKWholesaleSupplier component can affect the High Security quality attribute when this component has rapidly increased its execution time (an unknown context event).

3. **Look for a Tactic to Face the Unknown Context Event:** In our approach, *tactics* are considered as the last resort to be used when the system does not have predefined adaptation actions to deal with arising context events. These tactics are expressed in a *requirements model.* This model is leveraged at runtime to count on the representation of the requirements, including quality attributes, which the context-aware system must preserve at runtime. Requirements in this model have to be fulfilled despite arising unknown context events. Since we are particularly interested in keeping quality attributes or non-functional requirements (NFRs) at runtime (e.g. performance), the Evolution Planner uses GRL [11].

Figure 3 depicts the requirements model for the on-line product shopping system at EuroTech. This model has softgoals that describe the quality attributes to be kept by the context-aware system in order to reach the top-level goal (e.g. the High Security softgoal). It also contains tasks that specify specific surviving tactics to reach the softgoals (e.g. the Deception task). Since tasks represent core assets to keep the QoS of the system, they make a positive contribution to softgoals. The tactics with the highest contributions (i.e., the ones with "++") are chosen first by the Evolution Planner.

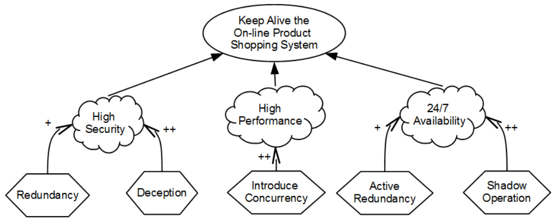


Figure 3. Requirements model

4. **Generate an Initial Population:** In this step, an initial population of chromosomes is auto-generated by the GA Evolver at runtime. Each chromosome is given a random collection of genes. Each gene represents a particular software component (see Figure 4).

A *software component* is a software element that conforms to a component model and can be independently deployed and composed without modification according to a composition standard. Each chromosome in the initial population represents a fully-functional software architecture.

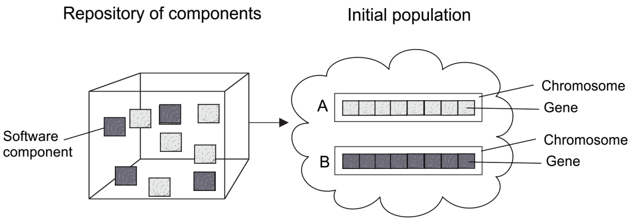


Figure 4. Generation of the initial population of components

Genes are taken from a repository of components. A *repository* is a storage site for objects of some sort. In other words, a repository stores information about an organization's assets. It can store information, store-in-depth documentation, support for versioning and change control, generate name conventions, etc.

For example, in our repository we have common and variant software components. *Common components* have to be present in all the generated chromosomes. However, *variant components* are present in particular chromosomes. For example, on one hand, the GoogleAuthService is a common component that is present in all the chromosomes. On the other hand, the functionality to look for products can be implemented by means of two alternative variant components, the HKWholesaleSupplier and the American Supplies Co. components. Any of these variants can fulfill the expected functionality.

In order to decide which genes (components) have to be present in each chromosome, we make use of a feature model [12]. This model describes the dynamic software architecture configurations, the variants of the software architecture, and constraints among functionalities [8]. Feature modeling was chosen because it offers coarse-grained variability management and has good tool support.

5. **Crossover and Mutation:** Crossover combines the properties of two chromosomes of the previous population to create new chromosomes (software architectures in our case). Mating is achieved by selecting two parents and taking a "splice" from each of their gene sequences. Mutation is used to introduce new genetic material into the population. In our approach, mutations are the tactical functionalities that are used to evolve a chromosome in order to preserve the quality attributes that can be negatively impacted by unknown context events.

For instance, Figure 5 shows how software architectures A and B are mated. The result is software architecture C. In the second step of this section, we described an example in which our solution discovers an unknown context event: the HKWholesaleSupplier component can affect the High Security quality attribute. In this case, GAs are used to insert the functionality of the Deception tactic into the resulting software architecture (see Figure 3). We consider this insertion as a mutation. This tactical mutation will try to preserve the security quality attribute in a particular generated architecture.

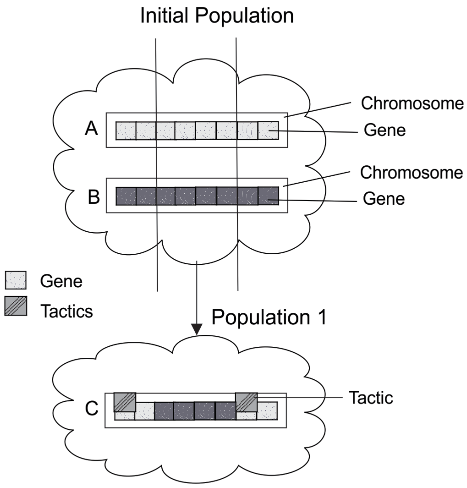


Figure 5. Crossover and mutation of tactical functionalities

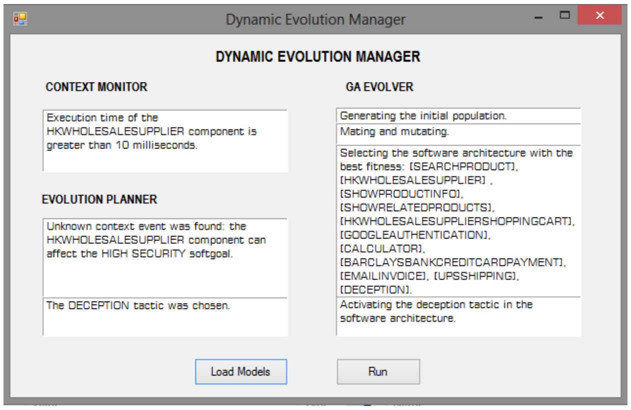
6. **Selection:** In this step, the population is subjected to a selection process that favors individuals better adapted. Each chromosome in the population must be evaluated. This is done by evaluating its "fitness" or the quality of its solution. The fitness is determined through the fitness (or utility) function, making a summation of the functionality (together with the mutated tactics) of the genes that make up the chromosome. Based on the fitness level of each chromosome, it is possible to select the chromosomes that will mate, or those that have the "privilege" to mate in further generations. For example, security levels can be evaluated with different metrics in each resulting chromosome. The chromosomes with the best results will have the best fitness.

Steps 5 and 6 are repeated until the best solution has not changed for a preset number of generations. Finally, our GA approach returns a software architecture that can be used to face the arising unknown context events.

5 Tool Support

A prototype system validates the feasibility of our approach. The Context Monitor and the Evolution Planner are described in our previous work [4, 8, 9]. The Context Monitor uses the SPARQL Protocol and RDF Query Language (SPARQL)[1] to query the ontology with context information. The Evolution Planner uses the EMF Model Query (EMFMQ)[2] to manage the feature model and the requirements model at runtime [4]. The GA Evolver is implemented in Java. Changes in the evolved software architecture can be applied on the underlying system by means of different technologies, such as OSGi bundles[3]. These bundles can be activated or deactivated at runtime according to the evolutionary process.

Figure 6 shows the screenshot of our running prototype, a dynamic evolution manager that implements the components of IBM's MAPE-K loop for the open world. This prototype presents the logs of the activities that are carried out by the Context Monitor, the Evolution Planner, and the GA Evolver when an unknown context event is detected.



Execution time of the HKWHOLESALESUPPLIER component is greater than 10 milliseconds.

Unknown context event was found: the HKWHOLESALESUPPLIER component can affect the HIGH SECURITY softgoal.

The DECEPTION tactic was chosen.

Generating the initial population.

Mating and mutating.

Selecting the software architecture with the best fitness: [SEARCHPRODUCT], [HKWHOLESALESUPPLIER] , [SHOWPRODUCTINFO], [SHOWRELATEDPRODUCTS], [HKWHOLESALESUPPLIERSHOPPINGCART], [GOOGLEAUTHENTICATION], [CALCULATOR], [BARCLAYSBANKCREDITCARDPAYMENT], [EMAILINVOICE], [UPSSHIPPING], [DECEPTION].

Activating the deception tactic in the software architecture.

Figure 6. Running prototype

In the scenario of Figure 6, the Context Monitor detects that the execution time of the HKWholesaleSupplier component is greater than 10 milliseconds. This is an event that was not predefined at design time. Afterwards, the Evolution Planner realizes that this unknown context event can negatively affect the High Security quality attribute. As a result, it looks for a tactic to preserve this quality attribute and it finds the Deception tactic. Then, the GA Evolver uses this information to choose the software architecture with the best fitness. At the end of the process, our GA-oriented approach mutates the original software architecture with the Deception tactic. This tactic will try to deceive the hackers that are trying to violate the security of the HKWholesaleSupplier component at a particular time. As a result, the expected High Security quality attribute is preserved.

[1] http://www.w3.org/TR/rdf-sparql-query/: SPARQL
[2] http://www.eclipse.org/modeling/emf/: EMF Model Query
[3] http://www.osgi.org/Main/HomePage/: OSGi

Preliminary evaluation results show that using GAs to evolve the software architecture is a promising option for the unpredictable open world. This is because of the fact that GAs can be used to optimize a large set of non-linear systems (software architectures) with a large number of variables (software components).

6 Related Work

Over the past decade, a large number of research works have been concerned with self-adaptation. We give an overview of relevant approaches that support self-adaptation.

The MUSIC project [13] focuses on developing self-adaptive mobile applications. The authors use an explicit representation of the environment, in particular an ontology-based model. MUSIC adopts utility functions as adaptation mechanism. MUSIC uses a system model based on a system component meta-model representing the system structure. The variability is implicit in the system model. The main variability mechanism consists in loading different implementations for each component type of the architecture. The system, environment and adaptation representations are fixed at design time. However, MUSIC does not provide mechanisms to manage unexpected changes in the architecture at runtime.

Morin et al. [14] propose a combination of model-driven engineering and aspect-oriented modeling to support self-adaptation. This approach keeps an explicit representation of the system. The system is modeled using a base model that contains the common functionalities and a set of variant models that can be composed with this base model. The variant models capture the variability of the adaptive system. An adaptation model specifies which variant have to be selected depending on the environment of the running application. As adaptation mechanisms they use adaptation rules that specify how the system should adapt to its context. All the models are fixed at design time and cannot be extended at runtime to incorporate unanticipated elements.

RAINBOW [15] is an architecture-based framework to support self-adaptation of software systems. At runtime, the authors use an explicit representation of the system, specifically an abstract architectural model. The adaptation mechanism is explicit and it is based on ECA rules. Since the variability is implicit, it is encapsulated into the adaptation rules. RAINBOW also uses an implicit representation of the context. RAINBOW does not provide mechanisms to allow the modification of the context, system, and rules representations at runtime.

Zhang et al. [16] introduce an approach for creating formal models to support self-adaptive system behavior. Specifically, they use state machine based models (such as Petri nets) to model the system's adaptive behavior. Contextual changes guide the transitions among system states. Then, the adaptation mechanism is explicit through rules. In this work, the underlying mechanisms for dynamic adaptations are fixed at design time and cannot be extended at runtime.

PLASTIC [17] focuses on service-oriented systems. PLASTIC maintains an explicit model of the system. Application alternatives are stored in a repository. However, no new alternatives can be added at runtime. PLASTIC does not provide mechanisms to extend the system, the context and system variants at runtime.

CAPucine [18] builds adaptive systems based on services. CAPucine considers the environment implicitly inside the system. The variability is managed explicitly by means of a feature model. The system is represented explicitly through models. The adaptation mechanism is explicit. A series of rules are stored in a repository. This approach defines all the underlying elements at design time.

CASA [19] provides a framework for enabling dynamic adaptation of applications executing in dynamic contexts. Adaptation mechanisms are defined explicitly by a set of adaptation policies. CASA does not provide mechanisms to modify or extend the specification of the software architecture at runtime.

In the aforementioned approaches, adaptation is fully foreseen at design time. Systems have a fixed set of adaptation actions and new behaviors cannot be introduced during runtime in the software architecture. The trend has focused on self-adaptive mechanisms that are not open to evolution in the open world. According to our best knowledge, our work presents the first generic approach based on GAs to handle unknown context events through the dynamic evolution of the software architecture.

7 Conclusions and Future Work

In this paper, we have presented an approach for the dynamic evolution of software architectures in the open world through GAs. Specifically, our approach deals with unexpected context events and has several benefits: 1) it can guide the creation of context-aware systems that self-evolve when facing unknown context events in order to preserve quality attributes; 2) it can be applied to different domains; and 3) human workload is reduced thanks to the autonomic evolution of the system.

As future work, we will answer the following questions related to the verification of the evolved software architecture: Does a merged tactic accomplish its objective at runtime? Or does the quality attribute continue decreasing? How to verify that the software architecture does not grow excessively with a large amount of merged tactics, which can make the software architecture complex or slow? In this way, some tactics could be automatically removed when the software architecture has reached a stable state.

8 References

[1] J. O. Kephart, D. M. Chess, 2003. The vision of autonomic computing, Computer, vol. 36, no. 1, pp. 41–50.

[2] A.K. Dey, 2001. Understanding and Using Context. Personal and Ubiquitous Computing Journal, Vol. 5 (1), pp. 4–7.

[3] M. Prehofer, C. Schäfer, W. Schlichting, R. Smith, D. Sousa, J. Tahvildari, L. Wong, K. Wuttke, J., 2013. Software engineering for self-adaptive systems: A second research roadmap. In: Lemos, R., Giese, H., Müller, H., Shaw, M. (Eds.), Software Engineering for Self-Adaptive Systems II. Vol. 7475 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 1–32.

[4] G. H. Alférez, V. Pelechano, 2012. Dynamic evolution of context aware systems with models at runtime. In: R. France, J. Kazmeier, R. Breu, C. Atkinson, (Eds.), Model Driven Engineering Languages and Systems. Vol. 7590 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 70–86.

[5] G. Blair, N. Bencomo, and R. B. France, October 2009. Models@run.time, Computer, vol. 42, pp. 22-27.

[6] D. Ashlock, 2006. Evolutionary Computation for Modeling and Optimization. Springer.

[7] M. Mitchell, 1996. An Introduction to Genetic Algorithms. MIT Press.

[8] G. H. Alférez, V. Pelechano, 2011. Context-aware autonomous Web services in software product lines. In: Proceedings of the 2011 15th International Software Product Line Conference. SPLC'11. IEEE Computer Society, Washington, DC, USA, pp. 100–109.

[9] G. H. Alférez, V. Pelechano, R. Mazo, C. Salinesi, D. Diaz, May 2014. Dynamic adaptation of service compositions with variability models, Journal of Systems and Software, Volume 91, Pages 24-47.

[10] L. Bass, P. Clements, R. Kazman. 2013. Software Architecture in Practice, Third Edition. Pearson Education, Inc.

[11] L. Liu, E. Yu, 2004. Designing information systems in social context: a goal and scenario modelling approach. Inf. Syst. 29, 187–203.

[12] D. Batory, 2005. Feature Models, Grammars, Propositional Formulas. In Software Product Lines Conference, ser. Lecture Notes in Computer Sciences, vol. 3714, Springer-Verlag. Springer-Verlag, 2005, p. 7-20.

[13] R. Rouvoy, P. Barone, Y. Ding, F. Eliassen, S. Hallsteinsen, J. Lorenzo, A. Mamelli, U. Scholz, 2014. MUSIC: Middleware support for self-adaptation in ubiquitous and service-oriented environments, in Software Engineering for Self-Adaptive Systems, ser. Lecture Notes in Computer Science, B. Cheng, R. de Lemos, H. Giese, P. Inverardi, J. Magee, Eds. Springer Berlin / Heidelberg, vol. 5525, pp. 164–182.

[14] B. Morin, O. Barais, G. Nain, J. M. Jezequel, 2009. Taming dynamically adaptive systems using models and aspects. In IEEE 31st International Conference on Software Engineering. ICSE'09. pp. 122–132.

[15] D. Garlan, S.-W. Cheng, A.-C. Huang, B. Schmerl, P. Steenkiste, oct. 2004. RAINBOW: architecture-based self-adaptation with reusable infrastructure, Computer, vol. 37, no. 10, pp. 46–54.

[16] J. Zhang, B. H. C. Cheng, 2006. Model-based development of dynamically adaptive software, in Proceedings of the 28th international conference on Software engineering, ser. ICSE'06. New York, NY, USA: ACM, pp. 371–380.

[17] M. Autili, P. Di Benedetto, P. Inverardi, 2009. Context-aware adaptive services: The PLASTIC approach, in Fundamental Approaches to Software Engineering, ser. Lecture Notes in Computer Science, M. Chechik, M. Wirsing, Eds. Springer Berlin / Heidelberg, vol. 5503, pp. 124–139.

[18] C. Parra, X. Blanc, L. Duchien, 2009. Context awareness for dynamic service-oriented product lines, in Proceedings of the 13th International Software Product Line Conference, ser. SPLC'09. Pittsburgh, PA, USA: Carnegie Mellon University, pp. 131–140.

[19] A. Mukhija, M. Glinz, 2005. Runtime adaptation of applications through dynamic recomposition of components, in Systems Aspects in Organic and Pervasive Computing. ARCS'05, ser. Lecture Notes in Computer Science, M. Beigl, P. Lukowicz, Eds. Springer Berlin / Heidelberg, vol. 3432, pp. 124–138.

SESSION

ENERGY EFFICIENT SOFTWARE DESIGN, REQUIREMENTS ENGINEERING, COST ESTIMATION

Chair(s)

TBA

Automation of Energy Performance Evaluation of Software Applications on Servers

Jasmeet Singh, Veluppillai Mahinthan, and Kshirasagar Naik
Dept. of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario, Canada

Abstract—*Although the hardware subsystems, namely, processors, memory, disk, and network interfaces of a server actually consume power, it is the software activities that drive the operations of the hardware subsystems leading to varying dynamic power cost. There are a number of ways to optimize application programs at their design stages but it is difficult for the developers to analyse their applications in terms of power cost on the real servers. In this paper, we present the design of an automated test bench to measure the power cost of an application running on a server. We show how our test bench can be used by software developers to measure and improve the energy cost of two Java file access methods. Another benefit of our test bench has been demonstrated by comparing the energy costs of compression and decompression features provided by two popular Linux packages:* 7z *and* rar. *Overall, this paper makes a contribution to reduce the perception gap between high level programs and the concept of energy efficiency.*

Keywords: energy cost, automated test bench, synchronization, energy efficient software design, file access methods

1. Introduction

Electrical energy is a key resource consumed by all computing platforms [1], [2], and the design of a software application has a significant impact on the power consumption [3]. Various techniques have been suggested to reduce the power consumption of software systems in [3] and [4]. Considering the fact that power bill accounts for a significant portion of the cost to run a data center, it is useful to analyse and minimize the energy cost of applications running on large systems, namely, servers. Although there are a number of ways to optimize the application at its design stage, developers generally do not consider the energy cost of their software while making important design decisions. They find it difficult to measure the energy cost incurred by their workload and know how it behaves on real servers inside data centers. In addition to this, the measurement process takes a lot of human effort and time.

In this paper, we present the design of an automation system, to measure the energy cost of an application running on a server, with the following properties: (i) a power automation software tool (PAST) is developed for automating the measurement process; (ii) the PAST runs on a monitoring computer which is the same machine used by the developer and different than the server under test; (iii) both the application running on a server (Load) and the power measurement instrument are remotely controlled by PAST for synchronization purpose; and (iv) for statistical data collection of power performance, the PAST can repeat a test on the Load multiple times without human intervention.

By using the automated test bench, developers can upload their application to the server and measure the energy cost of running it for the various design choices. By this way, they can concentrate more on the development, without wasting time on the measurement process. By means of our test bench, we validate the claim in the reference [5] that the energy cost of one Java file read method, FileInputStream (M1) is more than the other method, BufferedInputStream (M2). Then we study the impact of introducing a programmer *buffer* to both the methods, by measuring their energy cost of reading a file from the disk with varying buffer sizes. Although M1 consumes more energy than M2, their energy cost is same for a wide range of buffer sizes. In addition, the energy cost of M2 is further reduced after selecting the optimal *buffer size*. The other benefit of our test bench is that it can be used to compare the various functions of software in terms of their energy cost. Nowadays there are many software applications in the market providing the same functionality. The information regarding the energy cost of the same operation by different software applications allows us to chose the energy efficient ones in data centers. We analyse the energy cost of the compression and decompression features of two famous Linux packages: *7z* and *rar*.

The rest of the paper is organized as follows. In Section 2, we briefly present the related work and compare our approach with the other energy measurement tools developed recently. In Section 3, we explain the system model of the automation framework. Implementation details of the automation framework have been explained in Section 4. In Section 5, we explain how the automated system has been used to conduct experiments. Some concluding remarks and directions for future work are provided in Section 6.

2. Related Work

The techniques for understanding the power cost of servers can be categorized into three major groups: (i) *direct measurement* by means of instrumentation of the hardware [6]; (ii) *estimation* by means of power models [7], [8];

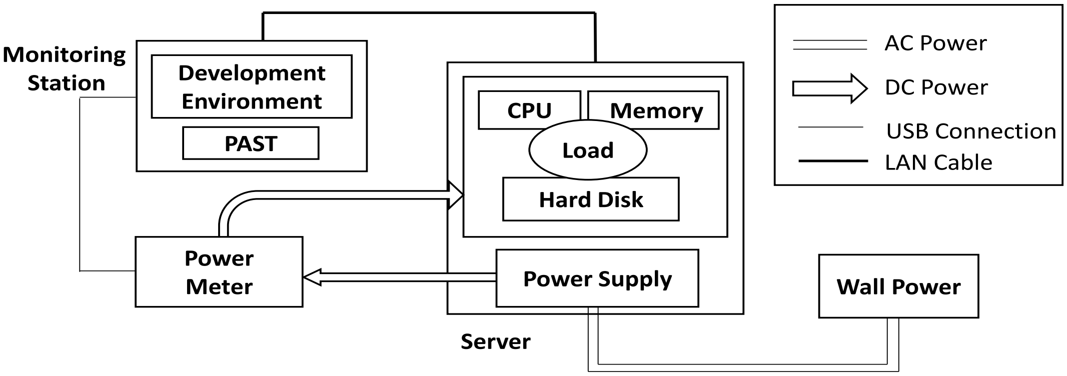


Fig. 1: System Model

and (iii) *software measurement* by means of various tools and application programming interface (APIs). A deeper understanding of power costs of computing subsystems, namely, memory, processor, hard disk, and other peripherals, enables better use of storage encryption, virtualization, and application sandboxing [9], [10]. The authors of the paper [11] studied in detail the effects of abstraction layers and application development environments on the energy efficiency of software. Their results indicate that greater use of external libraries is more harmful in terms of energy cost for large scale applications. Ardito et al. [5] developed the concept of introducing the energy efficiency into SQALE (Software Quality Assessment based on Lifecycle Expectations), one of the software quality models to monitor the impact of software on energy consumption during its development. They identified some energy efficient software guidelines and translated them into measurable requirements of the model. Although direct measurement of power consumption is expensive, it gives more accurate results than estimation models [7].

Our work also falls in the category of direct measurement. The tools used by developers to measure the energy cost of their applications rarely exist. The authors of the paper [12] presented a new tool for mapping software design to power consumption and describe how these mappings are useful for the software designers and developers. In reference [13], a comprehensive survey of different energy measurement approaches has been done. Based on this survey, the authors have come up with four recommendations for the efficient energy measurement approaches: (i) accurate measurements for better precision; (ii) fine-grained power models to trace how and where the energy is being used in software; (iii) reduce user experience impact - the measurement tools should not require manual modifications of source code of the applications; and (iv) software-centric approaches for better evolution and flexibility.

We have also reviewed those efforts similar to our framework which discuss automation of energy measurement. PowerPack [14] and pmlib software [15] have automated the energy profiling of parallel scientific workloads by software code instrumentation. These tools have a set of user level APIs which one can insert before and after the code region of interest to create its energy profile. Both these tools did not talk about the applicability of their APIs to the target code of all programming languages. PowerPack requires additional sensing resistors for each of the power lines in addition to the power meter. Moreover, these tools can not be used to measure the energy cost of closed source applications. In contrast, our framework does not need the manual modification of source code of the application for its energy measurement. In another recent work [16], they have designed a framework called software energy footprint lab, which executes the software of interest on the server and output the power consumed during the execution on a separate machine. Their approach requires manual effort to start the software under test and sending the commands to their Data Acquisition System right before the software is executed and another one right after it terminates, for synchronization. Our approach is different from them as PAST controls both the execution of the software as well as the measurement process. The process of synchronization between the server and the meter is automated in our approach. In addition, the measurement process of the same application can be repeated a number of times for statistical significance.

3. System Model of Test Bench

The system model of the automation framework has been shown in Figure 1. The definitions of all the terms used in the figure are given below.

Server: A system for which we are interested in evaluating the energy cost of running an application.

Load: A software application that runs on the server, and we measure the energy cost of running that application.

Power Meter: A data acquisition unit used for measuring power. We used a Lab-Volt 9063-00 Data Acquisition and Control Interface as a power meter.

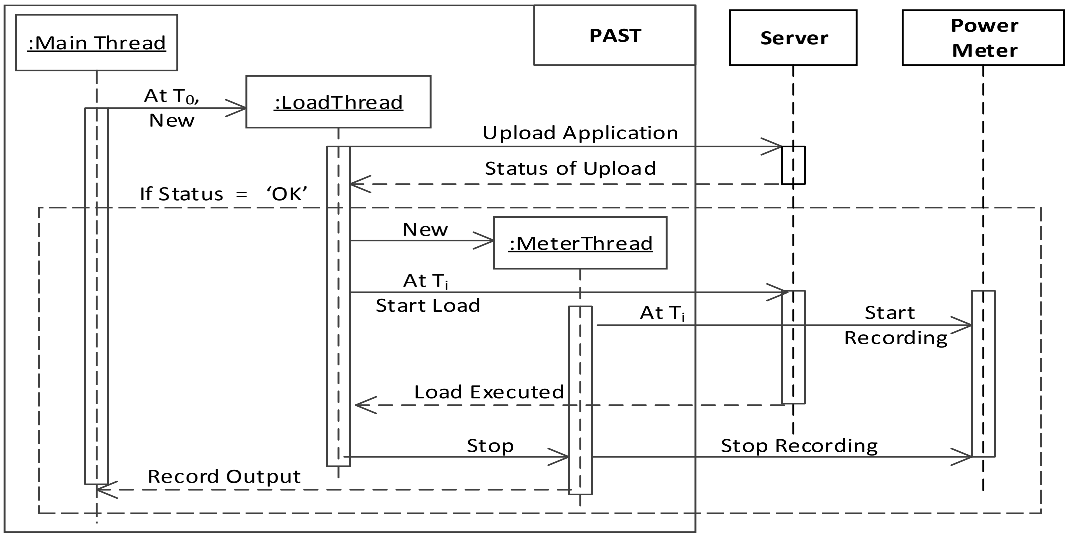


Fig. 2: Message Sequence Chart

Wall Power: Supplies AC power to the Server.
Monitoring Station: A computer equipped with the PAST system which controls both the Server and Meter. A programmer is developing his application on this machine and can run PAST to upload the application to the Server and measure its cost. By running PAST on a separate machine, it starts executing the Load on the Server as well as starts the Meter to record current and voltage values simultaneously.

The Monitoring Station is connected to the Meter via an USB (Universal Serial Bus) interface and to the Server through a LAN (Local Area Network).

Our test bench can be used to measure: (i) the power consumed by a server's individual subsystems, namely, memory, disk, and processor, if their power lines are easily accessible; and (ii) the total power cost of a server. Only the total power can be measured for a server where one cannot identify the power lines to its individual subsystems. To set up the test bench for power measurement of individual subsystems, we examined the different power lines from the ATX 24 pin connector which powers the whole motherboard of desktop computer. The power lines to the processor (CPU: central processing unit) operate at $12V$, first fed to the voltage regulator module which converts the voltage to the actual voltage required by the processor [17]. From the 24 wires of ATX connector, one yellow wire of $12V$ is feeding power to the processor. The other two yellow wires are from the ATX 4 pin $12V$ Power Connector (ATX v2.2) dedicated for the processor. The disk (Hard Disk) is getting power from a Molex 4 pin power supply connector which operates at two voltage levels, $5V$ and $12V$. The memory (RAM: random access memory) system is getting power over three lines from the 24 pin connector, and the voltage level is $3.3V$. The total power cost can be measured from the AC (Alternating Current) power lines to the server power supply.

4. Automated Test Bench

In our test bench, we use a Lab-Volt 9063-00 Data Acquisition and Control Interface system, known as the Meter in this paper. To read power samples from Meter, the device supports APIs in the form of Microsoft Dynamic Link Library (DLL). Therefore, the PAST is developed in Visual Basic. In the remainder of this section, we explain the design of PAST by means of its behaviour, which is represented as a message sequence chart, and then the key problems faced in the design of the PAST.

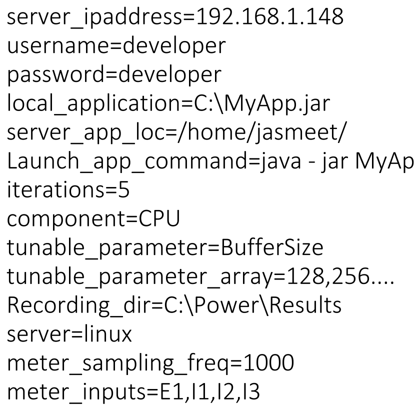
4.1 Message Sequence Chart

Figure 2 shows the sequence of steps of the PAST executed during the whole process of measurement.

PAST is a multi-threaded system, with three threads: MainThread, LoadThread and MeterThread. The PAST is launched on the Monitoring station with the location of the configuration file as its input parameter. A configuration file is a text file that is stored on the Monitoring Station, it contains both the Server and Meter information. Figure 3 shows some entries from configuration file. The behaviours of the three threads is described below: **MainThread:** MainThread first reads the configuration file for the $server\_ipaddress, username, password$ and the location on the server $(server\_app\_loc)$ where the developer wants to upload the application. $Launch\_app\_command$ contains the command to start an application on the server. $meter\_inputs$ in the configuration file tells which current and voltage input channels of the Meter and at what sampling frequency $(meter\_\ sampling\_freq)$ the Meter should produce those values. It then starts the LoadThread and waits for the other threads to finish.
LoadThread: This first uploads the application onto the server. It stops the process if the application is not uploaded

successfully. If upload is successful, then it initializes the Meter with the meter information being read from the configuration file. If the Meter is ready to read then it starts a new thread MeterThread and starts the application on the Server.

MeterThread: This starts recording the current and voltage values from the Meter by using meter API calls. LoadThread ensures that the MeterThread is recording the values till the application is running on the server. And finally it saves all the values in to the file inside directory $(Recording\_dir)$ on the Monitoring Station.



```
server_ipaddress=192.168.1.148
username=developer
password=developer
local_application=C:\MyApp.jar
server_app_loc=/home/jasmeet/
Launch_app_command=java - jar MyApp.jar
iterations=5
component=CPU
tunable_parameter=BufferSize
tunable_parameter_array=128,256....
Recording_dir=C:\Power\Results
server=linux
meter_sampling_freq=1000
meter_inputs=E1,I1,I2,I3
```

Fig. 3: Sample Configuration file

From the recorded values, energy cost of running an application is computed by using the expression:

$$Energy\_cost = \sum_{\forall i} V(i).I(i).\Delta t \qquad (1)$$

where $V(i)$ and $I(i)$ are the i^{th} voltage and current samples, respectively, and Δt is the sampling interval.

4.2 Key Challenges

There are some practical problems in measuring the energy cost of an application at the subsystem level, namely, processor, memory, and hard disk. There are only 4 current and voltage inputs to the meter. Therefore, at a time only 4 power channels can be measured. However, in case of our desktop computer, for all the three components (processor, memory and disk), there are a total of 8 power lines needed to be monitored. Therefore, we measured the power cost of the three subsystems in three repeated experiments.

5. Experiments and Results

In this section, we show how software developers can use our test bench to evaluate the energy performance of running an application on a server with various design options. We compare the energy cost of two Java file access methods: (i) M1 using FileInputStream only and (ii) M2 using BufferedInputStream. Ardito et al. [5] intuitively claim about the energy efficiency of these two methods without any measurements. First, we validate their claim by measuring the energy cost of the methods on our test bench. Then we revise the two methods by introducing a *buffer* into them and measure their energy cost with varying buffer sizes. We also compare the revised methods to read extremely large files in terms of their energy cost. Next, we compare the energy performance of two packages *7z* and *rar* with respect to compression and decompression. Table 1 shows the configurations of two machines used in our experiments.

Table 1: Server Machines Configuration

Parameter	Desktop (ASUS P4P800-VM)	Real Server (Dell PowerEdge 2950)
Processor	Intel Pentium 4, 3.2 GHz	7x Intel Xeon, 3 GHz, 4 cores per processor
Hard Disk	80 GB IDE	1.7 Tera Bytes SAS
Main Memory	2 GB DIMM	32 GB DIMM
Operating System	Linux (Ubuntu 13.10)	Linux (Ubuntu 13.10)

5.1 Example of using test bench to make important design decisions

Listing 1 and Listing 2 in Figure 4 describe M1 and M2, respectively. We measure the energy cost of CPU, memory and disk for reading a video file of size 512 MB (Mega Bytes) with M1 and M2 on a desktop machine. Figure 4 shows the results of our measurements by comparing the energy cost of all the three components for both M1 and M2. The reason behind the less energy consumption by M2, for all the components is that it reads a file of any size in larger chunks equal to the size of its internal buffer from the disk, whereas M1 reads a single byte of data in one read operation. It is clear from the results that CPU consumes the maximum energy in reading a file.

We further study the impact of introducing a programmer defined *buffer* into both the methods. Listing 3 and Listing 4 describe the modified code of the two methods, and they are denoted by M1' and M2' corresponding to M1 and M2, respectively. In both M1' and M2', line #2 shows the definition of *buffer* as an array of type byte, and its size is equal to *bufferSize*. Line #4 and #5 of M1' and M2' respectively, show that in one call *read* operation reads several bytes of data of size, *bufferSize*. Therefore *bufferSize* is a tunable parameter which the developer can vary and run these methods to read a file. We measure the energy cost of CPU, memory and Disk for both M1' and M2' with buffer size ranging from 1 Byte to 64 Mega Bytes (MB).

Figure 5 shows the evaluation of the total energy cost of all the three components for both M1' and M2'. The results in Figure 5 show that after introducing a programmer buffer into M1 and M2, the total energy cost of all the three components, is maximum at buffer size 1 byte. It started

```
FileInputStream fis = new FileInputStream(fileName);
int b,cnt = 0;
while ((b = fis.read()) != -1)
{
        if (b == ’\n’)
                cnt++;
}
fis.close();
```

Listing 1. **M1**: File Reading using FileInputStream

```
FileInputStream fis = new FileInputStream(fileName);
BufferedInputStream bis = new BufferedInputStream(fis);
int b,cnt = 0;
while ((b = bis.read()) != -1)
{
        if (b == ’\n’)
                cnt++;
}
fis.close();
```

Listing 2. **M2**: File Reading using BufferedInputStream

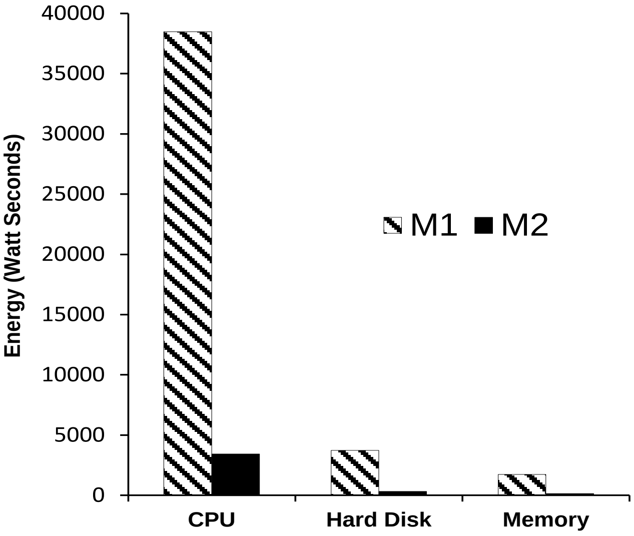


Fig. 4: Energy cost evaluation of CPU, Memory and Disk for M1 and M2 on the Desktop

```
FileInputStream fis = new FileInputStream(fileName);
byte[] buffer = new byte[bufferSize];
int b,cnt = 0;
while ((b = fis.read( buffer )) != 1)
{
        if (b == ’\n’)
                cnt++;
}
fis.close();
```

Listing 3: **M1’**: Introducing user buffer in M1

```
FileInputStream bis = new FileInputStream(fileName);
byte[] buffer = new byte[bufferSize];
BufferedInputStream bis = new BufferedInputStream(fis);
int b,cnt = 0;
while ((b = fis.read( buffer )) != 1)
{
        if (b == ’\n’)
                cnt++;
}
fis.close();
```

Listing 4: **M2’**: Introducing user buffer in M2

decreasing with the increase in the buffer size till 128 bytes. We expanded the graphs of Figure 5 in Figure 6 to show the energy cost of individual components along with their total energy cost at the buffer sizes from 128 bytes to 64 MB.

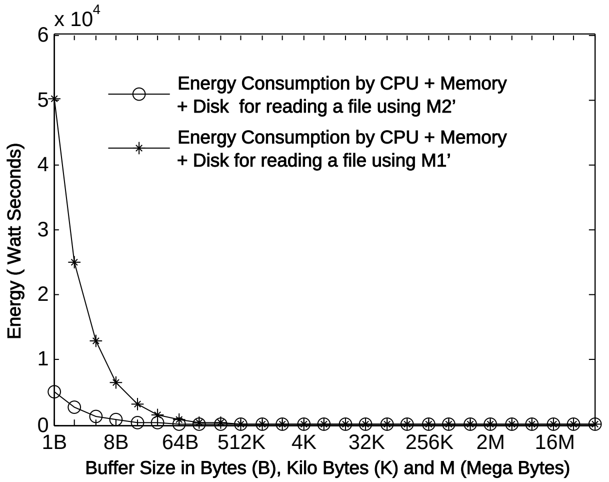


Fig. 5: Total energy cost by M1’ and M2’ with different buffer sizes on the Desktop

Figures 6(a), 6(b) and 6(c) show the energy cost of CPU, memory and disk respectively, and their total energy cost in 6(d). The energy cost behaviour between M1’ and M2’ is same as between M1 and M2 for buffer sizes from 128 bytes to 8KB; in other words, energy cost of M2’ remains less than M1’. Then, energy is constant for both the methods ranging from 8KB to 128KB, except that there is a sharp increase at 32KB by M1’ for disk. It started increasing from 128KB to 1 MB then decreases and remains constant till 64MB. Both M1’ and M2’ consume almost the same energy for all the three components from 8KB to 64MB and consumes minimum energy at 16KB. Moreover, this energy is even less than M2. Therefore, it is clear from our measurements that there is a further opportunity to decrease the energy cost of M1 and M2 by introducing a programmer buffer into them. Both the methods consume almost the same energy at buffer sizes ranging from 8K to 64 MB which contradicts the claim by Ardito et. al [5] that M1 always consumes more energy than M2. In addition to this, 16K is the optimal buffer size for all the three components.

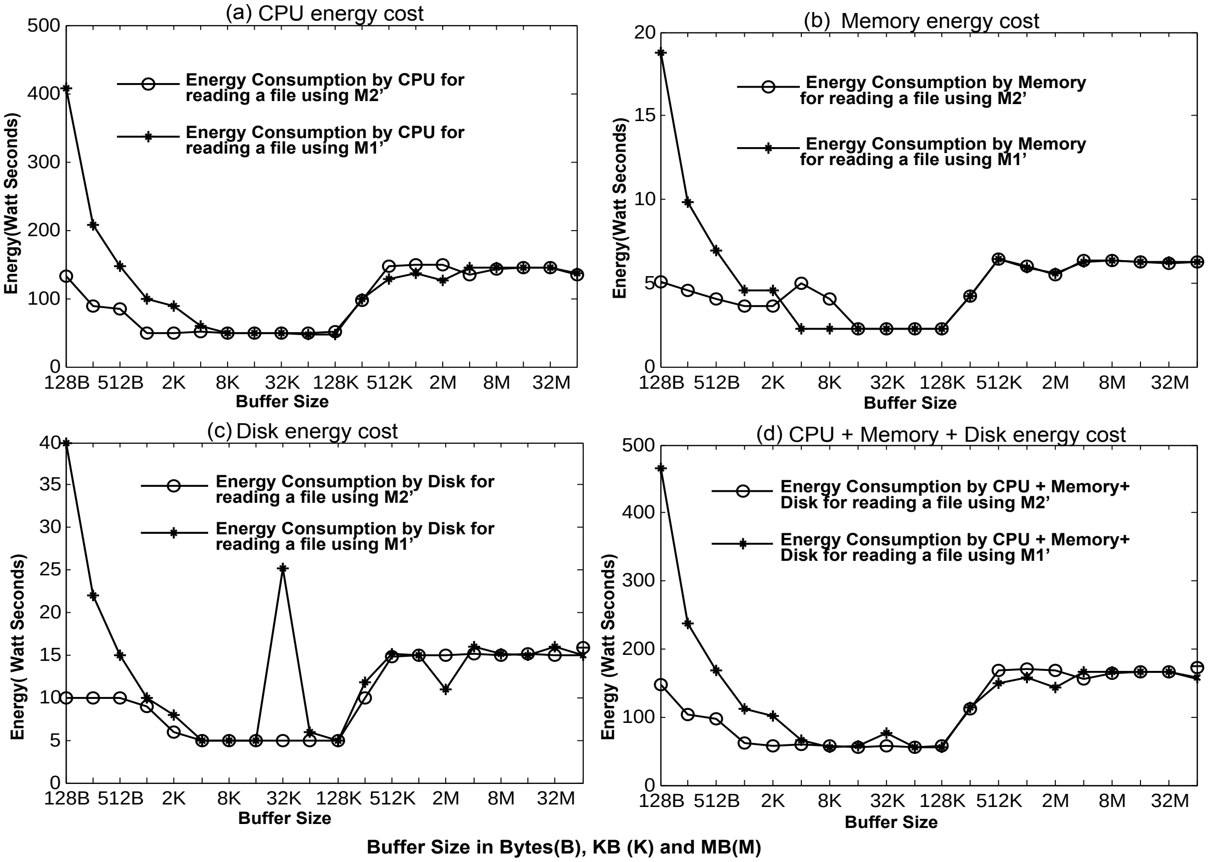


Fig. 6: Energy cost of M1' and M2' with different buffer sizes on the Desktop

To gain additional insights into the behaviours of M1' and M2' while reading extremely large files, we perform the experiments on the same desktop machine to read files ranging from 1MB to 32 Giga Bytes(GB), while keeping the buffer size fixed at 16KB. Figure 7 shows the graphs plotted for the total AC (Alternating Current) energy cost as function of different file sizes for both the methods at 16KB buffer size. It is clear from the graph that both the methods consume the same energy at 16KB buffer size. We close this section by noting the above results enable the developer to chose the right method for reading a file with appropriate buffer size during the design stage.

5.2 Using the test bench in function level energy cost measurement

As discussed in Section 1, the test bench can also be used to measure the energy cost of a specific function of an application software whether it is open source or closed source. To validate this functionality of our test bench we conducted the experiments on a real server (Table I) from a data center. We consider two popular Linux packages, namely, *7z* and *rar* to compress and decompress files. Both the packages output compressed files in *.rar* and *.7z* formats and can decompress the same to the original files. A video file of size 512 MB is used in our experiment for compression. Figure 8 shows the total AC (Alternating Current) energy cost of a server for the four functionalities of both the packages. The results show that the *7z* package consumes more energy in compressing the files to *.rar* and *.7z* formats compared to the *rar* package. However the *rar* package consumes less energy in producing *.rar* files than it consumes to produce in *.7z* format. In case of decompression from *.rar* format, *7z* consumes more energy than it consumes while converting from *.7z* format while *rar* consumes the same energy in decompressing *.rar* and *.7z* formats to produce the original files. Further investigation is required to find the causes of energy cost differences of the same operations of two packages.

6. Conclusion and Future Work

In this paper we presented an automation framework to measure the energy cost of servers while running software applications. The framework's infrastructure mainly contains a power meter, target server and control software (PAST) for synchronization and monitoring. By using the test bench, we performed actual measurements to verify the claim in a previously published paper [5] that energy cost of reading

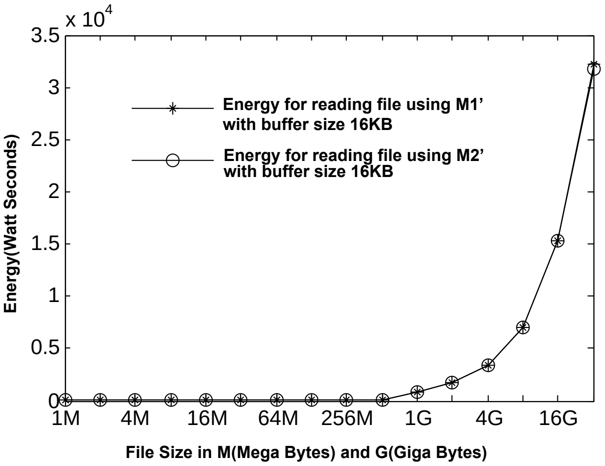


Fig. 7: Energy cost of M1' and M2' for different file sizes on the Desktop

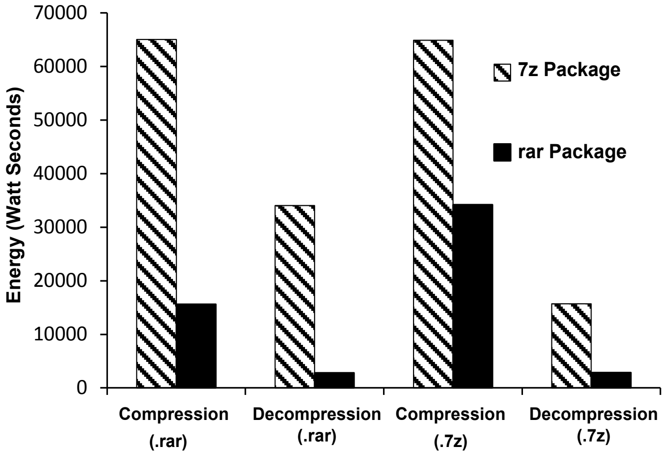


Fig. 8: Energy cost of 4 functions of *rar* and *7z* packages on the real server (Table I)

files by the method FileInputStream (M1) is greater than the BufferedInputStream (M2) method. However this claim is not valid in certain cases, if we introduce a programmer buffer in both the methods. It holds good for buffer sizes ranging from 128 bytes till 8KB, but these two methods consume almost the same energy at buffer sizes from 8KB to 64MB. Also, the introduction of *buffer* in M2 has further reduced its energy cost. Finally, we compared the energy costs of the same functionality provided by different software applications by measuring the energy costs of compression and decompression features of two Linux packages: *7z* and *rar*. The *7z* package consumes more energy than rar in compressing and decompressing files. However, *rar* consumes more energy in compressing to *.7z* format than to *.rar* format. The automation framework can be used by programmers to evaluate the energy cost of their applications. More work is required to be done to find out the causes of energy cost differences of the same operations of two packages.(Figure 8)

References

[1] T. Mudge, "Power: A first class design constraint for future architectures," in *High Perf. Computing*. Springer, 2000, pp. 215–224.

[2] K. Naik and D. S. Wei, "Software implementation strategies for power-conscious systems," *Mobile Networks and Apps*, vol. 6, no. 3, pp. 291–305, 2001.

[3] M. Sabharwal, A. Agrawal, and G. Metri, "Enabling green it through energy-aware software," *IT Professional*, pp. 19–27, 2013.

[4] D. J. Brown and C. Reams, "Toward energy-efficient computing," *Communications of the ACM*, vol. 53, no. 3, pp. 50–58, 2010.

[5] L. Ardito, G. Procaccianti, A. Vetro, and M. Morisio, "Introducing energy efficiency into sqale," in *ENERGY 2013, The Third Intl. Conf. on Smart Grids, Green Communications and IT Energy-aware Technologies*, pp. 28–33.

[6] T. Stathopoulos, D. Mclntire, and W. J. Kaiser, "The energy endoscope: Real-time detailed energy accounting for wireless sensor nodes," in *Information Processing in Sensor Networks, 2008. IPSN'08. International Conference on*. IEEE, pp. 383–394.

[7] J. C. McCullough, Y. Agarwal, J. Chandrashekar, S. Kuppuswamy, A. C. Snoeren, and R. K. Gupta, "Evaluating the effectiveness of model-based power characterization," in *USENIX Annual Technical Conf*, 2011.

[8] Y. Sun, L. Wanner, and M. Srivastava, "Low-cost estimation of subsystem power," in *Green Computing Conference (IGCC), Intl.* IEEE, 2012, pp. 1–10.

[9] P. A. P. D. S. William, J. Kaiser, and P. L. Reiher, "Investigating energy and security trade-offs in the classroom with the atom leap testbed," 2011.

[10] D. McIntire, K. Ho, B. Yip, A. Singh, W. Wu, and W. J. Kaiser, "The low power energy aware processing (leap) embedded networked sensor system," in *Proceedings of the 5th intl. conf. on Information processing in sensor networks*. ACM, 2006, pp. 449–457.

[11] E. Capra, C. Francalanci, and S. A. Slaughter, "Is software green? application development environments and energy efficiency in open source applications," *Information and Software Technology*, vol. 54, no. 1, pp. 60–71, 2012.

[12] C. Sahin, F. Cayci, J. Clause, F. Kiamilev, L. Pollock, and K. Winbladh, "Towards power reduction through improved software design," in *Energytech, IEEE*, 2012, pp. 1–6.

[13] A. Noureddine, R. Rouvoy, and L. Seinturier, "A review of energy measurement approaches," *ACM SIGOPS O.S. Review*, vol. 47, no. 3, pp. 42–49, 2013.

[14] R. Ge, X. Feng, S. Song, H.-C. Chang, D. Li, and K. W. Cameron, "Powerpack: Energy profiling and analysis of high-performance systems and applications," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 21, no. 5, pp. 658–671, 2010.

[15] S. Barrachina, M. Barreda, S. Catalán, M. F. Dolz, G. Fabregat, R. Mayo, and E. S. Quintana-Ortí, "An integrated framework for power-perf. analysis of parallel scientific workloads," in *ENERGY 2013, The Third Intl. Conf. on Smart Grids, Green Communications and IT Energy-aware Technologies*, pp. 114–119.

[16] M. A. Ferreira, E. Hoekstra, B. Merkus, B. Visser, and J. Visser, "Seflab: A lab for measuring software energy footprints," in *Green and Sustainable Software(GREENS), 2nd Intl. Workshop*. IEEE, 2013, pp. 30–37.

[17] C. Isci and M. Martonosi, "Runtime power monitoring in high-end processors: Methodology and empirical data," in *Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 2003, p. 93.

Environment for Requirements Elicitation Supported by Ontology-Based Conceptual Models: A Proposal

J. Valaski, S. Reinehr, and A. Malucelli
PPGIa, Pontifícia Universidade Católica do Paraná, Curitiba, Paraná, Brazil

Abstract - *Requirements Elicitation is a Requirements Engineering activity that aids understanding of the customer's needs. The lack of understanding of the problem domain, the poor communications among stakeholders and a lack of consensus regarding the use of terms, are some of the main challenges of this activity. Ontologies are a type of formalism that can be applied to aid understanding and reach a consensus in a knowledge domain. In this context, the goal of this study is to present a proposal for an environment that supports requirements elicitation from the early stages, when communications are more informal, to the point where a list of software requirements is obtained. This environment will be supported by natural language processing techniques and ontology-based conceptual model represented in OntoUML. An experiment is described with the objective to discuss the opportunities and difficulties that arise on the way to achieve the desired results.*

Keywords: Requirements Elicitation; Ontology; Natural Language Processing; OntoUML**.**

1 Introduction

In the context of software development, Requirements Engineering is the field that supplies the appropriate mechanisms to understand customer's needs [1]. Requirements Engineering can be characterized as an iterative process of discovery and analysis to produce for producing a clear, complete and consistent set of requirements of software [2] that can be represented in different forms. The initial result of the Requirements Engineering process is the production of several personal and not very clear visions of a system represented by informal languages, while the final result is a complete specification of the system represented formally, which all those involved have achieved and accepted. At the beginning of the process, it is common to see languages that are more informal, while at the end the specification is normally represented in formal languages [3].

Poor understanding of the business on the part of the requirements engineers and poor communication between the business specialists and computing specialists can compromise the quality of information [4]. Therefore, during software development, especially in its early stages, it is necessary to use a common language that enables shared understanding among stakeholders to aid the smooth flow of information that is obtained from different sources [5]. Broader understanding of the problem domain is also fundamental in terms of communication and quality.

The conceptual model is an instrument that enables the use of a common vocabulary and facilitates the comprehension and discussion of elements that may appear in the system. One of the most known conceptual metamodel is the Entity-Relationship (ER). Nevertheless, the reason of the ER popularity is also its main weakness: the metamodel is simple and this feature helps the conceptual modelers, however, the metamodel does not present high expressivity. The UML is also a well-known language to build conceptual models, but it presents the same problem of expressivity.

Ontologies have been an important element to support the construction of more expressive conceptual models. The main objectives of ontologies in this context include enabling a common shared language [6], aiding understanding of the problem domain [7], supporting the analysis of the expressivity of the language [8] and providing a language that represents a domain that is as close as possible to the real world [9].

Guided by these matters, Guizzardi [10] proposed OntoUML, a language used to represent ontology-based conceptual models. As the language is ontology-based, the assumption is that the conceptual models constructed in OntoUML are more expressive and represent the real world of the domain more faithfully than other languages of conceptual representation. The constructs proposed in OntoUML prevent the overload and redundancy found in other languages, such as, UML. Therefore, the problem of understanding the domain for correct requirements elicitation could be minimized if these models represented in OntoUML were used to derive the requirements. However, as OntoUML is a more expressive language, it proposes a larger set of constructs that are not easily identified, especially by novice modelers [11].

In this context, the present study aims to propose an environment that supports the construction of conceptual models by processing texts to derive initial requirements from a system to be developed. The presentation of this proposal is organized as follows: in Section 2, the basic concepts of the proposal are outlined; in Section 3 the works related to this proposal are presented; Section 4 presents the architecture of the proposed environment; an experiment is conducted and described in Section 5 to demonstrate and discuss the viability of the proposal; and in Section 6 the final considerations and recommendations for future works related to the proposal are presented.

2 Theoretical framework

2.1 Ontology in Requirements Engineering

With their origins in philosophy, ontologies were employed to categorize knowledge and represent it through taxonomies. The current definition of an ontology has been improved and adapted according to the field in which it will be applied. In computer science, one of the most well-known definitions is given by Gruber [12]: “An ontology is a specification of a conceptualization”.

In the field of Requirements Engineering, ontologies have been applied for different purposes: a common shared vocabulary [6]; the reuse and restructuring of knowledge [13]; understanding the problem domain [7]; analyzing language expressivity [8]; representation of the domain that is more faithful to the real world [9], and improvement of communications among specialists in different domains [14].

According to the objectives of the application of ontologies, the approach can be conceptual or computational. On this subject, Guizzardi [15] suggests a need for two classes of representation language in ontology engineering. The first class, referred to in this work as “conceptual” has to do with languages that are philosophically well founded, focusing on expressivity and conceptual clarity. The second class, referred to in the present study as “computational”, are languages that focus on computing matters such as decidability and automated reasoning. The class of language used in this work is conceptual.

2.2 OntoUML

The OntoUML language proposed by Guizzardi [10] was motivated by the need for an ontology-based language that would provide the necessary semantics to construct conceptual models with concepts that were faithful to reality. The classes proposed in OntoUML are specializations of the abstract classes of the Unified Foundational Ontology (UFO) and extend the original metamodel of UML.

Although OntoUML represents several types of categories, due to space limitations, in this paper only the main constructs that make up the object type category will be presented [11]. In this category, constructs are more closely related to the static conceptual modeling of a domain. The hierarchical structure of these models is described in Fig.1.

The Object Type constructs can be Sortal and Non-Sortal. The sortal provides identity and individualization principles to their instances, while the non-sortal does not supply any clear identification principles.

The Sortal constructs are classified as Rigid Sortal and Anti-Rigid Sortal. A Sortal is said to be rigid if it is necessarily applied to all its instances in all possible worlds. A Sortal is said to be anti-rigid if it is not necessarily applied to all its instances. To demonstrate this difference, the “Person” concept can be used as a Rigid Sortal and the “Student” concept as an Anti-Rigid sortal.

The Rigid Sortal includes the Kind and Subkind categories. A Kind is a Rigid Sortal, and therefore has intrinsic material properties that provide clear identity and individualization principles. It determines existentially independent classes of things or beings and are said to be functional complexes. These can be natural (dog, person) or artifacts (television, house). A Subkind is also a rigid type that provides the identity principle and has some restrictions established and related to the Kind construct. Every object in a conceptual model must be an instance of only one Kind; Subkinds of one Kind must appear in the form of a disjoint partition, e.g., the “Person” concept is a Kind and the concepts of “Man” and “Woman” are Subkind.

There are two sub-categories of Anti-Rigid Sortal: Phases and Roles. In both cases, the instances can change their types without affecting their identity. Whereas during the Phase construct the changes can take place as a result of changes of intrinsic properties, e.g., “Child”, “Adolescent”, “Adult,” which are concepts phases related to the “Person”. The alteration of intrinsic property age of a person causes a change in phase. In Role construct, the changes take place because of relational properties, e.g., “Student”, “Husband” and “Wife” which are concept roles related to “Person”. In the case of student, the relationship with a learning institution determines the role. The same applies to Husband and Wife.

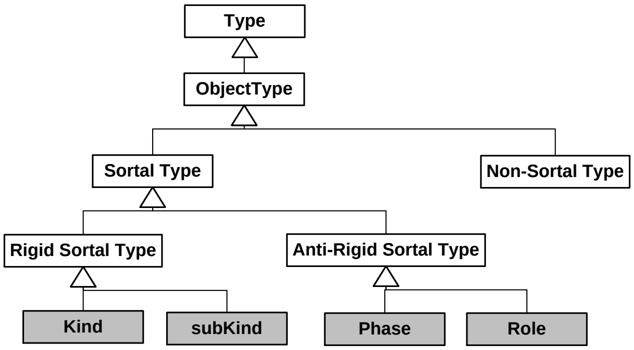


Figure 1 Fragment of metamodel (Guizzardi, 2005)

In comparison with UML, OntoUML has a larger set of constructs, enabling greater expressivity of conceptual models and avoiding overload and redundancy [10]. Nevertheless, it is more complex to use it than using traditional languages such as UML, especially for novice modelers [11]. One of the difficulties of constructing a model represented in OntoUML is identifying the correct construct for a given concept to be represented. In this sense, it is important to develop automatic or semi-automatic mechanisms that help the domain modeler to identify this concept and its correct construct. A linguistic approach with a semantic focus can be applied to aid comprehension of the concepts to be modeled [16].

2.3 Semantic types and disambiguation

Dixon [17] proposes a semantic organization for words in classes of meaning known as semantic types. In this proposal, semantic types handle nouns, adjectives and verbs. Generally, in conceptual modeling, nouns are the semantic types that indicate important concepts in a conceptual modeling. That being the case, due to space limitation, only some categories of semantic types related to nouns will be given as example. For the semantic noun, five main categories are proposed: Abstract Reference, Activities, Concrete Reference, States and Speech Acts. Each of these categories may have sub-categories. Concrete Reference, has the following categories, for instance:

- Animate: living animal beings, but not human beings;
- Human: living human animals;
- Parts: nouns whose referents are components parts of an individual; and
- Inanimate: nouns referring to inactive, non-living things.

The semantic types can be mapped to the constructs of OntoUML [16] and thereby enable semi-automatic support for their identification using Natural Language Processing (NLP). However, one of the challenges of automatic identification of the semantic type is the disambiguation of the term [16][18].

A word can have several meanings and the correct identification of its meaning may depend on the context in which it is used. The task of computationally identifying the meaning of words based on their context is known as Word Sense Disambiguation (WSD) [19]. There are techniques and algorithms for disambiguation, such as TargetWord, which is applied only in the case of a target word in a sentence, and the AllWord, which is applied to all words in a sentence. An example of a disambiguation technique is Wordnet::SenseRelated [19]. Wordnet:SenseRelate is based on Wordnet, which is a lexical base for the English language. This tool performs the disambiguation of a term found in the base and also identifies the corresponding semantic type.

3 Related work

One of the first works found in literature proposing the use of ontologies for constructing conceptual models is presented by [20]. In this work, a methodology was proposed for knowledge acquisition based on the use of natural language and mental models. The methodology is composed of an interview, diagrams and a conceptual analysis. The conceptual analysis was conducted through the construction of an ontology in which what constitutes the discourse domain is defined. The ontology was applied to limit the vocabulary that the computer would have to understand.

In [21] proposed domain ontologies and task ontologies to be used in a domain-oriented software development environment. The domain and task ontologies were built in the fields of cardiology, acoustic propagation and entomology (the study of insects). The ontology was represented using Prolog. In this proposal the role of the ontologies was to aid understanding of the domain and its tasks, contributing with elicitation and specification, such as what is written in use cases.

In [22] texts and use cases were processed to extract relevant terms from the problem domain. An ontology was then created that would later be used to identify the system object model. The ontology was generated in XML and RDF. The objective of the ontology was to provide metadata that would offer controlled vocabulary from the domain and transform the textual requirements into an object model.

In [23] a methodology was proposed for the development process of multi-agent software. The methodology considers d. The use of OntoUML helps on understanding of the discourse universe, i.e., the domain, without concern over the software to be developed.

The ontology-based conceptual model support the understanding of the domain as it is organized. Fig. 2 shows the aspects from the requirements analysis up to the implementation of the system. In the requirements stage, the use of the Problem Ontology Domain (POD) was proposed to obtain a general view of the problem domain. The ontology helps to identify concepts during the requirements phase through class diagrams. The role of the ontology was to provide an associated vocabulary to the problem domain and reuse knowledge.

The creation of a domain ontology derived automatically from texts described in Controlled Natural Languages (CNL) was proposed in [24]. The domain ontology model was later transformed into a UML class model. The ontology was generated using the Protégé tool and represented in OWL. The objective of the ontology was to support the creation and validation of artifacts generated from descriptions in CNL up to the class diagram. A linguistic approach with a semantic focus on comprehending the concepts to be modeled adopting OntoUML was proposed in [16]. As an extension of this proposal, a semi-automatic method was proposed for the learning of well-founded ontologies through disambiguation of terms [18].

Out of the works that were analyzed, those most closely related to this proposal are [24], [16] and [18]. In proposal [24] the modeling language is UML, which limits the building of ontology-based conceptual models. In proposals [16] and [18], OntoUML is used for conceptual modeling, but the method that is proposed has no environment that provides computational support from the building of the conceptual model to the derivation of requirements.

4 Proposed environment

As shown in the previous section, works can be found in literature that have already proposed and applied ontologies to support activities related to Requirements Elicitation using conceptual modeling as a base. However, no proposals were found for the derivation of requirements from ontology-based conceptual models. Neither was any proposal found for an environment that provides computational support to all tasks involved in the elicitation process up to the derivation of requirements using the ontology-based conceptual model. Therefore, this study is motivated by the proposition of this environment.

The proposed environment should support the identification of important terms pertaining to a domain, the disambiguation of relevant terms, the building of the conceptual domain model and the derivation of requirements in this domain. Identification of important terms can be done through electronic messages or the processing of documents exchanged by project stakeholders.

NLP tools could be used to identify the relevant terms and identify the semantic types. The initial lexical base could be the WordNet. To build the ontological conceptual model the language could be OntoUML as it is ontology-base

architecture and flow of tasks to be semi-automated by the proposed environment.

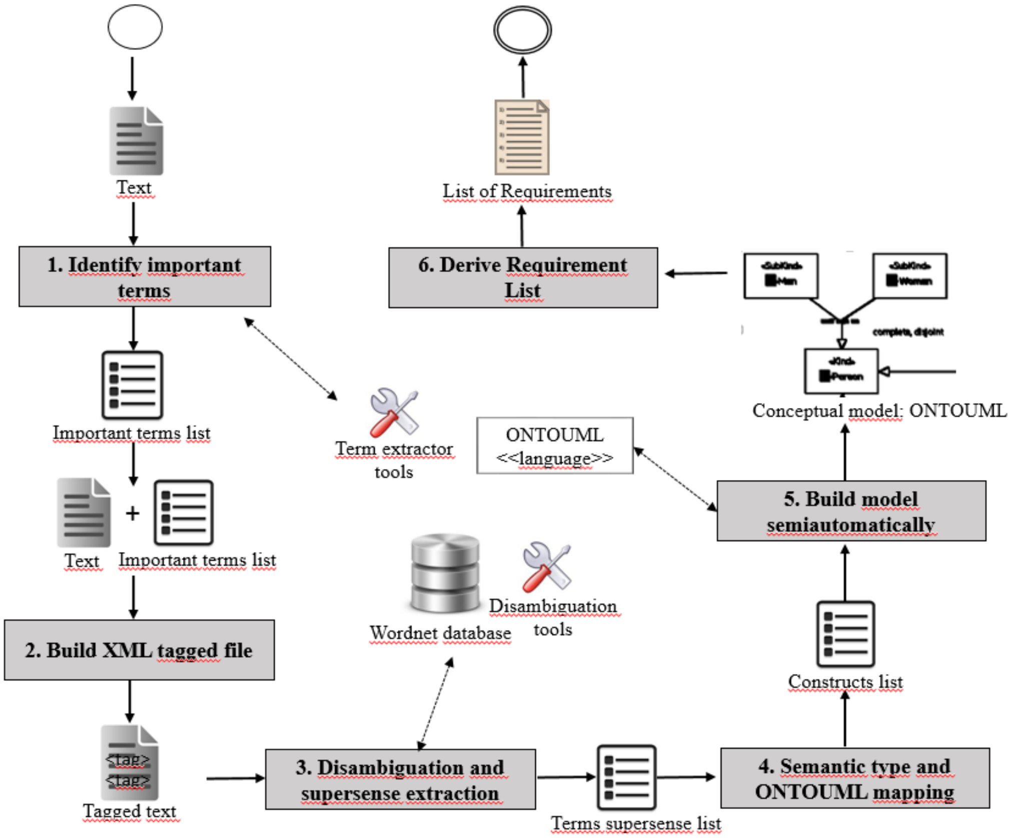


Figure 2 Architecture of the proposed environment

Part of the flow is based on the model proposed by [18]. The text to be processed could be originated from dialogues among stakeholders or documents (laws, internal documents, e-mails, etc.) that describe the domain in question. Using this text, the following tasks will be executed by the environment: i) Identify important terms: when a text is received in order to be used by NLP, the relevant texts will be extracted automatically; ii) Build XML tagged file: with the important terms identified and the original text, an automatic routine will be executed to create an XML file with the tagged terms; iii) Disambiguation and supersense extraction: with the XML file generated, algorithms will be executed for the disambiguation of the tagged terms using WordNet as a lexical database; iv) Semantic type and OntoUML mapping: with the terms disambiguated the constructs will be identified automatically using mapping of the semantic types and the OntoUML constructs; v) Build model semi-automatically: with the main constructs identified, routines will be developed to support the semi-automatic building of the model. The process will not be totally automated because the algorithms available cannot identify the correct semantic type in every situation. The identification of the relationships should also be a task that is performed semi-automatically. Here the guide based on Design Patterns proposed in [11] could be used; and vi) Derive list requirements: from the conceptual model a list of the possible requirements deduced from the relationships established among the concepts will be generated automatically.

5 Experiment

To demonstrate and discuss the viability of the proposed environment, an experiment was conducted. The method used for its execution and the results obtained are presented below, followed by a discussion of the findings.

5.1 Method

The experiment was conducted based on the stages of the method proposed in [18]:

i) Identifying important terms: in this stage, a text was selected to describe a knowledge domain and from this the conceptual model could be built. The text was extracted from [25] in which the domain of a bus route is described. In this work, there is also an entity relationship (ER) model that corresponds with the text. In this way, the elements proposed in the ER model were used to compare some results. The text used in the experiment is shown in Table I;

ii) Determining the meaning of important terms: to select the important terms, the topia.termextract version 1.1.0, developed by Python (https://pypi.python.org/pypi/topia.termextract/) was used. This tool was chosen because it is easy to use, it gives satisfactory results and its use is free;

iii) Identifying the semantic types of each term: following the identification of the relevant terms, an XML file was prepared with the original text tagged with the important terms

Copyright © 2014 CSREA Press, ISBN: 1-60132-286-0; Printed in the United States of America

that had been identified. To process the file, the algorithm for disambiguation TargetWord [26] was used with the help of Wordnet::SenseRelate. After the test with the metrics (lch, hso, wup, res, lin, jcn and lesk), the lesk was the metric that was used to execute it as it had the most satisfying results. A previous test was made using the example illustrated in [18]; and

iv) Choosing the OntoUML for each semantic type: after the semantic type of each term had been identified, the Supersense indicated by the WordNet base was obtained, using the mapping proposed in [16].

In the original method [18] another stage is proposed: Building the OntoUML model". This stage was not executed as no available mechanisms were located to enable the automatic building from the constructs that were identified.

TABLE 1. SELECTED TEXT

Text
There are two ways for people to travel with Voyager. Either passengers can make a reservation on a trip, or passengers can show up at the boarding gate without a reservation and purchase a ticket for an unreserved seat. Passengers with a reservation are assigned a reservation date, whereas, passengers without reservations are assigned a boarding date. The name and addresses of all passengers are collected. Telephone numbers are collected where possible. All bus trips are organized into daily route segments. All daily route segments have both a start time and an end time. Each daily route segment. Voyager organizes is classified as a route segment with a segment number, start town, and finish town. Voyager offers a range of trips, and each trip is made up of one or more route segments. For every trip there is a trip number, start town, and finish town. If the trip is organized around a special event, the event name is also associated with the trip. Each daily route segment that Voyager offers is part of a dally trip. A daily trip is undertaken by one or more bus drivers. The name, address, and employee number of all drivers is collected. Voyager also records information about absent drivers. When a driver is absent. Voyager records the absence start date and the details about the absence. The absent driver provides one or more reasons for being absent and each reason is assigned a detail number and a short description. Voyager also collects information about the buses used for daily trips. Buses have a make, model, and registration number. For buses in use, the average daily kilometers is collected. If a bus requires maintenance, Voyager notes the date on which the bus entered maintenance and records the one or more problems with the bus. Voyager assigns a problem number and a short description for every maintenance problem. Finally, the average cost to repair all problems with a bus in maintenance is also recorded.

5.2 Results and discussion

i) Important terms: the algorithm was executed using topia.termextract and returned 33 important terms, as shown in Table II.

TABLE II. IMPORTANT TERMS

Terms
bus; bus drivers; bus trips; date; detail number; driver; employee number; end time; event name; maintenance; maintenance problem; name; number; passenger; problem; problem number; record; records information; registration number; reservation; reservation date; route segment; segment; segment number; telephone numbers; town; trip; trip number; unreserved seat; voyager records; Voyager; Voyager notes.

To check the accuracy of the results, the elements proposed in the ER model [25] and the terms obtained automatically were mapped. Table III shows the elements in the ER model and the relationship with the automatically identified important terms.

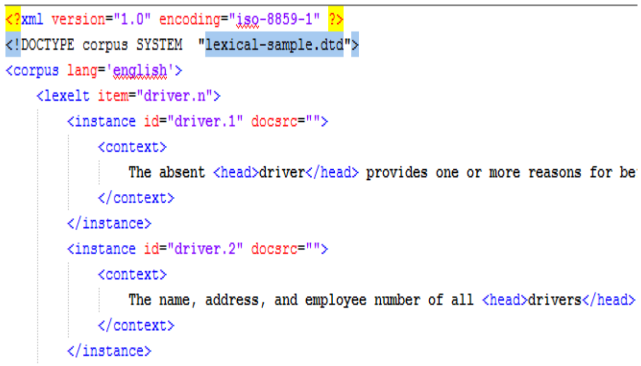
TABLE III. MAPPING BETWEEN ELEMENTS OF THE ER MODEL AND THE RELEVANT TERMS

Concept (ER model)	Element type	Relevant term mapped
Passengers	**Entity**	**Passenger**
Name	Attribute	Name
Address	Attribute	-
Telephone	Attribute	Telephone number; number
Reservation Date	Attribute	Reservation date; date
Boarding Date	Attribute	Date
Daily Route Segment	**Entity**	**Route segment; route**
Start Time	Attribute	-
Finish Time	Attribute	End time
Aver. No. Of Passengers	Attribute	-
Aver. No. Of Reservations	Attribute	-
Daily Trips	**Entity**	**Bus trips**
Finish Time (duplicate)	Attribute	End time
Start Time (duplicate)	Attribute	-
Route Segment	**Entity**	**Route segment; route;**
Segment#	Attribute	Segment number; number
Start Town	Attribute	Town
Finish Town	Attribute	Town
Bus	**Entity**	**Bus**
Aver. Daily Kilometers	Attribute	-
Reg.	Attribute	Registration number
Make	Attribute	-
Model	Attribute	-
Date Entered Maintenance	Attribute	Date
Trip	**Entity**	**Trip**
Trip#	Attribute	Trip number
Start Town (duplicate)	Attribute	Town
Finish Town (duplicate)	Attribute	Town
Special Event	Attribute	Event Name
Driver	**Entity**	**Driver; bus Driver**
Employee#	Attribute	Employee number; number
Name (duplicate)	Attribute	Name
Address (duplicate)	Attribute	-
Absence Start Date	Attribute	-
Maintenance Problems	**Entity**	**Maintenance Problems;**
Problem#	Attribute	Problem number; number
Description#	Attribute	-
Average Cost to Repair#	Attribute	-
Absence Details	**Entity**	**-**
Details#	Attribute	Detail number
Description# (duplicate)	Attribute	-

Only the elements of the entity and attribute type are shown. The mapping was done just to compare if the automatic extraction process was accurate. Of the 34 different concepts presented in the ER model (9 entities and 25 distinct attributes), only 11 distinct concepts (32.5%) had no mapping with an important term. While 23 distinct concepts had mapping (67.5%). It is important to emphasize that some terms found in the ER model were not found in the text (Aver.No.Of Reservations and Aver.No.Of Reservations).

On the other hand, of the 33 automatically identified important terms, only 7 (21.2%) terms (record, records information, unreserved seat, voyager records, Voyager and Voyager notes) had been mapped with the concepts presented in the ER model. Of these 7 terms, 3 were related to the name of the tourist company (Voyager), which is conceptually not relevant. While 26 (78.8%) terms had been mapped. These results proved to be satisfactory, enabling the experiment to proceed to the next stages.

ii) Important terms: sense, semantic type and the OntoUML construct. Of the 33 important terms, the terms that involved the name of the tourist company (voyager records, Voyager, Voyager notes) were excluded because they were not interesting terms for disambiguation. The simple terms with composite correspondents and which were not used in the text separately (route, segment, number) were also excluded. In the end, disambiguation was performed on 27 terms. Fig. 3 shows a sample of the XML file with the important terms tagged in the text.



```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE corpus SYSTEM  "lexical-sample.dtd">
<corpus lang='english'>
    <lexelt item="driver.n">
        <instance id="driver.1" docsrc="">
            <context>
                The absent <head>driver</head> provides one or more reasons for be
            </context>
        </instance>
        <instance id="driver.2" docsrc="">
            <context>
                The name, address, and employee number of all <head>drivers</head>
            </context>
        </instance>
```

Figure 3 XML file with tagged text

The meanings identified by the TargetWord algorithm are shown in Table IV.

TABLE IV. IMPORTANT TERMS AND MAPPING WITH ONTOUML

Important Term	WordNet Sense	Supersense	Semantic	OntoUML
Bus	#1: a vehicle carrying many passengers; used for public transport	artifact	artifact	kind
Bus drivers	#1: someone who drives a bus	person	rank	role
Bus trips	-	-	-	-
Date	#6: a particular day specified as the time something happens	time	time	datatype
Detail	-	-	-	-
Driver	#1: the operator of a motor vehicle	person	rank	role
Employee	-	-	-	-
End time	-	-	-	-
Event name	-	-	-	-
Maintenance	#1: activity involved in maintaining something in good working order	act	activity	kind
Maintenance Problem	-	-	-	-
Name	#1: a language unit by which a person or thing is known	communication	activity	kind
Passenger	#1: a traveler riding in a vehicle (a boat or bus or car or plane or train, etc.) who is not operating it	person	rank	role
Problem	#3: a source of difficulty	cognition		manual
Problem Number	-	-	-	-
Record	#7: a document that can serve as legal evidence of a transaction	possession	activity	kind
Records Information	-	-	-	-
Registration Number	#1: the number on the license plate that identifies the car that bears it	communication	activity	kind
Reservation	#2: a statement that limits or restricts some claim	communication	activity	kind
Reservation Date	-	-	-	-
Route Segment	-	-	-	-
Segment Number	-	-	-	-
Telephone Number	-	-	-	-
Town	#1: an urban area with a fixed boundary that is smaller than a city	location	place	datatype
Trip	#1: a journey for some purpose	act	activity	kind
Trip number	-	-	-	-
Unreserved	-	-	-	-

The disambiguation of terms related to entities, was more accurate than the disambiguation of terms related to attributes. Of 8 important terms related to entities, 5 (62.5%) (bus, bus drivers/driver, maintenance, passenger and trip) were disambiguated and the mapping was done between the important terms and the OntoUML constructs. On the other hand, of 19 important terms related to attributes, only 7 (36.8%) terms were disambiguated. It also was observed that sense identification for composite terms are less efficient.

From this viewpoint, it can be concluded that the method and tools are more efficient for identifying simple terms and terms related to entities. For complete identification of all the concepts, human intervention is required. It is also important to observe that the WordNet is a database of generic terms. In specific domains, disambiguation performance may not be possible due to the lack of record of all senses related to the terms to be disambiguated. Thus, there is a need to develop mechanisms in order to support the establishment of a semantic database for specific domains, which terms will not be found in the WordNet database. In addition to aiding the disambiguation of the terms in specific domains, additional information can be recorded in order to support the development of the ontology-based conceptual model.

6 Final considerations

Up to now, software requirements elicitation remains based on informal communications. For this reason, this stage is characterized by problems such as deficient communication, lack of consensus concerning the use of terms and lack of knowledge of the domain. To resolve part of these problems, studies are proposing the use of ontologies, a formalism with the purpose of facilitating communications by way of a common vocabulary and sharing knowledge of a knowledge domain.

However, in these proposals there is no environment to support requirement elicitation based on ontological conceptual models. The use of an ontological conceptual model would enable a closer view of the domain to be represented. However, the building of an ontological conceptual model is not an easy task, especially for novice modelers. OntoUML is a language for building ontology-based conceptual model, but it is considered more complex than other languages such as UML.

Therefore, it is necessary to create mechanisms to aid people in the building of this model. The use of NLP techniques is proposed to identify these constructs in the communications among those involved and thus support the semi-automatic construction of this model. It is assumed that from the conceptual model it is possible to automatically generate a list of requirements, since the ontology-based model reflects a domain more faithfully. The future works related to this proposal are: i) Improving the environment proposed integrating techniques to support the process; ii) testing with NLP algorithms for comparing results; iii) to propose a method in order to update a semantic database in a specific domain; iv) development of semi-automated support for building the conceptual model; v) and to propose a method for the derivation of requirements from the conceptual model;

REFERENCES

[1] R.H., Thayer, and M. Dorfman, “Software Requirements Engineering”, 2d Ed. IEEE Computer Society Press, 1997.

[2] W., Robinson, and S. Pawlowski, “Managing requirements inconsistency with development goal monitors”, IEEE Transactions on Software Engineering, 25, 1999.

[3] K. Pohl, “Requirements engineering: An overview”, In Encyclopedia of Computer Science and Technology. A. Kent, and J. Williams, Eds.Marcel Dekker, New York, NY, vol. 36, suppl. 21, 1997.

[4] J. Luis, D. Vara, and J. Sánchez, “Improving Requirements Analysis through Business Process Modelling : a Participative Approach”, 1, pp 165–176, 2008.

[5] S.W. Lee, and R. Gandhi, “Ontology-based active requirements engineering framework”, in Engineering Conference, APSEC, 2005.

[6] G.N. Aranda, A. Vizcaíno, A. Cechich, and M. Piattini, “A Methodology for Reducing Geographical Dispersion Problems during Global Requirements Elicitation”, in WER, pp 117–127, 2008.

[7] L. Li, “Ontological modeling for software application development”, Advances in Engineering Software, 36, pp 147–157, 2005.

[8] M. Harzallah, G. Berio, and A.L. Opdahl, “New perspectives in ontological analysis: Guidelines and rules for incorporating modelling languages into UEML”, Information Systems, 37, pp 484–507, 2012.

[9] H. Zhang, R. Kishore, R. Sharman, and R. Ramesh, “Agile Integration Modeling Language (AIML): A conceptual modeling grammar for agile integrative business information systems”, Decision Support Systems, 44, pp 266-284, 2007.

[10] G. Guizzardi, “Ontological Foundations for Structural Conceptual Models”, Telematica Instituut Fundamental Research Series 15, Universal Press, 2005.

[11] G. Guizzardi, A. Graças, and R.S.S. Guizzardi, “Design Patterns and Inductive Modeling Rules to Support the Construction of Ontologically Well-Founded Conceptual Models in OntoUML”, 3rd Workshop on Ontology Driven Inf. Systems (ODISE 2011), London.

[12] T.R. Gruber, “Toward Principles for the Design of Ontologies Used for Knowledge Sharing”, International Journal of Human and Computer Studies, 43(5-6), pp 907-928, 1995.

[13] R. Girardi, and A. Leite, “ A knowledge-based tool for multi-agent domain engineering”, Knowledge-Based Systems, 21, pp 604–611, 2008.

[14] H. Kilov, and I. Sack, “Mechanisms for communication between business and IT experts” Computer Standards & Interfaces, 31, pp 98–109, 2009.

[15] G. Guizzardi, “On ontology, ontologies, conceptualizations, modeling languages, and (meta) models”, Frontiers in artificial intelligence and applications, pp 18–28, 2007.

[16] L. Castro, “Abordagem Linguística para Modelagem Conceitual de Dados com Foco Semântico”, Msc Dissertation, Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro, Brazil, 2010.

[17] R.M. Dixon, “A Semantic Approach to English Grammar”, 2nd ed. Oxford University Press, USA, 2005.

[18] F. Leão, K. Revoredo, and F. Baião, “Learning Well-Founded Ontologies through Word Sense Disambiguation”, in proceeding of: 2nd Brazilian Conference on Intelligent Systems (BRACIS-13), 2013.

[19] T. Pedersen, and V. Kolhatkar, “WordNet:: SenseRelate:: AllWords: a broad coverage word sense tagger that maximizes semantic relatedness”, Human Language Technologies, 2009.

[20] S. Regoczei, and E.P.O Plantinga, “Creating the domain of discourse: ontology and inventory”, International Journal of Man-Machine Studies, 27, pp 235–250, 1987.

[21] K.M. Oliveira, F. Zlot, A.R. Rocha, G.H. Travassos, C. Galotta, and C.S. Menezes, “Domain-oriented software development environment”, Journal of Systems and Software, 72, pp 145–161, 2004.

[22] W. Vongdoiwang, and D.N Batanov, “An ontology-based procedure for generating object model from text description”, Knowledge and Information Systems, 10(1), pp 93–108, 2006.

[23] M. Cossentino, N. Gaud, V. Hilaire, S. Galland, and A. Koukam, “ASPECS: an agent-oriented software process for engineering complex systems’, Autonomous Agents and Multi-Agent Systems, 20(2), pp 260–304, 2009.

[24] P.F. Pires, F.C. Delicato, R. Cóbe, T. Batista, J.G. Davis, and J.H Song, “Integrating ontologies, model driven, and CNL in a multi-viewed approach for requirements engineering”, Requirements Engineering, 16(2), 2011.

[25] A. Gemino, and Y. Wand, “Complexity and clarity in conceptual modeling: Comparison of mandatory and optional properties”, Data & Knowledge Engineering, 55(3), pp 301–326, 2005.

[26] TargetWord, accessed: http://search.cpan.org/~sid/WordNet-SenseRelate-TargetWord-0.09/lib/WordNet/SenseRelate/Target Word.pm, date: 2014 January 3

Empirical investigation of Systems Cost Estimation Models in the Limpopo province of South Africa: A requirements modelling problem

Benson Moyo[1], Magda Huisman[2]
[1]Computer Science and Information Systems Department, University of Venda,
Thohoyandou, Limpopo, South Africa;
[2]School of Computer, Statistical and Mathematical Sciences, North-West University,
Potchefstroom Campus, South Africa

Abstract - *There are many factors believed to be* important to *systems* development *cost estimation. However an in-depth analysis* demonstrates requirements as central cost drivers. *The various transformations requirements go through from candidate requirements to released* response *is the most intricate part of* systems development *cost estimation. Requirements exist independent of systems development methodologies. Requirements may be viewed from bespoke or market driven perspectives. The former assumes a traditional economic agent theory* view where *a client* organisation *requests* for a service from the systems *develop*ment organisation*. The later, market-driven requirements elicitation entails predicting requirements by the* systems *develop*ment organisation *based on market research output.* Irrespective of the perspective the systems development cost estimation is imperative. *The study investigates adoption and usage of cost estimation models* by *the systems development companies in the Limpopo province of South Africa. The paper introduces a requirements transition state diagram and pinpoints informal cost estimation models as predominant. In this article we also present the results of our survey findings and the discussion of those results as well as the recommendations for further work*

Keywords: *System development, requirements, cost estimation*

1 Introduction

Requirements exist independent of systems development methodologies. Requirements form the basis for the contract among the developer, the client and the user in a traditional economic agent theory based development. In market-driven systems development, requirements are predicted by the development organisation from market research output. This gives two ramifications of requirements; one based on a particular organisation where client and users are accessible and the other where clients and the users are the universe market or a market segment hard or expensive to access if not impossible. The nature of requirements influences the selection of systems development methodologies. The purpose of systems development methodologies is to guide the development team successfully translate prioritised set of requirements into systems solution. They facilitate the development of technological frames to align expectations of technology and minimise incongruence, in systems development and reduce uncertainty in requirements determination [18]. Systems development methodologies are prospected to define work breakdown structure [19], provide support for improved product quality, productivity, human resource control and cost control [18]. Within the systems development methodology adopted by an organisation, forecasting and controlling systems development costs is crucial. Cost estimation is a systems development methodology activity that allows management to justify the relevance of a systems project in terms of reputation, social responsibility and more importantly financial value.

Research in systems development is not short of findings on software crisis that emerged from the 1960s. The crisis is claimed to stem from deficiencies in requirement determination emanating from inconsistencies, omissions, errors and ambiguities associated with requirements management. Systems development is basically governed by the requirements of the target system. The aforesaid requirement deficiencies are viewed as the key cause of systems development project failure. Failure means that the systems neither performs to specifications, nor meets budgetary constraints, nor is delivered within specified schedule, nor satisfies user needs and expectations. Cost estimation is primarily based on requirements. Inaccurate requirements identification, analysis, prioritisation, abstraction, triage, tracking and specification constitute the main causes for failure. In order to derive cost and schedule estimates requirements are the main input into the estimation process.

Cost estimate entails effort and schedule cost required to complete the system that meet client expectations, and satisfy developer starting from consultation to deployment. The client might be the universal market or market segment or a particular organisation (economic agent theory [30]). Requirements elicitation, analysis and management are complex processes that inject complexity to software cost estimation.

Costing is governed by requirement based on both internal and external factors. For example from internal factors client and user requirements, systems quality, and contractual obligations whereas on the external factors we have the legal, statutory and regulatory requirements. The dilemma is on deciding which factors to include and which transformation to apply in order to map the factors into a single financial value. The growth in size, importance and complexity of software has exacerbated cost estimation [1]. No one cost model can address all the cost estimation situations of systems development. Literature reviewed reveals that there is still a gap between the use of systems cost estimation models and the actual systems costing practices in organisations. Logically systems cost estimation models should permit systems project managers reliably approximate systems cost. However, systems projects are complex in nature and it is challenging to accurately estimate their costs. From a causal wisdom perspective, a miniature perturbation in the development process for instance, may cause a huge change in systems costs. A small perturbation on the systems development methodology may also result in the deterioration of communication which in turn might lead to conceptual incongruence within the development team. Systems development conceptual incongruence degenerate into assignment scope challenges, quality issues and schedule slippages which in most cases, trigger considerable costs. Project managers have their fears on two extremes on systems project cost estimation: over costing or under costing. Over costing may damage the company's reputation and lead to failure to win systems development contracts (or fail to penetrate the market in case of a market driven systems development). On the other hand under estimating the cost may lead to loss of money and decrease in organisational profitability.

Cost estimation is a multidimensional construct that require identification and definition of numerous attributes. These attributes are attached to a weight which is then converted into cost drivers which in turn are transformed into financial values. In systems cost estimation there is no uniform set of attributes or parameters for every project. Each project is unique and may require contextual consideration to define the parameters. Developing a mobile phone game application does not exert the same demands as an online banking application, nor a nuclear reactor safety monitoring systems. Failure of each of the three mentioned systems has different consequences. For instance mobile phone game system failure may lead to disappointment; the banking system malfunction may culminate in financial loss, whereas the nuclear safety monitoring system failure may be catastrophic. Requirements important in one system application may not be so important in another system application. Even if the same requirements are found in different systems, their weight intensities may differ. The different importance levels, rejection and prioritisation from one systems project to another makes it difficult to develop generic cost estimation models. Each cost estimation model is based on researcher's assumptions on a particular problem domain. Research has proposed a number of parameters but the underlying factors are requirements.

There is no simple way to make an accurate estimate of the effort required to develop systems [1]. Estimates are generated from requirements definitions or market research. The estimates define the effort, duration and staffing and other resources Alterations can be done to attain a trade-off between effort and duration. At the same time the systems product should respond to the requirements otherwise it might neither penetrate the market nor pass acceptance test even if the release date is met. Focusing on release date may have an implicit maintenance burden as some of the features may not be developed properly.

There are many requirements that need to be met by systems cost estimation models, in order to be adopted by an organisation as satisfactory in capturing the systems project costs measures. Systems cost estimation model maps various systems metrics and measurement into a single financial value. As the systems development process evolves so cost estimation should improve. There are many systems cost estimation models in existence, and companies in Limpopo are presented with the challenge to find the appropriate costing models, practices, techniques and tools. Systems cost estimation include specialised planning such as: quality plan which measures quality procedures and levels that will be used in a systems development, verification and validation plan which caters for the relevance degree of solution approximation by the system developed, configuration management plan which focuses on the alignment of management procedures and structures to be used, maintenance plan which predicts the maintenance burden, defect density which forecast the cost of defect discovery and removal, reviews and inspections, and change management plan that shows how the skills, experience, fears, and perceptions of project team members and users will be considered.

Systems cost estimation models strive to approximate the solution of systems cost estimation problem. Systems cost estimation is prediction of the cost of the resources that will be required to complete all of the work of a systems project [1]. It is important for project managers to use the appropriate model that will enable a successful completion of a systems project. In this work we investigate the actual adoption of the systems cost models in practice.

Khativi and Jawawi[8] state that the main reason for project failure is imprecision in cost estimation. However this is a high level of abstraction as one might ask for the causes of this imprecision. Systems project managers strive to acquire as much information on existing cost estimation models as possible. There is no exhaustive repository of projects that can enable success and failure comparisons of cost estimation models, despite claims by cost estimation model authors that such repositories exist. The failure and success histories of cost models have limited scope as each project responds to specific requirements. Specific situational characteristics make it so difficult to design a ‘one size fits all’ costing model [13], though model developers believe the use of templates and frameworks can solve the problem.

Cost estimation is critical for organisations both that specialise on systems and those that outsource systems development. The purpose of this research is to investigate existing cost estimation models and their adoption in the systems development industry in Limpopo province of South Africa.

A survey on companies is carried out and the results show low usage of different systems project costing models in existence. Mostly formal systems project costing models are used when dealing with government systems projects. This may be that in order to win a tender from the government there is need to have rigorous cost estimation algorithms. The paper is organised into five sections. The first provides an overview of cost estimation characteristics. The second outlines the rationale behind cost estimation. The third describes briefly our research approach, the forth section gives a discussion of the findings and finally, the fifth section provides conclusions and recommendations for further work.

2 Systems Cost estimation

Systems cost estimation is embedded into systems development. In this section we present justification, need and examples of systems costing models.

2.1 The rationale behind cost estimation

In order to make strategic decisions managers need some information about the resources required for the project. However this information is usually not available at the time it is needed. Estimates provide an approximation on the effort, schedule and other constraints needed. With estimates decisions on whether to proceed with a project can be made. Estimates serve at the intelligence and choice phases of the decision making process. One of the purposes of performing a cost estimate is to have a means by which the development costs can be monitored and controlled. Monitoring and control may be performed either at micro level or macro level. At micro level progress on addressing requirements is checked and at macro level progress is assessed by checking feature developed.

Costs estimates make the basis for the management and the development company to approve a project proposal or reject it. It is a crucial factor in determining when and how the project should be carried out. The project planning, controlling, resource allocation and roles in the project and overall activities of the project are linked to the cost estimate. A system is an investment and therefore it should demonstrate financial, technical and social feasibility. Systems cost estimates are critical to developers, clients and users [3]. In a bespoke development environment they can be used for generating request for proposals so that the client can have the clue of the approximate amount required for the whole system that is contract negotiations between the client and the developer, scheduling between the programmers and the project managers, monitoring and control. On a market-driven systems development the developer makes assumptions on client and user requirements based on market research and work out the justification to commit resources.

The reasons for performing a cost estimate dictate what to estimate, how to estimate, when to carry out the estimation and the degree of accuracy. In principle cost estimation is iterative in nature in order to continuously update management on the project status. Suri and Ranjan[7] assert that small projects can be easily estimated and accuracy is not very important. But as the size of project increases, requirements become hard to elicit and analyse. The increases and dynamism of requirements lead to complex dependences and complicated relationships among them.

2.2 Requirements in cost estimation

Requirements present challenges as they come from different stakeholders that may even have conflicting objectives. The huge volume of requirements inflow and the need to select and reject some is a challenge. For example a selected requirement can be dependent on a rejected requirement. On another hand considering all requirements is not feasible as some may be contradictory and conflicting. Inconsistency, errors, ambiguity, incompleteness and different levels of abstraction are some of the challenges. Despite all these challenges cost estimates are based on requirements as they form part of the contract and the main link between the client and the developer.

Systems cost estimation involves measurement. Like any other measurement it depends on the perspective about the phenomenon to measure in this case requirements. The perspective is dependent on the operational definition of the concept. However, requirements as aforementioned are presented at different levels of abstraction making it difficult to calibrate them. The operational definition compounds the difficulties of uniform measurement of requirements as they may fall on nominal scale or ordinal scale or interval scale, or ratio scale. They may also be presented in varying degrees of abstraction of course this is due to different stakeholder linguistic capabilities. Requirements are a unifying concept. They go through a series of stages and sometimes are rejected due to change of environment. This has implications on cost estimation. Estimators model requirements into higher level features and functions and during this process of translation some requirements may be rejected without visualising dependencies. Unimportant requirements may be included. In some cases a complete misinterpretation during translation from requirements to model systems artefacts may result. Figure 1 shows a proposal of a highly simplified view of requirement transition state diagram. During the initial phases of a systems project, large number of candidate requirements will be collected. These would be subjected to a selection and rejection processes. In order to cost selected requirements there should be cost elements and unit measures. The cost unit measures constitute a complex set of factors that affect cost estimation. The difficulty and the level of accuracy of systems cost estimation are shaped by the different categories of requirements which are considered as cost factors. Requirements can be grouped under two broad classes; the functional and non-functional. Requirements can determine the following factors: systems quality, duration, team size, number of consultants, number and characteristics of stakeholders, legal statutes, regulatory statutes, mandatory statutes, organisational policy, criticality and complexity of the system, project risk factor, team expertise and experience, development platform, systems development methodology, techniques, and tools adopted, contingency plan and defect density.

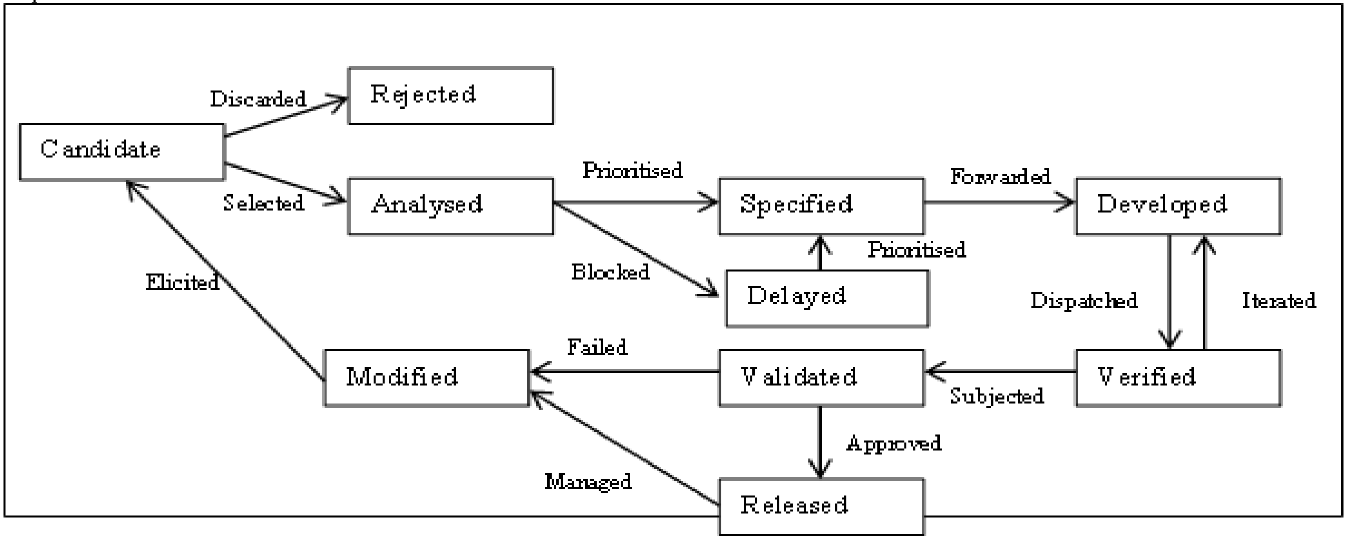


Figure 1: Requirements state transition diagram

The estimator works out the duration of the project in person-week or person-month based on the requirements analysis. Systems cost is directly proportional to project duration. The longer the duration between initial selected requirements and release, the more likely that there will be significant changes to the initial requirements resulting in more inaccurate cost estimates. This may occur due to continuous inflow of new requirements, modification of already known requirements, remodelling of requirements, change in user expectations, changes in the environment in which the system is to be installed, or change of technology.

Requirements also drive the size of the team and the level of expertise needed for the systems project. Cost increases with the size and composition of the team in other words the cost is directly proportional to the team size, expertise and experience. The more the team increases the more complex team coordination and communication will be among the members. Communication becomes less effective with the increase of the team and the project manager has to possess good management skills to keep the team productive. It is mostly the project manager's task to know the effort each team member and the capability of each member when scheduling the work that is meant to address the systems requirements. Therefore the increase in the team does not mean that the work is going to be completed earlier or best done.

Each systems project has risks associated with it. These risks may be associated with the integration of tools that offer source code generation, debugging, tests, document generation, diagraming, version controlling and other programming related services into the development process. Risks can also be due to lack of experience in the application domain, omitted requirements, misinterpreted requirements, mismanaged requirements, staff turnover and change of environment. The level of requirement uncertainty is used to establish the level of tolerance and estimate and allocate an estimate contingency fund.

Requirements leads to the selection of systems development methodology and in reverse the methodology addresses the requirements. The methods, techniques, tools, programming languages, programming paradigm, team skills, expertise, and experience are all unified by the methodology and a cost factors. Estimation is not a task done only once, at the project inception; it is a process where estimates and re-estimates are undertaken throughout the lifecycle of a project. The relevance of an estimator is not necessarily the accuracy of the initial estimates, but rather the degree to which the estimates converge towards the actual costs.

2.3 Existing systems cost estimation models

Organisations use different models to calculate cost estimates. These cost estimation models can be classified under two main categories: the formal mathematical models and informal experience based models. The formal models endeavour to quantify the cost factors and apply a set of relations that describe the mapping between the cost factors and the cost values. The mapping functions are formulated through analysis of historical data, assumptions and may be adjusted to each individual development context. On the other hand informal models are used by highly skilled experienced developers, expert or /and managers who have gained sufficient knowledge from previous systems development projects. The informal models rely on the history of past projects. A repository of detailed metrics and descriptions of characteristics recorded for each project. This repository may be a computerised database or manual record or simple organisational memory. The estimator can query the database searching for projects with similar characteristics and then benchmark the estimate on actual costs and process of the previous projects. Informal models are hard to understand as the experienced estimator may rely on tacit knowledge to obtain the estimate.

2.3.1 Formal models

A formal model may transform the systems requirements into a measure of the “size” of the systems in the form of Source Lines of Code (SLOC) as the basis for creating the cost estimates. Source Line of Code is an estimation parameter that illustrates the number of all commands, control structures, variable declarations, assignments, compiler directives, variable method definition and declarations excluding comments, blanks, and continuation lines. The advantage of SLOC is that estimating line of code seems intuitive. The lines of code are a parameter commonly used in formal model cost estimation models. It is also straightforward to count the lines of code in finished product which make it easy to compare the cost estimate and the actual cost. The disadvantage surfaces from deciding what to include as a line of code. The next challenge is when different programming languages are used it becomes difficult to use the SLOC model for cost estimation. Each language has its own number of lines of code to accomplish the same task. Line of code is not appropriate in a multiple programing language environment characterised by the current trends in systems development.

Instead of using the line of code, function points metrics can be used. It is a measurement based on functionality of systems[9]. It measures the amount of functionality in a system by counting and weighting inputs, outputs, queries, and logical flow, interfaces, files handling and device manipulation. Grouping the functions gives another measure referred to as the feature points. The feature points also consider algorithms as parameter and encapsulate control structures.

A function point based cost estimate known as systems functional size measurement is recognised by the International Organization for Standardization (ISO) and (International Electrotechnical Commission (IEC) as standard for measuring systems size. For example International Function Point Users Group (IFPUG) Function Point (FP) method [14] and the Common Software Measurement International Consortium (COSMIC) function point method [16]).

One most commonly known, well publicised and taught formal cost estimation models is Boehm’s Constructive Cost Model (COCOMO) [15]. COCOMO is widely practiced and popular among the systems development community because of its adaptability to different development environments. It predicts the length and effort of a project by drawing an association between the size of the systems and various cost drivers. The factors are assigned weights based on modelled requirements cost factors, project’s domain, environment, and constraints. The drawback on this model is that it requires too many parameters and simply selecting different values for the multipliers can vary the minimum and maximum estimates by a very high margin. Without historical data, it is difficult for an organization to determine the approximate values for these multipliers. The whole model fails when an organization is developing a system outside its immediate domain of expertise.

2.3.2 Informal models

An organisation may have a multi-stage estimation process. The contractor may present initial estimates. The estimate will be presented just to win the contract. The strategy is ‘pricing to win’. The estimate is made as low as possible so as to win the contract. Often times the estimate is done based on ambiguous and sometimes contradictory requirement. Once a company has been awarded the contract, it may then perform another more detailed estimate. The post contract winning estimate is done looking at the real systems problem to be solved. In most cases the posts-contract estimate is higher than the pre-contract estimate. The contractor may present this updated estimate to the client, and if management reject it there are a number of ways in which the contractor can go around the problem. They can develop a system with less functionality and suggest enhancements. If all fails then the contractor may accept the price to win estimate. Costs do not accurately reflect the work required and the rejection on acceptance test is common.

Estimation based on expert judgment is done by considering advice given by experts who have more experience in similar projects[8]. The work breakdown structure (WBS) organises activity patterns that vary from project to project, by defining assignment scope. It identifies activities needed to complete project development and the effort, staffing, duration of each task and the skills required. The size of activities defined within the WBS is dependent on the level of detail of the estimate and size of the project. A cost estimate frequently includes a complete work breakdown structure (WBS), which project team members use as a basis for understanding their roles and avoid bumping into each other's way. The sum of the direct individual activity costs and other indirect cost such as travel costs gives the overall project cost. The Delphi technique constitutes a team of experts tasked to generate systems cost estimates of a project given all available factors and constraints. Each expert provides an estimate without consulting other team members. The second iteration allows each expert to have access to the estimation information provided by other experts during the first iteration. The process continues until the expert estimates converge to a point.

The following table 1 shows a historical perspective of the formal models and indicates the modelling level of requirements abstraction that gives the cost unit measure. Avoid using too many capital letters. All section headings including the subsection headings should be flushed left.

Table 1: Trends in cost estimation

Year	Authors	Model	Cost drivers
1970	Boehm[1]	Rule of thumb	no specific factors
1975	Alberecht and Gaffhey[20]	Function Point Analysis	external input, external output, external inquiries, external, interfaces, internal files
1977	Park[1988]	PRICE-S	source lines of code, function points, predictive object points, defect prediction
1979	Putnam[21]	Putnam Model	manpower distribution, environment indicator, duration
1979	Albrecht[24]	Function Point	interfaces, forms, reports, database tables
1980	Jensen[25]	SEER-SEM	source line of code, effort, schedule, defect predict, risk, reliability
1981	Boehm[17]	COCOMO	source lines of code, effort
1983	Rubin[27]	ESTIMACS	function point, effort, staff count and deployment, risk, portfolio impact, customer complexity
1983	Symons[22]	Mark II function points	inputs, outputs, queries, and logical flow, interfaces, files handling, data processing
1986	ISO/IEC 20926[14]	IFPUG	function point
1992	Bergeron and St-Armaud[23]	Mark II Function Point	inputs, outputs, queries, and logical flow, interfaces, files handling, data processing
1995	Boehm[15], Boehm et al[26]	COCOMO II	object points, function points, lines of code, effort
1997	Jones[32]	Checkpoint	activity, task. estimates, deliverables ,defects, schedules
1998	Chatzoglou and Macaulay [28]	MARCS	time, effort, staff size

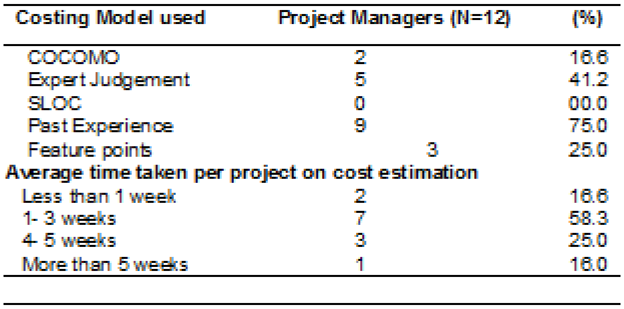
3 Methodology

The approach is essentially qualitative research based and uses instruments of interviews and questionnaire. Intensive and extensive literature survey is conducted to establish the status of systems cost estimation in industry. A questionnaire is administered to the project managers and freelancers in the Limpopo province of South Africa. It is important to acknowledge that the sample was purposeful and non-probabilistic. The guiding selection principle was to increase the probability of the presence of the phenomenon of study interest in the sample [13]. Participants were recruited from the Limpopo province systems development companies.

4 Results and discussion

Table 2 shows the usage trends of different costing models found in systems development houses in Limpopo. Experience based costing is more pronounced despite the dissemination of models, tools and techniques from the research community. For freelancers their cost estimates are based on the feature points. Cost estimation is allocated an average time of three weeks per project.

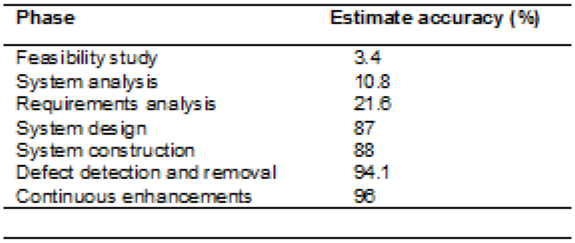
Table 2: Cost model usage



Costing Model used	Project Managers (N=12)	(%)
COCOMO	2	16.6
Expert Judgement	5	41.2
SLOC	0	00.0
Past Experience	9	75.0
Feature points	3	25.0
Average time taken per project on cost estimation		
Less than 1 week	2	16.6
1- 3 weeks	7	58.3
4- 5 weeks	3	25.0
More than 5 weeks	1	16.0

Table 3 shows the probability of approximating costs in different development phases of the development process. Estimates in the initial stages of the project are not reliable as the level of requirement understanding is low. Sometimes what may be established as feasible might not be as feasible as earlier thought. Development process iterates over requirements several times leading to the establishment of factors governing requirement dynamics. Requirements are regarded as the most difficult part of a project.

Table 3 *SDLC phase cost estimates accuracy*



Phase	Estimate accuracy (%)
Feasibility study	3.4
System analysis	10.8
Requirements analysis	21.6
System design	87
System construction	88
Defect detection and removal	94.1
Continuous enhancements	96

5 Conclusions

We found that systems cost estimation is based on and governed by the requirements of the target system. Requirements determination is an integral part of systems cost estimation and the basis for systems costing in Limpopo province. However, before cost estimation requirements are modelled into cost drivers in the form of function points, feature points or object points. The main challenges emerge during the requirement transformation into cost factors. In order to minimise the impact of error in cost estimation the estimators carry out the estimation process throughout the life cycle of a project. However, with requirements poorly identified, specified, and modelled failure probability is high irrespective of the number of times cost estimation is done. Timing of estimates, estimation constraints, systems development methodology used, experience and expertise are the basis for cost estimation in Limpopo, but all these are dependent on requirements.

Despite availability and publicising of formal cost estimation models, informal models are most preferred by the systems development industry in the province of Limpopo in South Africa. As further work we would replicate the research in the remaining eight provinces of South Africa and endeavour to find out the reasons for not adopting the formal cost models.

6 References

[1] I. Sommerville, in Software Engineering; Software Cost Estimation, pp. 1–58, 2004.

[2] T. N. Sharma, A. Bhardwaj, and A. Sharma, “A Comparative study of COCOMO II and Putnam models of Software Cost Estimation,” vol. 2, no. 11, pp. 1–3, 2011.

[3] J. Kaur, S. Singh, and K. S. Kahlon, “Comparative Analysis of the Software Effort Estimation Models,” pp. 485–487, 2008.

[4] M. M. Albakri and R. J. Qureshi, “Empirical Estimation of COCOMO I and COCOMO II Using a Case Study.” pp. 265–270, 2012.

[5] I. Myrtveit, E. Stensrud, and M. Shepperd, “Reliability and validity in comparative studies of software prediction models,” IEEE *Trans. Software Engineering*, vol. 31, no. 5, pp. 380–391, 2005.

[6] B. Boehm and C. Abts, “Software Development Cost Estimation Approaches”, 1998.

[7] P. K. Suri and P. Ranjan, “Comparative Analysis of Software Effort Estimation Techniques,” vol. 48, no. 21, pp. 12–19, 2012.

[8] V. Khativi and D. N. A Jawawi, “Software Cost Estimation Methods : A Review,” vol. 2, no. 1, pp. 21–29, 2010.

[9] S. L. Pfleeger, F. Wu, and R. Lewis, *Software Cost Estimation and Sizing Methods: Issues, and Guidelines (Google eBook)*. Rand Corporation, pp. 97, 2005.

[10] H.M Haddad, N.R Ross, and W. Kaensaksiri,’’Software Reuse Cost Factors’’ Int`l Conf. Software Engineering and Practice,pp.284-290,2012

[11] B. Boehm, C. Abts, and S. Chulani, Software development cost estimation approaches - A survey. Annals of Software Engineering, Vol 10, No1-4. Springer, Netherlands, 2000.

[12] C.F. Kemerer, An empirical validation of software cost estimation models. Communication of the ACM, Vol 30, No 5, 416-429, 1987.

[13] K. Eisenhardt, Building theory from case study research, Academy of management review, Vol 14, No 4, pp 532-550, 1989.

[14] ISO/IEC 20926 Software and systems engineering -Software measurement - IFPUG functional size measurement method 2009, 2nd , ISO, Geneva, 2009

[15] B. Boehm, Software Cost Estimation with COCOMO II, Prentice Hall PTR, Upper Saddle River, NJ, 2000.

[16] ISO/IEC 19761 Software engineering – COSMIC: a functional size measurement method, 2nd edition, ISO, Geneva, 2011.

[17] B. Boehm, Software Engineering Economics, Englewood Cliffs N.J., Prentice-Hall Inc. 1981.

[18] M. Huisman and J. Iivari, Deployment of systems development methodologies: Perceptual congruence between IS managers and systems developers, Information & Management , Vol. 43, No. 1, pp. 29-49, 2006.

[19] N.Jayaratna, Understanding and evaluating methodologies: A systemic framework. McGraw Hill, UK, 1994.

[20] A.J Alberecht and J. E Gaffhey, Software function, source lines of code and development effort prediction: A software science validation, IEEE transactions on Software Engineering.

[21] L. H Putnam, A general Empirical Solution to the macro software sizing and estimating problem, IEEE transaction on Software Engineering, Vol 4, No 4, pp345-361, 1981.

[22] C. Symons, Software sizing and estimation Mark II function points, Wiley, 1991.

[23] F. Bergeron and J. Y. St-Armaud, Estimation of information systems development effort: a pilot study, Information and Management, Vol 22, No 4, pp 239-254, 1992.

[24] A. J. Albrecht, “Measuring Application Development Productivity”, Joint SHARE,
GUIDE, and IBM Application Development Symposium, pp. 83-92, Monterey, 1979.

[25] R. Jensen, “An Improved Macrolevel Software Development Resource Estimation Model”, Proceedings of 5th ISPA Conference, pp. 88-92, April 1983.

[26] B. Boehm, B. Clark. E. Horowitz, C. Westland, R. Madachy, and R. Selby, “Cost Models for Future Software Life Cycle Processes: COCOMO 2.0”, Annals of Software Engineering, Software Process and Product Measurement, Science Publishers, Amsterdam, The Netherlands, Vol. 1, pp.57-94, 1995.

[27] H. Rubin, “ESTIMACS”, IEEE, 1983.

[28] P. D. Chatzoglou and L. A Macaulay, A rule based approach to the developing software prediction, Automated Software Engineering, Vol 5 No 2, pp211-243.

[29] M. Shin and A. L. Goel, “Emprirical data modeling in software engineering using radial basis functions”, IEEE Transactions on Software Engineering, pp. 567-576, 2000.

[30] C. Toffolon, S. Dakhli, “A Framework for Software Engineering Inconsistencies Analysis and Reduction”, In Proceedings 22nd Annual International Computer Software & Applications Conference (COMPSAC 98), IEEE Computer Society Press, pp. 270- 277.

[31] R. Park., “The Central Equations of the PRICE Software Cost Model,” Park R., 4th
COCOMO Users’Group Meeting, 1988.

[32] C. Jones, Applied Software Measurement, McGraw Hill, 1997.

.

SESSION

SOFTWARE ENGINEERING AND SAFETY, SOFTWARE QUALITY, ERROR CHECKING, TESTING METHODS, DEBUGGING METHODS

Chair(s)

TBA

UAH OnTrack: Precision Navigation System for Research on The Software Safety Issues of Positive Train Control

Scott Schiavone, Sjohn Chambers, Sunny Patel, Lee Ann Hanback,
David J. Coe, Jason Winningham, B. Earl Wells, George Petznick*, and Jeffrey H. Kulick
Department of Electrical and Computer Engineering, *Consulting Railway Expert
The University of Alabama in Huntsville, Huntsville, Alabama, USA

Abstract - *The Department of Electrical and Computer Engineering at The University of Alabama in Huntsville (UAH) is developing UAH OnTrack, a system for creating location and velocity aware model trains for teaching about software system safety. UAH OnTrack mimics capabilities of the congressional mandated Positive Train Control system, which will allow centralized analysis and control of US trains in case of imminent danger. The system provides location and velocity information to model trains in GPS challenged environments such as real trains might find in tunnels, urban environments and underground rail yards and stations. It also provides an opportunity to develop advanced scheduling algorithms mimicking the properties of the US airspace Required Navigation Performance capabilities by allowing trains to follow more closely than the traditional block scheduling system. It also provides a platform for development and verification of robust algorithms for monitoring of system safety and security.*

Keywords: Positive train control, software safety engineering, DO-178, inertial measurement unit, smart train

1 Introduction

Over the past few years, the Electrical and Computer Engineering Department at The University of Alabama in Huntsville has been developing a Software Safety and Security Laboratory for teaching software safety engineering. Students utilizing the laboratory have developed train scheduling software using an aviation safety standard, DO-178B to create high reliability software for scheduling trains [1]. Like the current system in use in the US, the system used block scheduling which only requires knowledge of a train's location to the nearest block. This is typical of the system in use in the US on real trains. However, in the Rail Safety Improvement Act of 2008, Congress mandated that a supervisory safety system be created with a centralized authority to detect and manage possibly dangerous conditions such as a halted train, an out of position train, or a train that ran a signal [2]. These systems are primarily based on GPS and as a result can only be used in areas with GPS coverage. GPS-based systems fail in challenged areas such as tunnels and underground yards as found in inner cities. Moreover, a control system that relies only upon GPS data may be spoofed by an attacker. To ameliorate these risks some systems plan on using inertial measurement units (IMU) to confirm data supplied by GPS and to provide GPS equivalent data in GPS challenged regions. Computer engineering students and faculty at UAH have been working on UAH OnTrack, a low cost system to provide model trains precise navigation capabilities similar to those found on full size trains and other systems.

The goals of the UAH OnTrack system are to provide all basic operating capabilities required to implement a positive train control system including occupancy detection, switch operation, train selection and movement, precise train localization, and accurate train velocity data. Using these basic capabilities provided by the OnTrack system, students enrolled in the software safety engineering course may then focus on the development and verification of smart train scheduling software and the safety supervisory system components required for positive train control. Before describing the UAH OnTrack system in detail, we discuss current US railway signaling, the Positive Train Control test bed developed for use with the OnTrack system, and relevant details of modeling train sensing, signals, and control.

2 Background

Current signaling in the US and on model railroads is a block-oriented system of signals. Rail systems are divided into blocks of track where block lengths may vary. To maintain safe separation between trains, the general rule is that only one train is allowed to occupy any given block. Signals are used to govern movement of trains from block to block. Signaling may be manually mediated, by wireless communications or by warrants – explicit orders that a train has permission to proceed. Automated Block Signaling is used to locally determine if a train is allowed to proceed into the next block. A train occupying a block results in a short circuit between the rails triggering display of the occupied signal.

It is important to note that prior to positive train control and precise navigation, train locations were known only down to the block level. Whether a block spanned a few miles to a several tens of miles, the exact location of the train within that block was unknown. With the precise navigation availed by positive train control signals using GPS or the precision location and velocity information provided by UAH OnTrack, signaling can enter an entire new arena which will allow multiple trains to safely occupy a single block simultaneously including multiple trains closely following as they travel in the same direction.

2.1 Positive Train Control Test Bed

Our previously reported software safety research and education efforts utilized our original 4' x 8' HO gauge model railroad software safety test bed [1][3]. As a result of experiments performed on this compact track layout, we determined that our new Positive Train Control (PTC) test bed must not only contain longer detection blocks, but it must also utilize faster and more accurate occupancy detection mechanisms to allow trains to operate at higher speeds and maximize track usage.

Figure 2 below shows a schematic of the new PTC test bed track layout, which consists of a Main Line and an East-West Interurban Line. Outer dimensions of the PTC test bed are 18' x 8'. The Interurban Line includes a single passing siding (bottom center) and two figure eight shaped reversing loops. The Interurban tracks cross the Main Line tracks at four points, but trains operating on one line cannot transition to the other line. The Main Line includes multiple passing sidings and two rail yards. Complexity of the scheduling and safety monitoring software will increase as the software must not only schedule trains on two different rail lines, but it must also prevent collisions within each line and between the trains operating on the two different lines.

To mimic modern railway systems utilizing ABS, the PTC test bed track has been divided into multiple blocks for both scheduling and occupancy detection. Each siding will have at minimum three blocks. Short blocks will be used for siding entry and siding exit detection while a longer block or set of blocks in between will provide adequate space for one or more model trains to park off of the main track. Track control electronics will provide as a baseline block-level occupancy data, but this low precision localization mechanism will be augmented by precise navigation to facilitate the development of advanced scheduling algorithms.

2.2 Model Trains

Model railroading has been around for almost 100 years. The accuracy of the models has varied over the years. The scale of the models (N, HO, O, etc.) has been used to convey the relative size of model trains to real ones. For example, the size ratio of an HO model train to a real train is 1:87 while O gauge has a size ratio of 1:43. The accuracy of the modeling varies as needed. For example, while the scaled size of windows on an HO gauge train can be close to the true size of the windows. In the real train, the wheel flange, which holds the wheel on the track is usually much larger than an accurate scale would allow for to ensure that the train would leave the track on the slightest curve.

2.2.1 Digital Signaling and Control

Signaling is another area where fidelity varies based upon the need. Policies and procedures, such as block signaling are often tightly adhered to while signaling methodology varies considerably. Most HO gauge trains today utilize a signaling and control system called digital command control (DCC). In DCC, unlike earlier model train systems, power is applied continuously to the track. Digital data packets are transmitted to the individual locomotives that have computer decoders installed. The decoders read the message and adjust locomotive features such as speed, direction, sound, train type (slow, heavy freight vs fast, light, passenger train). Power on the track is low voltage AC and the digital data packets are overlaid on the AC power signal.

Digital data is created in a unit called a command module. This is converted to DCC signals in a combined command/booster module, or conveyed to separate power boosters using a CSCDMA network called LocoNet. Output from the boosters can be conveyed directly to the track. Since the location of the individual locomotive is unknown, the signal is sent to all track sections.

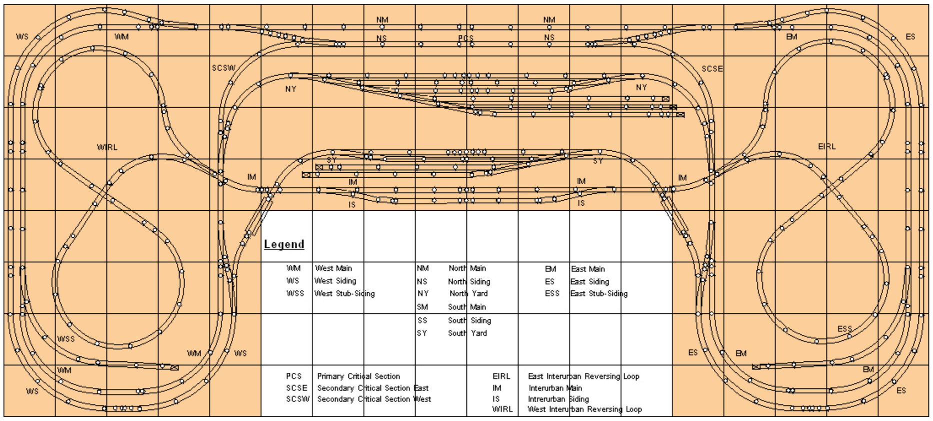


Figure 2 – Positive Train Control test bed track layout

Copyright © 2014 CSREA Press, ISBN: 1-60132-286-0; Printed in the United States of America

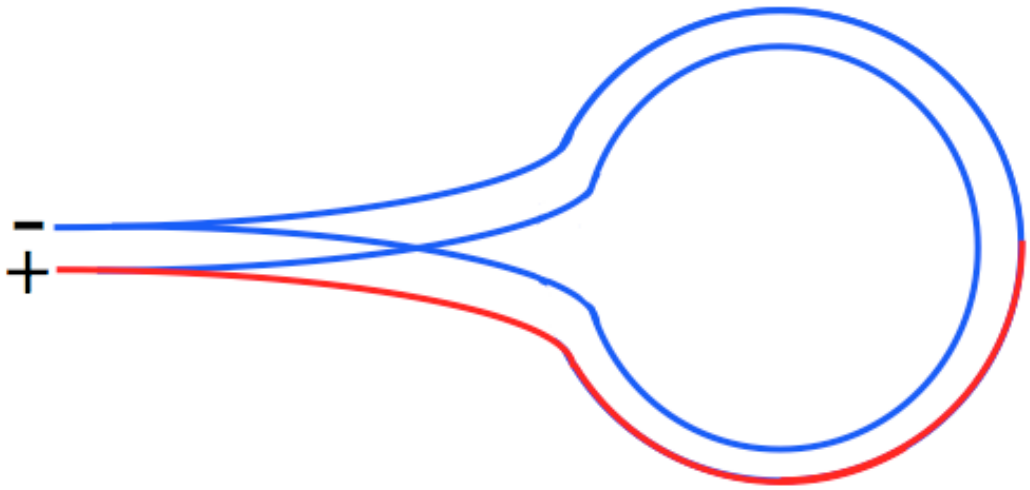


Figure 3 – Rail Diagram Showing Reversing Loop Polarity Short (counter-clockwise travel) [4]

2.2.2 Power Management, Protection, and Reversing Loops

Two power management strategies are also implemented at this level: power management and reversing. Power management, which can be implemented in a variety of ways, typically limits power on a track section to what is required by a few locomotives- typically 10 watts. When an overload or short is detected, the power management unit shuts down power to the offending section(s) of track. Reversing is a power management strategy that is needed when a train can reverse direction and traverse a section of track in both directions (going forward in both cases). In this case, as seen in Figure 3 above, if one track is hot and the other neutral, then when the track reverses hot and neutral are reversed. When a train wheel touches two sections of track that are reversed powered, a direct short occurs. To rectify this problem, a reversing power setup will reverse the polarity on one section of track upon detection of a short. If this does not clear the problem in a few milliseconds, the power management system kicks in and turns off power to that section.

Modern model train controllers have integrated coarse train location and identification capabilities. Train location power managers that distribute power to individual sections of track are available that if a train is detected in a block of track, a message is sent on LocoNet signaling that some power-consuming device is on that track section. The train itself is not identified and if this information is to be used then the controller system connected to the power manager has to keep track of which vehicle is in the section. As a vehicle traverses two sections then the system reports a vehicle is entering. As a controller continues to monitor the power manager, it maybe able to track the location of a train provided it knew its initial location and its speed and direction. Since there is no positive confirmation of this information at best systems of this type can only guestimate which train is in which section.

2.3 Transponding

A new development in model train power management is called transponding. In this case, the decoder on the locomotive can send information back into the rails while it is drawing power from the track. If the supervisory controller keeps careful note of which trains are entering and exiting a section of track and receives the transponder data contemporaneously, it is possible to track the entering and exiting of a particular train into and from a particular track section. However, the exact location of the train within the section cannot be determined by any current model railroad signaling system. In order to mimic current developments in positive train control on US railways, we have started development of a positive train control tested.

3 The UAH OnTrack System

3.1 System Overview

The UAH OnTrack system consists of an Android tablet application and an Arduino-based wireless controller module that is connected to Digitrax DCC hardware on the PTC test bed (Figure 4) [5]. The precise navigation sensor hardware mounted on each train sends data by Nordic NRF24L01+ radio to the wireless controller, which forwards this data back to the tablet via Bluetooth to accurately update the application display in real time. The tablet application allows a user to control operation of PTC test bed trains and track switches. The application utilizes two-way Bluetooth communications to interact with the wireless controller module. Train and switch commands (DCC and LocoNet) are forwarded from the tablet to the wireless controller, which forwards these commands via the Digitrax hardware by DCC or LocoNet to trains or switches as appropriate.

Copyright © 2014 CSREA Press, ISBN: 1-60132-286-0; Printed in the United States of America

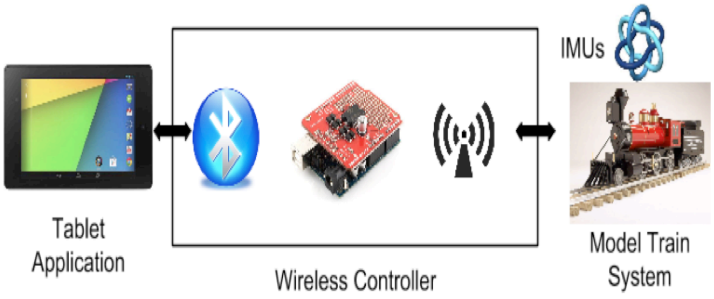


Figure 4 – Block diagram of UAH OnTrack system.

3.2 Key Design Goals

The goal of the UAH Software Safety and Security Laboratory in this effort is to allow students to study and advance safety and security aspects of the positive train control system. To accomplish this, train locations and velocities need to be determined more precisely than available through DCC block level signaling and detection. Positive train control systems use a variety of localization systems including GPS and inertial navigation units. Although it is possible to use GPS repeaters within a building GPS would still not yield precise enough location information (approximately 3 meters) for control of HO train systems. Furthermore, real trains must often operate in GPS challenged areas such as tunnels and underground yards in inner-city stations such as Grand Central Station in New York City. We therefore decided to look at an inertial navigation system scaled to the power and costs commensurate with model trains but providing extremely precise location and velocity information.

To reduce cost, the sensor package was based on an ArduIMU designed for model aircraft. This MEMS- based IMU includes 3-axis accelerometers, 3-axis angular rate sensing, and 3-axis magnetometers. MEMS-based IMUs are subject to a number of potential sources of error including noise, drift, and gravitational perturbations. A sample output of the x accelerometer, aligned in the direction of motion, is shown in Figure 5. As can be seen from this figure, the output of the accelerometer is quite noisy. A low pass filter would normally be used to obtain the envelope characteristics. However, the processor located in the gondola housing the IMU is extremely low power operating off batteries and had insufficient processing power to both filter and correct the data and maintain radio contact with the wireless controller.

Railroad tie counters have been used in real railway applications to obtain estimate of train speeds independent of wheel rotation. So, a tie counting system was integrated with the IMU sensor package to supplement the location and velocity information from the IMU. The tie counting system utilized a downward facing infrared source/sensor pair. It was determined experimentally that painting the cork roadbed under the track light grey reduced overall noise improving the accuracy of the tie counting system. The tie counting system can only provide speed information since the direction is unknown. The IMU data is used to determine train direction and location to within 1 cm is possible.

One additional output of the precise navigation system is which path of a switch was taken. When a train proceeds across a switch, we need to know not only the speed and direction (forward or backwards) but whether the train took the straight through or turnout direction. This information is also provided by the IMU system.

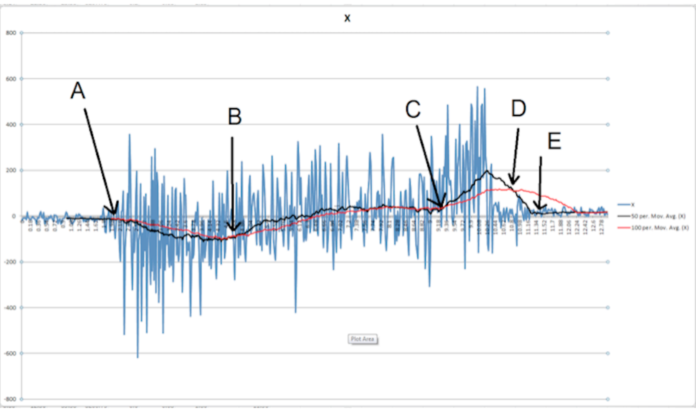


Figure 5 – IMU acceleration output in the x direction (blue trace).

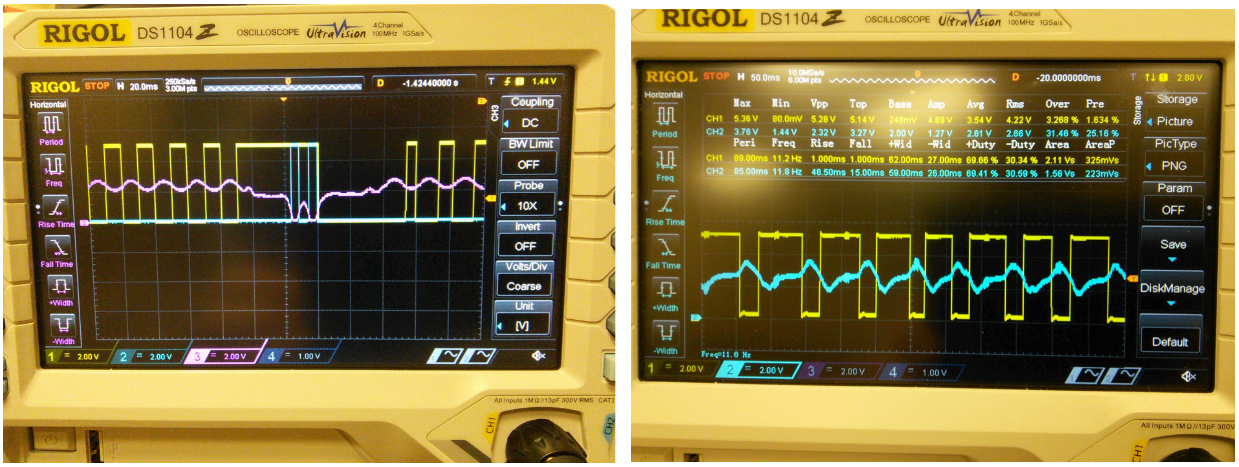
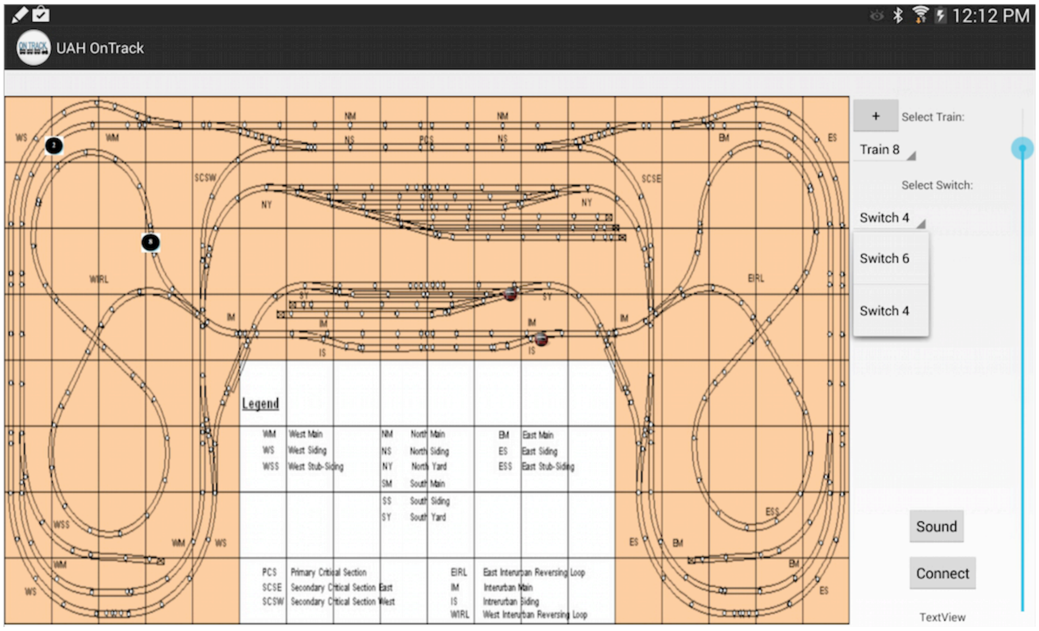


Figure 6 – Tie Counter: (A) Output in the Presence of Location Bar Code and (B) Bare tie output.



Legend

WM	West Main	NM	North Main	EM	East Main
WS	West Siding	NS	North Siding	ES	East Siding
WSS	West Stub-Siding	NY	North Yard	ESS	East Stub-Siding
		SM	South Main		
		SS	South Siding		
		SY	South Yard		

PCS	Primary Critical Section	EIRL	East Interurban Reversing Loop
SCSE	Secondary Critical Section East	IM	Interurban Main
SCSW	Secondary Critical Section West	IS	Intrerurban Siding
		WIRL	West Interurban Reversing Loop

Figure 7 – Sample screenshot of UAH OnTrack tablet app.

3.3 Track Bar Codes Re-zero Absolute Location

Two key issues are drift and initialization of the inertial navigation system. If an expensive IMU system, perhaps based on ring laser gyros, is used, then drift may not be significant. However, if a less expensive system, such as the MEMS based unit used in UAH OnTrack is used, then drift is a significant cause of error. In addition, the initial location of the IMU and attached locomotive needs to be established whenever the system is initialized.

Rather than require the locomotives be started in a fixed position, which would be a significant burden to the current system that might have 10's of locomotives on stub sidings, we approached the problem of drift and initialization with a single solution. In the UAH OnTrack system, bar codes are placed in between the tracks in known locations, such as on the entries and exits to switches and the end of stub sidings. The tie counter hardware was augmented with software to detect these barcodes in the track. Figure 6A below illustrates the signal output when the tie counter runs over a bar code and should be compared to Figure 6B where the tie counter is running over bare ties.

3.4 Tablet Application

The tablet application currently under development is a key component of the UAH OnTrack system since no existing software is capable of utilizing real time sensor package IMU and tie counter data for command and control. The application will support both layout acquisition and rail operations. For acquisition, the user overlays the locations of switches and bar codes on top of a photograph or schematic of the track layout. All switches, bar codes, and trains have integer identifiers. Trains are then added to the overlay displayed on the tablet screen by setting a numeric identifier and an approximate starting position. As with standard model train DCC controllers, the user may then use the tablet to command the physical train to move forwards, backwards or stop via a virtual on-screen throttle control. Data from the sensor package is used to correct the on-screen position of the train relative to the locations of the bar codes traversed. Gesture commands will be added to facilitate opening and closing of switches, changing focus of control from one train to the next, and emergency stop of all trains. Visual feedback to the user will include numbered train icons indicating current position and train identifier and the current setting of each switch. Figure 7 shows a screenshot of the UAH OnTrack application currently under development.

4 Conclusions and Future Work

The PTC test bed, in conjunction with the UAH OnTrack system, provides a low-cost research and education platform for researchers, graduate and undergraduate students. Efficient safety critical automatic scheduling algorithms will be key to full utilization of limited track resources in the future. The PTC test bed allows safe exploration of new scheduling algorithms to improve track utilization and techniques for verification of these algorithms. Users may also gain hands-on experience in software safety engineering by developing and verifying centralized safety supervisory systems via the PTC test bed. The test bed also provides a safe environment for students to explore the use of alternate technologies for precise localization and track inspection such as video tracking and ultrawide band radars.

5 References

[1] David J. Coe, Joshua S. Hogue, and Jeffrey H. Kulick, "Software Safety Engineering Education," *2011 International Conference on Software Engineering Research and Practice (SERP'11)*, WORLDCOMP 2011, July 18-21, 2011, Las Vegas, NV.

[2] Rail Safety Improvement Act of 2008, URL http://www.fra.dot.gov/eLib/Details/L03588

[3] Travis Cleveland, David J. Coe, and Jeffrey H. Kulick, "Video Processing for Motion Tracking of Safety Critical Systems," *2013 International Conference on Software Engineering Research and Practice (SERP'13)*, WORLDCOMP 2013, July 22-25, 2013, Las Vegas, NV.

[4] Reversing Loop Diagram, URL http://www.hmrg.co.uk/techtops/revloop.htm

[5] Digitrax, URL http://www.digitrax.com

Symbolic Model Checking Applied to Timing Diagrams

Amanda D. L. de O. Tameirao[1], **and Mark A. J. Song**[2]
[1]Department of Computer Science, FUMEC University
Belo Horizonte, Minas Gerais, Brazil
[2]Department of Computer Science, Pontifical Catholic University of Minas Gerais
Belo Horizonte, Minas Gerais, Brazil

Abstract—*During a software productŝ entire life cycle, errors and faults are introduced, which should be corrected as soon as possible. Among the most efficient forms of error correction is symbolic model checking. Symbolic model checking is a technique that enables developers to check for properties that do not conform to what was modeled, by going through every computation of a system. System modeling was shown to be an efficient aid in understanding the solution and minimizing errors. Among the proposed modeling methodologies is Unified Modeling Language (UML). UML is a visual modeling language to document, specify and build system artifacts. Although it has many resources, it lacks formal semantics, which hinders the use of symbolic checking. In this paper we discuss applications of model checking in UML through the translation of timing diagrams to symbolic model verification (SMV).*

Keywords: software engineering, model checking, timing diagram

1. Introduction

The UML visual modeling language is broadly used to specify, model, document and build systems. It has two specific model types that can be created: structural models, that highlight the physical organization; and behavioral models, that highlight a systems dynamic structure [1] [2]. Version 2.0 proposes 13 types of diagrams used to organize and understand every need in a development project. One of the proposed diagrams is the timing diagram, which is the main focus of this paper.

Timing diagrams are used to model systems for which time is the main execution factor and determines the action and response of every iteration.

One of the goals of this work is to guarantee that the model is drawn correctly and to avoid error propagation to subsequent process stages, like coding and testing. To achieve that, the modeled diagram will be translated into a verification model, and symbolic checking will be applied to validate the model.

Symbolic model checking is a technique used to assess whether a finite state model satisfies every property defined for it. If it does not, a counterexample is provided, allowing every path to modeled states to be traversed [3].

Associating both techniques is, however a challenge. UML is becoming more and more efficient in its modeling, aiding and supporting system development, but its lack of formal semantics makes it hard to apply model checking techniques, which use a mathematical language that is scarcely used by developers.

This study aims to bring the two techniques closer so that system development is more efficient and dependable. The rest of the paper is structured as follows, section 2 presents the diagrams contained in UML 2.0, highlighting the main diagram used in this paper, the timing diagram. Section 3 defines model checking. Section 4 there is a brief summary of the most relevant related work. Section 5 explains how timing diagrams are translated into model checking code. Section 6 contains translation examples and what kind of properties can be used. Finally, section 7 concludes the paper, and discusses future work prospects.

2. UML 2.0 Diagrams

A set of graphically represented elements is known as a diagram [2]. These diagrams can offer many perspectives for the visualization of a system. UML has, in version 2.0, 13 diagrams available for modeling a system [1] [2], split into two types, as follows:

- Structural diagrams, that focus on the physical structure of the modeled system. Represented by, class diagram, components diagram, composite structure diagram, development diagram and package diagram.
- Behavioral diagrams, that focus on the behavior of artifacts. Represented by, activity diagram, use case diagram, UML state machine diagram.
- Finally, interaction diagrams, which correspond to a subdivision of behavioral diagrams, and show the interaction between the components [4]. Represented by, sequence diagram, interaction overview diagram, communication diagram and timing diagram.

The timing diagram in this article is used together with the model checking.

2.1 Timing Diagram

The timing diagram was created to model systems for which time is of vital importance to the execution. It illustrates state changes that occur in a system, as well as interactions and restrictions existent between events. It is a behavioral diagram, and it is vastly employed in embedded and real-time systems.

Three elements can be used to model a timing diagram: lifelines, state conditions and time units.

Lifelines represent a systems main events. In Figure 1, we can see the indication of MailServer, that is, the indication that the systems main event is the server itself. The state conditions correspond to the state changes for each event. In Figure 1, these are represented by the inactive, authenticated and transmitting states, indicating that the MailServer lifeline can change into any of those states. Time units determine the temporal counting for the items. In Figure 1, they correspond to the progressive count from 0 to 19. All of the elements are connected and determine the systems actions, limits, and temporal restrictions.

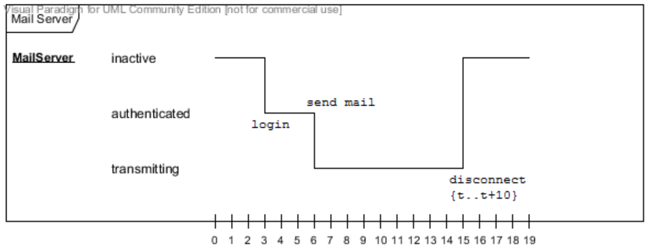


Fig. 1: Timing Diagram [5]

3. Model Checking

Formal verification is a technique used to spot errors that were not identified during usual testing. It provides a counterexample for cases where there is a divergence, allowing its origin to be traced [3]. It is largely used in critical systems, but even though is guarantees the reduction of flaws in a project, it is not very explored in common systems, due to the difficulty in using a specific formalism.

In this approach, a state transition graph models the system to be verified through formulae written in temporal languages, known as Computation Tree Logic (CTL), and Linear Temporal Logic (LTL). Each node corresponds to one of the systems states, determined by the values contained in each one of its variables. The edges correspond to the transitions between states. The verification procedure goes through all of the states and checks if they meet the defined required properties.

The following steps must be followed: first, the properties that a system must have to be considered correct must be specified. Then, a formal model to represent the system must be built. The model must include every property considered essential to the verification, leaving behind details that do not affect the correctness of said properties. Finally, the model checker must be run to validate the properties that were previously defined. In this way, the checker is employed to test if the model meets the specified requirements. If every property is satisfied, then the model is correct. Otherwise, a counterexample will be generated to demonstrate the failure.

4. Related Work

Formerly conducted studies concerning UML diagrams and their conversion to SMV were analyzed.

Amongst the most relevant studies, the papers by Hai-Yan [6] , Beato [7], Fernandes [8] and Santos [9] can be cited. This section will point out the main aspects of those studies.

In 2001, Hai-Yan et al [6] conducted a study about the translation of the UML state machine diagram to the SVM model checker. In this study, the authors used an automaton hierarchy formalism, and starting from it they translated the implementations into the SMV language, allowing properties to be subsequently inserted into the translation.

In 2005, Beato et al [7] defended the use of XMI to enable the translation of a diagram into SMV. In this study, they performed translations of class, activity and state machine diagrams. Properties were inserted using a tool developed by the authors.

In 2011, Fernandes [8] also used XMI, and by means of a self-developed tool, performed the translation of activity, state machine and sequence diagrams into the SMV checker, enabling properties to be inserted and tested at any time.

In 2014, Santos et al [9] transformed three behavior diagram, sequence, state machine and activity, to the SMV, to meet critical systems, in order to minimizer errors and quickly corrects them preventing them from propagating.

These studies demonstrate that diagram translation is viable and welcome, because currently only the development of critical systems have their efficacy guaranteed through the use of model checking.

5. Translation of th Timing Diagram to the SMV Checker

Figure 2 demonstrates this works main proposal, showing the involved stages and the environment created for the translations.

Proposal stages:

- Drawing the conceived timing diagram in a UML modeling tool;
- Exporting the diagram to XML Metadata Interchange (XMI);
- Including the XMI file in the environment developed by the authors for the translation into formal verification language, so that it can be evaluated by the SMV verifier;
- Once in possession of the translated diagram, setting, through the environment, the properties that should be evaluated and including them in the translated file;

Copyright © 2014 CSREA Press, ISBN: 1-60132-286-0; Printed in the United States of America

- Running the verification using the SMV tool, to obtain an answer.

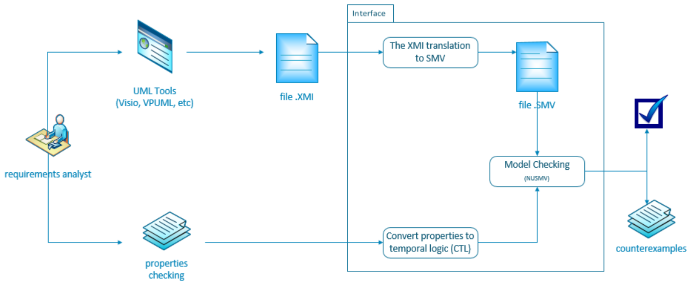


Fig. 2: Diagram-to-SMV translation stages

Following these steps, the translation of the diagram shown in Figure 1 was performed and the following results were obtained, in Table 1. Each item in the timing diagram is reported to the XMI file by tags and specific types, following the diagrams reading order:

Table 1: Translation of the Timing Diagram to SMV

UML	SMV	Initial value
Lifelines	Variables	First value of a conditional state
State conditions	Variable states	
Time units	Time when transition states occur	First value indicated in the diagram
Time instances	Determine the occurrence of state changes in lifelines	

Items taken from the XMI file are described in SMV as follows:

- In a timing diagram, lifelines represent the maximum number of states the diagram will be split into. In an XMI file, they are represented by the `timingFrameLifeline` type. Figure 3 demonstrates what the representation looks like in an XMI file, followed by translation performed.

```
<packagedElement name="mailServer" xmi:id="_YzaIkKGAqACBwTG"
                                   xmi:type="timingFrameLifeline">
```

Fig. 3: Lifeline description in an XMI file

In the translation made by the environment, the name is used to define the variable. Each timing diagram can have more than one lifeline (Table 2):

Table 2: Translation of the Lifelines to SMV Language

```
MODULE main
VAR
    mailServer : inactive, authenticated, transmiting;
```

- State conditions represent transaction changes that occur in each lifeline. In an XMI file, they are represented using the `stateCondition` (Figure 4).

```
<packagedElement name="inactive" xmi:id="J9raIkKGAqACBwTH"
                                 xmi:type="stateCondition">
    <xmi:Extension extender="Visual Paradigm for UML">
        <qualityScore value="-1"/>
    </xmi:Extension>
</packagedElement>
```

Fig. 4: State condition description in an XMI file

In the environment-generated translations, the names are used to define the possible states of a variable (Table 3):

Table 3: Translation of the State Conditions to SMV Language

```
MODULE main
VAR
    mailServer : inactive, authenticated, transmiting;
```

- Time units define the exact moment f the occurrence or replacement of an action. In an XMI file, they are represented by the `timeUnit` type. Figure 5 exemplifies how the representation is done in an XMI file, followed by the translation generated by the proposed environment (Figure 5).

```
<packagedElement name="0" xmi:id="8P3aIkKGAqACBwTK"
                          xmi:type="timeUnit">
    <xmi:Extension extender="Visual Paradigm for UML">
        <qualityScore value="-1"/>
    </xmi:Extension>
</packagedElement>
```

Fig. 5: Time unit description in an XMI file

The translation created by the proposed environment uses the names to define the possible states of a counting variable (Table 4):

Table 4: Translation of the Time Unit to SMV Language

```
MODULE main
VAR
    mailServer : inactive, authenticated, transmiting;
    counter : 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19;
```

- Time instances determine exactly when state changes will take place. They store the initial and final moments of every event. In an XMI file, they are represented using the `timeInstance` type. Figure 6 shows an example of such an XMI representation and is followed by an example translation by the proposed environment.

```
<packagedElement name="" stateCondition="ExBkIkKGAqACBwNk"
     timeUnit="P.xkIkKGAqACBwNm" xmi:id="WBxkIkKGAqACBwNq"
                                   xmi:type="timeInstance">
    <xmi:Extension extender="Visual Paradigm for UML">
        <qualityScore value="-1"/>
    </xmi:Extension>
</packagedElement>
```

Fig. 6: Time Instance description in an XMI file

The following is an example translation made by the environment, showing the states with their respective initial and final times, determining when they will occur (Table 5):

Table 5: Translation of the Time Instance to SMV Language

```
next(mailServer) := case
    counter >= 0 & counter <= 2 : inactive;
    counter >= 3 & counter <= 6 : authenticated;
    counter >= 7 & counter <= 14 : transmiting;
    counter >= 15 & counter <= 19 : inactive;
esac;
```

After all translations are completed, the final code generated from the XMI file shown in Figure 1 will be (Table 6):

Table 6: Translation timing diagram Figure 1 to SMV

```
MODULE main
VAR
    mailServer : inactive, authenticated, transmiting;
   counter : 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19;
ASSIGN
    init(mailServer) := inactive;
    init(counter) := 0;
    next(counter) := case
       counter = 0 : 1;
       counter = 1 : 2;
       counter = 2 : 3;
       counter = 3 : 4;
       counter = 4 : 5;
       counter = 5 : 6;
       counter = 6 : 7;
       counter = 7 : 8;
       counter = 8 : 9;
       counter = 9 : 10;
       counter = 10 : 11;
       counter = 11 : 12;
       counter = 12 : 13;
       counter = 13 : 14;
       counter = 14 : 15;
       counter = 15 : 16;
       counter = 16 : 17;
       counter = 17 : 18;
       counter = 18 : 19;
       TRUE : 0;
    esac;
    next(mailServer) := case
       counter >= 0 & counter <= 2 : inactive;
       counter >= 3 & counter <= 6 : authenticated;
       counter >= 7 & counter <= 14 : transmiting;
       counter >= 15 & counter <= 19 : inactive;
    esac;
```

6. Definition of Properties and Verification

This section presents some possibilities of formal verification applied to the timing diagram translation. For data representation, some properties were previously defined and verified by the checker.

6.1 It is globally true that the execution of an activity implies in the execution of another one in the future

SPEC AG ((mailServer = authenticated) -> AF (counter > 5))

Considered the example illustrated in Figure 1, the specification questions whether it is true that whenever the mail server is currently in the `authenticated` state it is implied that, in the future, the value of the `counter` variable will always be greater than 5. For this property, the checkers answer will be *TRUE* (Table 7).

Table 7: Result of the property

```
– specification AG (mailServer = authenticated -> AF counter > 5) is true
```

6.2 It is always true that, for the next state in which a state occurs, it is implied to be globally true that another state will occur

SPEC AX(mailServer = inactive -> AG(counter > 10))

Considered the example illustrated in Figure 1, the specification questions whether it is true that, in every situation, for the next state in which the mail servers action is `inactive`, it is implied that counter will always be greater than 10. For this property, the checkers answer will be *FALSE*, and in this case it provides a counterexample to back its answer (Table 8).

Table 8: Result of the property

```
– State: 1.1 <-
   mailServer = inactive
   counter = 0
– State: 1.2 <-
   counter = 1
```

6.3 It is globally true that the execution of an activity implies in the execution of another one in the future

SPEC AG ((mailServer = authenticated) -> AF (counter > 5))

Considered the example illustrated in Figure 1, the specification questions whether it is true that whenever the mail server is currently in the `authenticated` state it is implied that, in the future, the value of the `counter` variable will always be greater than 5. For this property, the checkers answer will be *TRUE* (Table 9).

Table 9: Result of the property

```
– specification AG (mailServer = authenticated -> AF counter > 5) is true
```

6.4 There is some computation for which, in the future, if a state occurs, another one will occur

SPEC EF(mailServer = transmiting & EG(counter <= 5 | counter > 15))

Considering the example illustrated in Figure 1, the specification questions whether there exists a path in which the mail server currently has the `transmitting` action and counters value is less than or equal to 5, or greater than 15. For this property the checker will return *FALSE*, and provide the following counterexample to corroborate the answer (Table 10).

Table 10: Result of the property

```
– State: 1.1 <-
  mailServer = inactive
  counter = 0
```

6.5 There is some computation for which, in the future, an activity depends on some other activity

SPEC EF(mailServer != authenticated & counter <7)

Considering the example illustrated in Figure 1, the specification questions whether there exists a path in which, in the future, the mail server will have the `authenticated` action and counters value will be less than 7. For this property, the answer provided by the checker is *TRUE* (Table 11).

Table 11: Result of the property

```
– specification EF (mailServer != authenticated & counter < 5) is true
```

7. Conclusion and Future Work

The aforementioned related work demonstrates that the insertion of symbolic model checking in usual systems adds more reliability to them, as well as financial gain prospects, by avoiding errors that were not previously found.

This paper presents a contribution by translating the timing diagram, allowing, through the developed environment, any system to be validated and verified with a greater probability of success, which guarantees more dependability. The developed environment has the goal to unburden analysts involved in projects by making their work a bit easier, by taking from them the requirement of knowing the formalisms that accompany temporal languages.

As future work, it will be possible to develop a plugin that will enable automatic model verification. From an XMI file generated by a UML modeling tool, the environment will generate the SMV translation, allow the definition of properties, add them to the previously generated translation and check the model, eliminating some of the steps that comprise the approach proposed here.

Another possible approach would be pointing out errors in the diagrams. Since the checker can generate counterexamples, they could be read and a path inversion could be generated, pointing out to the analyst the exact contention point, broadening the collaboration for a more assertive development process.

These improvements will provide analysts with more precision in work carried out during the requirements identification and system modeling activities, while also minimizing error propagation to future project stages.

Acknowledgements

The authors acknowledge the financial support received from the Foundation for Research Support of Minas Gerais State, FAPEMIG.

References

[1] Pender, T.; UML Bible. Wiley. 2003.

[2] Booch, G.; Rumbaugh, J.; Jacobson, I. The Unified Modeling Language User Guide. Addison-Wesley. 2005.

[3] Clarke, E. M.; Grumberg, O.; Peled, D. A. Model Checking. The MIT Press, 1999.

[4] Object Management Group. (2014). UML 2.4.1 Superstructure Specification. [Online]. Available: http://www.omg.org/spec/UML/2.4.1.

[5] Pilone, D.; Pitman, N. UML 2.0 in a Nutshell. OŔeilly Media, 2005.

[6] Hai-yan, C.; Weil, D.; Ji, W.; Huo-wang, C. V̈erify UML statecharts whit SMV.̈ Wuhan University Journal of Natural Sciences. China, vol. 6, p 183-190, 2001.

[7] Beato, M.E. ÜML automatic verification tool with forma methods.̈ Eletronic Notes in Theoretical Computer Science. Spain, vol. 127, p 3-16, 2005.

[8] Fernandes, F. G.; Song, M. A. J. V̈erification of UML Behavioral Diagrams using Symbolic Model Checking.̈ In: IADIS Applied Computing. Brazil. p. 1-8. 2011.

[9] Santos, L. B. R.; Santiago JÃžnior, V. A.; Vijaykumar, N. L. T̈ransformation of UML Behavioral Diagrams to Support Software Model Checking.̈ In: Formal Engineering Approaches to Software Components and Architectures. France. p. 133-142. 2014.

A Quality Assurance Approach to Technical Debt

Zadia Codabux, Byron J. Williams, Nan Niu
Department of Computer Science and Engineering
Mississippi State University
Starkville, Mississippi

Abstract - *Technical debt is a metaphor used to reason about lingering issues in software development. It is the result of decisions taken during the development process, which are often made due to resource constraints and aggressive schedules. The consequences of technical debt are that it can impede future development and incur increasing costs. However, technical debt is not always bad as, in some cases, respecting deadlines is more important than clean code. Nevertheless, for achieving quality software, it is crucial to prevent the amount of debt from increasing. In this paper, we describe a method for addressing technical debt using a quality assurance (QA) classification scheme and focus on prevention, reduction, and containment activities. We also highlight techniques and processes that are used to apply quality assurance to technical debt.*

Keywords: technical debt, software quality, software testing

1. Introduction

Software development is prone to failure. One of the root causes of this failure is the use of predictive processes for complex software. Predictive processes attempt to fully define all requirements upfront. Traditional software development models such as the waterfall model, which involves predictive processes, are not the most appropriate when business needs and technology change rapidly. It is often difficult to establish a complete vision for the software product at inception, list all the requirements clearly, and devise a detailed plan to convert these requirements into the finished product [1].

A shift in thinking occurred when development began to focus on customer needs and the product rather than the processes. This shift gave rise to agile software development, which involves an incremental, iterative approach to project development where the requirements and their solutions evolve through collaboration between development teams and customers [1]. In addition, as stated by Williams and Cockburn, agile development is “about feedback and change,” and agile methodologies are developed to “embrace, rather than reject, higher rates of change [2].”

One of the main foci of agile software development is quick release of functionality. While so doing, technical debt can arise whenever issues not solved in the current release will have to be addressed in later releases [1]. This technical debt metaphor was coined by Ward Cunningham in the 1992 OOPSLA experience report to describe how long-term code quality is traded for short-term gains such as increased productivity. Cunningham stated, “Shipping first time code is like going into debt. A little debt speeds development so long as it is paid back promptly with a rewrite. Objects make the cost of this transaction tolerable. The danger occurs when the debt is not repaid. Every minute spent on not-quite-right code counts as interest on that debt. Entire engineering organizations can be brought to a stand-still under the debt load of an unconsolidated implementation, object-oriented or otherwise” [3].

In other words, technical debt is the consequence of tradeoffs made during software development, as illustrated in Figure 1. These tradeoffs include not preserving architectural design; not using good programming practices, conventions and standards; no longer updating documentation; and not performing thorough testing [4]. However, every minute spent on “not-quite-right” code counts as interest towards the debt [3]. The metaphor has since been extended and refers to any imperfect artifact in the software development life cycle [5].

Technical debt can be classified as unintentional or intentional. Unintentional debt is acquired inadvertently, for example, due to low quality of work or the lack of adherence to good programming practices. However, intentional debt is accumulated for strategic reasons, e.g., compromising on proper coding standards in order to deliver software on time. Intentional debt is further classified as short-term or long-term debt [6].

Another classification of technical debt is based on traditional lifecycle phases such as testing debt, design debt, and documentation debt [7]. Documentation debt is the result of inadequate or outdated documentation. Design debt is the result of any anomaly or imperfection in the source code. Testing debts are tests that were planned but not implemented or executed.

Over the last decade, technical debt has gained a lot of attention from the software engineering research community.

Copyright © 2014 CSREA Press, ISBN: 1-60132-286-0; Printed in the United States of America

However, the minimal amount of empirical studies existent in the field indicates that there is still need to bridge the gap between research and practice [9]. This can be achieved by carrying out more empirical studies involving practitioners and identifying best practices from the software engineering literature to address technical debt.

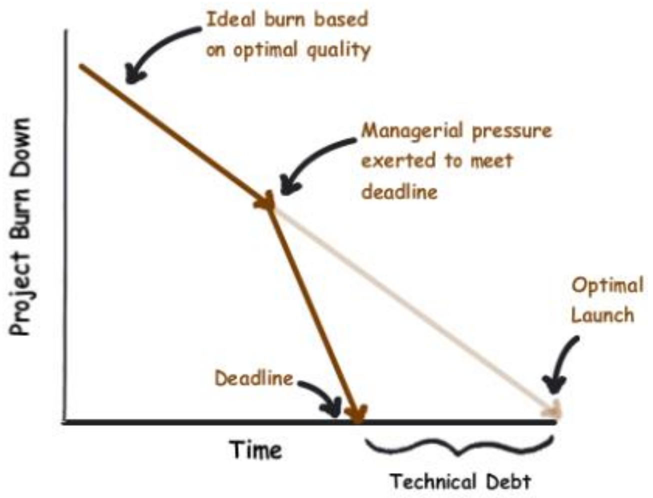


Figure 1: Technical Debt [8]

Technical debt if not handled, can become a problem to organizations. Seaman et al. [10] enumerated some examples of the consequences of technical debt as

- disastrous consequences
- large cost overruns
- quality issues
- inability to add new features without disrupting existing ones
- premature loss of a system

This paper proposes different practices to prevent, reduce and contain technical debt in an agile software development environment context.

The rest of the paper is structured as follows: Section 2 addresses the impact of technical debt. Section 3 focuses on the relation between quality assurance and technical debt while presenting a quality assurance approach to managing technical debt. Section 4 presents some recommendations for managing technical debt. Finally, Section 5 concludes the paper.

2. The Impact of Technical Debt

At the start of a new project, it is common to intentionally incur technical debt in order to achieve some goals. When technical debt is incurred for strategic reasons, it is due to the opportunity cost of releasing software now compared to some point in the future. For instance, McConnell points out that when time to market is critical, incurring technical debt might be a good business decision. Other instances where debt can be incurred may be due to time and resource constraints where the software or feature needs to get out of the door and the software will be “fixed” after the release. In addition to time-to-market factor, he also explained how preservation of startup capital contributes towards technical debt. In a startup company, expenses that can be delayed should be, as opposed to expending startup funds on technical debt now. Another case where it might be justifiable to incur technical debt would be near the end of a system’s lifecycle because when a system retires, all the debts retire with it [6].

As explained earlier, technical debt is not always bad if it allows a business to achieve a competitive edge and market share. In such cases, it makes sense to delay the moment when the software is brought in line with standards and best practices. Good technical debt has one or more of the following characteristics [11]:

- It has a low interest rate – technical debt with low interest rate can be supported for a longer time frame as the development costs won’t increase over time. Repaying the debt at a later time might cost insignificantly more than it will cost to repay it now.
- It is being regularly serviced - when technical debt is regularly serviced, the amount of debt will decrease over time, thereby increasing the quality of the software.
- The work that preceded it had a very high opportunity cost – if there is a high opportunity cost related to the debt, it is preferable to release the software and incur some debt, rather than not releasing and losing money.

The decision to assume the debt must be made explicit and should be the result of a collective decision based on the key stakeholders concerned. Such a decision will normally be taken after assessment of the risks and benefits identified. In addition, the stakeholders must ensure that a process exists to manage the debt and that it is paid back within a set period of time.

However, non-strategic debt can be detrimental to the quality of the software. Such situations might include when the debt is taken on without stakeholder approval and the impact of the debt is not properly assessed. It is harder to track and manage technical debt accumulated due to shortcuts taken as a result of the lack of quality assurances processes [6]. This type of debt is comparable to credit card debt as it can be easily incurred and adds up very fast due to compounding interest.

Nonetheless, a process to pay back technical debt is needed. The longer the debt repayment is deferred, the harder and more costly it will be to pay it back as the interest charges keep compounding. Technical debt is often quantified as principal and interest. Principal refers to the cost of completing the task at present. Interest is the extra cost added to the principal that will be needed to complete the task at a later time. These are the basic factors for calculating the Return on Investment (ROI) on resolving the debt if the costs can be determined.

Over time, technical debt can lead to degeneration of the system’s architecture. The lack of prompt debt payment can result in technical bankruptcy where an organization's resources are spent dealing with the inefficiencies created by the debt that has accumulated over time and can no longer keep up [12]. In the worst case, the software might need a complete redesign or need to be rewritten, entire departments are outsourced, customers and market shares lost and customer confidence is lost as the company will be spending more resources on debt servicing rather than focusing on new features [13].

A famous example is Netscape Navigator, which experienced architecture decay over a short time period. Netscape developers wanted to release a newer version of the software but could not because the code was harder to change than expected and the system became unmaintainable. The architecture was difficult to understand and it became almost impossible to add new components [14]. Another example is Visual Query Interface (VQI), a software package that degenerated as the programmers made changes to the system without following the architectural guidelines provided. The programmers introduced design pattern violations which cause unnecessary couplings, misplaced classes (i.e., classes placed in the wrong package), and imported classes not used in the package [15]. This degeneration is referred to as code decay [13].

Code decay can have several root causes. One cause is violating architectural design principles. For example, in a strictly layered system, where a layer can only use the services provided by the layer below. If a developer does not follow the constraint, this change is considered a violation of the architecture.

Other causes for code decay include:

- Time pressure that causes programmers to knowingly postpone refactoring
- Writing code without following proper programming conventions
- Debugging code improperly
- Taking shortcuts to get a working solution as fast as possible

The above examples illustrate that technical debt gives rise to code decay, which makes code changes harder than they should be.

Hochstein et al. [13] reported that in order to find areas of code decay, they used a tool to detect code smells which helped to identify code areas where good design principles were breaking down. A code smell is a surface indication that usually corresponds to a deeper problem in the system [16]. As a result, code smells are useful to identify areas accumulating technical debt.

As illustrated above, technical debt can be disastrous if not handled properly. The next section proposes different techniques to handle technical debt with the aim of producing better quality software.

3. Technical Debt and Quality Assurance

Technical debt can be used to reason about software quality. Seaman et al. pointed out that metrics such as cohesion, coupling, complexity and depth of decomposition for software quality can be used to quantify technical debt. Some examples include: the software has defects and needs to be reworked, the software does not fulfill its requirements, all test cases have not been executed, or missing documentation. Example indicators of technical debt include the number of defects in the system, the number of requirements that remain unfulfilled, the number of test cases that need to be executed or automated and documentation that requires completion updating. Technical debt can be characterized by the amount of work that needs to be done and associated costs in order to address these problems [17].

In order to better select, adapt and use QA techniques most appropriate for specific applications, Tian proposed the following Quality Assurance classification scheme for defects [18]:

- Defect prevention
- Defect reduction
- Defect containment

Defect prevention activities consisted of activities whose aim is to prevent certain types of faults from being injected in the software. These activities include eliminating technical ambiguities and correcting human misconceptions about functionality. Another activity is fault prevention or blocking that involves directly preventing missing or incorrect human actions.

Defect reduction activities have the objective of detecting and removing faults once they have been injected in the system. This is commonly achieved using inspections, testing, static analysis or observations during dynamic execution.

Defect containment activities involve containing the failures to local areas to limit their impact. Activities include using fault tolerant techniques to break the relationship between fault and failure.

The QA classification described above has been adapted to technical debt. The following section will depict how technical debt can be prevented, reduced and contained with the overall aim of managing the amount of technical debt in the software, thereby increasing its quality.

3.1. Technical Debt Prevention

The aim of technical debt prevention activities is to proactively reduce the chance of technical debts being injected in the software. Different strategies are proposed to handle the different sources of debt injections.

- *Education and Training* - In cases that human misconceptions are sources of error, the software professionals can be trained on the organization's development methodology, technology and tools, and development process. Education and training will reduce the probability of implementing the wrong solutions and subsequent rework in design, coding and testing phases. These issues are potential contributors to the technical debt backlog.

- *Pair Programming* - Pair programming is an agile technique whereby two software developers work side by side at a workstation, one writing the code while the other reviews the code. A study carried out by Cockburn et al. found that pair programming improves design quality, reduces defects and enhances technical skills [19]. Such best practices will help reduce unclear, unreadable, duplicated code which is often the source of technical debt.

- *Test-driven Development* - Test-driven development is an approach to software development that requires writing a failed automated test before writing the code that makes the test pass. Once the code passes the test, the cycle is repeated including refactoring the existing code to ensure clean code that always works. Thus, test-driven development reduces risk because developers have a better understanding of what the software should achieve and the code has fewer defects. Consequently, this code does not contribute to the technical debt backlog.

- *Refactoring* - Refactoring is a technique for restructuring code by improving its internal structure without affecting its external behavior. Refactoring improves not only aspects of code quality but also productivity [20]. Examples of refactoring involve deleting unused code and duplicated code.

- *Continuous Integration* - Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily, leading to multiple integrations per day [21]. Continuous integration speeds development and the team is able to deliver at any moment a version of the software suitable for release. In addition, it encourages feedback between customers and the development team, which assists with ensuring proper functionality before set deadlines.

- *Conformance to process and standards* - When the source of the debt is due to non-conformance to processes, standards and conventions then enforcement either at the organization or project level will help prevent technical debt injections. Such activities theoretically contribute to high quality software, thereby reducing the amount of defects.

- *Tools* - Moreover, tools that can aid in reducing the chance of debt injections should be utilized; for example, a syntax-directed editor can help to reduce syntactical problems.

- *Customer Feedback* - Getting customer feedback can help clarify any requirements misinterpretations early. This feedback will assist in preventing the wrong solution from being implemented, thus increasing technical debt. Prototyping is a tool that can be used to aid in getting customer feedback by allowing them to interact with the "working system" and communicating their satisfaction or concern before the software is actually deployed [22].

- *Others* – There are other best practices that can contribute to technical debt prevention:
    - Bug bashes which involve the entire team putting aside their daily activities to review and exercise a part of the code base with the objective of quickly revealing defects.
    - Having dedicated teams whose primary focus is on reducing technical debt [9].
    - Having each development team dedicating one iteration during a set release period towards debt reduction [9].
    - Allowing Slack – planning some lower priority tasks that can be dropped if the team gets behind. In this way, the team will deliver the most important functions.
    - Reflective improvement – developers take a break from software development and try to find ways to improve their processes. This helps to pinpoint processes that are and are not working and modify them to develop a strategy that works better.

However, none of the techniques listed above are 100% foolproof and cannot guarantee prevention of technical debt. In addition, it is very difficult for a company to adopt each and every best practice suggested above. For example, prototyping can be a costly technique and might not be suitable for small companies. A company with limited manpower might not find pair programming a very attractive solution. Refactoring requires some level of expertise and experience with the code. Novice programmers may not be able to refactor large portions of the code. A company might adopt one or more of the suggested techniques and best practices, according to what they think will be most fruitful to them, based on the resources at their disposal and the types of system being developed.

3.2. Technical Debt Reduction

The aim of technical debt reduction activities is to remove as much of the injected debt as possible. There are numerous good practices in agile software development that can benefit technical debt reduction. Such agile practices include code reviews (using static analysis tools), test automation and customer feedback.

- *Code reviews* using static analysis tools assess the quality of the code without executing the code to reveal violations of recommended programming practices that might affect the quality of the software [4]. These tools help to pinpoint problematic code areas such as long methods, large classes, and duplicated methods and help reduce technical debt [23].

- *Automating unit tests* act as a safety net by providing immediate feedback to the team. In addition, automating regression testing in an automated build process permits growth of solid code while quickly iterating. Automated tests protect the software in the sense that it prevents certain defects from reoccurring without being detected [23].

- *Customer feedback* is an essential agile practice. After an iteration, the new features should be demonstrated to the customers for feedback. Such an initiative will ensure that the development team is on the right track and not accumulating any specification debt.

Performing regular code reviews and automated testing are techniques that can be used to identify and reduce technical debt. Involving the customer provides the added benefit of validating the software before its release. However, despite the availability of static analysis tools, it might not always be possible to carry out extensive code reviews. Often, these tools report many false positives that limit the effectiveness of the tools. Similarly, when all unit tests cannot be automated due to resource constraints, a team can accumulate test debt. Based on their resource availability, a company can decide which of the proposed techniques work best for them.

3.3. Technical Debt Containment

Technical debts that bypassed the debt prevention and reduction activities intentionally or unintentionally remain present in the software. It is common practice to have working software deployed at a customer's site with known technical debt present. For these situations, technical debt containment is the quality assurance approach that must be taken. It is imperative to isolate the impact of known debts so that other parts of the software are not impacted.

While there is a lack of techniques proposed in existing literature to contain technical debt, N-version programming (NVP) is one applicable approach.

- *N-version programming* – NVP is the independent generation of functionally equivalent programs from the same initial specification [24]. It is a technique that can be adapted to contain technical debt by introducing duplications in the software. Often, developers are hesitant to refactor complex code for fear of introducing additional defects. When there are independent versions of the software, programmers will be able to refactor one version of the code. If there is a failure in that particular version, the software will still function correctly, due to the multiple versions.

NVP is a technique that can be used for technical debt containment. However, NVP has its limitations. First, it is a technique that relies heavily on the accuracy of the requirement specifications. Also, it is based on the assumption that the software built differently will result in few similar errors. Lastly, the cost with implementing N versions of a program instead of one increases development cost [25]. These restrictions may prevent a challenge to the adoption of NVP, but the need to contain the impact of lingering debts makes it a viable solution.

4. Recommendations For Managing Technical Debt

Despite a company's desire to prevent and reduce technical debt, it is difficult to get rid of all the debt. Gradually, the software development team needs to pay off the debt as a commitment towards ensuring quality software. Laribee proposed a four-phase approach to tackle technical debt [26]:

1. Identify the technical debt and its location.
2. Prioritize the debt with the help of the concerned stakeholders and the team members.
3. Fix the debt.
4. Repeat phases 1-3.

Phase 1 is a crucial step as it is imperative to perform a technical debt analysis to assess how much technical debt is present in the code base and make it visible. Free tools such as Sonar [27] can aid in the process of code quality transparency. Sonar has a technical debt plug-in that identifies potential issues before they affect delivery and shows the approximate cost to pay back all the technical debt in a module or across different modules as in Figure 2.

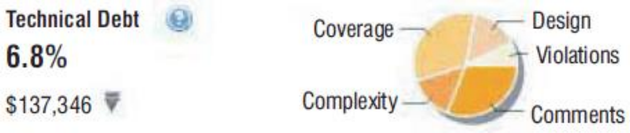

Technical Debt
6.8%
$137,346

Figure 2: Sample Information Extracted from Sonar [23]

Such tools can help build up a technical debt list, each item in the list representing an uncompleted task. For each debt item, the list should contain a description of where the debt is, why the tasks need to be addressed, estimates of the principal and interest, and effort estimates. It is common practice to maintain the list within the company's defect tracking system as a debt backlog.

Once technical debts have been identified, the next step is to communicate with the concerned stakeholders so that they are aware of the issues. Next, the decision can be made on how to prioritize the technical debt list such that the most critical items are addressed first.

Seaman el al. [10] propose 4 distinct decision–making approaches for prioritizing technical debt. In the Cost/Benefit Analysis approach, the principal, interest and interest probability of each technical debt item is assigned ordinal scales of measurements such as low, medium and high, which can help make coarse-grained preliminary decisions. For example, a company may decide to tackle 75% of debt with high interest and 25% of debt with medium interest and defer the ones with low interest. Secondly, the Analytic Hierarchy Process (AHP) assigns weights and scales to different criteria which are used to measure technical debt. Then, a series of pair-wise comparisons is performed between the alternatives to get a prioritized ranking of the technical debt items. The Portfolio approach is based on the return on maximization of investment value and investment risk minimization to decide what technical debts should be addressed first. Lastly, the Options approach is analogous to investing in refactoring the debt item with the long-term objective of facilitating maintenance in the future and thereby saving money.

Snipes et al. assess deferred defects to determine the amount of technical debt and prioritize fixes. [28]. They identified the factors that influence the decision as follows:

- *Severity* - assesses the impact of the defect on the customer’s normal system operations
- *Existence of a workaround* - defers the defects to a later release and temporarily fixing the issue using an alternate approach
- *Urgency of fix required by a customer* - occurs when the customer requests a fix for a particular defect
- *Effort to implement the fix* - refers to the effort and time required to fix and test the defect
- *Risk of the proposed fix* – refers to the extent to which the code and functionality will be changed when the defect is fixed
- *Scope of testing required* - determines whether regression tests need to be run on the resulting defect fix

The factors listed above are of decreasing order of importance.

After technical debt has been identified and prioritized, the technical debt items need to be integrated into the existing development backlog. Identifying and prioritizing technical debts are the initial phases in the whole process. The real challenge for the team is to actually work on the debt backlog to reduce the debt and transform a system of marginal quality into a sustainable, high quality set of artifacts.

5. Conclusion

Technical debt is part of every project and affects different artifacts in the software development life cycle. It is important that technical debt is visible to all concerned stakeholders. An additional good practice is to repay the technical debt within a tolerated time frame. Indeed, if not handled promptly, the consequences can be drastic. Proactively applying prevention and reduction QA activities will ensure that quality artifacts are being produced. Reactively, the technical debt needs to be identified, located and contained.

In this paper, we have proposed a set of quality assurance activities that focus on prevention, reduction and containment of technical debt. Prevention activities include pair programming, test-driven development, continuous integration, formal methods, amongst others; reduction activities include customer feedback, code reviews and automated testing; and containment activities include N-version programming.

This paper provides a single source for common best practices for technical debt and can help organizations that have newly adopted agile methods to better manage their technical debt and prevent the debt backlog from growing. More mature companies can also benefit from this work as they may decide to include more best practices into their iterations or shift from one technique to another and this list can help guide their choices.

6. References

[1] K. Schwaber and J. Sutherland, Software in 30 Days: How Agile Managers Beat the Odds, Delight Their Customers, And Leave Competitors In the Dust. John Wiley & Sons, 2012.

[2] L. Williams and A. Cockburn, “Agile software development: it’s about feedback and change,” Computer, vol. 36, no. 6, pp. 39–43, 2003.

[3] W. Cunningham, “The WyCash portfolio management system,” in Addendum to the proceedings on Object-oriented programming systems, languages, and applications (Addendum), Vancouver, British Columbia, Canada, 1992, pp. 29–30.

[4] A. Vetro, “Using automatic static analysis to identify technical debt,” in 2012 34th International Conference on Software Engineering (ICSE), 2012, pp. 1613 –1615.

[5] C. Seaman and N. Zazworka, “IEEE/Lockheed Martin Webinar on Identifying and Managing Technical Debt,” Nov-2011. [Online]. Available: http://www.nicozazworka.com/research/technical-debt/.

[6] S. McConnell, “Technical Debt.” [Online]. Available: http://www.construx.com/10x_Software_Development/Technical_Debt/. [Accessed: 14-Apr-2013].

[7] J. Rothman, “An Incremental Technique to Pay Off Testing Technical Debt.” [Online]. Available: http://www.stickyminds.com/sitewide.asp?Function=edetail&ObjectType=COL&ObjectId=11011. [Accessed: 14-Apr-2013].

[8] G. Lipka, “The UX of Technical Debt,” Feb-2011. [Online]. Available: http://commadot.com/the-ux-of-technical-debt/. [Accessed: 14-Apr-2013].

[9] Z. Codabux and B. Williams, “Managing technical debt: An industrial case study,” in 2013 4th International Workshop on Managing Technical Debt (MTD), 2013, pp. 8–15.

[10] C. Seaman, Y. Guo, N. Zazworka, F. Shull, C. Izurieta, Y. Cai, and A. Vetro, “Using technical debt data in decision making: Potential decision approaches,” in 2012 Third International Workshop on Managing Technical Debt (MTD), 2012, pp. 45–48.

[11] S. Chin, E. Huddleston, W. Bodwell, and I. Gat, “The Economics of Technical Debt,” Cutter IT Journal, vol. 23, no. 10, 2010.

[12] T. Theodoropoulos, “Technical Debt – Part 1: Definition.”

[13] L. Hochstein and M. Lindvall, “Diagnosing architectural degeneration,” in Software Engineering Workshop, 2003. Proceedings. 28th Annual NASA Goddard, 2003, pp. 137–142.

[14] M. W. Godfrey and E. H. S. Lee, “Secrets from the Monster: Extracting Mozilla’s Software Architecture,” in In Proc. of 2000 Intl. Symposium on Constructing software engineering tools (CoSET 2000, 2000, pp. 15–23.

[15] R. T. Tvedt, P. Costa, and M. Lindvall, “Does the code match the design? A process for architecture evaluation,” in International Conference on Software Maintenance, 2002. Proceedings, 2002, pp. 393–401.

[16] M. Fowler, K. Beck, J. Brant, W. Opdyke, and D. Roberts, Refactoring: Improving the Design of Existing Code, 1st ed. Addison-Wesley Professional, 1999.

[17] C. Seaman and Y. Guo, “Chapter 2 - Measuring and Monitoring Technical Debt,” in Advances in Computers, vol. Volume 82, Marvin V. Zelkowitz, Ed. Elsevier, 2011, pp. 25–46.

[18] J. Tian, “Quality Assurance Alternatives and Techniques: A Defect-Based Survey and Analysis,” 01-Jun-2001. [Online]. Available: https://secure.asq.org/perl/msg.pl?prvurl=http://mail.asq.org/pub/sqp/past/vol3_issue3/sqpv3i3tian.pdf. [Accessed: 10-Mar-2014].

[19] A. Cockburn and L. Williams, “The Costs and Benefits of Pair Programming,” in In eXtreme Programming and Flexible Processes in Software Engineering XP2000, 2000, pp. 223–247.

[20] R. Moser, P. Abrahamsson, W. Pedrycz, A. Sillitti, and G. Succi, “A Case Study on the Impact of Refactoring on Quality and Productivity in an Agile Team,” in Balancing Agility and Formalism in Software Engineering, B. Meyer, J. R. Nawrocki, and B. Walter, Eds. Springer Berlin Heidelberg, 2008, pp. 252–266.

[21] M. Fowler, “Continuous Integration,” May-2006. [Online]. Available: http://martinfowler.com/articles/continuousIntegration.html. [Accessed: 12-Mar-2014].

[22] E. Allman, “Managing technical debt,” Commun. ACM, vol. 55, no. 5, pp. 50–55, May 2012.

[23] B. Barton and C. Sterling, “Manage Project Portfolios More Effectively by Including Software Debt in the Decision Process,” Cutter IT Journal, vol. 23, no. 10, 2010.

[24] L. Chen and A. Avizienis, “N-Version Programming: A Fault-Tolerance Approach to Reliability of Software Operation,” in Twenty-Fifth International Symposium on Fault-Tolerant Computing, 1995, Highlights from Twenty-Five Years, 1995.

[25] V. Bharathi, “N-Version programming method of Software Fault Tolerance: A Critical Review,” in National Conference on Nonlinear Systems and Dynamics, NCNSD, 2003.

[26] D. Laribee, “Using Agile Techniques to Pay Back Technical Debt.” [Online]. Available: http://msdn.microsoft.com/en-us/magazine/ee819135.aspx. [Accessed: 22-Apr-2013].

[27] “SonarQubeTM.” [Online]. Available: http://www.sonarqube.org/. [Accessed: 16-Jan-2014].

[28] W. Snipes, B. Robinson, Y. Guo, and C. Seaman, “Defining the decision factors for managing defects: A technical debt perspective,” in 2012 Third International Workshop on Managing Technical Debt (MTD), 2012, pp. 54 –60.

Towards Automatic GUI Testing Using Task and Dialog Models

Eman M. Saleh
Software Engineering Department, Applied Science Private University, Amman, Jordan

Abstract - *Testing GUIs for correctness can enhance the entire system's safety, robustness, and usability. The major difficulty in testing a GUI comes from fact that it is impossible to cover all possible interactions with all possible paths they might lead to. This paper introduces a model-based testing framework of GUIs that can automatically generate test cases from the task and dialog models.*

Keywords: Model-based UI testing; ConcurTask Trees; Dialog models; State charts

1 Introduction

Graphical user interfaces (GUIs) are the most dominant way in interacting with today's software systems. The ease of using computers and handheld devices through GUIs makes them a daily need for people to do their work with a small amount of training, or even no training at all. On the other hand, a special attention should be taken in designing and implementing the user interface as it represents the first line of interaction between the users and the underlying software. This implies the importance of testing GUIs as they affect both the acceptability and the success of the software as well as the entire system's safety, robustness and usability [1].

Despite their widely usage and their increased importance; GUIs remain the least considered aspect in software testing research area. Traditional testing techniques that are used to test conventional software are not appropriate to test GUIs which is due to two main reasons [1]: (1) The extremely large input domain when considering user clicks, selections and screen touches at all possible valid and invalid input situations which leads to test case explosion problem [2] (2) traditional coverage criteria do not work well for GUIs. For example considering white box coverage criteria do not work with testing GUIs because GUI software differs from the underlying application code in its level of abstraction so mapping between GUI events and the underlying code is not straightforward [1]. (3) Test oracles usage in traditional software differs from the way should be used for testing GUIs; in traditional software testing test oracles are used to compare test results against the oracle (expected results) after the testing is completely done.

Model-based testing enables the testing of an implemented software artifact against a model that represents the expected results or "what it should be" which is known as the oracle. In the case of model-based user interface testing, the models should be expressed in an abstract way and, on the other hand, it should contain elements that are closely related to the GUI structure and behavior. Models that are developed during the analysis phase of the software engineering process are too abstract to express the UI in an adequate way, such as the task model or the use-case model.

In this paper the model-based GU testing framework is based on the ConcurTask Tree (CTT) task model [3] and a test oracle, namely, the Dialog-States Model (DSM) [5], which is a dialog model derived automatically from the task model and represents all interactions and state transitions for a given (CTT). The states in the DSM represent abstract screens or windows and the navigation between them for the application under test.

This paper is structured as follows: sections 2 introduces related work to model-based testing in general, section 3 introduces CTT task models and the Dialog States Model (DSM) and section 4 describes how the DSM can be used as an oracle in a model-based testing technique of GUIs; the paper ends with conclusions and ideas for future work in section 5..

2 Related work

Test automation using models has been extensively studied in recent years; especially the use of UML models. The closest work is the work in [7] where test cases are derived from UML state charts. The main difference between their work and the proposed work in this paper is that their sate chart is derived by transforming the textual use cases descriptions while we derive the state chart from the navigational behavior of the GUI relying on the task model (CTT) which is enriched with temporal operators that do not only specify the tasks that are carried out by the actors but also the order and information dependencies between these tasks. Another work that matches the idea of this paper is IDATG [8] (Integrating Design and Automated Test Case Generation) environment for GUI testing. IDATG supports the generation of test cases based on both a model describing a specific user task with is similar to our task model by specifying the order of tasks execution and a model capturing the user interface behavior which is also similar to the DSM

Copyright © 2014 CSREA Press, ISBN: 1-60132-286-0; Printed in the United States of America

used in this paper for to capture the behavior of the system, our framework goes beyond the idea of the behavior by defining a more abstract model that can derive test cases for any user interface taking into consideration multiple target platforms; ideally this is inspired by the fact the DSM model was built with multi-platform GUIs are to be generated [9].

In [10], Marlon Vieira, et. al, proposed a technique to derive test cases from activity diagrams after annotating them with test data requirements; this annotation is time consuming and subject to errors which may lead to faults in the generated test cases, compared to the proposed method of this paper, the transitions on the Dialog States Model (DSM) are computed automatically based on the CTT task types and the semantics of the temporal operators between the tasks. The details of this algorithm are out of the scope of this paper and can be found in details in our previous work in [9].

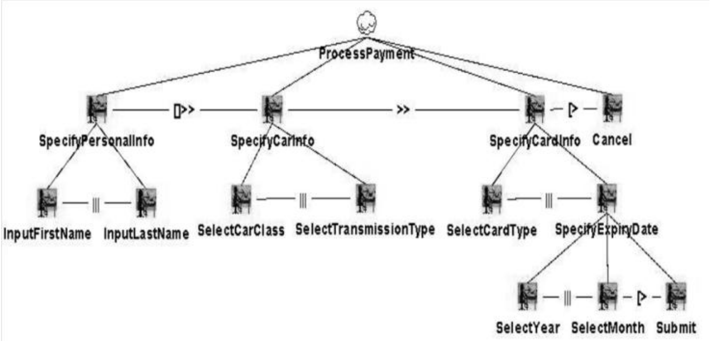
3 CTT Formalism and the Dialog-States Model

Our methodology is built around model based design and development of user interface, hence, we assume that the user interface is generated by defining set of models starting by the CTT task model is our starting point but is not suitable as a test oracle nor to derive test cases since it is an abstract representation of the tasks performed while interacting with the GUI. It may not help to represent the event response states of the GUI. As we mentioned the model to be used as a test oracle should be closest to GUI nature (e.g. events, transitions and new states), for this purpose the test cases are to be derived from the Dialog-States model. The following sub-sections will explain briefly both of these models.

3.1 The CTT Task Model

CTT notation [3] is a hierarchical task model, it is considered as the most usable specification for task modeling in multiplatform model-based UI design and development [11]. It provides a graphical syntax, a hierarchical structure and a notation to specify the temporal relation between tasks, an example of CTT task model is shown in Fig. 1.

Fig. 1. Simple Example of CTT Task Model



With this notation, tasks can be classified into four categories: abstract tasks ,interaction tasks ,user tasks and application tasks . Tasks at the same level can be can be connected by temporal operators like choice ([]), independent concurrency (|||), concurrency with information exchange (|[]|), disabling ([>) , enabling (>>), enabling with information exchange ([]>>), suspend/resume (|>) and order independence (|=|). The precedence of these operators from highest to lowest is: [] > {|||, |[]|}> {[>,|>} > {>>,[]>>} [28].

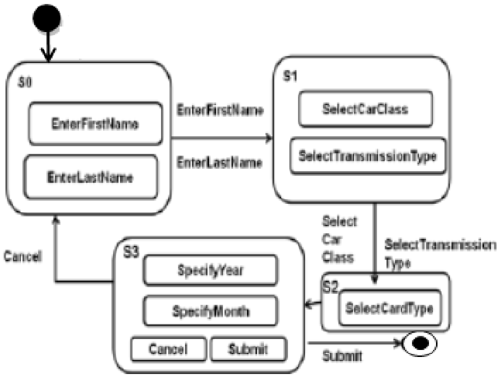
3.2 The Dialog-States Model

We originally defined the Dialog-States model as a transition point from the abstract model toward more concrete models [5] that represent the final GUI. The idea behind this is to define the navigational structure of the GUI in a platform independent matter which implies the possibility to customize it through transformations to a concrete GUI that suits the target platform or device, where a platform is triple of: <hardware device, Operating System, Users>.

The DSM is represented as StateCharts where it is automatically derived from the task model using the algorithm described in [9], it finds abstract screens generated by analyzing the CTT task model depending on the semantics of the temporal operators. The algorithm finds all states including the start and the final states; another algorithm finds all transitions as well as, all guards that label these transitions.

The following Example, in Fig. 2 which is extracted from [5] shows the corresponding DSM for the ConcurTask Tree in Fig. 1. As mentioned before an algorithm is used to derive the states and the transitions between them

Fig. 2. The Dialog-States model for the CTT in Fig. 1 [5]



4 Model Based GUI Testing Using DSM

Basic requirements for an automated software testing criterion are a test case generator, a coverage criteria and a test oracle.

In the proposed framework the DSM represents the test oracle as it specifies the expected behavior of the GUI. In GUIs test oracle usage need to be interleaved with test execution as various user actions may lead to different states of the system.

Test coverage is the most important aspect in conventional software testing, and more important and difficult in GUI testing as it is hard to exhaustively test every action performed by the use (mouse movements, clicks, double clicks, menu selections, …etc.) which might lead to test case explosion problem. To cope with this we propose to automatically derive test cases from the DSM upon a test coverage criterion defined by the tester.

The DSM is capable to define and extract test cases depending on basically three test coverage criteria:

1. State coverage: where the minimum requirement of the derived test sets is to cover every state defined on DSM, to prevent test case explosion and to minimize the number of test cases every state has to be covered once by the test suite.

2. Transition coverage: Where every transition between two states has to be covered at least once.

3. Path coverage: Every path in the DSM has to be covered at least once by the test suite. This implies both state and transition coverage.

5 Conclusion and Future Work

In this paper we presented a framework to automatic test case generation of GUIs using a model based approach. The framework is based on the CTT [3] model and the DSM [5]. The main contribution is solving one of the problems of model-based technique; which is the need for creating a model that that will serve as the test oracle; this is due to the automatic derivation of the DSM. We also show how the derivation nature of DSM will help to derive test cases based on different coverage criteria.

The DSM was built around the idea of designing of multi-platform user interfaces with any screen sizes taking into consideration devices with very small screen size; hence, the DSM derivation algorithm suffers from the large number of states derived. Future work is needed to modify the DSM derivation process to minimize the number of the generated states and hence the number of generated test cases.

Extensive work is needed to implement a testing tool that integrates the DSM with an IDE in order to run tests and find test results by interleaving the execution of the test oracle (DSM) and the actual GUI interactions.

6 Acknowledgment

Eman Saleh is grateful to the Applied Science Private University, Amman, Jordan, for the financial support granted to cover the publication fee of this research article.

7 References

[1] A. M. Memon, M. E. Pollack, and M. L. Soffa, “Hierarchical GUI Test case generation using automated planning,” IEEE Trans. on Soft. Eng. (TSE), vol. 27, no. 2, pp. 144-155, February 2001.

[2] C. R. Paiva, N. Tillmann, J. C. P. Faria, and R. F. A. M. Vidal, "Modeling and Testing Hierarchical GUIs", in Proceedings of the ASM 2005 - 12th International Workshop on Abstract State- - 12th International Workshop on Abstract State Machines, Paris - France, March 8-11,2005.

[3] F. Paternò, C. Mancini, and S. Meniconi, "ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models", in Proceedings of the Interact'97, 1997.

[4] Jan Peleska and Wen-ling Huang: "Exhaustive Model-Based Equivalence Class Testing". In Yenigün, Hüsnü and Yilmaz, Cemal and Ulrich, Andreas (eds.): Testing Software and Systems, Proceedings of the ICTSS2013. Springer, LNCS 8254, pp.49-64, 2013.

[5] Eman Saleh, A. kamel, and A. Fahmy, “Dialog States a multi-Platform Dialog model”, ECS journal, pp. 1-8, vol. 33, Sep. 2009.

[6] P., A. C. R. P., Automated Speci_cation-Based Testing of Graphical User Interfaces," Ph.D. thesis, Engineering Faculty of Porto University, Department of Electrical and Computer Engineering (2007). URL: http://www.fe.up.pt/~apaiva/PhD/PhDGUITesting.pdf.

[7] P. Fröhlich, J. Link, “Automated Test Case Generation from Dynamic Models”. In: Bertino, E. (Ed.): Proceedings of the ECOOP 2000 pp.472–491, 2000.

[8] A. Beer, S. Mohacsi, and C. Stary: “IDATG: An Open Tool for Automated Testing of Interactive Software”. Proceedings of the COMPSAC '98 - 22nd International Computer Software and Applications Conference, pages 470-475, August 19-21, 1998

[9] Eman Saleh, Amr Kamel and Aly Fahmy, “An MDE Design Approach for Developing Multi-Platform User Interfaces”, WSEAS Transactions On Computers Journal, Issue 5, Volume 9, ISSN: 1109-2750, ACM press, pp. 536-545, May, 2010.

[10] Marlon Vieira, et. al. "Automation of GUI testing using a model-driven approach", Proceedings of the 2006 international workshop on Automation of software test, pp. 9-14, ACM, 2006.

[11] Giulio Mori, Fabio Paterno`, and Carmen Santoro, "Design and Development of Multidevice User Interfaces through Multiple Logical Descriptions", IEEE Transactions on Software Engineering, VOL. 30, NO. 8, pp. 1-14, August, 2004.

An environment for automatic tests generation from use case specifications

Carolina D. Cunha[1] **and Mark A. J. Song**[1]
[1]Computer Science Department, PUC MINAS, Belo Horizonte, Minas Gerais, Brasil

Abstract—*This paper proposes an environment for the specification of use cases (UCs) and their derivation in automated functional tests for web applications. The proposed environment is based on our own domain specific language (DSL), formalized by means of a BNF grammar (for the Xtext framework), offering all the amenities and usability of the Eclipse IDE. The proposal seeks to tighten the relation between system specifications and tests, aiming to facilitate and incentivize the use of automated tests and their continuation, since they are frequently abandoned due to cost and deadline limitations. All of the tools used in the process, from specification to generation, execution and management of tests, are free and in the public domain.*

Keywords: functional-tests, use-cases, automation

1. Introduction

Tests are fundamental activities in software development, because they are the last product evaluation resource before delivery to end users. Among the various types of tests, functional tests, in which systems are evaluated concerning their inputs and outputs, without considering their internal workings, stand out. Because they are based on system's specifications, functional tests are largely used, because software products must behave like what was defined with users during requirement analysis [1].

The demand for faster development of software products while retaining quality calls for the very frequent execution of tests, especially regression tests, to assure that new functionality did not compromise the behavior of previously implemented and validated requirements [2]. Although regression testing is important, it is one of the most costly stages of software development and maintenance. According to [3], software tests can account for 30 to 50% of total development cost, and according to [4] it is estimated that regression testing is responsible for up to 80% of total testing costs, and up to 50% of total software maintenance costs.

The realization of automated tests, in addition to reducing time spent testing the software, allows for an indirect increase in software quality, since efforts can be focused in other types of tests or in tests that cannot be automated [5]. Automating repetitive tasks not only reduces costs, but also improves precision, since humans are slow and prone to error when dealing with such tasks.

It is perceivable that the generation of functional test cases is still a predominantly manual and arduous task [6], as is their execution, according to experience obtained in large companies - an especially complicated scenario in large-scale and long-lasting projects [5].

For test automation, test scripts are written that replicate manual tests. The generation of functional test scripts can be done by manual coding or by a recording tool (like the Selenium IDE [7]), using the "Capture and Replay" technique.

At first sight, the technique seems like a feasible solution, because it requires no programming skills. However, if a test needs to be altered, so does the recorded script - which would require such skills - or a new recording must be manually made [5]. Besides, scripts generated by those tools are linear, with hard-coded inputs and comparisons, with no possibility of reusing a script to compose new tests, not to mention being hard to write and maintain. Furthermore, the link between original and generated code is weak, making it difficult to pinpoint specific reasons for an error identified during testing.

In spite of the advantages in using automated code, they are considerably costly to maintain, because for every alteration in the system under test (SUT), tests must be reviewed and updated as needed. If they fail to be updated due to cost or deadline reasons, they become worthless. According to [8], only a small part (sometimes less than 20%) of tests are automated because of these difficulties.

One way to reduce testing efforts while still guaranteeing effectiveness would be to automatically generate Test Cases (TC) from artifacts that were previously used during development [9]. One of the most widely used ways to capture software requirements is through Use Case (UC) description, a central mechanism recommended in Unified Process (UP) and in many modern methods [10], [11]. TCs derived from UCs can be used to check if system requirements (UCs) were correctly implemented [3], [9].

The goal of this work is to generate test scenarios from UCs and, from these scenarios, generate executable TCs that are converted into automated test scripts that retain a connection to the specification.

2. Related Work

Many studies seek to use UCs to generate TCs or test scenarios. In [9], test scenarios with sequencing based on

UC's textual pre and post conditions are generated and described in controlled language. Generated scenarios have a high abstraction level, opposite to what is proposed here, a more concrete-level approach.

In [10], textual UCs described in a controlled language and a domain model are used as inputs. Data inconsistencies, like entities being described with multiple names, UC operations that do not refer to a concept operation in the domain model, among others. Scenarios are generated by UC composition, and their execution is simulated in a state machine with a tool called UCEd. Like in [10], we propose a domain model with possible value information and intend to generate scenarios with UC sequencing, but for real test execution.

In [12] and [13], a UC specification language named SilabReq, based on Xtext [14], is presented. Domain model, system operation list, UML UC model and state, sequence and activity diagrams are generated from UCs. [13] proposes dividing the specification into different abstraction levels, since UCs are used by people of different roles, with different needs, during software development, ranging from end users, requirement engineers, to designers, developers and testers. Like [12] and [13], we propose a UC specification language, with varied abstraction levels, using Xtext as base. However, since our focus is generating executable tests, the proposal encompasses elements beyond what is explored in SilabReq.

According to [15], to contemplate the needs of every role involved in a software development project, it is necessary to specify UCs in many notations. A notation set is thus proposed, based on structured text, and interaction and activity diagrams, with defined rules to convert one into the other. A test-friendly notation is mentioned as an important and integral part of the set, but was not mentioned in the paper, apparently because it was not worked on by the group.

In [16], a tool called TestGen is used to generate activity diagrams and possible scenarios from UCs. A correctness study is conducted for the generated TCs using mutant analysis. Our proposal also uses the aforementioned tool, but distinguishes itself by generating executable test scripts.

In [17], UCs are specified in an environment called Text2Test, based on the Xtext framework, in which validations vary according to user profile. We also propose an Xtext-based environment with validations, but focusing on test execution.

[2] presents Webspec, a visual language with diagrams, used to capture browsing, interaction and interface characteristics for web applications. Webpsec was built as an Eclipse plugin and uses mockups (UI doodles) for simulations. It enables the generation of code and Selenium tests. Like [2], our solution generates executable tests for Selenium and uses Eclipse, but is based in a descriptive language instead of diagrams.

Finally, in [18], an approach to generate test scenarios from UCs with contracts formalized in OCL and sequence diagrams. The required formalization in [18] elevates the technical level necessary to understand and maintain UCs.

3. Proposed Approach

For the present work, we developed a DSL for UC description, formalized in a BNF grammar (fig 1). The grammar, fed to the Xtext framework, is used in the generation of the language's concrete syntax (by means of a meta-model), for the proposed environment.

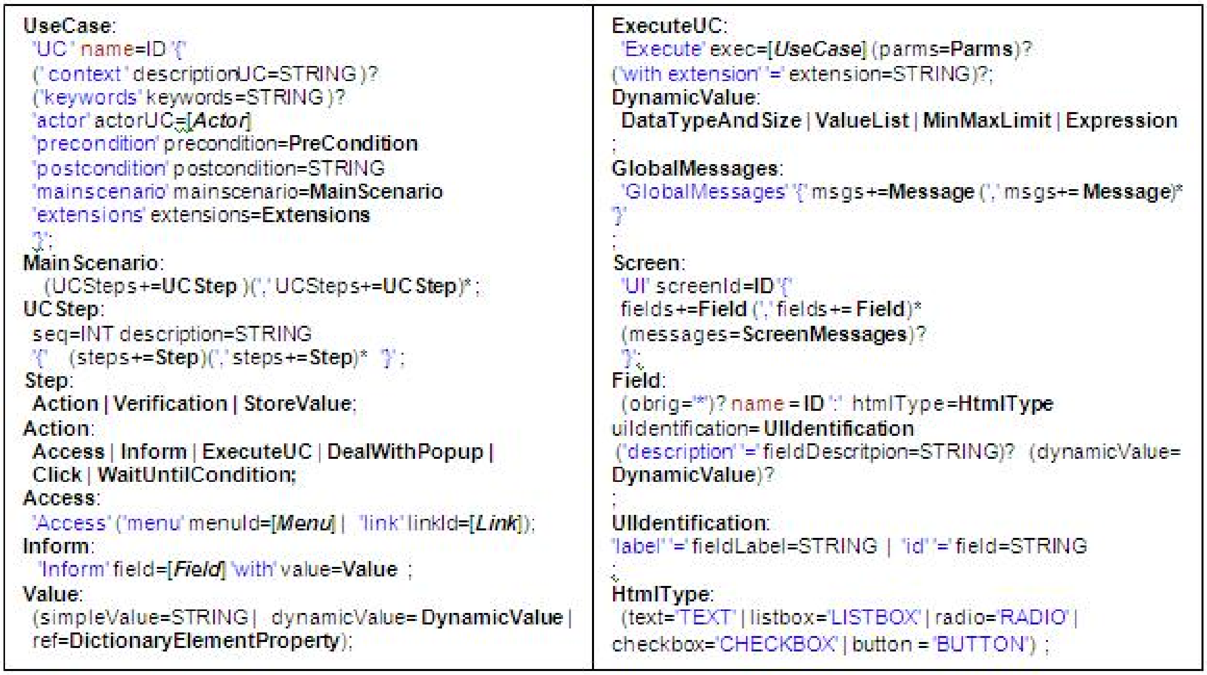
We opted to use the "Fully Dressed" use case template, presented in [19], which has a positive response in practical experiments reported by the author. This template uses one text column and numbered steps with a convention of numbers and letters for alternate steps. This numbering structure supports identification and generation of scenarios.

UCs can be described with varying levels of detail, according to the project's stages. In [11], three detailing levels for writing UCs are presented, and it is suggested that UCs be written in an essential style especially in the beginning of specification - when the focus must be on intention, and not on UI - and in a concrete/detailed style in the following stages, when the UI project is available.

In this paper it is proposed that every step of the UC in the abstract level be followed by its detailing, where UI details are inserted. By adding UI information to UC steps, test scripts can be generated. The use of an environment where UCs are kept linked to tests helps reflecting into the latter changes made in the former. Even if the changes are not entirely automatically realized, they become motivated by the connection of UC steps to their detailing and by validation-generated warnings. Besides, UCs produce useful debugging information when a test fail (fig 8). So, as can be seen in figure 1, UCSteps are composed of Steps, that can be of types: Action (User interactions with the screen), Verification (Assertions that validate system responses) or Storage (Storing values for later use).

In addition to UCs, UIs must also be described - with their elements and menus - optionally with a domain model (data dictionary), that helps to resolve ambiguity, registering composite terms and the relationship between the elements. Changes to the dictionary or UI generate warnings to users if they break references in UC steps (or vice-versa) (fig 5). Fields and messages are referenced in the use cases through internal IDs. When changes occur, they only need to be reflected in the UI specification, which facilitates maintenance (PageObject pattern [20]).

When indicating an input value to be inserted into a UI element, fixed values can be used directly, or rules for dynamic generating input in execution time can be stipulated. Dynamic values can stem from field type description (type, minimum and maximum values), from an expression or from a list of values. The rule specification for dynamic value generation can be made in a UC step, in a determined



```
UseCase:
 'UC' name=ID '{'
 (' context ' descriptionUC=STRING )?
 ('keywords' keywords=STRING )?
 'actor' actorUC=[Actor]
 'precondition' precondition=PreCondition
 'postcondition' postcondition=STRING
 'mainscenario' mainscenario=MainScenario
 'extensions' extensions=Extensions
 '}';
Main Scenario:
   (UCSteps+=UC Step )(',' UCSteps+=UC Step)* ;
UC Step:
 seq=INT description=STRING
 '{'   (steps+=Step)(',' steps+=Step)*  '}' ;
Step:
 Action | Verification | StoreValue;
Action:
 Access | Inform | ExecuteUC | DealWithPopup |
 Click | WaitUntilCondition;
Access:
 'Access' ('menu' menuId=[Menu] |  'link' linkId=[Link]);
Inform:
  'Inform' field=[Field] 'with' value=Value  ;
Value:
 (simpleValue=STRING |  dynamicValue= DynamicValue |
 ref=DictionaryElementProperty);
```

```
ExecuteUC:
 'Execute' exec=[UseCase] (parms=Parms)?
('with extension' '=' extension=STRING)?;
DynamicValue:
 DataTypeAndSize | ValueList | MinMaxLimit | Expression
;
GlobalMessages:
 'GlobalMessages' '{' msgs+=Message (',' msgs+= Message)*
'}'
;
Screen:
 'UI' screenId=ID '{'
 fields+=Field (',' fields+= Field)*
 (messages=ScreenMessages)?
 '}';
Field:
 (obrig='*')? name = ID ':'  htmlType=HtmlType
uiIdentification= UIIdentification
 ('description' '=' fieldDescritpion=STRING)?  (dynamicValue=
DynamicValue)?
;
UIIdentification:
'label' '=' fieldLabel=STRING  |  'id' '=' field=STRING
;
HtmlType:
 (text='TEXT' | listbox='LISTBOX' | radio='RADIO' |
checkbox='CHECKBOX' | button = 'BUTTON') ;
```

Fig. 1: Some rules of proposed DSL .

screen field or a determined property in the data dictionary. If specified in the dictionary, it is possible to vary the specification for ranges, called equivalence classes. Should a rule be used in multiple UCs, it should be defined in a single location (the dictionary or the screen) and be referenced in the UC steps. If this rule is used in various screens, it becomes more interesting be defined in the data dictionary

Test generation must be run every time an alteration to the specification is realized. For test generation, it is first necessary to generate test scenarios, traversing path possibilities in the basic and alternative flows (an example is shown in section 4).

Among the many options to generate the paths of a UC, like a graph or FSM (finite state machine) conversion, we opted to treat the UC like an activity diagram, because it is easily understood by users, helping them visualize and validate the completeness and correctness of UC paths.

3.1 The environment

This paper presents an environment for the specification of UCs - and their unfolding into TCs - based on the open-source framework Xtext. Through this framework, it is possible to obtain a specific editor for a language, on the Eclipse IDE [21], with all the characteristic support, which favors productivity and usability features, like: support to "folding", "autocomplete", "syntax highlighting", "structure outline" and "cross-referencing", in addition to easily managing artifact versions.

The environment validates, in editing time, the informed text, according to the supplied grammar. During data input, the allowed options in each point are presented ("autocomplete" resource). When it is a cross-reference point, options for the referenced type are offered. When specifying a menu access, for example, a list containing all declared menu options is shown. Semantic validations are also considered during editing, and added to the code generated by the framework. An example of such validation is the verification that every mandatory field (indicated by ’*’) were filled, in a UC’s steps, before the step that requires submitting the screen. When an error is found, it is indicated in the text and in a tab that helps fixing it ("Quick Fix" resource) (fig 5). Due to the fact that validations happen in editing time, the author can iteratively refine the UC. Standardization in writing UCs is aided and guided by the environment, improving its quality and facilitating the identification of its components, that can be used for automatic transformations afterwards [12].

For scenarios generation, we opted for the TestGen [16] tool, which takes the high-level UC steps as input and generates test scenarios (.tol files) and an activity diagram (.xmi file) as outputs, for each UC. Test scenarios are generated for every path obtained from the diagram.

Test scripts are generated in Java, and begin their execution using JUnit [22]. The scripts use, for interaction with the UI, the Selenium WebDriver [7] tool, for simulating user navigation. This tool was chosen because it is a free tool and in the public domain.

Test executions are done in a declarative way, using the

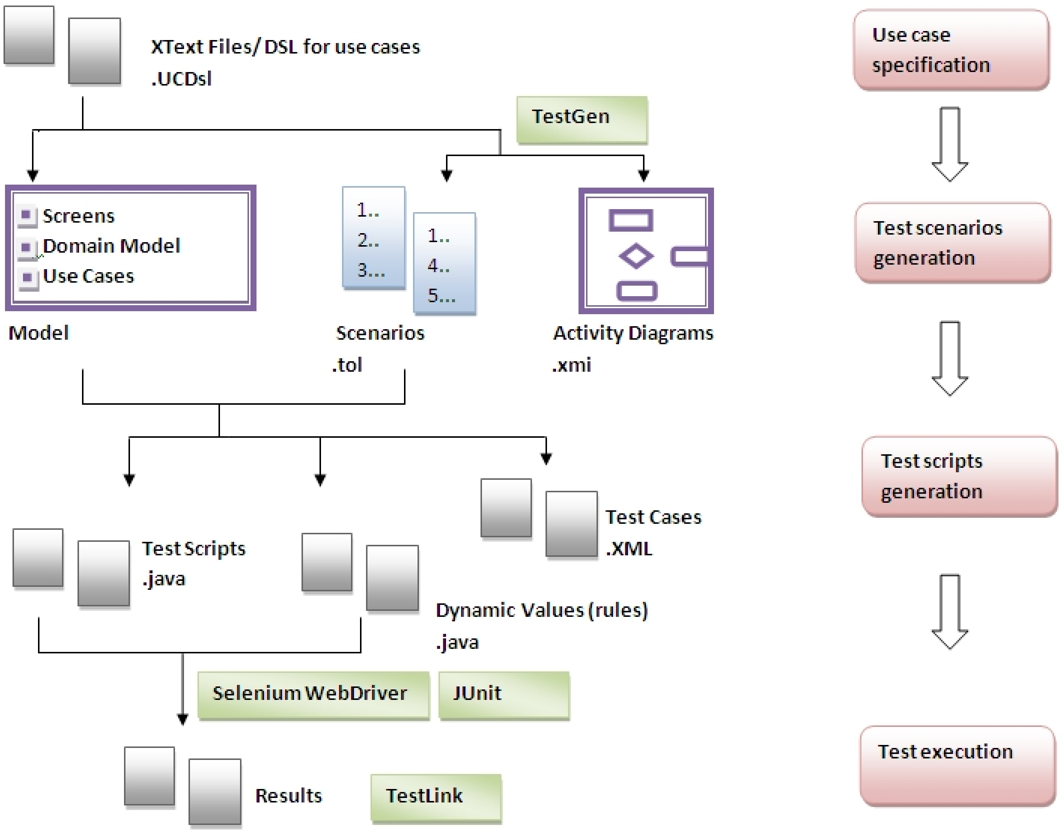


Fig. 2: Proposed environment structure.

UC name. It is also possible to execute all or a set of tests, formed by UCs that correspond to keywords informed in the execution. Keywords related to each UC are previously added to the specification, in addition to UC name. It is possible to test a UC's main scenario or every possible scenario. UC sequencing is allowed, by stating so in the pre-conditions or within steps of the scenario. UCs invoked from other UCs have the main scenario executed, unless extensions are indicated he should go through.

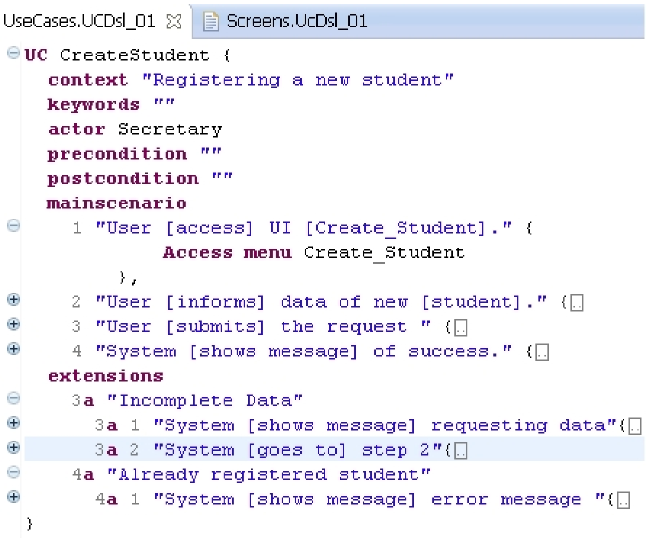
For each execution, dynamic values are generated according to the rule specified for each field.

Results are stored in the TestLink [23] environment, because it is a free and complete tool for test management, that allows users to centralize automated and manual tests, keep an execution history (with evidence logging), and obtain management reports. Each execution is recorded, along with its results. Errors results in the logging of the following evidence: a screenshot and a Java error stack trace, containing information about the UC, scenario and the step in which the error occurred, in addition to the runtime flaw (fig 8). These evidences, with the SUT's logs, help identifying what caused the errors.

Figure 2 shows the structure of the environment proposed and artifacts used in each phase.

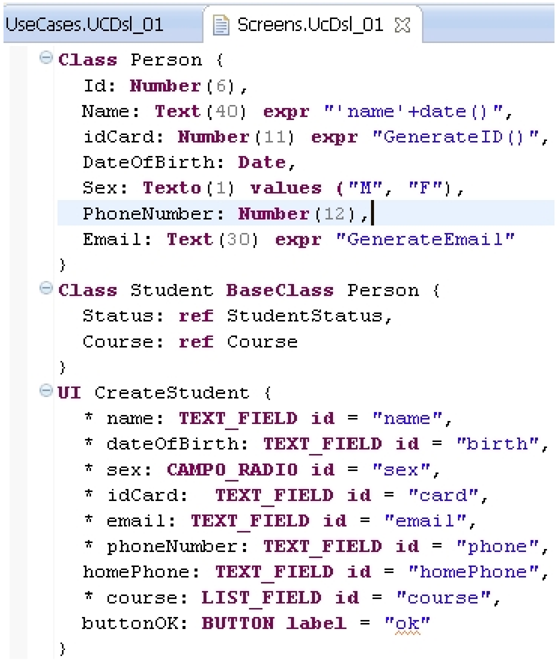
4. Experiments

In this section we present a viability study, for the UC "Create Student", presented in figure 3. Data dictionary and user interface specification are also presented (fig 4).



```
UC CreateStudent {
  context "Registering a new student"
  keywords ""
  actor Secretary
  precondition ""
  postcondition ""
  mainscenario
     1 "User [access] UI [Create_Student]." {
              Access menu Create_Student
          },
     2 "User [informs] data of new [student]." {..
     3 "User [submits] the request " {..
     4 "System [shows message] of success." {..
  extensions
     3a "Incomplete Data"
       3a 1 "System [shows message] requesting data"{..
       3a 2 "System [goes to] step 2"{..
     4a "Already registered student"
       4a 1 "System [shows message] error message "{..
}
```

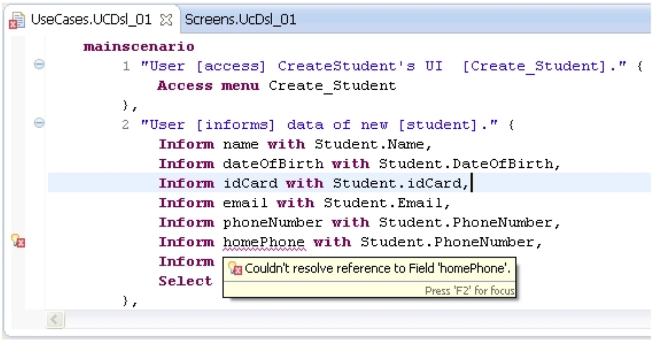
Fig. 3: Example of UC: Create Student.



```
Class Person {
  Id: Number(6),
  Name: Text(40) expr "'name'+date()",
  idCard: Number(11) expr "GenerateID()",
  DateOfBirth: Date,
  Sex: Texto(1) values ("M", "F"),
  PhoneNumber: Number(12),
  Email: Text(30) expr "GenerateEmail"
}
Class Student BaseClass Person {
  Status: ref StudentStatus,
  Course: ref Course
}
UI CreateStudent {
  * name: TEXT_FIELD id = "name",
  * dateOfBirth: TEXT_FIELD id = "birth",
  * sex: CAMPO_RADIO id = "sex",
  * idCard:  TEXT_FIELD id = "card",
  * email: TEXT_FIELD id = "email",
  * phoneNumber: TEXT_FIELD id = "phone",
  homePhone: TEXT_FIELD id = "homePhone",
  * course: LIST_FIELD id = "course",
  buttonOK: BUTTON label = "ok"
}
```

Fig. 4: Example of data dictionary and UI specifications.

Simulating a specification modification motivated by an alteration of the SUT, it was removed, from the IU description, the field "homePhone". An error in the UC that references the removed field is pointed out by the environment (fig 5).



```
mainscenario
    1 "User [access] CreateStudent's UI  [Create_Student]." {
      Access menu Create_Student
    },
    2 "User [informs] data of new [student]." {
      Inform name with Student.Name,
      Inform dateOfBirth with Student.DateOfBirth,
      Inform idCard with Student.idCard,
      Inform email with Student.Email,
      Inform phoneNumber with Student.PhoneNumber,
      Inform homePhone with Student.PhoneNumber,
      Inform
      Select
    },
```

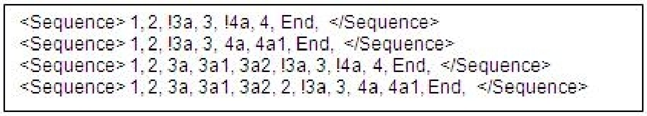
```
Couldn't resolve reference to Field 'homePhone'.
```

Fig. 5: Error pointed out by the environment.

Scenario generation for the UC CreateStudent resulted in 4 scenarios, as can be seen in figure 6. Test scripts were generated for the scenarios and tests were ran with positive results.

So, to simulate the discovery of an error in a regression test, we inserted a flaw in the SUT, in the routine that reads course data, causing a collateral effect in the routine that adds a student, which conducts some validations regarding courses to allow the addition of a student.

Figure 7 shows the execution results: the first one ended normally and the second one failed with an error. By clicking on the UseCaseName_StackTrace evidence, details about the error can be seen (fig 8). As can be noticed, class names that appear on the error stack bring useful information to aid in discovering the reasons behind the error, pointing to which step/scenario/UC the error occurred in. In this case, in step 4, scenario 1_2_3_4 of the CreateStudent UC.



```
<Sequence> 1, 2, !3a, 3, !4a, 4, End,  </Sequence>
<Sequence> 1, 2, !3a, 3, 4a, 4a1, End,  </Sequence>
<Sequence> 1, 2, 3a, 3a1, 3a2, !3a, 3, !4a, 4, End,  </Sequence>
<Sequence> 1, 2, 3a, 3a1, 3a2, 2, !3a, 3, 4a, 4a1, End,  </Sequence>
```

Fig. 6: Scenarios generated for the UC Create Student.

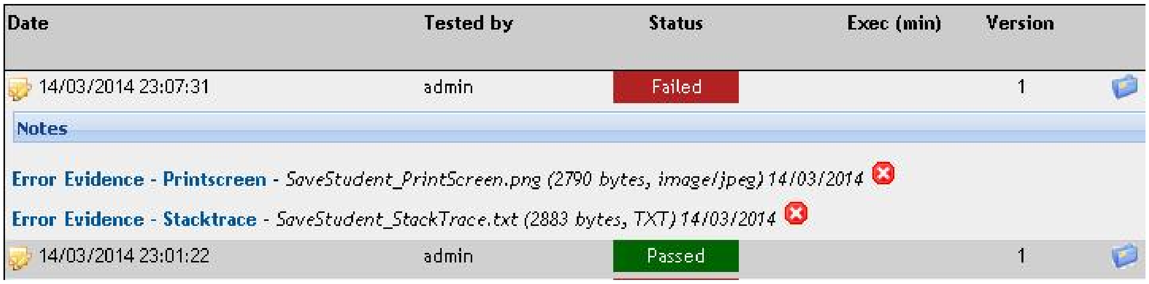
5. Conclusion and Future Work

It was concluded that using the proposed environment was feasible for a real system, according to case study. This study covered the CRUD functionality, and, in a second phase, will treat varied and more complex features. Between the two stages, it was decided to develop some mechanisms to facilitate the specification, as a new kind of step that informs all fields of a screen and submit it, a way to describe variations of a screen by means of extensions, and a mechanism to parameterize instances of similar use cases. Some limitations were found during the experiments, like the non-treatment of asynchronous steps and lack of permission for variations of more than one level, inside the alternate flux, situations that can be seen in UC22, in [19].

As future work, first and foremost, we intend to consider this second level of variation (which involves altering the TestGen tool) and the above mentioned enhancements. Additionally, we intend to: develop a mechanism to automatically capture the screen structure; allow the execution of tests for only UCs that suffered changes, between two versions; allow the execution of tests with "pairwise" technique, dealing with boundary-value analysis and equivalence class partitioning techniques (the element specification format already allows the generation of this variation); deal with screen modeling, like in [2], since the employed technologies are related; generate mechanisms to facilitate importing specifications created with other tools; generate the activity diagram in formats compatible with free UML tools, using XMI as base.

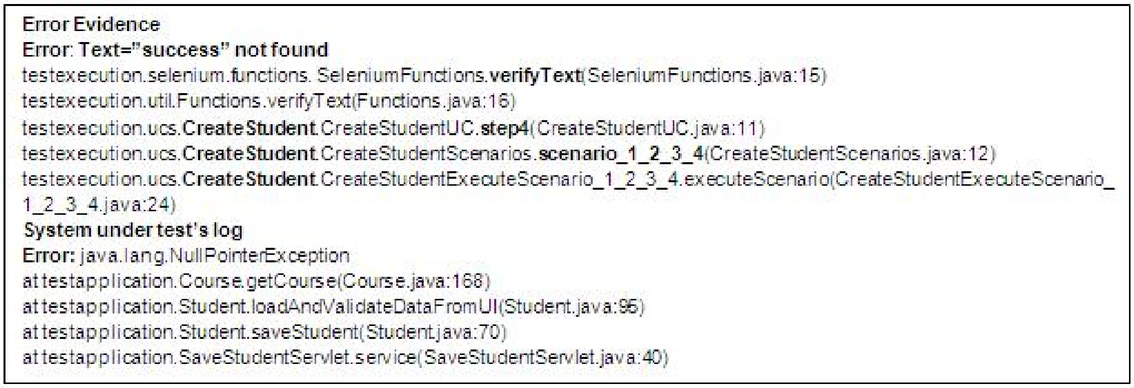
6. Acknowledgements

We would like to thank FAPEMIG for the financial support.



Date	Tested by	Status	Exec (min)	Version
14/03/2014 23:07:31	admin	Failed		1
Notes				
Error Evidence - Printscreen - *SaveStudent_PrintScreen.png (2790 bytes, image/jpeg) 14/03/2014*				
Error Evidence - Stacktrace - *SaveStudent_StackTrace.txt (2883 bytes, TXT) 14/03/2014*				
14/03/2014 23:01:22	admin	Passed		1

Fig. 7: Execution results.



```
Error Evidence
Error: Text="success" not found
testexecution.selenium.functions. SeleniumFunctions.verifyText(SeleniumFunctions.java:15)
testexecution.util.Functions.verifyText(Functions.java:16)
testexecution.ucs.CreateStudent.CreateStudentUC.step4(CreateStudentUC.java:11)
testexecution.ucs.CreateStudent.CreateStudentScenarios.scenario_1_2_3_4(CreateStudentScenarios.java:12)
testexecution.ucs.CreateStudent.CreateStudentExecuteScenario_1_2_3_4.executeScenario(CreateStudentExecuteScenario_
1_2_3_4.java:24)
System under test's log
Error: java.lang.NullPointerException
at testapplication.Course.getCourse(Course.java:168)
at testapplication.Student.loadAndValidateDataFromUI(Student.java:95)
at testapplication.Student.saveStudent(Student.java:70)
at testapplication.SaveStudentServlet.service(SaveStudentServlet.java:40)
```

Fig. 8: Error evidence in the execution of the CreateStudent UC.

References

[1] R. Pressman, *Software Engineering: A Practitioner's Approach*, 7th ed. Bookman, 2011.

[2] E. R. Luna, G. Rossi, and I. Garrigós, "Webspec: a visual language for specifying interaction and navigation requirements in web applications," *Requirements Engineering*, vol. 16, no. 4, pp. 297–321, 2011.

[3] J. Heumann, "Generating test cases from use cases," *The rational edge*, vol. 6, no. 01, 2001.

[4] M. J. Harrold, "Reduce, reuse, recycle, recover: Techniques for improved regression testing," in *Software Maintenance, 2009. ICSM 2009. IEEE International Conference on*. IEEE, 2009.

[5] P. Pedemonte, J. Mahmud, and T. Lau, "Towards automatic functional test execution," in *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*. ACM, 2012, pp. 227–236.

[6] L. Mariani, M. Pezzè, O. Riganelli, and M. Santoro, "Autoblacktest: a tool for automatic black-box testing," in *Proceedings of the 33rd International Conference on Software Engineering*. ACM, 2011, pp. 1013–1015.

[7] Selenium, "Selenium," http://www.seleniumhq.org, Mar. 2014.

[8] S. Thummalapenta, S. Sinha, N. Singhania, and S. Chandra, "Automating test automation," in *Software Engineering (ICSE), 2012 34th International Conference on*. IEEE, 2012, pp. 881–891.

[9] S. S. Some and X. Cheng, "An approach for supporting system-level test scenarios generation from textual use cases," in *Proceedings of the 2008 ACM symposium on Applied computing*. ACM, 2008, pp. 724–729.

[10] S. S. Somé, "Supporting use case based requirements engineering," *Information and Software Technology*, vol. 48, no. 1, pp. 43–58, 2006.

[11] C. Larman, *Applying UML and Patterns An Introduction to object - oriented analysis and design and iterative develop*, 3rd ed. Prentice Hall, 2005.

[12] D. Savic, I. Antovic, S. Vlajic, V. Stanojevic, and M. Milic, "Language for use case specification," in *Software Engineering Workshop (SEW), 2011 34th IEEE*. IEEE, 2011, pp. 19–26.

[13] D. Savic, A. R. da Silva, S. Vlajic, S. Lazarevic, V. Stanojevic, I. Antovic, and M. Milic, "Use case specification at different levels of abstraction," in *Quality of Information and Communications Technology (QUATIC), 2012 Eighth International Conference on the*. IEEE, 2012, pp. 187–192.

[14] Xtext, "Xtext," http://www.eclipse.org/Xtext, Mar. 2014.

[15] M. Smialek, "Accommodating informality with necessary precision in use case scenarios." *Journal of Object Technology*, vol. 4, no. 6, pp. 59–67, 2005.

[16] J. J. Gutierrez, M. J. Escalona, M. Mejias, J. Torres, and A. H. Centeno, "A case study for generating test cases from use cases," in *Research Challenges in Information Science, 2008. RCIS 2008. Second International Conference on*. IEEE, 2008, pp. 209–214.

[17] A. Sinha, S. Sutton, and A. Paradkar, "Text2test: Automated inspection of natural language use cases," in *Software Testing, Verification and Validation (ICST), 2010 Third International Conference on*. IEEE, 2010, pp. 155–164.

[18] C. Nebut, F. Fleurey, Y. Le Traon, and J.-M. Jezequel, "Automatic test generation: A use case driven approach," *Software Engineering, IEEE Transactions on*, vol. 32, no. 3, pp. 140–155, 2006.

[19] A. Cockburn, *Writing effective use cases*. Pearson Education, 2001.

[20] PageObject, "Page object," http://martinfowler.com/bliki/PageObject.html/, Mar. 2014.

[21] Eclipse, "Eclipse," http://www.eclipse.org/, Mar. 2014.

[22] Junit, "Junit," http://junit.org/, Mar. 2014.

[23] Testlink, "Testlink," http://testlink.org/, Mar. 2014.

Experimental Evaluation of Hybrid Algorithm in Spectrum based Fault Localization

A. Jonghee Park[1,2], **B. Jeongho Kim**[2], **and C. Eunseok Lee**[2]
[1]Samsung Electronics, Suwon, Gyeonggi-do, South Korea
[2]Sungkyunkwan University, Suwon, Gyeonggi-do, South Korea

Abstract— *During debugging process in software development cycle, fault localization is inevitable work. Diverse approaches have been proposed, such as program slicing, machine learning, and data mining for fault localization. In this paper we propose an effective hybrid fault localization algorithm based on a spectrum that enables fault detection in every statement. This algorithm distinguishes the location of a bug that causes a false positive score through the relationship between a test case and statement hit information. We also provide a fault localization tool named SKKU Fault localizer which enables source code instrumentation, test automation, test result comparison, extraction of distinct data, and fault ratio display. We applied it to the bug detection in Siemens test suite. Empirical results show that the hybrid algorithm not only decreases the amount of code to be reviewed by the programmer but also increases the effectiveness.*

Keywords: Program debugging, Spectrum based Fault Localization, Execution trace, Suspicious code, Fault localization

1. Introduction

Fault localization is time-consuming and costly, but, it is a very important task in the software development process. It is the hardest and most boring work for the programmer but it is essential work, needed to get rid of program bugs. To date, many studies have been carried out as people in the software industry have always been interested in fault localization. Firstly, program slicing [1] with a static and dynamic method was introduced. After that, various machine-learning methods were developed, such as artificial neuron network [2], SVM [3] and K-NN [4], data mining [5], and applied to the fault localization field. In addition, a major method called spectrum-based fault localization utilizes the relationship between the test result of a test case, and statement hit information. Tarantula [6], AMPLE [7], Jaccard [8], and Heuristic III [9] are representative algorithms in spectrum-based fault localization, which we will introduce in Section 2.

The main goal of this paper is to reduce the reading code coverage that the programmer should review, and effectively detect the exact location of any program bug. In addition, all processes are to be conducted with minimum human intervention. Then, we expect improved software product quality as well as reduced human workload in debugging activity. Therefore, we focused on spectrum-based fault localization which can provide suspiciousness that shows the probability of a program bug in every statement. In particular, we proposed a Hybrid fault localization technique which combines advantages from previous equations which were introduced by former researchers. In addition, we applied distinct data extraction technique to remove redundant duplicated data among test result.

The rest of this paper is as follows. In Section 2, we present some backgrounds and related work on spectrum-based fault localization. In Section 3, we propose methodology including the Hybrid technique and SKKU Fault localizer that we have developed for fault localization automation. In Section 4, we discuss the results of experimentation from two perspectives regarding reading code coverage and correctness of fault localization. Finally we concluded in Section 5, and presented future work.

2. Related work

Spectrum-based fault localization provides fault suspiciousness ratio by analyzing the relationship between a test result (pass or fail) and the visiting information of a statement. We assume that if failure test case happened, a fault exists among statements that were visited during a test in runtime. However, we cannot expect to determine the exact fault location only by the fail test case. Therefore, the pass test case was also utilized, to narrow down the fault statement.

Table 1. describes some notations that are commonly used in the fault localization field. h_i contains binary information as to whether the statement was visited or not. e_i contains binary information to describe the test result (pass or fail). If test case (T_i) which is one of the test cases in the test pool, was executed in runtime and the test result was fail, a certain statement (s_i) can be described as a_{11} (this line was visited) or a_{01} (not visited). In the same way, if the test result was pass, a certain statement (s_i) can be described as a_{10} (this line was visited) or a_{00} (not visited). Therefore, according to test result, every statement will be counted with four types of notation (a_{11}, a_{10}, a_{01}, a_{00}).

Representative equations have previously been introduced in spectrum-based fault localization methods, such as Tarantula, AMPLE, Jaccard, CBI [10], Ochiai [7] and Heuristic

Table 1: Relation between statement hit and test result

	Value		Description	
Notation	h_i	e_i	Statement hit	Test result
a_{11}	1	1	O	Fail
a_{10}	1	0	O	Pass
a_{01}	0	1	X	Fail
a_{00}	0	0	X	Pass

III. They calculate suspicious fault ratio in a different way, as below.

$$\text{Tarantula } S_j = \frac{\frac{a_{11}}{a_{11}+a_{01}}}{\frac{a_{11}}{a_{11}+a_{01}} + \frac{a_{10}}{a_{10}+a_{00}}} \quad (1)$$

J. A. Jones and M. J. Harrold (2005) developed Tarantula [6] which is aim to show suspiciousness in every statement. In addition, they conducted an experimental program which is based on C language.

$$\text{AMPLE } S_j = |\frac{a_{11}}{a_{01}+a_{11}} - \frac{a_{10}}{a_{00}+a_{10}}| \quad (2)$$

AMPLE [7] was developed to collect information about the hit spectra of method call sequences. Therefore, it is known to check faults in object-oriented language, such as Java and C++ language.

$$\text{Jaccard } S_j = \frac{a_{11}}{a_{11}+a_{01}+a_{10}} \quad (3)$$

Jaccard [8] developed from similarity coefficients in the mathematics field. It normally was used to find meaningful data among sets that consist of nominal elements. In particular, it was used in the Pinpoint framework.

$$\begin{aligned}\text{Heuristic III } S_j = {} & [(1.0)*n_{f1}+(0.1)*n_{f2}+(0.01)*n_{f3}] \\ & - [(1.0)*n_{s1}+(0.1)*n_{s2}+\alpha * X_{(F/S)} * n_{s3}] \quad (4)\end{aligned}$$

Wong et al. (2010) recently introduced Heuristic III [9]. They considered additional failed and passed test cases, and how they contribute to locating program fault. They divided failed and passed test cases into three groups (see more details in reference paper). In addition, they employed scaling factor α. Heuristic III (a), (b) and (c) were made according to the α value 0.01, 0.001 and 0.0001, respectively. Through various experiments, they presented Heuristic III (c) as the best equation.

3. Methodology

In spectrum-based fault localization field, there are approximately 30 types of existing equations [11][12]. We observed their behaviors and found out the characteristics of each algorithm. For the same software program, the suspicious fault ratio was different in every equation, according to their peculiarity. In this paper, we considered each equation's strength and finally proposed a Hybrid algorithm. To prove the proposed Hybrid equation, we developed the SKKU Fault localizer.

3.1 Hybrid algorithm

Hybrid algorithm is basis of two assumptions.

- *Assumption 1: Each algorithm has strength and weakness at the same time.*
  Therefore, once we get advantages of their algorithms, general outperformed Hybrid algorithm can be generated.
- *Assumption 2: To remove redundant duplicated test result makes better effectiveness of suspiciousness.*
  Test result data is huge and contains redundant data. Once we remove unnecessary data, it localizes exact bug location.

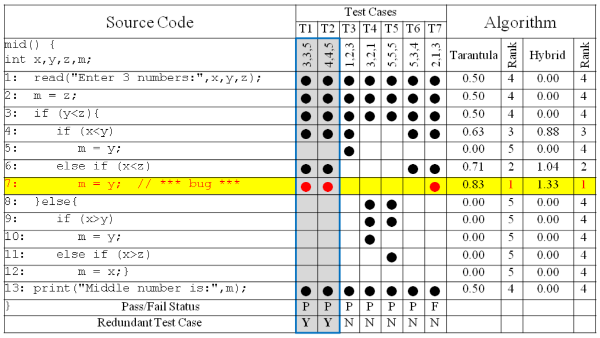
We figured out that Tarantula and Jaccard show zero suspiciousness when the numerator a_{11} was zero. This means that suspiciousness should be zero in statement (s_i), if there are no fail test cases. Otherwise, it shows suspiciousness over zero even if all test cases were hit. In addition, AMPLE described zero suspiciousness, if all test cases were visited in statement (s_i). We realized that this characteristics makes variation of fault localization effectiveness when we applied those algorithms to several test programs. Finally, we develop the Hybrid algorithm to combine the Tarantula, AMPLE, and Jaccard characteristics as follows.

$$\text{Hybrid } S_j = \begin{cases} \text{if } |\frac{a_{11}}{a_{01}+a_{11}} - \frac{a_{10}}{a_{00}+a_{10}}| = 0, 0 \\ \\ \text{else} \frac{\frac{a_{11}}{a_{11}+a_{01}}}{\frac{a_{11}}{a_{11}+a_{01}} + \frac{a_{10}}{a_{10}+a_{00}}} + \frac{a_{11}}{a_{11}+a_{01}+a_{10}} \end{cases} \quad (5)$$

We present the mid function in the example in Figure 1, which is well known for describing fault localization. There is a program bug in the 7th statement. Only three test cases hit that statement, in addition to one test case being failed, and the others being passed. This means that four types a_{11}, a_{10}, a_{01}, a_{00} mapped to 1, 2, 0, and 4. Based on the formula, Tarantula shows 0.84, and Hybrid indicates 1.33. In addition, Tarantula has six statements to be reviewed (over than zero suspicious fault ratio), and Hybrid has only three statements. It shows the Hybrid equation not only reduced the reading code coverage, but also detected the exact fault location in an early rank. Removing redundant data, such as T1, T2 (T7 is not redundancy cause test result was fail.) is important because redundant data affects suspiciousness. Only distinct data, such as T1 shall be used for fault localization.

3.2 SKKU Fault localizer tool

Software debugging is time-consuming work, where the programmer rectifies error; but it is essential work, in order to fulfill the required quality of software, and completeness of product. Therefore, those who work in Research and Development develop their own tools, or purchase commercial tools. However, those tools are not customized as much as they want, and even, require manual intervention from the user in many places, while they operate the tool. Therefore,



Source Code	Test Cases							Algorithm			
	T1	T2	T3	T4	T5	T6	T7				

mid() {
int x,y,z,m;
	3,3,5	4,4,5	1,2,3	3,2,1	5,5,5	5,3,4	2,1,3	Tarantula	Rank	Hybrid	Rank

1: read("Enter 3 numbers:",x,y,z);	●	●	●	●	●	●	●	0.50	4	0.00	4
2: m = z;	●	●	●	●	●	●	●	0.50	4	0.00	4
3: if (y<z){	●	●	●	●	●	●	●	0.50	4	0.00	4
4: if (x<y)	●	●	●			●	●	0.63	3	0.88	3
5: m = y;			●					0.00	5	0.00	4
6: else if (x<z)	●	●				●	●	0.71	2	1.04	2
7: m = y; // *** bug ***	●	●					●	0.83	1	1.33	1
8: }else{				●	●			0.00	5	0.00	4
9: if (x>y)				●	●			0.00	5	0.00	4
10: m = y;				●				0.00	5	0.00	4
11: else if (x>z)					●			0.00	5	0.00	4
12: m = x;}								0.00	5	0.00	4
13: print("Middle number is:",m);	●	●	●	●	●	●	●	0.50	4	0.00	4
} Pass/Fail Status	P	P	P	P	P	P	F				
Redundant Test Case	Y	Y	N	N	N	N	N				

Fig. 1: Source code, test cases and suspiciousness statement (Mid function)

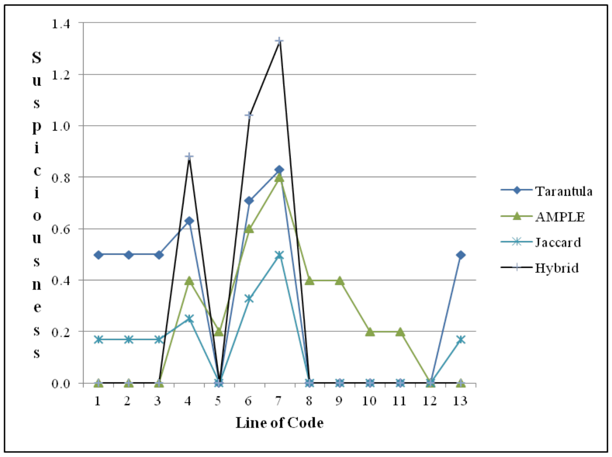
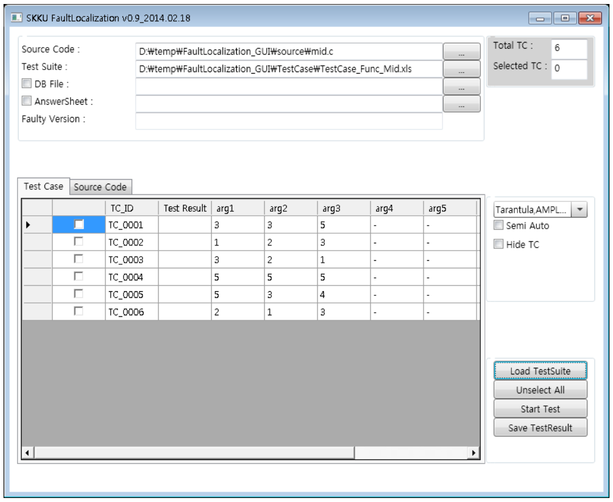


Fig. 2: Suspicious fault ratio of Talantula, AMPLE, Jaccard, and Hybrid

this paper introduces the SKKU Fault Localizer tool. If there is original source code, this tool can be applied to applications operated by Windows OS.

SKKU FL tool has many benefits, as follows. First, it saves time, because many procedures are automated, such as build source code, instrumentation, execution test, and visualization of suspicious fault ratio. Furthermore, it minimizes user invention, when the user utilizes the tool. Second, it reduces source codes that the developer needs to examine. Generally, when problem was happened, the developer debugs every statement that the program passed. However, SKKU FL tool executes many test cases, and automatically displays the most suspicious line. Then, the developer just checks the highly suspicious lines, which are shown with red background fill. Third, it supports a function to extract the test result as an Excel file. Then, it can be utilized in analyzing a test case and fault localization information data.

The SKKU FL tool GUI consists of an input space for source code, database file, answer sheet and test suite from the user, and displays space for test cases and test results of source code. It provides a function to extract Excel file that belongs to the display space (Test case, Source code, Result).



Source Code : D:₩temp₩FaultLocalization_GUI₩source₩mid.c
Test Suite : D:₩temp₩FaultLocalization_GUI₩TestCase₩TestCase_Func_Mid.xls
□ DB File :
□ AnswerSheet :
Faulty Version :

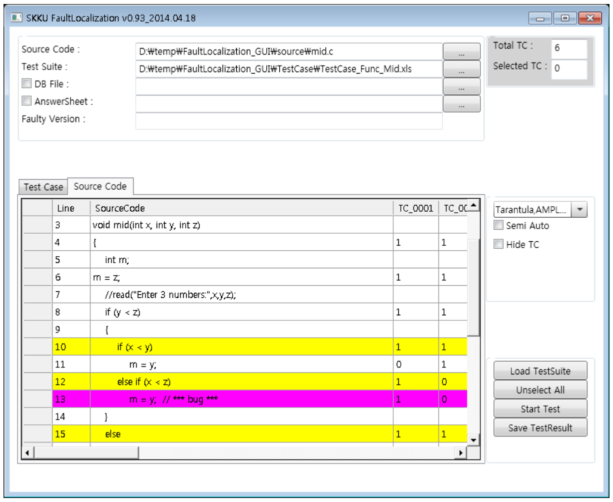
Total TC : 6
Selected TC : 0

□ Semi Auto
□ Hide TC

	TC_ID	Test Result	arg1	arg2	arg3	arg4	arg5
□	TC_0001		3	3	5	-	-
□	TC_0002		1	2	3	-	-
□	TC_0003		3	2	1	-	-
□	TC_0004		5	5	5	-	-
□	TC_0005		5	3	4	-	-
□	TC_0006		2	1	3	-	-

Fig. 3: Input screen : test case

- Input space: location of source code, database, answer sheet, test case, algorithm of suspicious fault ratio, start and load button
- Display space: view of test case, source code added suspicious fault ratio and rank



Source Code : D:₩temp₩FaultLocalization_GUI₩source₩mid.c
Test Suite : D:₩temp₩FaultLocalization_GUI₩TestCase₩TestCase_Func_Mid.xls
□ DB File :
□ AnswerSheet :
Faulty Version :

Total TC : 6
Selected TC : 0

□ Semi Auto
□ Hide TC

Line	SourceCode	TC_0001	TC_0C
3	void mid(int x, int y, int z)		
4	{	1	1
5	int m;		
6	m = z;	1	1
7	//read("Enter 3 numbers:",x,y,z);		
8	if (y < z)	1	1
9	{		
10	if (x < y)	1	1
11	m = y;	0	1
12	else if (x < z)	1	0
13	m = y; // *** bug ***	1	0
14	}		
15	else	1	1

Fig. 4: Display screen : source code and suspicious fault ratio

In Figure 4, it displays line number, statements and passed information as well as colored suspicious fault ratio. Yellow means it is potential risky code, and red means it is highly risky code. As a result, it isolates the fault location, even guides the path where statements are passed, and reducing code lines unrelated to the fault. The functionality of SKKU FL tool has been continuously updated, in order to apply

Table 2: Test case format

TC_ID	Test result	arg1	arg2	arg3	Expected value	Actual value
TC_0001	Pass	3	3	5	3	3
TC_0002	Pass	1	2	3	2	2
TC_0003	Pass	3	2	1	2	2
TC_0004	Pass	5	5	5	5	5
TC_0005	Pass	5	3	4	4	4
TC_0006	Fail	2	1	3	2	1

various test programs, such as the Siemens test program, as well as to utilize database.

3.3 Test case design

The test case is managed in an Excel sheet, and it defines the TC_ID with a unique identifier, test result, arguments value, expected value, and actual value returned from the program. The test result and actual value are automatically inserted by SKKU FL tool, because it can decide pass and fail, by comparing the expected and actual value. If semi-auto is checked, the test result will be inserted by user decision.

3.4 Test process

The SKKU FL is operated in the following seven steps.

- Step 1: The user should select the test cases and source code to be examined. (This can be replaced with a database, if there is previous test case pool, such as test results and statement hits.)
- Step 2: Select an algorithm, such as Tarantula, AMPLE, Jaccard, Heuristic III (c), and Hybrid. If it is hard to reach a pass or fail verdict in the test result in the target application, check the Semi-Auto checkbox. Then it will ask for user input for the verdict after the test case is finished, in order to decide pass or fail.
- Step 3: Load the test case to the display screen.
- Step 4: Select as many test cases as you want to test, and click the Start button.
- Step 5: The tool instruments the original source code, and makes a binary file, through building an instrumented source code. After that, it passes parameters, in launching the binary file. Next, it monitors the test program running and recording line information, which is passed, until the program is terminated.
- Step 6: Compare the expected value with the actual value that the program returned. SKKU FL tool automatically decides pass or fail, if it does not require a user decision. Otherwise, it asks for user input, as to whether it is Pass or Fail.
- Step 7: SKKU FL tool computes suspicious fault ratio by using the line of statement passed and test result of pass or fail and finally displays suspicious fault ratio.

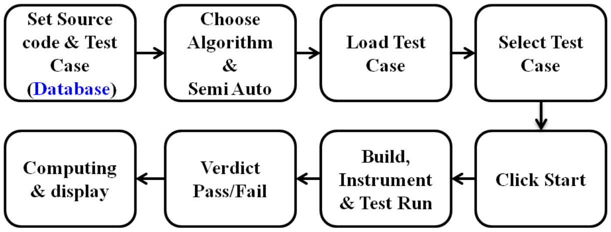

Set Source code & Test Case (Database)
Choose Algorithm & Semi Auto
Load Test Case
Select Test Case
Click Start
Build, Instrument & Test Run
Verdict Pass/Fail
Computing & display

Fig. 5: Process of the SKKU Fault localizer tool

4. Experimental results

4.1 Test environment

To set up the test environment, two major procedures are required. The first procedure is that test case designer creates a test case file, as introduced in Section 3. When making the test case file, we used the Siemens input file that they provided as a text file. After that, all test cases are run using the original source code, in order to collect the correct test result, which will be used to obtain a pass or fail verdict test result.

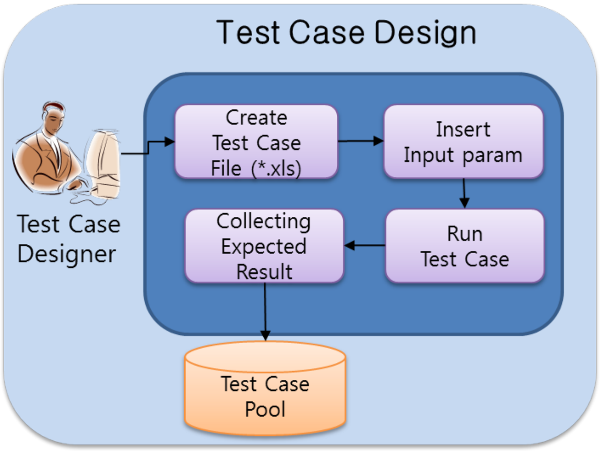


Fig. 6: Test case design architecture

The second procedure is that test executor runs SKKU Fault localizer, and collects the test result. We choose a faulty version source, and load the test case from the test case pool. After that, all test cases are run, and suspiciousness is obtained, using various algorithms.

4.2 Siemens test suite

We used 123 faulty versions in Siemens test suite seven test programs [13][14] among 132 faulty versions, which contain a single bug, seeded from the original non-faulty version. Test programs were designed to apply realistic software program commonly used in industry. We used only the available test programs. Therefore, we excluded 9 faulty versions: version 4 and 6 of printtokens because there is no

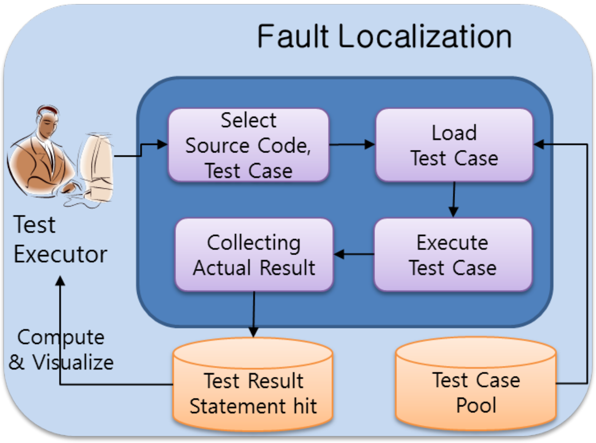


Fig. 7: Fault localization architecture

Table 3: Siemens test program

Program	Ver.	LOC	Test Cases	Description
printtokens	7	565	4140	lexical analyzer
printtokens2	10	510	4071	lexical analyzer
replace	32	563	5542	pattern recognition
schedule	9	412	2650	priority scheduler
schedule2	10	307	2680	priority scheduler
tcas	41	173	1578	altitude separation
totinfo	23	406	1054	information measure

change between the original and faulty version and version 6, 10, 19, and 21 of totinfo and version 38 of tcas and version 12 of replace because there is only change in declaration variable, such as define statement. We also exclude version 9 of schedule2 because there is no fail test case. Each faulty version was aimed at various human errors that they usually make in their work [15], such as missing code, wrong position of switch-case, or wrong boundary value.

4.3 Results and analysis

We have performed an experiment using the Siemens test suite. There are two aspects, when comparing the effectiveness regarding fault localization among equations. One is EXAM score which was proposed by Wong et al. (2010). EXAM score consists of Best and Worst. Best means that during review, the fault statement may be found first, between the same suspiciousness. Worst means that the fault may be found last. The other aspect is the RCC (Reading code coverage) that we proposed. This means that a statement that has a suspiciousness of over zero should be reviewed by the programmer. Of course, in both measurements, the score is better when it closes to zero.

A comparison of effectiveness between the previous studies and Hybrid is shown in Figure 8~10. We choose Tarantula and Heuristic III (c), to avoid describing many

Table 4: Measurement method

Name	Formula	Description (lower is the best)
EXAM (Best)	B_r/Total line	B_r : Rank of Faulty line
EXAM (Worst)	W_r/Total line	W_r : Rank of faulty line + number of same rank lines - 1
RCC (Reading code coverage)	Z_+/Total line	Z_+ : number of lines over than zero suspicious ratio

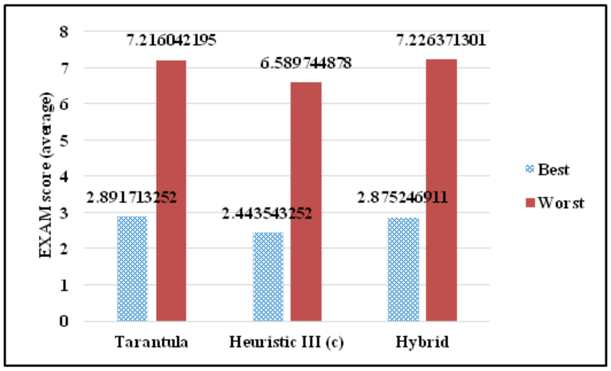


Fig. 8: Original EXAM score (average)

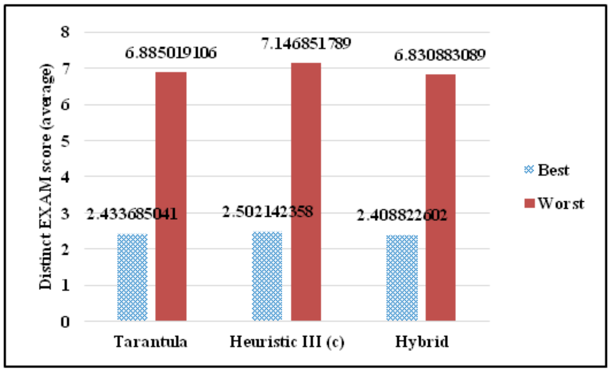


Fig. 9: Distinct EXAM score (average)

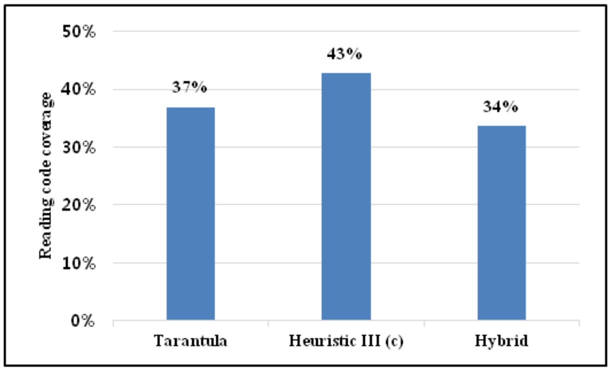
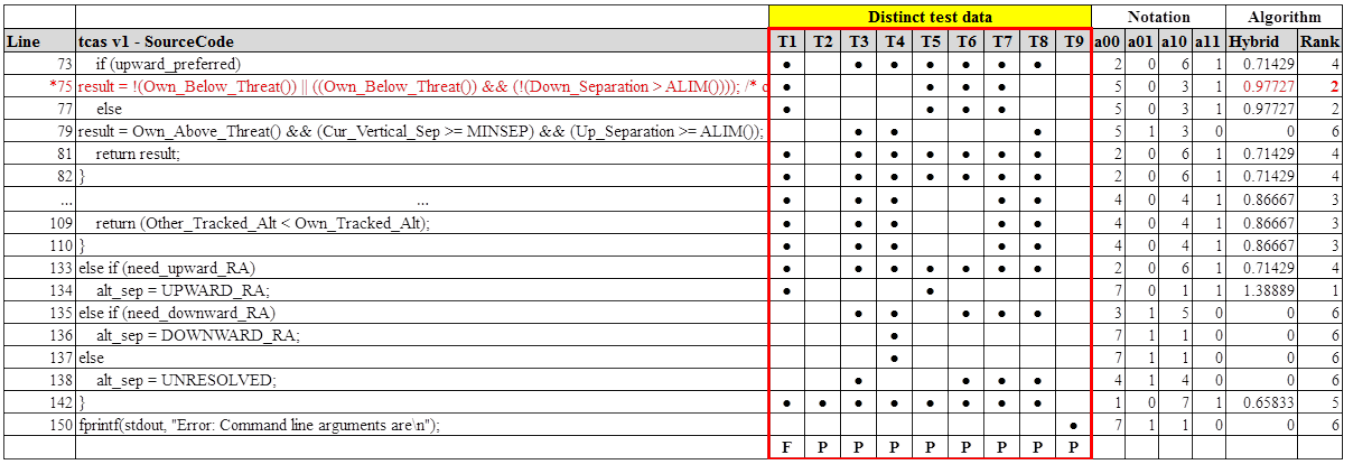


Fig. 10: RCC (Reading code coverage on average)

Copyright © 2014 CSREA Press, ISBN: 1-60132-286-0; Printed in the United States of America



		Distinct test data									Notation				Algorithm	
Line	tcas v1 - SourceCode	T1	T2	T3	T4	T5	T6	T7	T8	T9	a00	a01	a10	a11	Hybrid	Rank
73	if (upward_preferred)	●		●	●	●	●	●	●		2	0	6	1	0.71429	4
*75	result = !(Own_Below_Threat()) || ((Own_Below_Threat()) && (!(Down_Separation > ALIM()))); /* c	●				●	●	●			5	0	3	1	0.97727	2
77	else	●				●	●	●			5	0	3	1	0.97727	2
79	result = Own_Above_Threat() && (Cur_Vertical_Sep >= MINSEP) && (Up_Separation >= ALIM());			●	●				●		5	1	3	0	0	6
81	return result;	●		●	●	●	●	●	●		2	0	6	1	0.71429	4
82	}	●		●	●	●	●	●	●		2	0	6	1	0.71429	4
...	...	●		●	●			●	●		4	0	4	1	0.86667	3
109	return (Other_Tracked_Alt < Own_Tracked_Alt);	●		●	●			●	●		4	0	4	1	0.86667	3
110	}	●		●	●			●	●		4	0	4	1	0.86667	3
133	else if (need_upward_RA)	●		●	●	●	●	●	●		2	0	6	1	0.71429	4
134	alt_sep = UPWARD_RA;	●				●					7	0	1	1	1.38889	1
135	else if (need_downward_RA)			●	●		●	●	●		3	1	5	0	0	6
136	alt_sep = DOWNWARD_RA;				●						7	1	1	0	0	6
137	else				●						7	1	1	0	0	6
138	alt_sep = UNRESOLVED;			●			●	●	●		4	1	4	0	0	6
142	}	●	●	●	●	●	●	●	●		1	0	7	1	0.65833	5
150	fprintf(stdout, "Error: Command line arguments are\n");									●	7	1	1	0	0	6
		F	P	P	P	P	P	P	P	P						

Fig. 11: Examaple of how Hybrid makes a distinct data (faulty version 1 of tcas)

previous algorithms. From Jones et al. (2005), the Tarantula method is more effective than set-union, set intersection and nearest-neighbor. Wong et al. (2010) presented Heuristic III (c) as being more effective than Tarantula in EXAM (Best) and EXAM (Worst). Through Figure 8: Original EXAM score, Heuristic III (c) outperforms Tarantula and Hybrid on average in both EXAM (Best) and EXAM (Worst). Hybrid only outperforms Tarantula in EXAM (Best). However, in the case of Figure 9: Distinct EXAM score, Hybrid is more effective than Tarantula and Heuristic III (c) in both EXAM (Best) and EXAM (Worst). Therefore, the Hybrid technique is more effective overall in EXAM score.

With respect to RCC, Hybrid always outperforms Tarantula and Heuristic III (c). In Figure 10, approximately 34% of the code has suspicious fault ratio. This means that when bug was found during testing the programmer should just review a third of the original source code. Hybrid decreased the reading code coverage to be checked.

4.4 Additional consideration

We found that there are similar test cases that contain the same test result, among thousands of test cases. To increase the effectiveness, we thought that distinct data (not a same statement hit information and test result) makes a better EXAM score, than the original experiment. In particular, faulty version 1 of tcas, there are 1578 test cases when calculating suspiciousness. Many redundant duplicated test data make lower suspiciousness. In Figure 11 shows that after applying distinct technique, test cases were reduced to only 9. Therefore, before calculating suspiciousness, except for unique one, the other test cases that have same statement information and result should be removed.

After removing duplicated data in the original data, Figure 12 indicates that most EXAM scores were decreased in every algorithm. In particular, Hybrid EXAM (Best) was improved by 17 percent over the original EXAM score.

Table 5: Comparison of EXAM score (average)

	Tarantula		Heuristic III (c)		Hybrid	
EXAM	Original	Distinct	Original	Distinct	Original	Distinct
Best	2.892	2.434	2.444	2.502	2.875	2.409
Worst	7.216	6.885	6.590	7.147	7.226	6.831

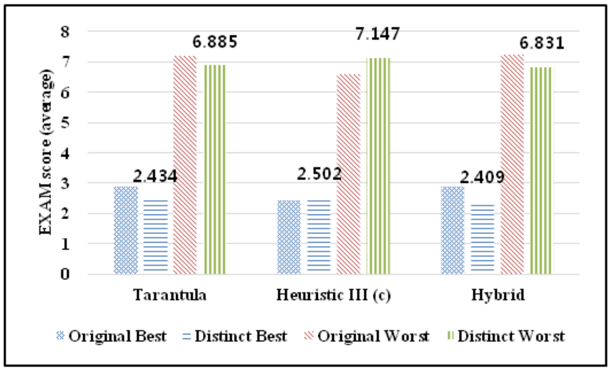


Fig. 12: Comparison of EXAM score between original and distinct data

5. Conclusion and future works

We proposed Hybrid algorithm and developed SKKU Fault localizer tool, in order to not only localize the exact fault location, but also to reduce the reading code coverage. Hybrid was proposed to overcome each characteristic of equation characteristic because there was no superior algorithm in every test program. Hybrid shows that it was more effective generally among existing algorithms. We presented distinct test data to clarify unique test data and get rid of redundant test data which degrades performance of algorithm performance. Furthermore, SKKU Fault localizer can help people to automatically perform all test procedures. In addition, it is easy to install and simple to operate the

GUI, because it is based on Windows OS.

For future work, we would like to extend the Hybrid algorithm to all of the Unix test suite, as well as a large-scale real world program. At the same time, we will analyze the program characteristics, in order to improve Hybrid algorithm according to the program. In addition, we figured out that distinct test data makes the algorithm better. However, it is still time-consuming and burdensome work for the software quality assurance department in a company. People who are in charge of software quality would like to do regression test, to avoid reproducing the same problem. They really want to reduce the amount of test cases that have the same capability as the original test cases. Therefore, we will do research regarding the reduction of test case.

6. Acknowledgment

This research was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MEST)(No. 2012008240)

References

[1] Agrawal, Hiraral, et al. “Fault localization using execution slices and dataflow tests.” *Proceedings of IEEE Software Reliability Engineering* (1995): 143-151.

[2] Wong, W. Eric, and Yu Qi. “BP neural network-based effective fault localization.” *International Journal of Software Engineering and Knowledge Engineering* 19.04 (2009): 573-597.

[3] Ascari, L. C., et al. “Exploring machine learning techniques for fault localization.” *Test Workshop, 2009. LATW’09. 10th Latin American.* IEEE, 2009.

[4] Renieres, Manos, and Steven P. Reiss. “Fault localization with nearest neighbor queries.” *Automated Software Engineering, 2003. Proceedings. 18th IEEE International Conference on.* IEEE, 2003.

[5] Nessa, Syeda, et al. “Software fault localization using N-gram analysis.” *Wireless Algorithms, Systems, and Applications.* Springer Berlin Heidelberg, 2008. 548-559.

[6] Jones, James A., and Mary Jean Harrold. “Empirical evaluation of the tarantula automatic fault-localization technique.” *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering.* ACM, 2005.

[7] Abreu, Rui, Peter Zoeteweij, and Arjan JC Van Gemund. “An evaluation of similarity coefficients for software fault localization.” *Dependable Computing, 2006. PRDC’06. 12th Pacific Rim International Symposium on.* IEEE, 2006.

[8] Chen, Mike Y., et al. “Pinpoint: Problem determination in large, dynamic internet services.” *Dependable Systems and Networks, 2002. DSN 2002. Proceedings. International Conference on.* IEEE, 2002.

[9] Eric Wong, W., Vidroha Debroy, and Byoungju Choi. “A family of code coverage-based heuristics for effective fault localization.” *Journal of Systems and Software* 83.2 (2010): 188-208.

[10] Liblit, Ben, et al. “Scalable statistical bug isolation.” *ACM SIGPLAN Notices.* Vol. 40. No. 6. ACM, 2005.

[11] Wong, W. Eric, and Vidroha Debroy. “A survey of software fault localization.” *Department of Computer Science, University of Texas at Dallas, Tech. Rep. UTDCS-45-09* (2009).

[12] Naish, Lee, Hua Jie Lee, and Kotagiri Ramamohanarao. “A model for spectra-based software diagnosis.” *ACM Transactions on software engineering and methodology (TOSEM)* 20.3 (2011): 11.

[13] Do, Hyunsook, Sebastian Elbaum, and Gregg Rothermel. “Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact.” *Empirical Software Engineering* 10.4 (2005): 405-435.

[14] The Siemens test suite. SIR (Subject Infrastructure Repository) website. [Online]. Available: http://sir.unl.edu/portal/index.php

[15] Hutchins, Monica, et al. “Experiments of the effectiveness of dataflow- and controlflow-based test adequacy criteria.” *Proceedings of the 16th international conference on Software engineering.* IEEE Computer Society Press, 1994.

Prioritization of Artifacts for Unit Testing Using Genetic Algorithm Multi-objective Non Pareto

Eduardo Noronha de Andrade Freitas[1], **Auri Marcelo Rizzo Vincenzi**[2], **and Celso Gonçalves Camilo-Júnior**[2]
[1]Department of Informatic, Instituto Federal de Goias, Goiânia, Goiás, Brazil
[2]Institute of Informatic, Universidade Federal de Goiás, Goiânia, Goiás, Brazil

Abstract— *The choice of units for application of write unit testing in level of methods can be seen as a combinatorial problem. The time demanded to write unit tests becomes a challenge for the testing professional considering the constraints of time existent. Adopting the time to write unit testing as a cost, the goal this work is to present a multi-objective evolutionary approach that look for a subset of artifacts such that minimizes the cost at the same time that maximizes their strategic importance. Both metrics, static and dynamic such as cyclomatic complexity, operational coverage, and frequency of modification were used to define the strategic importance. The motivation to use multi-objective evolutionary approach is that growth of fault proneness is inversely proportional to the cost. The experiments were done in real context of industry system, and the results found confirmed the benefits of proposal.*

Keywords: Software testing, unit testing, effort test, prioritization unit test, Search Based Software Test (SBST), multiobjective evolutionary optmization.

1. Introduction

Among the activities of the Software Engineering, the verification and validation are the more expensive, representing 50% to 80% of the total cost of a project, and software testing is the most common practice for software verification and validation.

The development of unit testing in this context is an especial challenge for the professionals who find themselves in difficulty in deciding which artifacts must be the firsts, once the resources available to conclude this activity are limited. In this sense, the development and the application of unit test on the whole system with high level of coverage may be impractical, if considered the resources constraint. The identification of artifacts really relevant of system become essential and strategic. It is especially in the case of legacy systems, large systems, and systems with high level of maintenance.

In this work, the problem of prioritization of effort of unit tests will be discussed with techniques of Search Based Software Engineering (SBSE) [7]. In SBSE the problems are modeled as optimization problem, and after that they are solve utilizing concepts, techniques, algorithms, and strategies of search. The objective is to identify among all possible solutions a set of solutions, which will be sufficiently good according to a set of appropriate metrics.

The validation of the proposed approach was carried out some software companies seeking a process of real experimentation in context of high criticality. Some decision variables were considered to the experimentation of the algorithms such as: cyclomatic complexity, frequency of changes found on repository, and operational coverage. The constraint of the problem will be considered in terms of hours, considering the availability of time to the development of unit tests to homologation of a new release of the system.

The rest of the paper is organized as follows: In the Section 2 is presented the concept of software testing and evolutionary optimization utilized to the development this work. In the Section 3 is described the characterization of the problem of selecting code units and its complexity. In the Section 4 is presented the details of the approach and of the experiments realized. Finally, in the Section 5 is presented the conclusion of the work developed.

2. Fundaments

The goal this Section is to present part of the concepts utilized in the development of the work.

2.1 Software Testing

According Delamaro et al. [5], the software testing consists in an activity of quality assurance in order to verify if the product in development is in conformity with your specification. The [8] defines software testing as a dynamic verification of the program's behavior, utilizing a set of finite test, correctly selected of the executions' domain, to verify if it's according with the expected.

Many techniques and test's criteria had been developed to permit the selection of a subset of the entrance domain to be effective in reveal the presence of the existent defects. This occurs because mostly of time the exhaustive test is impracticable due to constraints of time and cost.

According to Pressman [9], the testing activity can be considered as a incremental activity realized in three phases: unit testing, integration testing, and high level testing.

Unit tests, also called of unit testing has its focus in the minor code unit in order to ensure that the implementation's aspects are correct. In this phase, seeks to ensure greater coverage and maximum defect's detection in each module.

The scope this work is focused on building an alternative to find among the artifacts at the unit level, which should be selected to the development of structural tests in the method level. The prioritization must be carried out considering the possibility of combining two or more code's unit as priority for a specific moment. So, this work doesn't focus the prioritization of test cases.

2.2 Evolutionary Optimization

In mathematic or in computation, the term optimization is used as reference to study of problems that seek maximize or minimize a function by choosing values to the variables of the problem.

However, some problems own combinatorial characteristic and can be computationally intractable due the your dimension of exponential order. In that case, the classical optimization models become limited to present an optimal solution in feasible time.

The concept of evolutionary computation has been used by means of so-called evolutionary algorithms. They apply the process of natural evolution defended by Charles Darwin, as paradigm to implementation of the algorithms. The evolutionary heuristics give up the warranty of global optimal for ensuring a set of approximated solutions.

Some problems are considered as multiobjective. A multi-objective problem consists in finding a set of variables that satisfy some constraints, while optimizing two or more objectives. The solution of a multi-objective problem is composed by a set of solutions that represent a commitment among the objectives, according to Azuma [1].

According to Coello [2], a multi-objective optimization problem can be formulated as:

$$\min[f_1(x), f_2(x), ..., f_k(x)]] \quad (1)$$

subject to *m* inequality constraints:

$$g_i(x) \leq 0 i = 1, 2, ..., m \quad (2)$$

and to *p* equality constraints:

$$h_i(x) = 0 i = 1, 2, ..., m \quad (3)$$

where *k* is the number of functions, e $x = [x_1, x_1, ..., x_n]^T$ the array of decisons' variables. The problem becomes to be to determine values among the set *F* of all the array's that satisfy the Equations 2 e 3, the particular set of values $x_1^*, x_2^*, ..., x_n^*$ that provide optimal values of all functions. So, in context of prioritization of code unit, a solution is considered as a set of code units.

One important concept in multi-objective optimization is associated the dominance of the solutions. A solution *x* dominates another solution *y*, if both condition follows happen in a optimization problem: if the solution *x* is better or equals to *y* in all objective functions, and The solution *x* is é strictly better than *y* in at least one objective function.

There are two different spaces in multi-objective optimization. The first one is associated with the variable of the problem, while the second is associated with the objectives.

Each point enumerated in space of the variables correspond to code units of a arbitrary system. The mapping these points in solutions in objective space can be comprehended by a strong analogy to classic Knapsack Problem. The decision consists in choice the better combination of methods to be exercised in unit testing. To do it, the function of mapping will go utilizing some metrics that they contain strong correlation interdependent with strategic importance, according to will be detailed in Section 4.

In Figure 1 are presented some of these solutions. Each solution present in the objective space may have 3 distinct classifications: the first when in which the solution dominates, is dominated, and when it is irrelevant. In Figure 1 a small example of mapping the space of objectives, containing 2 goals is presented. On the first objective we seek to find a set of code units with high values of strategic importance associated. In the second objective the aim is to find a set of code units that require the least amount of time to develop unit tests. These requirements confer to the problem a multi-objective characterization, once mostly cases the higher the strategic importance of a unit takes longer time for the development of unit testing.

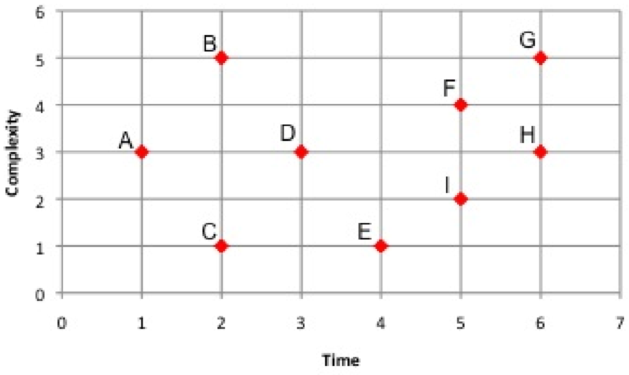


Fig. 1: Analysis of the space of objectives, considering time and cyclomatic complexity.

The solution *A* shown in Figure 1 represents a set of code units that after the mapping to the space solution presents the value 3 to complexity, requiring 1 unit of time (for example, hours) to the development of unit testing.

Some methods are used to measure the objective functions such as the weighted sum of the objectives, and also, the method ε-restrict. In this work was adopted a multi-objective evolutionary approach with a genetic algorithm that uses weighted sum.s

3. The Problem

The objective this Section is to present the characterization of the problem of prioritization of software artifacts in level of methods or function, and the challenge inherent to it.

In the context of software testing, specifically unit testing, we seek to find a subset of code units (methods) that together maximize the strategic importance in software. Another objective consists in minimizing the time used to developing test cases to these code units. The existence these two conflicting objectives justifies the necessity of the use of SBST to solve this problem, especially with multi-objective approach.

To calculate all distinct ways to choice elements without repetition is necessary to realize the sum of distinct combinations of r elements, according to Equation 4.

$$\sum_{r=1}^{n} C(n,r) = \frac{n!}{r!.(n-r)!} \qquad (4)$$

To illustrate the exponential behavior of the problem, the Figure 2 shows the curve containing the number of possible combinations to systems with different quantity of code units.

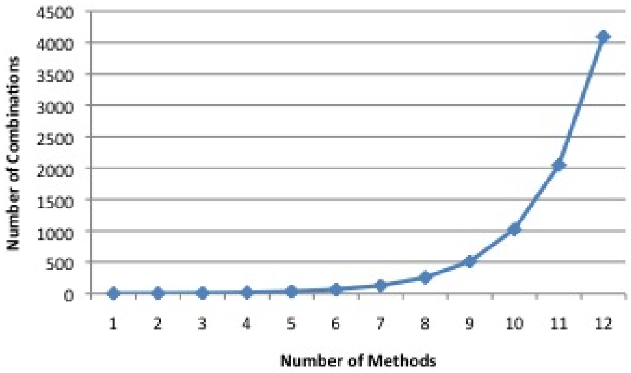


Fig. 2: Exponential growth possibilities for selection of units for unit testing code.

In Section 4 will be presented the main characteristics of a multi-objective optimization problem, and how to prioritize units of code to the development of unit testing can be solved by this approach.

4. Proposed Approach

The activity of software testing has reveled challenging problems. Identifying strategic interesting areas to reduce the effort in testing is one of them. In Elberzhager et al. [6] is presented a systematic mapping of existing approaches able of reducing the effort in testing, one being the prediction of areas with most prone to defects. The essence that approach is the consideration of metrics. The key metrics adopted in that studies reviewed were: product, process, object oriented, and defects.

Also, others approaches to minimize the effort in testing were found. In Shihab [11] is presented a comparative analysis of some heuristics in order to prioritize software artifacts for unit testing in legacy systems. According to the study, the application of unit tests on artifacts that have undergone corrective changes more frequently, have greater potential to reveal defects. Other approach is found in Ray and Mohapatra [10] in which a method is proposed to prioritize testing effort in order to guide the tester during the development life cycle of software. Amount of influence, average execution time, structural complexity, and severity and value, were considered relevant factors. In the work of Czerwonka et al. [4] a system for failure prediction, risk analysis and prioritization of tests is presented to supporting the aspects of maintaining legacy systems from Microsoft, particularly Windows Vista. However, it is also applied prioritization of test cases.

The main shortcoming with these approaches is that it isn't seen as a combinatorial problem. The works that focus on identifying priority artifacts to application of unit testing, using simple heuristics, whose result is a list of artifacts in descending order of importance. A trend in predicting defects is explores the metrics and their combinations. However, a small number of papers have focused the combination of static and dynamics metrics.

The objective this section is to characterize the proposal of prioritization of code based in multi-objetive evolutionary approach. The research made use of experiments to validate the proposal. This section shows the context and strategy for prioritization followed by a detailed of the experiments utilizing Genetic Algorithm Multiobjective Non Pareto using aggregating functions of same weights [3], combining all objectives into a single one.

The proposed approach to selecting units of code for application of unit testing is supported in Search Based Software Engineering (SBSE) more specifically Search Based Software Testing (SBST). Considering a system with several modules, its dependencies, and a great quantity of methods, the test's planning for a new release must be optimized, considering limited resources for the development of the testing activies.

The amount of metrics available to be used to establishing a strategy to optimize the selection of units of code is very large. For this work was defined as the first objective function the maximization of strategic importance of the units selected. The main goal is to try to find out a selection of units (methods) with higher strategic importance. Basically, three main components are used to address it: cyclomatic complexity, operational coverage, and frequency of modification.

The mensuration of the strategic importance (si) of each method is calculated based on the product and process metrics. Some works influenced the choice of the metrics utilized by algorithm, as Ray and Mohapatra [10] which have showed some important metrics such as time of execution, influence's value of a component, structural complexity, responsibility per class and LCOM4, type and severity of failures. In Shihab et al. [11], the authors present a comparison of some heuristics to prioritize artifacts to application of unit testing in legacy systems. The application of unit testing in artifacts that receive corrective changes most frequently own more potential to reveal failures. Thus, we aim at to figure out which characteristics exist among units more recently corrected. This analyze amongst the evidences shown in [6] concluded that they own high values of cyclomatic complexity. Based in this, the cyclomatic complexity was used initially as a general guide for defect-prone.

The following metrics were considered to calculating the strategic importance of a subset of units of code for application of testing activies, mainly unit testing: Cyclomatic Complexity, Number of corrective changes, and Operational Coverage that measures the intensity the artifacts are exercised in operational environment.

Some techniques and technologies were used to extract metrics to permit the execution of techniques of prioritization. To obtain the quantity of changes was utilized the software *svnstat* and the software *jdiff*. Also, was developed a software to realize a parser in the files of the repository to shows how many times an artifact was changed, and what the intensity that modification. The intensity was measured by number of lines of code changed.

The operational coverage was found utilizing the software Cobertura, and execution of an instrumented version in production environment. This metric was considered highly important, once it allows identifying which lines of code are really executed by user, and the frequency of utilization. Thus, even an artifact which metrics that indicates the needed of writing unit testing, it is discouraged if no actual use in a production environment occurs. On the other hand, an artifact can be strongly recommended to be tested, if it is very exercised sin a production environment.

All metrics were extracted and persisted in a database modeled specifically to allow the heuristics in their executions. The Figure 3 illustrates this process.

The first objective function is defined according to Equation 5.

$$si = \sum_{i=1}^{N} (NScc_{(i)}*Wcc_{(i)})+(NSoc_{(i)}*Woc_{(i)})+(NSfm_{(i)}*Wfm_{(i)}) \tag{5}$$

where:
si: Strategic importance;

N: Quantity of units (methods) in the system;

NScc: Normalized score of cyclomatic complexity of the method;

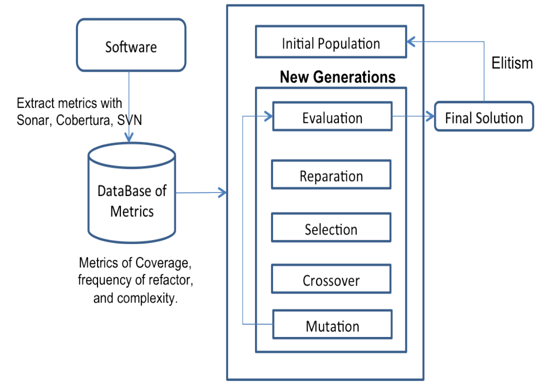


Fig. 3: Elitist Genetic Algorithm

NSoc: Normalized value of operational coverage of the method;

NSfm: Normalized value of Frequency of Modification on repository of version of the method;

Wcc: Weight considered for cyclomatic complexity;

Woc: Weight considered for operational coverage;

Wfm: Weight considered for Frequency of Modification on repository of version;

x: It receives value 1 if the artifact *i* has been considered, and 0 otherwise.

For each artifact the normalized scores *NS* (*NScc*, *NSoc*, *NSfm*) are calculated as in order to assign values between 0 and 100, according to example shown in Table 2. There are 2 kind of metric: bigger is better, and smaller is better. For the first one, the normalization is given by the following equation:

$$NS = \frac{(mv - min)}{(max - min)} * 100 \tag{6}$$

As an example, the metric of cyclomatic complexity presented in the Table 2, the higher its value is more interesting to choose this artifact to the application of unit testing. For the metric type the smaller is better, the normalization is performed by the following equation:

$$NS = 1 - \frac{(mv - min)}{(max - min)} * 100 \tag{7}$$

where:
NS: Normalized score;
mv: Absolute value of metric;
min: Lowest Absolute value of the metric between units;
max: Largest absolute value of the metric between units;

Each component (NScc, NSoc, NSfm) assumes a weight, and the sum of the weights must be equal to 1.

Therefore, the selected units will be those whose combination maximizes the objective function presented. Logically, if there is no restriction, all artifacts will be selected for employment of unit testing. However, this work owns as second objective function the Equation bellow:

$$ar = \frac{(\sum_{i=1}^{N} tt_{(i)} * x_{(i)}) - ha}{ha} \tag{8}$$

where:
ar: Availability of resources *tt*: time required for the development of unit testing activities (planning, implementation and execution) for the method *i*;
ha: hours avaiable.

The calculation of time *tt* is done based on the number of lines of code, and assuming as productivity average, 1 hour for every 10 lines of code. Therefore, if a method has 180 lines of code, the estimated time for the development of unit activity for this test method will be 18 hours.

The objective is not to exhaust the discussion on what are the best strategic components for selecting units of code, but provide flexibility to the software testing community to choose on different environment metrics that they consider appropriate in the selection model.

In order to address some penality for the selection that violates the time constraint, the fitness function may be calculated as:

$$F = si + (si * ar) \tag{9}$$

5. Experiments

For these experiments was considered critical software of industry responsible for the integration of a chain of services in accounting and tax segment. The software analyzed was developed in Java and has 808 methods, totaling 11,300 lines of code.

To obtain the operational coverage of the software was generated an instrumented version of the system. This version was put in a production environment for 4 months, and approximately 4,000 users used it during this period of time. Information for the last three months of the versions of each artifact to calculate the frequency of changes in repository were collected. The cyclomatic complexity of each artifact was obtained from the latest stable version of the system, using the software Sonar and metrics plugin.

Thus, considering arbitrarily the productivity in developing cases of tests as 10 lines per hour, these artifacts together would demand an effort of work around 1,130 hours. For sensitivity analysis using EA in this context, 100 experiments were performed, each considering a different availability of resources for the development of unit tests. These values were established in terms of percentage of total 1,130 hours varying from 1% to 100%.

It is expected that the optimization model minimizes the amount of hours wasted by not exist or can't find one or more artifacts to be incorporated in the list of selected artifacts. For example, if the time available for the development of unit test is 300 hours, and the best combination of artifats found by a heuristic demande 285 hours, 15 hours can be considered in this case as a waste of resources.

Table 1: Example of Representation of a Individual.

1	0	0	1	1	1	0	0	1	1

Table 2: Model of Normalization of Metrics.

Artifact	Complexity	Normalized Value
1	5	57.1
2	2	14.3
3	3	28.6
4	7	85.7
5	1	0.0
6	8	100.0
7	2	14.3
8	5	57.1
9	3	28.6
10	8	100.0

The strategy chosen for representation of the algorithm is binary. Each individual has *n* genes, and each gene can assume the values 0 or 1, indicating the presence or absence of an artifact in the solution candidate. The value of *n* is the number of methods in the project being evaluated.

Initially, the algorithm randomly generates the initial population by assigning each gene, a value: 0 or 1, according to the adopted representation. After that it holds the evolution of the population for generations. In each generation individuals are initially evaluated and assigned fitness to each one.

According to the representation of individuals presented in Table 1, this representation considers the presence of 6 artifact (positions 1, 4, 5, 6, 9 and 10) concerning genes with value 1. Using the metric complexity of these artifacts as an example, the model of normalization presents the values showned in Table 2.

Once evaluated individuals, the feasibility verification of each one is accomplished by ensuring that the time available for the development of testing activities is sufficient to test the artifacts represented in each individual. In cases which an individual violates the constraint of time available, one or more artifacts are removed from the individual, with the objective of making it feasible.

The algorithm worked with operators of crossover of 1 point, mutation operator per gene, and method of the roulette as selection operator to crossover, and the application of elitism value of 1 for the canonical algorithm.

5.1 Analysis of Results

The Figure 4 shows the average behavior of the standard deviation of the average time required for the development of unit testing to the artifacts prioritized. The behavior of the curve can be explained by the EA attempt to optimize the combination of artifacts. With a smaller amount of time available to allocate heuristic search in prioritizing artifacts with low amount of lines of code because they allow fine-tune the complementation.

In the first executions, whose available time is reduced, the selection of artifacts that consume more time for development is practically impossible, and allocating smaller artifacts to use the time available. This behavior is repeated until the heuristic can optimize your choice.

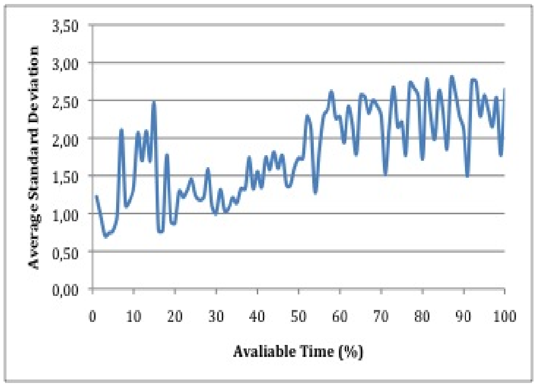


Fig. 4: Standard Deviation of Size of Artifact.

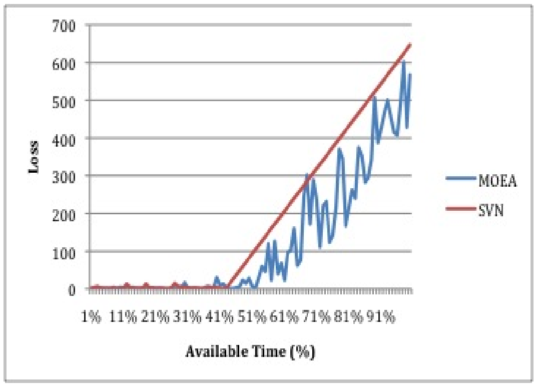


Fig. 5: Analises of Loss of Time.

The Figure 5 shows the performance of the heuristics compared to the non-use of the resource of time available, here called waste. The comparison chart shows that the first 50 simulations considered by heuristic called here by SVN [11], the EA proposed has been more effective in 31 of them (62 %). In all the last 50 simulations EA presented, the EA proposed own better results, totaling 81 simulations with better gain in relation to another strategy.

Another important feature in the first simulation 50 is related to the ability to complement the list of prioritized using artifacts from those, which have a small number of lines of code. This wide existence of artifacts with this feature allows the so-called *waste* to be less.

Figure 6 shows the cyclomatic complexity of the solutions during the experiments. It is noticed that the presence of artifacts with greater complexity is perceived only from the experiment with over 20% of available time. The artifacts with this characteristic are avoided in the presence of low availability of resources, since there is a direct correlation between cyclomatic complexity and number of lines of code.

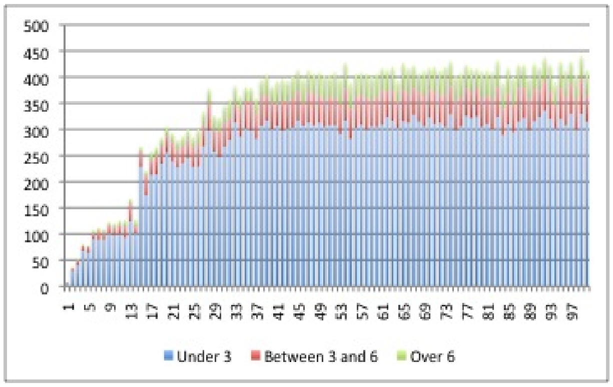


Fig. 6: Characteristics of complexity of the methods during the Experiments.

6. Conclusions

The main goal this work was present a new approach to prioritize artifacts in level of methods to application of unit testing. The approach is based in Multi-objective Evolutionary Algorithms as work of SBST. The framework proposed is guided by a flexible and dynamic set of metrics according to interest of the professional. In this work were considered 3 metrics: cyclomatic complexity, operational coverage, and frequency of changes on version's repository. The initial results were compared with one research in literature. We hope to experiment others techniques of multiobjective evolutionary algorithm as SPEA and NGSA.

We are conducting a research to try to figure out which metrics own the better power to be used by this framework to reveal defects. Techniques of data mining have been applied to address it. The initial results has been promising. Also, the experimentation of the implementation of EA in parallel environment is being tested.

7. Acknowledge

Grateful thanks are owed to the company Oobj Technology of the Information by its invaluable help in sharing the data, systems, and some workers for the development this work. This work was partially supported by FAPEG.

References

[1] R. M. Azuma. Otimização multiobjetivo em problema de estoque e roteamento gerenciados pelo fornecedor. 2011.

[2] C. A. C. Coello. Recent trends in evolutionary multiobjective optimization. In *Evolutionary Multiobjective Optimization*, pages 7–32. Springer, 2005.

[3] C. A. C. C. Coello. A short tutorial on evolutionary multiobjective optimization. In *Evolutionary Multi-Criterion Optimization*, pages 21–40. Springer, 2001.

[4] J. Czerwonka, R. Das, N. Nagappan, A. Tarvo, and A. Teterev. Crane: Failure prediction, change analysis and test prioritization in practice–experiences from windows. In *Software Testing, Verification and Validation (ICST), 2011 IEEE Fourth International Conference on*, pages 357–366. IEEE, 2011.

[5] M. E. Delamaro, J. C. Maldonado, and M. Jino. Conceitos básicos. In M. E. Delamaro, J. C. Maldonado, and M. Jino, editors, *Introdução ao teste de software*, pages 1–7. Rio de Janeiro: Elsevier, 2007.

[6] F. Elberzhager, A. Rosbach, J. Münch, and R. Eschbach. Reducing test effort: A systematic mapping study on existing approaches. *Information and Software Technology*, 2012.

[7] M. Harman and B. F. Jones. Search-based software engineering. *Information and Software Technology*, 43(14):833–839, 2001.

[8] IEEE. *Software Engineering Body of Knowledge (SWEBOK)*. 2004.

[9] R. S. Pressman and D. Ince. *Software engineering: a practitioner's approach*, volume 5. McGraw-hill New York, 1992.

[10] M. Ray and D. P. Mohapatra. Code-based prioritization: a pre-testing effort to minimize post-release failures. *Innovations in Systems and Software Engineering*, 8(4):279–292, 2012.

[11] E. Shihab, Z. Jiang, B. Adams, A. Hassan, and R. Bowerman. Prioritizing unit test creation for test-driven maintenance of legacy systems. In *Quality Software (QSIC), 2010 10th International Conference on*, pages 132–141. IEEE, 2010.

SESSION

SOFTWARE DEVELOPMENT STRATEGIES, AGILE TECHNOLOGY, BUSINESS MODELS, REUSE + CLOUD AND TOOLS

Chair(s)

TBA

Copyright © 2014 CSREA Press, ISBN: 1-60132-286-0; Printed in the United States of America

Contemporary Build Systems and Techniques for Improvement

Jason White and Kay Zemoudeh
School of Computer Science and Engineering,
California State University, San Bernardino,
California, United States

Abstract—*A build system plays a critical role in the software development process; particularly on large software projects. Having a build system that is fast, reliable, and easy to use can have a large, positive impact on a project. Iteration times can be reduced, giving more immediate feedback on changes; complete rebuilds can become less common or non-existent; and describing the build can become easier and more flexible.*

Background on how build systems work will be given; the strengths and weaknesses that apply to a vast majority of existing build systems will be shown; and methods by which contemporary build systems can be improved to achieve significant gains in developer productivity will be discussed.

Keywords: Build System, Software Tool, Automation

1. Introduction

Build systems (or build automation software) are tools that automate the execution of other tools, such as a compiler or linker. This is useful for automatically performing the tasks necessary to generate executables from source files.

In large software projects with thousands of source files and millions of lines of code, the build system can easily become a burden and bottleneck. A great deal of time and money is spent by organizations to streamline their software creation process. This involves reducing the time it takes to transform source files into deliverables such as executables or dynamic libraries. After all, software development is an iterative process and the faster the build system can run, the faster a developer can test their creation.

It is therefore highly beneficial to have a build system that enables its users to work faster and more easily.

2. Background

2.1 Fundamentals

In its most basic and abstract form, a build system is simply a job scheduler. It schedules jobs based on the dependencies between them. If, for example, job B cannot be performed until after job A is performed, then job A must be performed first. Once job A completes, then job B can be performed. The sequence of jobs should then be $[A, B]$.

2.1.1 Graphs

As shown in Figure 1, this scheduling problem can be represented using a Directed Acyclic Graph (DAG) where each task is a node and each dependency is a directed edge.

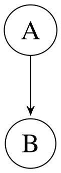


Fig. 1: A very simple DAG where B depends on A.

Most build systems use a graph to model the dependencies between nodes.

2.1.2 Nodes

Nodes can be further categorized as either *tasks* or *resources*. [2] A task might be a shell command to execute while a resource might be a file on the file system. In Figure 2, tasks are represented by rectangular nodes and resources are represented by elliptical nodes. A task takes zero or more resources as inputs. Based on those inputs, a task creates one or more resources as outputs. A task is then dependent on the resources it uses.

A task does not necessarily have to be a shell command. Tasks can be any procedure that produces one or more resources. Similarly, a resource does not necessarily represent a file on the file system. A resource can be any type of data. This includes files, environment variables, shared memory, and even the input or output from a device.

2.1.3 Edges

An edge between two nodes represents a dependency. In Figure 2, two edges are drawn between the resource '`A.h`' and the tasks '`gcc -c A.c`' and '`gcc -c B.c`'. Whenever the resource '`A.h`' changes, both of the tasks that depend on it should be re-executed.

Edges impose an ordering on when tasks can be executed. An ordering only exists when there are no cycles or loops in the graph. A graph has a cycle when a node ultimately depends on itself. A graph containing a cycle is illustrated in Figure 3. Build systems require an ordering so that builds

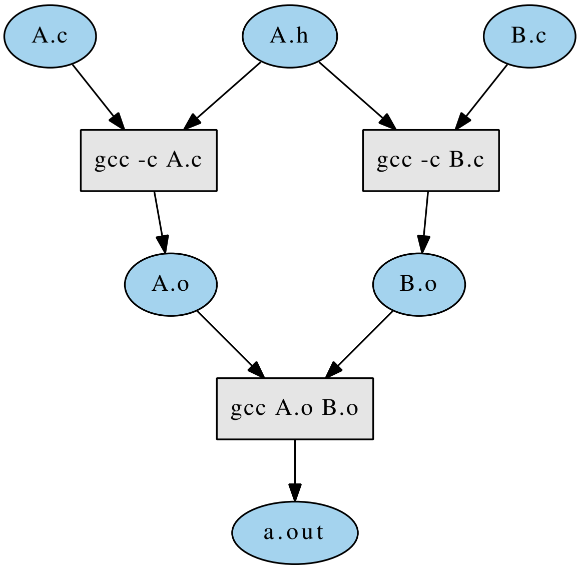


Fig. 2: A DAG of a simple C program.

can be completed in finite time. Thus, cyclic dependencies are usually forbidden.

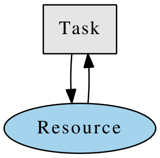


Fig. 3: A simple cyclic graph. Cycles are not always quite so obvious.

Cyclic dependencies are often the result of having a dependency on a resource which is too coarse-grained. Granularity is the extent to which an entity can be subdivided. Consider a file on a file system. A file consists of bytes and those bytes consist of bits. Suppose a task reads one region of a file and writes to a disjoint region of the same file. If the resource is considered to be an entire file, there would be a cyclic dependency. On the other hand, if the resource is considered to be a range of bytes within a file, then there is no cyclic dependency. Thus, cyclic dependencies can usually be resolved by adjusting the granularity of a resource.

2.2 Existing Build Systems

Quite a few build systems exist today. Some of them were created to serve a singular purpose or project, some for use with a certain language or domain, and some for the general case.

Single-purpose build systems are *ad hoc* build scripts that automate a build-related task. These are typically written in shell script, Python, or Perl. For example, a shell script containing a series of compiler commands is an *ad hoc* build script.

Language-specific or domain-specific build systems are designed specifically for a single programming language or family of programming languages. These build systems may work for any language, but they are engineered to work closely with one or more languages and their idiosyncrasies. For example, *Apache Ant* and *Apache Maven* are designed with Java in mind while *Boost.Build* and *Visual Studio* are designed primarily for C++.

General build systems are not tied to any particular language or implementation of that language. They merely run tasks based on the dependencies between them. *Make*, *SCons*, and *Tup* are all general build systems. This paper focuses on general build systems. However, many of the ideas and concepts described here can apply to domain-specific build systems as well.

2.3 Evaluating Build Systems

Existing build systems can be evaluated based on three criteria: speed, usability, and reliability. [1]

2.3.1 Speed

For most existing build systems, when the size of a project grows, so does the time it takes to update it. That is, the time complexity is $O(n)$ where n is the number of resources.

Ideally, the time it takes to update should depend only on the number of *changed* resources. That is, $O(m)$ where m is the number of changed resources. In the worst case, where $m = n$ and every resource has been changed, the time taken would be the same. However, in the average case, only one or two resources will have changed between builds. Thus, the amortized time will be on the order of $O(1)$.

How the build system scales with the size of a project is often more important than how well it can be parallelized. If the build system does not scale well, then throwing more processors at it will not bring significant gains.

2.3.2 Reliability

Reliability is the measure of how accurate a build system is at detecting changes and producing or maintaining a set of correct outputs given a set of corresponding inputs.

Reliability is heavily affected by the specification of dependencies (or lack thereof). A perfectly reliable build system would have all possible dependencies specified for every node. Such dependencies may include files, environment variables, command-line options, the current time, standard system libraries, or even other running processes. Specifying all possible dependencies is unrealistic and unnecessary, however. Some dependencies rarely change, some are inconsequential, and others are difficult to detect altogether.

Avoiding manual specification of dependencies and instead opting for automatic dependency analysis aids *usability*.

2.3.3 Usability

The usability of a build system is how easy it is to use. This includes running the command that invokes the build system, creating or maintaining build scripts, or simply learning how to use it.

The user of a build system can be a developer or another piece of software. For either type of user, it should be as easy and convenient as possible to use the build system. The build system should never get in the user's way by imposing unnecessary restrictions or requiring tricky workarounds for some obscure problem.

Unlike speed and reliability, usability is not as easy to measure. It is a subjective topic and depends on the skills and experiences of the user. It can, however, be measured in terms of how much time is spent dealing with the trivial matters of the build system by the *average* user.

2.3.4 Comparing Build Systems

The most influential build system, particularly with Unix systems, is *Make*. It was created at Bell labs and initially released in 1977. Since then, many variants and derivatives of *Make* have been created. Today, the most widely used variant is *GNU Make*.

Because *Make* is the *de facto* standard build system, many build systems are compared against it. In this paper, build systems will instead be compared against the *Ideal Build System*. The *Ideal Build System* is one that is fast, reliable, and easy to use. That is, updates do not depend on the *total* number of nodes, but instead on the number of *changed* nodes; dependencies never need to be manually specified, but are automatically derived; and little time is spent getting it to work. Of course, the *Ideal Build System* is, in reality, an unfeasible prospect. Creating a build system to approach these ideals is, however, quite feasible.

Comparisons of existing popular build systems to the *Ideal Build System* are shown in Table 1. There is not a single build system that does well in all three areas. *Tup* does well in all areas except usability. Although, at the time of this writing, *Tup* has been slowly improving in that area as well. Unsurprisingly, *Make* does poorly in all three areas. With *Make*'s widespread adoption and native inclusion in many Unix-based systems, its usage does not appear to be waning.

3. Improving Build Systems

The three facets of a build system—its speed, reliability, and usability—are not independent of each other. Each one affects the other. Increasing speed may decrease reliability and usability. Likewise, improvements to usability may take place at the expense of speed and reliability. There are many methods and techniques of improving speed, reliability, and usability. Each of these methods have their own advantages and disadvantages, and none of them are completely bulletproof.

3.1 Improving Speed

Increasing the speed of a build system is usually a matter of minimizing the number of tasks to perform and maximizing the throughput of the tasks that *are* performed. This is done by employing *incremental builds*, *parallel builds*, or *distributed builds*. Incremental builds reduce the number of steps required to perform the build by reusing the results from previous builds. Parallel builds and distributed builds utilize multiple threads or machines to run tasks simultaneously.

3.1.1 Discovering Changes

In order to enable incremental builds, the build system must first know what changes have occurred since its last invocation. As we will see in Section 3.2, the ability to discover changes has a large impact on the reliability of a build system as well.

Make, and most other build systems, discover changes the naïve way. They simply compare the timestamps of every output file with its corresponding input file. If the input file is newer than the output file, then the output file is updated using the associated shell command. This has the disadvantage of needlessly querying the file system for the modification time of every single file. For large projects, even if nothing has changed, this can take a very long time.

3.1.2 Change Notifications

Instead of *polling* for changes, the build system should be *notified* of changes. To be notified of file system changes, a user-space program can tell the operating system that it wants to be notified if a specific directory or file changes. This can be done by using the `inotify` API on Linux, the `ReadDirectoryChangesW` system call on Microsoft Windows, or the `kqueue`/`kevent` system calls on Mac or BSD. The build system can use a daemon (a process that runs in the background) that waits for notifications.

This method has the disadvantage of missing changes when the daemon is not running. Thus, the file system must be polled for changes the slow way when the daemon initially starts. This may be an acceptable trade-off, however, because it would only occur the first time the build system is invoked.

3.1.3 Change Journals

Certain file systems store a list of recent file changes on each volume in a *change journal*. Change journals were primarily designed for use by indexing or backup services. Since change journals are maintained by the operating system and not a user-space program, changes are not missed. A user-space program can read the change journal and scan for changes to the files it cares about.

The two main drawbacks of this approach are the requirement of root access and the dependence on a particular

Build System	Speed	Reliability	Usability
Ideal Build System	Updates always take minimal time, regardless of project size.	All possible dependencies are derived. Outputs are always correct. Orphaned outputs are deleted.	Intuitive and easy to use. Little time is spent getting it to work.
Make	The time to update is dependent on the total number of files.	Most dependencies must be manually specified. Some tools exist to generate implicit dependencies. Complete rebuilds are common due to incorrect outputs.	Uses a DSL (domain-specific language). The syntax can be cryptic and difficult to read and write. Makefiles are not platform independent.
SCons	Updates are dependent on the total number of files. Some speed is traded for reliability.	Checksums are used to detect file changes. Implicit dependencies can be automatically discovered for C++.	Python scripts are used to describe the build. Build scripts can be made platform independent.
Tup	Updates are only dependent on the number of changed files.	Implicit file dependencies are automatically discovered for any tool.	Uses a DSL as well as Lua. Imposes restrictions on where build description files must be. Not all features work on all platforms.

Table 1: Comparisons of popular build systems based on the criteria of speed, reliability, and usability.

file system. Build systems should not require administrative privileges in order to discover changes to files it normally has access to. Moreover, not all file systems support change journals nor do all file systems implement change journals in the same way. Depending on the environment, this may not be a reliable method for detecting changes.

3.1.4 Parallelizing Tasks

The other main method of improving the speed of a build system is the execution of build processes in parallel. For example, C source files do not usually depend on each other and can be compiled in any order. Therefore, they can be compiled simultaneously without any conflicts. This is a common optimization in most modern build systems. Builds where multiple threads are utilized to execute tasks in parallel are known as *parallel builds*.

Multiple networked machines can also be used to increase the speed of a build. Subsets of the build can be farmed out to other computers in a network. These types of builds are called *distributed builds*. Distributed builds are typically used to perform complete rebuilds as opposed to incremental builds. Change notifications and reliable dependency management are not necessary or useful when running builds from scratch. Therefore, this method becomes less important if the build system can be notified of changes and if dependencies are managed in a reliable manner.

Distributed builds have the disadvantage of executing in a different environment where subtle differences can result in different outputs, ultimately leading to unreliable builds. If execution in an identical environment can be guaranteed, as it is with parallel builds, then this disadvantage disappears.

3.2 Improving Reliability

The reliability of a build system is the direct result of how it manages dependencies, detects new dependencies, and discovers changes to those dependencies.

An unreliable build system might fail to run tasks because it lacks dependency information or lacks the ability to detect changes to certain dependencies altogether. For example, *Make* does not detect changes to the Makefile itself. If the Makefile is modified to change the options passed to a compiler command, the change will not be detected and the command will not be re-executed. This becomes problematic when, at a later time, the linker is still linking old object files because the compiler was never re-invoked with new command-line options. Bugs resulting from this problem can be particularly frustrating to track down.

3.2.1 Dependency Management

The reliability problems of *Make* stem from how it detects changes to its internal graph. It does not know when a node is added to or removed from the graph. Indeed, it *cannot* know because it has no knowledge of what the graph looked like from previous invocations.

This problem can be solved by storing the graph in a persistent database. When the build system is invoked, it can determine if nodes have been added or removed by comparing the differences between the old graph and the new graph. Special action can then be taken based on these differences. Resources that were generated by removed tasks can be deleted as they are no longer needed. This also improves the usability of a build system.

3.2.2 Detecting New Dependencies

C and C++ source files can include other source files. The output of the compiler command that compiles a source file is also dependent on all the other files that the source file includes. Dependencies such as these are called *implicit dependencies*. As opposed to *explicit dependencies*, implicit dependencies are not usually specified by the user. A build system should be able to detect and handle implicit dependencies automatically.

Most current build systems use *ad hoc* tools for extracting implicit dependencies. For example, `gcc` has the option `-M` to generate dependency information for *Make*. *SCons* on the other hand, uses regular expressions to scan source files for `#include` directives.

Each of these methods have major disadvantages. They are not general solutions and do not detect all possible implicit dependencies.

A more general approach would be to intercept all system calls that a process makes. For example, when a process attempts to open a file in read mode, the build system would know the task should have a dependency on that file. Likewise, if the process attempts to write to a file, the build system would know that an output file has been created by that task. Detecting dependencies by intercepting system calls would work for all tools, not just `gcc` for example. However, intercepting system calls is not always easy, efficient, or platform independent.

On Linux, the `ptrace` system call is primarily used by debuggers to determine exactly what a process is doing and which system calls it is making. Unfortunately, `ptrace` is slow, difficult to use, poorly designed, and not available on other platforms. Its performance implications alone make it impractical to use for detecting dependencies where speed is a necessity. The build system `fabricate` uses this method for automatically detecting dependencies on Linux.

Another method for automatically obtaining dependencies is called *system call patching*. When a process is created, its code segment is scanned for system calls and replaced with an instruction to jump to a subroutine of one's choosing. This subroutine can then establish a dependency and hand control off to the real system call. This method is fairly involved and requires writing at least part of a disassembler. Additionally, the initial overhead of patching the system calls can overshadow the total running time of the process. Thus, long-lived processes benefit the most from this approach.

Another method for automatic dependency detection is through *DLL injection*. A shared library is "injected" such that it is loaded in place of another shared library. The process then makes calls to the injected library instead of the intended one. This could allow calls such as `fopen()` to be intercepted and used to detect dependencies on files. A disadvantage to this method is that it will not intercept all desired dependencies for all processes. For example, not all processes will use `fopen` from `glibc`; they may use the system call `open()` directly.

For only detecting dependencies on files, a custom file system can be used. On Linux, *FUSE* can be used to create a file system in user-space. That is, a custom file system is mounted in a specified subdirectory. Whenever another process attempts to read or write files on this file system, the *FUSE* server is notified and can decide what information to return. Most importantly, the *FUSE* server knows the ID of the process that is requesting file information. From this, the build system can determine the files that a particular task depends on and the outputs it generates. The disadvantage of this approach is that it is not platform independent. For Microsoft Windows, a custom driver must be created that allows the creation of file systems in user-space. *Dokan*, a port of *FUSE*, is available, but is unstable and prone to crashes.

3.2.3 Detecting Changes

As discussed in Section 3.1.1, file changes can be discovered from file system notifications. But we must also be able to detect changes to other types of resources such as environment variables. For the types of resources that a build system cannot be notified of, a *polling* approach must be used. If the build system stores the graph and the state of all its nodes (see Section 3.2.1), changes to resources can be detected by polling for them on each build invocation. This is what *Make* does for detecting changes to files.

To reduce the number of changes that must be detected, the granularity of resources can be adjusted. [2] By grouping similar, rarely changed, resources together, such as standard library header files, the size of the graph and the number of changes that must be detected can be drastically reduced. All the files in a single directory can be grouped together into a single resource. When a change to the directory is detected, all the tasks that depend on it can be updated.

Care must also be taken to detect changes to the build description files themselves. Build description files can be treated as a file resource and their parsing as a task. Thus, if the file changes, it should be re-parsed.

3.3 Improving Usability

The usability of a build system is critical to the productivity of the user. There are two main sources that affect the usability of a build system: how the build is described and how the user interacts with it.

3.3.1 Build Descriptions

For all build systems, the user must be able to describe the build—that is, the tasks to run and which resources those tasks depend on. The usability of a build system is greatly affected by how easy it is to write build descriptions.

Following in the footsteps of *Make*, many build systems use a domain-specific language to describe the build. This build description language is often very limited compared to traditional programming languages. For example, in *Make*, complex conditional statements are difficult to write because `‘else if‘` is not supported. Small problems such as this can make domain-specific languages difficult to write and maintain.

SCons uses Python to communicate instructions to the build system. This provides great flexibility and extensibility.

The processing power and flexibility available to Python is near-impossible to mimic in domain-specific languages. Using an established and widely used programming language also has the added benefit of being easier to learn, maintain, and write.

The only downside to using a general programming language is the possibility of too much verbosity in the build description. Domain-specific languages are tailored to the task at hand and can provide terser build descriptions. The syntax of general programming languages is not optimized for describing builds. In practice, however, these are minor issues that are quickly dwarfed by their expressive power.

3.3.2 Interaction

Making the build system, as a whole, easier to use also greatly impacts its usability. This includes trivialities such as how the build system is invoked, how often is invoked, or simply how good the documentation is.

One major boost to usability is the automatic invocation of the build system. By being notified of a file change, the build system can automatically run the necessary tasks that are dependent on the file. This could allow the build system to sit in the background, automatically recompiling the necessary files as they change, without ever being manually asked to do so.

4. Conclusion

Significant improvements can be made to existing build systems. By improving speed, build times for large software products can be dramatically reduced. By improving reliability, the need for complete rebuilds can be eliminated. By improving usability, time spent fiddling with the build system can also be eliminated. Such improvements would help facilitate the creation of large scale software.

References

[1] Mike Shal. *Build system rules and algorithms*. http://gittup.org/tup/build_system_rules_and_algorithms.pdf, 2009.

[2] Derrick Coetzee, Anand Bhaskar, and George Necula. *A model and framework for reliable build systems*. http://arxiv.org/pdf/1203.2704.pdf, February 2012.

A Survey of Building Robust Business Models in Pervasive Computing

Osama Khaled
Computer Science and Engineering Department
The American University in Cairo
Cairo, EG
okhaled@aucegypt.edu

Hoda M. Hosny
Computer Science and Engineering Department
The American University in Cairo
Cairo, EG
hhosny@aucegypt.edu

Sherif G. Aly
Computer Science and Engineering Department
The American University in Cairo
Cairo, EG
sgamal@aucegypt.edu

ABSTRACT

Pervasive computing is one of the most challenging and difficult computing domains nowadays. It includes many architectural challenges like context awareness, adaptability, mobility, availability, and scalability. There are currently few approaches which provide methodologies to build suitable architectural models that are more suited to the nature of the pervasive domain. This area still needs a lot of enhancements in order to let the software business analyst (BA) cognitively handle pervasive applications by using suitable tasks and tools. Accordingly, any proposed research topic that would attempt to define a development methodology can greatly help BAs in modeling pervasive applications with high efficiency. In this survey paper we address some of the most significant and current software engineering practices that are proving to be most effective in building pervasive systems.

Author Keywords

Business Analysis, Elicitation, Design techniques, requirements gathering, Survey

ACM Classification Keywords

D.2.1 [**Requirements/Specifications**]

D.2.2: [**Design Tools and Techniques**]

INTRODUCTION

The pervasive computing concept was first introduced by Mark Weiser [1] in 1991 as if he was reading into the future of computing in the 21st century. He was convinced that personal computers are not satisfactory for integration into humans' lives in a natural way. He was convinced that computation will converge to become ubiquitous. In other words, computation will be present "everywhere" and will be featured by its invisibility to the human eyes, yet available for people to use unconsciously. This vision may have been impossible to achieve during the 90s of the last century, but we do nowadays have all the technologies that we need to achieve Weiser's vision. We have advanced wireless networks distributed in many areas, GSM networks across all countries, hand-held and mobile devices with integrated sensors, appliances with embedded computers and wireless controllers, and more importantly industry and universities are more willing to spend money on research in these areas. MIT Oxygon, IBM, and AT&T researches are just examples for huge research investments [2].

The idea is attractive for many researches and has proven its success in many forms. Mobile technology is considered one type of pervasive computing, although not fully ubiquitous, but is considered a very successful model. Today, people are so closely attached to their cell phones and to their applications. Moreover, people who experience the luxury of modern new cars that sense their owners, warn drivers on parking actions, or take preventive actions to avoid accidents will really appreciate this futuristic technology. People need this kind of technology that facilitates their life without losing the main goal or purpose that they want to achieve. It is only natural, psychologically, to focus on goals and utilize activities to achieve the purpose as described in the activity theory [3]. It is not just luxurious, but it frees the user's mind for a more important goal to be achieved.

Researchers that work in this domain face many challenges, however. Pervasive computing descended from other computing fields, like distributed systems, and mobile technologies and hence inherited their existing challenges. It is characterized by the common appearance of factors like context-awareness, system adaptability, and volatility. In addition to the above, researchers are concerned with privacy, security, safety, and limited resources as main issues that must be resolved. As understood from the term ubiquitous, personal information may be collected and distributed without permission from its owner. This can raise legalization issues that must be resolved within the information distribution laws. In addition, if security can be breached for devices, appliances, or cars, this may cause high risks to their users, which results in safety threats that must be handled as well [4]. The challenge of limited resources is inherited from the mobile technology, but it will be more apparent with pervasive computing since the processing requirements will constantly increase. This can also lead to higher consumption for devices' resources like batteries.

In pervasive computing, there are many smart objects that have computation capabilities and can interact with each other using different network channels and can sense the changes in their surrounding world using their sensors and this is called **context-awareness.** A pervasive application can be stimulated by many things like light, sound, movement, gravity, temperature, or system changes. If the smart object reacts towards that change, then it has a feature called **adaptability.** For example, a pervasive computing solution can detect the existence of a teacher in a class and based on the saved teacher's profile, which makes the class switch on the light and start the smart board, then starts up the class computer.

Although many researches focus on filling the design gap in pervasive computing by focusing on frameworks and modeling languages, there are still other successful practices that need some attention as well. Software business analysts must be able to transform business dreams into real implementations. Accordingly, the analyst must know how to deal with business needs as well as design constraints. This survey paper is an attempt to provide as much information as possible about the current software engineering practices in the domains of software requirements engineering, high level design and architecture, and design patterns that are useful for building pervasive systems. We believe these knowledge areas are equally important for today's business analyst and architect.

The paper is organized as follows. In section 2 we cover software requirement engineering practices. Section 3 surveys the socio-cultural aspects of ubiquitous computing and their reflection on design decisions. Section 4 surveys some of the existing design and architecture practices. Section 5 discusses some of the ongoing research work on design patterns.

Requirements Engineering Approaches for Pervasive Computing

Requirements Engineering (RE) is one of the most important and difficult tasks in software engineering. It is the step during which one realizes the needs for building a new system. The analyst studies the technical, economic, and cost-benefit aspects of system needs. The job of the analyst is to come up with a clear analysis model of the stakeholders' needs that can be easily answered in the design phase. As IBM [5] puts it : "business analysis is the corner stone of any project success."

The International Institute of Business Analysis (IIBA) defines the business analyst's role as "a liaison among stakeholders in order to elicit, analyze, communicate and validate requirements for changes to business processes, policies and information systems. The business analyst understands business problems and opportunities in the context of the requirements and recommends solutions that enable the organization to achieve its goals [6]."

Analysts can approach pervasive computing systems using the traditional requirements engineering methods. However, according to the IIBA, the business analyst must improve the process continuously and provide high quality systems and products [6]. The proposed requirements engineering approaches evolve mainly around elicitation, and analysis techniques proposed in the pervasive computing domain.

Many researchers realized the need for special techniques, which are more suitable for pervasive systems. Lyubov et al. [7] claim that existing requirements engineering techniques are not enough to engineer requirements for pervasive systems. They propose procedures to help the analyst in eliciting and analyzing requirements properly. The following steps represent their approach

1. Identify system stakeholders and engage with them to capture the required needs.

2. Build a detailed business model for the environment derived from the information captured from stakeholders.

3. Hold workshops with stakeholders which are close to brainstorming sessions where stakeholders set their perceptions on the pervasive system.

The authors used the following pervasive system contextual properties to serve as guidelines in the different engagement sessions with stakeholders. These contextual properties are:

1. The spatio-temporal context: it describes properties like time, location, direction, and speed.

2. The environment context: which describes objects around the user like services, persons, and noise.

3. The personal context: which describes the user's physiological and mental state.

4. The task context: which reveals the user's explicit goals, tasks, and actions.

5. The social context: which describes the user's relations with others and his/her role at work.

6. The information context: which describes the global and personal space available.

There are other researchers who followed a similar elicitation practice, but on a completely different theoretical background. For example, Afridi and Gul [8] adapted the activity theory in the field of psychology. The activity theory says that when individuals engage and interact with their environment, new tools are produced. These tools are considered forms of mental processes, and as these mental processes are manifested in tools, they become more readily accessible and communicable to other people, thereafter becoming useful for social interaction [3].

Another definition for the activity theory is that it is a descriptive framework that considers an entire work/activity system (including teams, organizations, etc.) beyond just one actor or user. It Accounts for the environment, history of the person, culture, role of the artifact, motivations, complexity of real life action, etc. The user performs an activity to achieve a certain goal. The theory includes cultural and technical mediation of human activity, artifacts in use (and not in isolation). Activities consist of goal-directed actions that are conscious. Constituents of an activity are not fixed; they can dynamically change [3].

Afridi and Gul [8] argue that elicitation techniques such as group-driven elicitation or model-driven elicitation have some drawbacks in context-aware systems as they do not address the emergent time-model, the priorities of context-aware scenarios nor the scenarios' constraints. Their research proposed specific procedures that help in eliciting requirements:

1. Enlist all the tasks in operations.

2. Define primary and secondary activities for the system domain.

3. Develop an activity chart to complete the activity life cycle.

4. Identify where to enable, the technology or activity. And enlist the key activities for which context should to be used.

5. Define how context benefits the productivity and efficiency in terms of resources (time, HR, equipment, labor, physical activity, computation).

6. Establish context variables required for the context awareness i.e. time, location, bandwidth etc.

The above procedure uses the same classical elicitation techniques, but with special focus on context as the main driver. It addresses also the cost-benefit of using context to automate a mobile computing system.

Munoz and Pelechano [9] rather preferred to adapt the existing UML analysis model and customize it for pervasive computing systems. They introduce some interesting approaches in the software development life cycle. They proposed an analysis model approach based on UML where the analyst has to build the services model, the structure model, and the interaction model. The services model is based on the UML class diagram, and they model the behavior by using the state-transition diagram. They went deeper and described the acting component inside each service using the UML component diagram. They also used the UML sequence diagram to describe the interactions among services, and they recommended the design of a single diagram for every interaction. They link this approach with other steps towards the required system architecture (Figure 1)

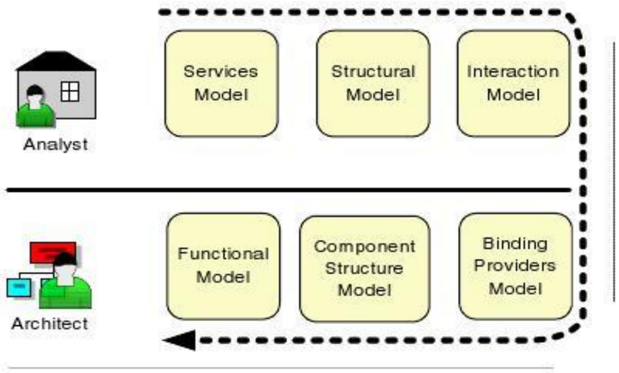


Figure 1: The six Model of Perv-ML [9]

Stéphane et al. [10] and Hen-I [11] took a specific aspect of the pervasive system to analyze. The first introduces a methodology for Trust Analysis and describes techniques to find inherent trust issues in the pervasive system that helps, as claimed, in guiding the system design. The second discusses safety issues and gives a deep analysis in order to be considered in the system design.

Trust Analysis by Loprestiel etal [10] recommends eleven trust issues categorized as subjective, data, and system. They present their approach as a matrix-based analysis. They propose 4 steps to realize and understand trust issues fully. They say the analyst must write the pervasive computing scenarios fully and ask subject matter experts to review them. Then, the analyst builds a trust-analysis matrix table that analyzes vignettes of the scenarios, and checks the trust issue value against each vignette. After that, the trust-analysis matrix goes into a peer-review session to enhance the scenarios, which is the fourth step. The fifth and the last step is to guide the design by identifying the most significant areas that need attention and match technology against design. The authors also present a trust-analysis matrix for common technologies used in pervasive computing which is quite interesting.

Francisca et al [39] introduced a different model for requirements engineering which requires active user interaction during the elicitation phase. The authors introduced this approach through a visualization tool which helps the user view the location of the devices in the smart space as shown in Figure 2. They help the user put his requirements through an elicitation process which the user would have to specify:

1. The scope of the context
2. Define system specifications using predefined list of characteristics in the system catalogue.
3. Refine system specification for those characteristics which are not found in the catalogue.
4. Validate the gathered requirements

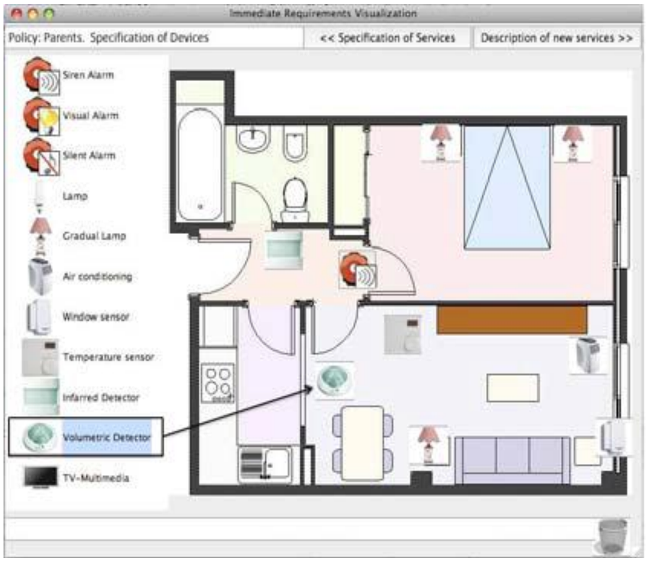


Figure 2: Snapshot of prototype visualization tool [39]

Yong and Helal [11] refer to a specific criteria or aspect of pervasive computing which is safety. They give a rich analysis for different risk scenarios that may cause safety hazards. For example, they describe the conflict that may happen between two different appliances if there are side effects in their computation of the temperature, which may cause severe hazards. They describe other scenarios that show different types of risks as well. Authors claim that any solution focusing on safety must focus on four main contributors in any pervasive computing environment: device, service, user, and space. They describe and analyze their role in pervasive computing in order to put proper solutions for safety and minimize risk of hazards for these items.

Socio-Cultural Analysis for Pervasive Computing

Analyzing cultural and social behavioral patterns takes a considerable space in ubiquitous computing. Business Analysts need to have a deep understanding of users' intrinsic behaviors and reasons behind them. This understanding represents the corner stones of all the work directed towards building an efficient pervasive computing solution.

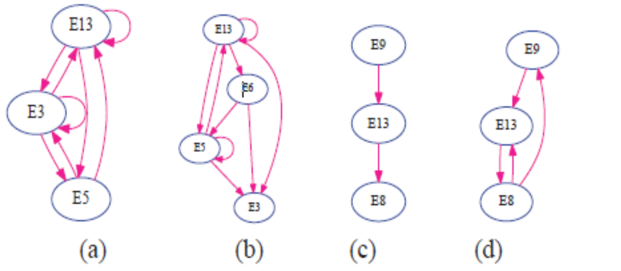


Figure 3: Example of causal graphs representing two different behaviors of a user in doing an activity. (a) and (b) represent behaviors of a person doing 'Use bathroom', while (c) and (d) represent behaviors of a person doing 'Get drink'. Nodes represent events. [31]

Several studies have been conducted within that context. For example, Chikhaoui et al. [31] introduce an attractive approach to build personal profiles by understanding users' behaviors and their relationships through a casual model. The researchers visualize the model as an undirected graph linking major behavioral patterns with each other to help in design decisions as shown in Figure 3.

Another research by Kawsar et al. [32] attempts to understand how people use technology in households especially those connected with the Internet. Their findings show that the role of devices such as desktop PCs diminished to be used for special purposes like working from home or game playing, while tablets and smart phones are being used now on a larger scale especially with internet-related services. Moreover, locations like kitchen and bathroom are common places for several computing activities.

In another similar example, Takayama et al. [37] studied sources of satisfaction in home automation systems. Their research team worked to answer some key questions related to the purpose, meaning, and usability of the home technology. The answers to these questions represent important values of the user, which they found to include things like personalization, and entertainment and making impression for others.

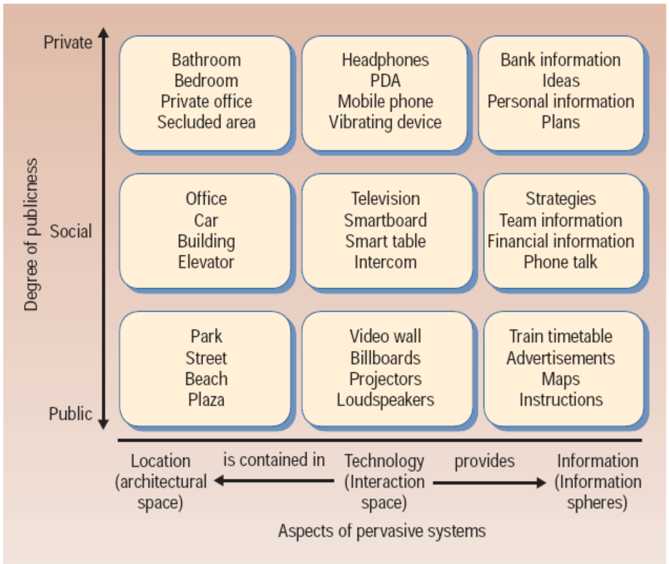
Grönvall et al. [33] approached household ubiquitous technology in healthcare applications based on a deep understanding of the non-functional aspects surrounding it. They focused their study on people, resources, places, routines, knowledge, control and motivation. The outcome of the research shows, for example, that patients and care-networks need to be aware of their health situation through learning and reflection on non-regular settings.

Tian et al. [34] studied user behavior in video-chatting services and got to understand behavioral trends with respect to many aspects such as the duration of the chat, usage of the camera, and the misbehaving users. The study shows that normal users directly face the camera in opposition to misbehaving users who hide their faces. They also show those strategies for selecting the proper partner has to be developed as chat durations are short mainly because of failing to select such a partner.

Mainwaring et al. [35] conducted a very interesting study to understand the usage of digital cash solutions using Sony FeliCa NFC (Near Field Communication) smartcard technology in Japan. They found that the Japanese society prefers to use this NFC technology rather than using credit cards as they tend to save time which is consistent with their cultural habits taught to people to avoid commotion as much as possible.

Lin et al. [36] researches the privacy concerns of the users who install Android applications with respect to permissions needed to access phone resources. Their approach focuses on bridging the gap between the expectation of the user from the application through what is known as the mental model, and the actual features and permissions needed by the application to access mobile sensitive resources. This kind of understanding prompted them to build a new privacy summary interface to help the users take a proper decision via reading past misconceptions of the users.

Kostoko et al [12] introduce an interesting conceptual framework for privacy/public issues in pervasive systems within urban areas. They divided the publicness into public, social, and private aspects and related them to three selected aspects of pervasive systems which are location, technology, and information. The analysis of this approach is shown in Figure 4. A Social degree means that it is neither public nor private, and may indicate that there is a group access rather than individual access. Figure 4 shows situations at which locations, technology, and information can be public, social, or private. For example, headphones are considered a technology that imposes privacy. Train-time table is a public piece of information. A person talking in the elevator is in a social location.



	Location (architectural space)	Technology (Interaction space)	Information (Information spheres)
Private	Bathroom Bedroom Private office Secluded area	Headphones PDA Mobile phone Vibrating device	Bank information Ideas Personal information Plans
Social	Office Car Building Elevator	Television Smartboard Smart table Intercom	Strategies Team information Financial information Phone talk
Public	Park Street Beach Plaza	Video wall Billboards Projectors Loudspeakers	Train timetable Advertisements Maps Instructions

Figure 4: Publicness spectrum. The vertical axis represents the degree of publicness, while the horizontal axis describes three main features of pervasive systems and the relationship between them [12].

Architecture and Design Approaches in Pervasive Computing

The technical community members agree that system architecture and design are considered key success factures for any system and solution. This section discusses the design issues, profound architecture approaches which address these design issues and more, technologies that can be used with pervasive computation and finally different architecture models which address key issues in pervasive systems.

Pervasive computing is not new in terms of technology, but is considered an innovative paradigm. It inherits its design issues from distributed systems, and mobile computing [12]. They characterize the fields that architects should deal with to provide suitable designs. We will focus here on the distributed system design issues as they are also major design issues in pervasive computing. The following points are considered the main design issues in any distributed system [13]:

1. **Heterogeneity**: the system should be designed to work through different types of computers, networks, operating systems, programming languages, and applications implemented by different developers

2. **Openness**: characterized by the number of published key service interfaces, which are possibly built over heterogeneous hardware and software resources

3. **Security**: is concerned with protecting data from being leaked to unauthorized individuals, protecting data from corruption and alternation, and ensuring accessibility to data whenever requested

4. **Scalability**: this issue describes the degree of the system efficiency whenever the number of resources or users increases

5. **Failure Handling**: is concerned with detecting failure points of the distributed system and the ability of the system to handle them through masking them or tolerating their failure, and how efficient it is when recovered from failure.

6. **Concurrency:** the system design must ensure proper performance and correct behavior of shared resources under concurrent access from different clients.

7. **Transparency:** user should not be aware of the system details and should deal with it as one unit. For example, the user should not worry about the location of services, and their failure. The user should not also worry about replication of services.

8. **Quality of Service:** it is a very important design issue which provides constraints on the provided services in order to get the required quality. For example, there could be deadlines for system response time. There could also be boundaries for system availability and security.

Dargie et al. [38] identified a number of challenges specific for any future ubiquitous computing system, namely:

1. **Adaptive control**: where ubiquitous devices may need to make decisions using uncertain data

2. **Reliability and accuracy**: where future work needs to address accuracy of recognition algorithms and the possibility of making use of cloud computing resources.

3. **Security and Privacy**: how a device can recognize other sensing devices and employee proper security and privacy strategies as well.

4. **Hybrid Intelligence**: mixture of non-deterministic and deterministic intelligence mechanisms to reason about context types.

5. **Unified architecture**: where a rapid and common architecture is required.

6. **Tool Support**: the need is still there to have tools to support rapid development of context-aware Systems

The question here is what are the design issues that are critical for pervasive systems? There are two major design constraints in pervasive systems, i) context-awareness and ii) quality of service. The main characteristic of the pervasive system is to adapt to context changes. This means that a pervasive system must have the capability to detect its surrounding environment (context) according to the scope of the system, and adapt itself to changes that may occur. Context-awareness covers design issues related to device location, motion, network availability, information access, device energy [2].

The quality of service is an inherited design issue from distributed systems. However, quality issues are more obvious in pervasive systems as processing, memory, and disk space are just adequate for the mobile device to operate. Moreover, client applications may be hosted on mobile devices and appliances that in many cases change their context, e.g. change location, which leads to disturbance of the services as communication may be lost. In addition, mobile devices use batteries that run out of power according to the device utilization and to processing activities that also lead to service disconnection [2]. Hence, limitation, instability, and degradation of resources are all reasons that impact the quality of service.

It is important to mention that the Service Oriented Architecture (SOA) is considered an established approach that addresses the above mentioned issues in distributed systems. SOA is an architecture approach in which the system functionality is represented as a service and separated from the service consumers. The main characteristics of the SOA architecture are [14]:

- Services have well-defined interfaces and policies
- Services usually represent business functions
- Services have a modular design
- Services are loosely coupled
- Services can be discovered
- Services' location is transparent to service consumers.
- Services are independent from transport
- Services are independent from the platform.

Figure 5 shows the conceptual components of the SOA architecture

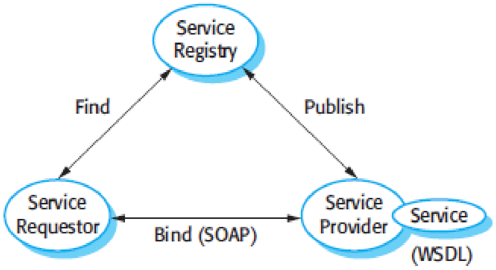


Figure 5: SOA conceptual components [15]

Many researchers used web-service technology to implement pervasive systems. Web Services technology is considered a standard XML-realization for the SOA architecture as it provides useful techniques that fulfill SOA guidelines [13]. For example, Ruimin et al. [16] used agent-based web services with web applications and mobile devices in a client-server model so that server-based web-services can recover if a client disconnects at any time. Ranganathan and McFaddin [18] used workflows to coordinate the execution of the web services in a pervasive system. On the other hand, some Researchers see that web-services incur an extra overhead of communication due to using XML in its messages which requires additional processing power to parse its content, and consume more network bandwidth than binary remote procedure calls [17].

There are other technologies that are designed specifically for embedded systems and adapt to SOA architecture guidelines. These technologies use native or binary procedure calls. Harihar [19] depicts Jini as an existing Sun Java-based technology already designed for embedded systems. As explained, Jini can satisfy all pervasive system's characteristics such as ubiquitous access, context-awareness, natural interaction, intelligence, security, and reliability. Architects designed Jini so that it fits in any hardware that has processing, memory, and network connectivity. The technology is portable in a way that it does not require a hardware driver nor a special protocol, and is not designed for a specific operating system.

The goal of the Jini technology is to turn the network into a flexible and easily administrated environment with respect to its resources, which are acquired by users. Resources can be either software programs, hardware devices, or a combination of both [19]. For example, the architecture of the Jini technology is based primarily on the lookup service which links both the client and the service provider to allow for service discovery. It adapts leasing in order to free unused resources, or services, to make them available for other clients (Figure 6) [19].

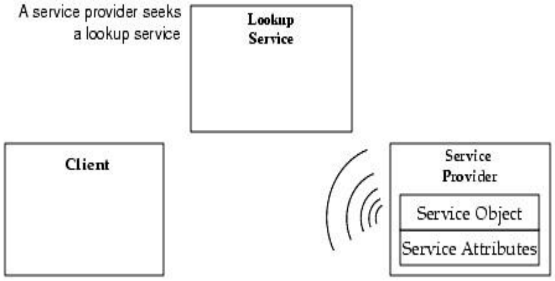


Figure 6: Jini Discovery Architecture Model [19]

There are other technologies provided by Microsoft and HP that are designed to implement pervasive systems. Microsoft implemented UPnP (Universal Plug and Play) as an open platform based on HTTP, XML, and SOAP. HP implemented JetSend which provides peer-to-peer capability between devices to allow information exchange [19]. It is important to note that every technology has its pros and cons and the selection of the technology to use must be done very carefully.

There are some frameworks that target the development of pervasive systems with different capabilities and are designed for different purposes. For example, the JCAF (Java Context Awareness Framework) [20] is a java based framework for implementing context-aware applications. It offers a high flexibility for programmers which allows them to implement varieties of pervasive systems running on different contexts. The framework followed some design principles like flexibility of distribution with loosely coupled services. It is designed also to show context-adaptive behavior according to context events. It provides privacy and security protection mechanisms for data although pervasive environments are not secured by nature. Additionally, they have programmer APIs for extensibility in order to allow for different types of customizations.

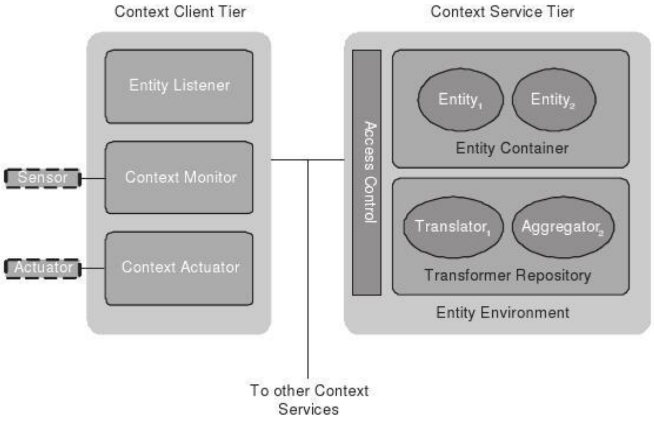


Figure 7: The Runtime Architecture of the JCAF Framework [20]

It is very useful to explore the JCAF runtime architecture to understand some concepts in pervasive computation. Its design stimulates thinking and shows a good level of abstraction. The runtime architecture (Figure 7) is composed of two tiers Context Service Tier and Context Client Tier. The Context Service is responsible for handling context in a specific environment and communicating with other services (peer-to-peer). It is ultimately a process running on the J2EE Application Server. Inside the context Service, we see the Entity Container which manages Entities. Entities respond to changes in the context. An Entity Container handles subscription to context events and notifies clients accordingly. The Entity Environment provides the required resources for entities. Access Control provides the required authentication to access the entity environment.

A Context Client is a client that can access a context service either via a normal request-response scenario, or by subscribing to context events on specific entities. It can monitor context changes via the sensors and update the entity accordingly. It can also change the context if it is an actuator in cooperation with other actuators.

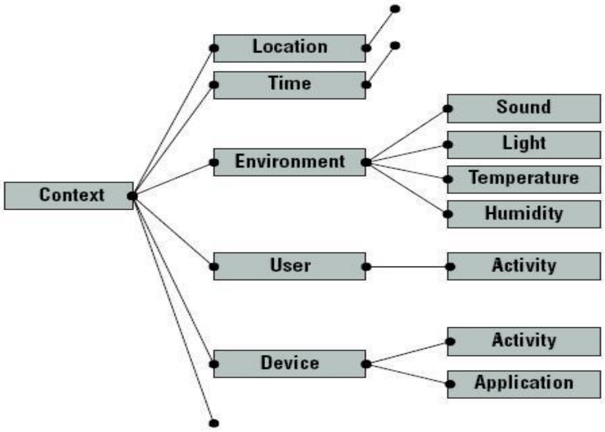


Figure 8: CMF Context Ontology Main Elements [21]

JCAF is so generic and does not provide all required advanced architecture functionalities for pervasive systems. Korpipää et al. [21] worked on a framework with open APIs called Context Management Framework (CMF) designed for Symbian mobile phones. It allows real-time context reasoning for information even if there is noise. Researchers use an expandable ontology which clients can use in different contexts. The framework design principles are built over security and event-based interaction with clients. The real power of the CMF framework is its capability for reasoning based on context variables. Figure 8 shows the main categories that the CMF reasons against. It is important to notice that the framework APIs allow the client to interrogate with context information to reason, subscribe for events, or change behavior according to the context variables CMF Ontology's main Vocabulary.

Other researchers focused on resource discovery and tried to refine its behavior to make it more efficient. For example, Kalapriya et al. [22] present a resource predictor mechanism along with the resource discovery in order to detect variability of resources if they are available. Resources, if available, may vary based on their location, and accordingly a mobile device should detect their variability as early as possible so that precautionary actions can be taken if the resource cannot meet device task requirements. They devised their research for mobile devices, which may lose resources upon changing location. They claim that it will also help in recovering from service disconnection and handoff resources smoothly if disconnected when changing location.

Petrus and Ravula [23] worked on more or less the same design issues as Kalapriya et al. [22]. Petrus and Ravula [23] provide their view for a fault-tolerant pervasive system by adapting principles of software architecture. They argue that by providing services and resource discovery, fault-tolerant, and component replication, the system will be more stable. They provide their own middleware solution which is called "Prism-MW" and they argue that it resolves the key architecture principles to achieve the required fault-tolerance. The research also addresses the limited computational resources in any pervasive system and the need for faster failure recovery. Accordingly, they adapt an active replication technique, which consumes more computational resources by nature, but provides analytical algorithms to identify the components to be replicated and achieves the best performance with less failure and less computational resources.

Hafez et al. [24] introduce context-aware architecture for pervasive systems which allows required services to adapt to quality of service requirements by clients. The research work highlights three major design issues in existing context-aware architecture solutions, namely: openness, scalability, and extensibility. Their proposed solution provides mechanisms for designers so that they can provide services that match client quality of service requirements. They offer a QoS-Broker, which is responsible for deciding on whether the served client received the required QoS or not. It also takes corrective actions and applies self-healing to rectify the situation, which may reach up to replacing the service with another one.

Finally, we discuss a modeling approach which gives the architect a view with simple UML notations. Figure 1 had shown an architecture modeling process by Muñoz and Pelechano [9] which allows the architect to have three models namely the Binding Providers Model, the Component Structure Model, and the Functional Model. The Binding Providers Model shows a set of devices or software systems that provide similar functionality without referring to the manufacturer specification (Figure 9). The Component Structure Model shows the objects that will build the system. For example there could be 3 lamps and a single Fluorescent Panel for building a lightening system. The Functional Specification Model describes the interaction of objects described in the Component Structure Model.

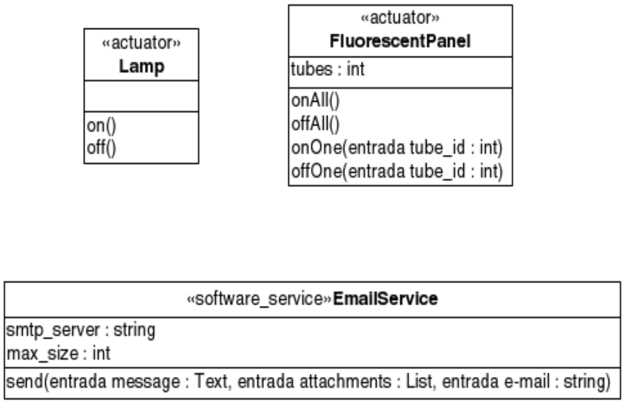

«actuator»
Lamp
on()
off()
«actuator»
FluorescentPanel
tubes : int
onAll()
offAll()
onOne(entrada tube_id : int)
offOne(entrada tube_id : int)
«software_service»EmailService
smtp_server : string
max_size : int
send(entrada message : Text, entrada attachments : List, entrada e-mail : string)

Figure 9 Some Elements of a Bindings Providers Model [9]

Researchers are aware that by resolving context-awareness and quality of service issues, they achieve a major step forward in providing a better pervasive system. Presented ideas and concepts are considered innovative and promising. All surveyed papers in this section depended on architecture approaches to resolve these design issues in addition to mathematical solutions. It is important to mention that advancement in hardware technologies will lead to better architecture solutions as well.

Patterns in Pervasive Computing

Design Patterns were first introduced in architecture engineering. Alexender [25] in 1979 introduced the concept in his book, *The Timeless Way of Building.* He defines a pattern as "'Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice" [25]. Although he wrote his book for architecture engineering, yet it became clearer that its effect was found useful in software engineering as well [25].

Later, in 1987, *Kent Beck* and *Ward Cunningham* published a technical paper describing how they used Alexender's concepts of patterns to accelerate the development of a user interface in one of their projects [26]. Patterns became then more popular when the "*Design Patterns Elements of Reusable Object-Oriented Software"* book was published by the Gang of four [27].

It is really difficult to capture a design pattern. Although, novel designs could be created from scratch, a design pattern has to come from experiencing a design and proving that it is worth using with other projects. A novel piece of design could be very successful in one application but it may fail in another. So, a design pattern will not be captured unless it is used in more than one project inside the same domain or other domains. These patterns need to be documented for future use [25].

More researchers contributed in the pervasive computing field to identify suitable and usable patterns. This area still needs more attention from researchers, since their contribution will help in simplifying development of solutions for complex environments by nature. We describe a pervasive computing environment as complex as it inherits this complexity from being distributed, and depends on non-permanent resources. There are various research papers that discuss patterns and their use in the pervasive computing field.

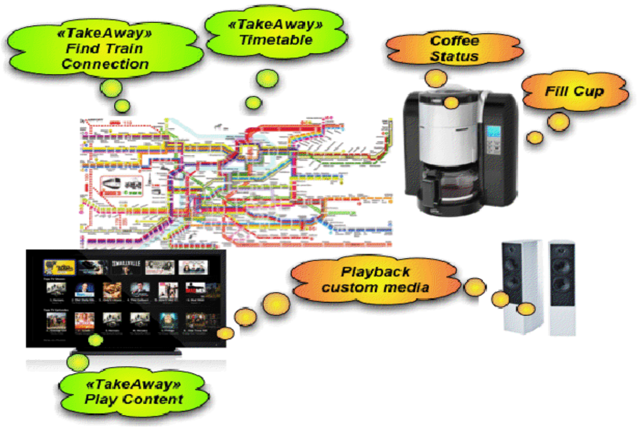


Figure 10: In this example, services with the takeaway-attribute are connected to a metro plan and a TV screen. The coffee-machine and speakers providing playback services but only allow direct interaction [28].

René Reiners [28] in his research work paves the way towards a pattern language for pervasive computing. This paper addresses the main design principles towards defining pattern and anti-design patterns for pervasive computing solutions. René Reiners gives definitions for the *Smart Object*, *Smart Service*, *Smart Environment*, and *Take-away* feature [28]. A *Smart Object* is defined as any object or device that is augmented with additional computational behavior to the object or device main purpose [28]. A *Smart Service* could be any computing service augmented to the physical object ranging from simple informative services to sophisticated applications [28].

In addition to the above explanation, objects can provide a take-away facility that can be available for smart services (Figure 10). This feature allows the offline collection of information for further retrieval and processing. However, the author highlights the risk of dealing with such a feature when working with appliances [28]. Finally, the author gives a definition of *a Smart Environment* which is a setup of arbitrary kinds of services attached with an arbitrary number of real-world objects. A *Smart Environment* can be broken down into sub categories to reflect the purpose of the provided services [28].

Kostakos et al. **[12]** described a conceptual framework for designing and analyzing a pervasive system and identified two patterns called *Insulating technology* and *Secrets revealed* out of their work. The *Insulating technology* pattern describes the use of technology that separates a user from his physical environment. This separation may be desirable or undesirable based on the user's context and activity. The designer must identify the system patterns where insulating technologies are appropriate and if not defined then it means that there is no individual or group privacy. The *Secrets revealed* pattern indicates situations at which private or social information is made public. This pattern may or may not be appropriate based on the user's context and activity. The designer must however, understand the situations that can make this pattern desirable.

Other researchers gave an overview for HCI (Human Computer Interaction) patterns in pervasive computing. Wilde et al. [29] show concrete examples and references for patterns that could be used for mobile phone applications. Researchers could not however introduce a single pattern for other areas like *Smart Environment*, and *Collaborative work*, which are considered, with the mobile phones, all the categories of patterns in pervasive computing, as recommended by the authors. They give real-world applications which were used in both the Smart Environment and Collaborative work categories [29]**.**

Sauter et al. [30] introduce an extension to the MVC design pattern towards a task-oriented development approach. They do this by extending the *Service to Worker* design pattern which adapts the MVC approach. The *Service to Worker* pattern tries to separate the business logic from the user's interaction with the implementation for web applications. They focus mainly on developing the required logic as separate from the design of the view according to the target device [30]. This approach handles the displayed attributes, style and actions performed to achieve the required task at the end. It is important to point out that this research has a concrete implementation in J2EE with mobile phones.

One can notice that researchers in pervasive computing did not introduce complete pattern languages in many categories. There are of course pattern languages inherited from other domain areas, which suit pervasive computing, but the additional characteristics of pervasive computing need to enrich this literature as well. An explanation for this limitation although the concept was introduced in the 90, is due to the lack of diversified applications that utilize all the pervasive computing theories. Pervasive computing requires more open mobile smart objects and services. Openness will allow for more applications, and hence more patterns.

Conclusion

In this survey paper we presented a variety of concepts, ideas, techniques, and practices in a number of pervasive domain areas which we believe can be very helpful for business analysts. The paper focused on associated requirements engineering and design research work. A business analyst may be able to use this knowledge to build his/her business or technical analysis models.

Pervasive computation has been a wide and attractive field for many researchers ever since Weiser dreamed about it. Researchers have been working on all technical aspects that can lead to its prosperity. However, it is notable that most of the researchers concentrated their work on the technical aspects of this domain field, trying to resolve some of its major design issues like context-awareness. A few researchers focused on areas like requirements engineering and patterns, in addition to other non-surveyed areas (e.g business process engineering).

An explanation for that trend may be because this field is still considered new and needs researchers to prove that it's worth the investment. There are, of course, many implementations for pervasive systems. But, they are still on limited domains and most of them are specific to manufacturers' technologies. This area needs more corporate organizations to invest in it, and if this happens, then the researchers will have a breadth of research opportunities on all different software engineering processes and practices. It is important to recall here, that the most successful research is the one driven by the business need.

Future research work should focus more on developing context-awareness related practices on all levels. Lyubov et al. [7] think that there is a need for requirements engineering methods focusing on context-awareness characteristics. It is also needed to have design pattern languages for pervasive computing [28]. Moreover, researchers need to keep working on resolving critical design issues that hinder service quality. Along the way, there is a need for new design approaches to provide better context-awareness solutions.

REFERENCES

[1] Mark Weiser. The Computer for the 21st Century. *In Scientific American*, (265) 3: 66--75, Year 1991

[2] M. Satyanarayanan. Pervasive Computing: Vision and Challenges. *In IEEE Personal Communications*, (8) 4: 10-17, IEEE, Year 2001.

[3] Kaptelinin, Victor and Kuutti, Kari and Bannon, Liam. Activity theory: Basic concepts and applications . *Human-Computer Interaction*. editor(s) Blumenthal, Brad and Gornostaev, Juri and Unger, Claus. Lecture Notes in Computer Science, (1015) 189--201, Springer-Verlag, Berlin/Heidelberg, Year 1995.

[4] Surendra Sharma. 2013. Embedded Systems -- A Security Paradigm for Pervasive Computing. In *Proceedings of the 2013 International Conference on Communication Systems and Network Technologies* (CSNT '13). IEEE Computer Society, Washington, DC, USA, 472-477.

[5] Requirements Management and Definition. *http://www.ibm.com*

[6] A Guide to the Business Analysis Body of Knowledge, Release 1.6. *International Institute of Business Analysis* (2006). http://www.theiiba.org

[7] Lyubov Kolos-Mazuryk, Gert-Jan Poulisse, and Pascal van Eck. Requirements Engineering for Pervasive Services. In *Workshop on Building Software for Pervasive Computing*, OOPSLA 2005.

[8] Ahmad Hassan Afridi, Saleem Gul. Method Assisted Requirements Elicitation for Context Aware Computing for the Field Force. In *Proceedings of the International MultiConference of Engineers and Computer Scientists* 2008.

[9] Javier Muñoz and Vicente Pelechano. 2005. Building a software factory for pervasive systems development. In *Proceedings of the 17th international conference on Advanced Information Systems Engineering* (CAiSE'05), Oscar Pastor and João Falcão e Cunha (Eds.). Springer-Verlag, Berlin, Heidelberg, 342-356.

[10] Stéphane Lo Presti, Michael Butler, Michael Leuschel, and Chris Booth. 2005. A trust analysis methodology for pervasive computing systems. In *Trusting Agents for Trusting Electronic Societies*, Rino Falcone, Suzanne Barber, Jordi Sabater-Mir, and Munindar P. Singh (Eds.). Springer-Verlag, Berlin, Heidelberg 129-143.

[11] Hen-I Yang and Abdelsalam Helal. 2008. Safety Enhancing Mechanisms for Pervasive Computing Systems in Intelligent Environments. In *Proceedings of the 2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications* (PERCOM '08). IEEE Computer Society, Washington, DC, USA, 525-530.

[12] Vassilis Kostakos, Eamonn O'Neill, and Alan Penn. 2006. Designing Urban Pervasive Systems. *Computer* 39, 9 (September 2006), 52-59.

[13] George Coulouris, Jean Dollimore, Tim Kindberg. Distributed Systems Concepts and Design. Fifth Edition. *Addison-Wesley Publishing Company* (2012).

[14] James McGovern, Scott W. Ambler, Michael E. Stevens, James Linn, Vikas Sharan, Elias K. Jo. A Practical guide to enterprise architecture. *Prentice Hall Professional Technical Reference* (2004).

[15] Ian Sommerville. Software Engineering. Ninth Edition. *Addison-Wesley Publishing Company* (2011).

[16] Ruimin Liu, Feng Chen, Hongji Yang, William C. Chu, and Yu-Bin Lai. 2004. Agent-Based Web Services Evolution for Pervasive Computing. In *Proceedings of the 11th Asia-Pacific Software Engineering Conference* (APSEC '04). IEEE Computer Society, Washington, DC, USA, 726-731.

[17] N. A. B. Gray. Comparison of web services, java-RMI, and CORBA service implementations. In *Fifth Australasian Workshop on Software and System Architectures. in Conjunction with ASWEC*, 2004.

[18] Anand Ranganathan and Scott McFaddin. 2004. Using Workflows to Coordinate Web Services in Pervasive Computing Environments. In *Proceedings of the IEEE*

International Conference on Web Services (ICWS '04). IEEE Computer Society, Washington, DC, USA, 288-.

[19] Karthik Harihar and Stan Kurkovsky. 2005. Using Jini to enable pervasive computing environments. In *Proceedings of the 43rd annual Southeast regional conference - Volume 1 (ACM-SE 43)*, Vol. 1. ACM, New York, NY, USA, 188-193.

[20] Jakob E. Bardram. 2005. The java context awareness framework (JCAF) – a service infrastructure and programming framework for context-aware applications. In *Proceedings of the Third international conference on Pervasive Computing (PERVASIVE'05)*, Hans-W. Gellersen, Roy Want, and Albrecht Schmidt (Eds.). Springer-Verlag, Berlin, Heidelberg, 98-115.

[21] Panu Korpipaa, Jani Mantyjarvi, Juha Kela, Heikki Keranen, and Esko-Juhani Malm. 2003. Managing Context Information in Mobile Devices. *IEEE Pervasive Computing* 2, 3 (July 2003), 42-51.

[22] K. Kalapriya , S. K. Nandy , Deepti Srinivasan , R. Uma Maheshwari , V. Satish, A framework for resource discovery in pervasive computing for mobile aware task execution, *Proceedings of the 1st conference on Computing frontiers*, April 14-16, 2004, Ischia, Italy.

[23] Petrus, Sharmila Ravula, "Exploring the Role of Software Architecture in Dynamic and Fault Tolerant Pervasive Systems," sepcase, pp.9, *First International Workshop on Software Engineering for Pervasive Computing Applications, Systems, and Environments* (SEPCASE '07), 2007.

[24] Hafez, Dina and Aly, Sherif G. and Sameh, Ahmed. A Context and Service-Oriented Architecture with Adaptive Quality of Service Support. . In *I. J. Comput. Appl*., (18) 1: 37-51, Year 2011.

[25] Osama M. Khaled, Capturing Design Patterns for Performance Issues in Database-Driven Web Applications. *M.Sc. Thesis, Computer Science Department, the American University in Cairo*, 2004.

[26] Beck, Kent and Cunningham, Ward. Using Pattern Languages for Object Oriented Programs. *Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*. Year 1987.

[27] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides .Design Patterns, Elements of Reusable Object-Oriented Software. *Addison-Wesley Professional* (1994)

[28] René Reiners. Towards a Common Pattern Language for Ubicomp Application Design. PATTERNS 2010, The *Second International Conferences on Pervasive Patterns and Applications* (2010).

[37] Leila Takayama, Caroline Pantofaru, David Robson, Bianca Soto, and Michael Barry. 2012. Making technology homey: finding sources of satisfaction and meaning in home automation. *In Proceedings of the*

[29] Wilde, Adriana G, Bruegger, Pascal and Hirsbrunner, Béat. An Overview of Human-Computer Interaction Patterns in Pervasive Systems. In *Conference i-USER 2010, IEEE*, University Teknologi Mara, Shah Alam - Malaysia, December (2010).

[30] Patrick Sauter, Gabriel Vögler, Günther Specht, and Thomas Flor. Extending the MVC Design Pattern towards a Task-Oriented Development Approach for Pervasive Computing Applications. In *Proceedings of International. Conference on Architecture of Computing Systems - Organic and Pervasive Computing* (ARCS 2004), Augsburg, 23.-26. March 2004, Spinger-Verlag, LNCS 2981, 2004, pp. 309-321.

[31] Belkacem Chikhaoui, Shengrui Wang, and Hélène Pigot. 2012. Towards causal models for building behavioral user profile in ubiquitous computing applications. *In Proceedings of the 2012 ACM Conference on Ubiquitous Computing (UbiComp '12)*. ACM, New York, NY, USA, 598-599.

[32] Fahim Kawsar and A.J. Bernheim Brush. 2013. Home computing unplugged: why, where and when people use different connected devices at home. *In Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing (UbiComp '13)*. ACM, New York, NY, USA, 627-636.

[33] Erik Grönvall and Nervo Verdezoto. 2013. Beyond self-monitoring: understanding non-functional aspects of home-based healthcare technology. *In Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing (UbiComp '13)*. ACM, New York, NY, USA, 587-596.

[34] Lei Tian, Shaosong Li, Junho Ahn, David Chu, Richard Han, Qin Lv, and Shivakant Mishra. 2013. Understanding user behavior at scale in a mobile video chat application. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing (UbiComp '13)*. ACM, New York, NY, USA, 647-656.

[35] Scott Mainwaring, Wendy March, and Bill Maurer. 2008. From meiwaku to tokushita!: lessons for digital money design from japan. *In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. ACM, New York, NY, USA, 21-24.

[36] Jialiu Lin, Shahriyar Amini, Jason I. Hong, Norman Sadeh, Janne Lindqvist, and Joy Zhang. 2012. Expectation and purpose: understanding users' mental models of mobile app privacy through crowdsourcing. *In Proceedings of the 2012 ACM Conference on Ubiquitous Computing (UbiComp '12)*. ACM, New York, NY, USA, 501-510.

2012 ACM Conference on Ubiquitous Computing (UbiComp '12). ACM, New York, NY, USA, 511-520.

[38] Waltenegus Dargie, Juha Plosila, and Vincenzo De Florio. 2012. Existing challenges and new opportunities in context-aware systems. *In Proceedings of the 2012 ACM Conference on Ubiquitous Computing (UbiComp '12)*. ACM, New York, NY, USA, 749-751.

[39] Francisca Pérez and Pedro Valderas. 2009. Allowing End-Users to Actively Participate within the Elicitation of Pervasive System Requirements through Immediate Visualization. In *Proceedings of the 2009 Fourth International Workshop on Requirements Engineering Visualization (REV '09)*. IEEE Computer Society, Washington, DC, USA, 31-40.

Java Core API Migration: Challenges and Techniques

Victor L. Winter[1], **Jonathan Guerrero**[2], **Carl Reinke**[3], **and James T. Perry**[4]
[1,2,3]Department of Computer Science, University of Nebraska at Omaha, Omaha, NE, USA
[4]Sandia National Laboratories, Albuquerque, NM, USA

Abstract— *The task of developing Java-based applications for embedded systems can be greatly enhanced by providing developers access to Java's Core APIs such as* `java.lang` *and* `java.util`*. Oftentimes, platforms used in embedded systems are scaled back versions of the JVM. As a result, Core APIs must be* migrated *in order to be compatible with a particular platform. A significant portion of such migration centers around the removal of field, method, or constructor declarations.*

This paper describes the challenges and techniques associated with the automated removal-based modification of Java source code. Driving this research is the need to migrate selected Java Core APIs to an embedded platform called the SCore processor. This migration is being performed using a tool called $\mathcal{M}onarch$.

Keywords: source-code analysis, program transformation, code migration

1. Overview

$\mathcal{M}onarch$ is a Java source-code migration tool being developed at the University of Nebraska at Omaha to assist in migrating Java Core APIs to the *SCore platform*, a hardware implementation of the JVM [1] being designed at Sandia National Laboratories for use in embedded systems. The SCore is not a full-blown JVM and places various restrictions on the class files it can execute. For example, the SCore does not support floating point arithmetic. Therefore, the compilation of migrated code may not contain any floating point bytecodes (a more detailed discussion of the SCore is given in Section 2). In a nutshell, the central problem that $\mathcal{M}onarch$ must confront is how to produce a (migrated) code base suitable for execution on the SCore.

At this time, the primary code base targeted for migration is a set of Core APIs belonging to the Standard Edition (SE) of the Java Platform. Specifically, a subset of Java SE 6 update 18 consisting of compilation units drawn from `java.io`, `java.lang`, and `java.util`.

In its totality, $\mathcal{M}onarch$ migration is comprised of the following three stages.

a) Re-implementation Stage: This manual stage involves the re-implementation of "must have" functionality within the Core APIs in order to remove unwanted dependencies. Re-implemented code fragments are encoded as program transformations which can then be automatically applied (or replayed). Further discussion of re-implementation lies beyond the scope of this paper. In this paper, we assume we are working with a target code base for which the re-implementation stage associated with migration has been completed.

b) Preparation Stage: This manual stage involves an expansion of the target code base with the goal of obtaining a *prepared code base* whose properties satisfy the preconditions of $\mathcal{M}onarch$'s static analysis system. Preparation is necessary to assure the correctness of static analysis. The specifics of the preparation stage also lie beyond the scope of this paper.

c) Removal Stage: This fully automated stage, which is the focus of this paper, consists of the application of program transformations expressed as *conditional rewrite rules*. These transformations remove field, methods and constructor declarations having dependencies on features not supported by the SCore platform. A novel feature of $\mathcal{M}onarch$ transformations is that they enable the conditional portions of rewrite rules to include nontrivial semantic properties (e.g., resolution of references and dependency analysis).

In this paper, the term *resolution analysis* is used to refer to static analysis whose purpose is to determine the relation between *references* to types and type members (e.g., fields, methods, and constructors) and their *declarations*. Resolution analysis is central to the removal stage of migration and our discussion assumes that the source-code analysis capabilities of $\mathcal{M}onarch$ can correctly perform resolution analysis [2] for target code bases that have been suitably *prepared*.

d) Contribution: This paper focuses on identifying and addressing correctness challenges arising from removal-based migration. The discussion and analysis takes into account features and properties of Java such as upcasting, shadowing, overriding, and overloading. Also included in the analysis are constraints imposed by the SCore.

e) Outline: The remainder of this paper is as follows. Section 2 gives an overview of the SCore platform. Section 3 describes resolution analysis. Section 4 justifies and articulates a *migration policy* governing the removal of type members so that resolution analysis of the pre- and post-migration versions of the targeted code base is consistent. This consistency is sufficient to assure that the compilation of migrated code, by the Java compiler, is functionally correct and an informal argument is made to this effect. Section 5 takes an in-depth look at research activities having an intersection with the problem discussed in this paper, and Section 6 concludes.

2. Background: The Scalable Core Platform

The Scalable Core (SCore) platform [3], [4] is a hardware implementation of the JVM [1] being designed at Sandia National Laboratories for use in resource-constrained embedded applications.

2.1 Software Development

From the perspective of *process*, developing code for the SCore is essentially identical to developing code for the JVM. Programmers can develop and debug programs on a desktop using an IDE such as Eclipse or Netbeans. Standard tools such as unit testers can be used to validate aspects of the software. It is important to mention that at this stage of development, the application interacts with the original (i.e., un-migrated) Core APIs such as `java.lang` and `java.util`. After this initial stage, SCore development is moved to a simulation environment where the application interacts with the migrated Core APIs. See [5] for a detailed discussion of the simulation environment.

2.2 $\mathcal{I}nterlude$: The SCore Classloader

A SCore application is a Java program that is compiled using a standard Java compiler. Migrated Core APIs are also compiled in this fashion. The resulting class files are then processed, as shown in Figure 1, by a classloader-like converter called $\mathcal{I}nterlude$[1] which combines all class files into a single significantly reduced file format called a *ROM image*. It is this ROM image that is executed by the SCore platform.

$\mathcal{I}nterlude$ assumes the class files given to it are produced by trusted Java compilers. This assumption is an entailment of the development process for SCore applications which require all class files to be produced in-house from source code. Due to this assumption, general byte-code verification is unnecessary. However, $\mathcal{I}nterlude$ does verify that the class files it processes satisfy a specific set of properties. Among other things, $\mathcal{I}nterlude$ will fail to produce a ROM image upon encountering a symbolic reference to a non-existent field, method, constructor, type or package. The fact that $\mathcal{I}nterlude$ performs these important checks is relied upon by $\mathcal{M}onarch$ (see Section 4.1).

[1] $\mathcal{I}nterlude$ is not the official name of the converter used by the SCore development team. It is a term introduced by the authors to more concisely reference the part of the SCore tool set that performs classfile conversion.

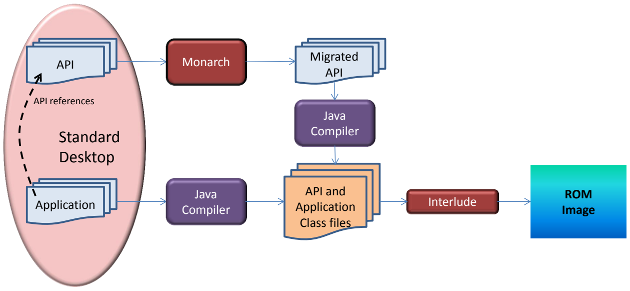


Fig. 1: An overview of SCore application development.

3. Core Analysis

The goal of the removal stage of $\mathcal{M}onarch$ migration is broadly stated as follows.

> *Migration Policy 1:* From the targeted code base, remove all fields, methods, and constructors having direct or indirect dependencies on (1) features that are not supported by the SCore, or (2) external references.

The removal stage is fully automated, transformation-based, and constitutes the heart of $\mathcal{M}onarch$ migration. For the remainder of this paper, we will use the term *migration* when referring to the removal stage of $\mathcal{M}onarch$ migration.

It is important to know that, for assurance purposes (e.g., to facilitate manual code review and traceability of migration), $\mathcal{M}onarch$ operates exclusively on Java source code. In particular, $\mathcal{M}onarch$ does not extend its resolution analysis to class files or jar files.

3.1 Resolution Analysis

We define a *reference* as a source-code expression (i.e., valid Java syntax) referring to a declared element. A reference is the mechanism by which types, arrays, fields, methods, and constructors can be denoted within Java source code. Such denotations can be in relative terms, in indirect terms (also known as aliases), and in absolute terms (also known as canonical forms).

On a conceptual level, a *reference* can be modeled as a (dot-separated) *sequence* consisting of one or more *atoms*, where an atom is either (1) a simple identifier denoting a package, type, generic type parameter, or field, or (2) an array reference, a method signature, or a constructor signature. This understanding of references as *sequences of atoms* leads to an approach to resolution analysis that is incremental in nature and atom-based: Atom sequences are resolved one atom at a time from left to right. We write $ref_{1..n}$ to denote a reference consisting of n atoms.

In this paper, the term *resolution analysis* is used to refer to static analysis whose purpose is to determine the *relation* $\mathcal{R}$ between element *references* and element *declarations*. We use the term *resolvent* to refer to the result (i.e., the value) produced when resolution analysis is applied to a reference occurring in a given environment. Let *ref* denote a reference and let T denote a canonical name of the type in which *ref* occurs. We formally express the resolution of (T, ref) as follows.

$$resolution(T, \mathit{ref}) = resolvent \qquad (1)$$

Note that in this paper, a resolvent is considered to be the canonical name of a type, array, field, method, or constructor.

3.1.1 Local Variables and Generic Type Parameters

When viewed in its entirety, resolution analysis must encompass references to local variables as well as generic type parameters. $\mathcal{M}onarch$ performs such analysis during its migration. However, the goal of this paper is to create a source-code *removal policy* that will correctly remove declarations and all their corresponding references. Furthermore, the stated policy does not permit the isolated removal of local variables (e.g., from the bodies of methods) or generic type parameters. The only way local variables and generic type parameters could be removed during migration would be through the removal of their enclosing declarations (e.g., method declarations). It is precisely these enclosing declarations that define the scopes of local variables and generic type parameters. For this reason, it is without loss of generality that we can restrict our discussion of resolution analysis to environments (T) denoting types. In particular, we abstract away from our discussion the environments associated with methods and constructors.

3.2 Primary versus Secondary Resolvents

For the purposes of our analysis we will, when necessary, use the terms *primary resolvent* and *secondary resolvent* to make finer distinctions between resolvents. Such distinctions are relevant because the complexity of Java allows for the creation of code structures in which certain declarations are *hidden* (i.e., not visible) from the environment (T) in which the reference occurs. When they exist, we refer to such hidden declarations as *secondary resolvents* of the reference. Declarations that are not hidden are called *primary resolvents*.

The Java compiler performs resolution analysis to determine reference bindings. $\mathcal{M}onarch$ performs resolution analysis to perform dependency analysis used to determine which members of a type can be safely migrated, and which members must be removed.

A central concern is whether the migration process, when seen as a whole, creates conditions for the re-classification, by the Java compiler, of a secondary resolvent as the (new) primary resolvent for a given reference. If this shift occurs, then migration is *not correctness preserving*. It is the omission of declarations that gives rise to this threat. The omission of a declaration can occur in one of two ways.

1) **Pre-migration**: The consideration of code bases for which there is tacit omission of some declarations (e.g., targeting a subset of an API for migration). This situation is addressed during the preparation stage.
2) **Post-migration**: The explicit removal of declarations during migration.

3.3 Points-to Analysis

In theory, cases exist where resolution analysis is unable to precisely determine the primary resolvent of a reference. This particular problem is known as the *points-to* problem, and its static analysis is undecidable.

The undecidable nature of points-to analysis implies that an approximating analysis must be used to prevent errors associated with the potential re-classification of secondary resolvents. This approximation must be conservative with respect to correctness. From a technical standpoint, this means that cases can arise where $\mathcal{M}onarch$ may need to remove both primary and secondary resolvents in order to assure migration correctness. The drawback of such an aggressive removal policy is that it results in a (possibly unnecessary) reduction in the functionality that is migrated.

3.4 Unresolvable References

At the source code level, brute-force attempts to expand a target code base with the goal of obtaining a code base that is *reference-closed* are impractical. For example, a simple "hello world" program when executed via the command `java -verbose:class` demonstrates that this tiny program loads (e.g., has dependencies on) over 400 classes. Thus, when combined with the space limitations of the SCore, $\mathcal{M}onarch$'s restriction to source-code level analysis gives rise to target code bases containing *unresolvable references*. We denote such resolvents by the constant `<unresolved>`.

When resolving $(T, \mathit{ref}_{1..n})$, if $\mathit{ref}_{1..n}$ has a proper prefix that lies outside of the target code base C, then $(T, \mathit{ref}_{1..n})$ is classified as an *external reference*.

$$\begin{aligned} &\mathit{externalReference}(C, (T, \mathit{ref}_{1..n})) \overset{def}{=} \qquad (2) \\ &\quad \exists i: \quad 1 \le i < n \\ &\qquad\qquad \wedge \\ &\qquad\qquad resolution(T, \mathit{ref}_{1..i}) = resolvent \\ &\qquad\qquad \wedge \\ &\qquad\qquad resolvent \notin C \end{aligned}$$

From the perspective of migration, if an element in C has a dependency on an external reference, the element must be removed during migration. Furthermore, for the purposes of dependency analysis, to conclude that a reference $(T, \mathit{ref}_{1..n})$ is `<unresolved>` it is sufficient to find a prefix $\mathit{ref}_{1..i}$ that is an external reference.

4. Removal Analysis

This section identifies and articulates changes to the basic migration policy, stated in Section 3, with the goal of producing an amended migration policy that is both implementable and correctness preserving. The central challenge here is to articulate a policy governing removal in such a way that the relation $\mathcal{R}$ between references and their primary resolvents is preserved in the migrated code.

Definition 1: Let C denote a target code base containing no unresolvable references (i.e., a compilable code base) and let C' denote the code base that results when C is migrated using $\mathcal{M}onarch$. Let $\mathcal{R}_C$ and $\mathcal{R}_{C'}$ denote the relations between references and their resolvents that exist in the code bases C and C' respectively. In order to be correct, removal-based migration must satisfy the following property:

$$migrationCorrect(C, C') \overset{def}{=} \mathcal{R}_{C'} \subseteq \mathcal{R}_C \qquad (3)$$

It is worth noting that removing *more* (e.g., removing arbitrary elements) does not compromise correctness, provided this is done consistently and comprehensively. For example, the relation for an empty code base C' is $\mathcal{R}_{C'} = \emptyset$, and in this case $migrationCorrect(C, C')$ will hold for all C. This observation gives rise to an important secondary migration goal (whose discussion lies outside the scope of this paper). Namely, that migration should strive to maximize the amount of code migrated. This is after all, in part, what is being addressed in the re-implementation stage mentioned in Section 1.

> *Policy Constraint 1:* In order to assure migration correctness, $\mathcal{M}onarch$ may remove arbitrary declarations during migration. However, such removal should be kept to a minimum.

4.1 Application Independence

A constraint that has been placed upon $\mathcal{M}onarch$ is that API migration be *application independent*. For example, let $\mathcal{C}$ denote a code base (e.g., a subset of the Java libraries) that has been targeted for migration, and let *App* denote an application code base that uses the functionality provided by $\mathcal{C}$. Even though the analysis of *App* may result in the migration of a larger portion of C, $\mathcal{M}onarch$ may not broaden it static analysis to include *App*. We refer to this constraint as *application independence*.

There are technical as well as security related reasons justifying the application independence of API migration. On the technical side, if $\mathcal{M}onarch$ migration was application dependent, then $\mathcal{M}onarch$ would need to be intimately integrated with *all* (future) application development for the SCore. From the perspective of security, if an application is proprietary, then the confidentiality policy of an organization may prohibit "externally developed" tools such as $\mathcal{M}onarch$ from accessing the application.

> *Policy Constraint 2:* Migration of a targeted code base $\mathcal{C}$ must be performed in an application independent fashion. Specifically, migration may not make any assumptions about how an application might use $\mathcal{C}$.

As mentioned in Section 2.1, application development for the SCore begins on a desktop environment utilizing a traditional JVM and only later transitions to the SCore platform. The application is also compiled with respect to this desktop environment. Migration must assure that applications developed and compiled on a standard JVM ultimately interact with migrated and *separately* compiled target code bases in a correct manner. From the perspective of migration, this approach to application development and separate compilation entails an additional challenge which is addressed by placing the following constraint on $\mathcal{I}nterlude$.

> *Policy Constraint 3:* The $\mathcal{I}nterlude$ classloader should fail to create a ROM image when the class files of an **application** contain a reference to a field, method or constructor that does not exist in the migrated target code base.

4.2 Shadowing

The following subsections examine the extent to which shadowing issues must be considered during migration.

4.2.1 Field Shadowing

In Java, it is possible for a subtype to (re)declare a field declared in its supertype. A re-declaration of a field x in a subtype T_2 (i.e., $T_2.x$) is said to *shadow* the declaration of the field x in its supertype T_1 (i.e., $T_1.x$). We also say the declaration $T_1.x$ is *shadowed* by the declaration in $T_2.x$, and that the declaration $T_1.x$ is *shadowing* the declaration $T_2.x$.

The semantics of field inheritance poses a threat to the correctness of field removal. For example, let (T, ref) denote a reference whose resolvent denotes the field re-declaration. In this case, the corresponding field declaration in the supertype constitutes a secondary resolvent. Thus, if the field re-declaration is removed, then the secondary resolvent will be re-classified as the primary resolvent for (T, ref). An example of this is shown in Figure 2.

> *Policy Constraint 4:* In order to assure the correctness of migration, when removing the field declaration $T.x$ it is also necessary to remove all field declarations shadowed by $T.x$.

4.2.2 Method Shadowing

Method shadowing is a phenomenon that only occurs between a nested class and its enclosing class(es). All

```
package p1; // Target code base
public class A {
    int y1 = new B2().x1;       // p1.B2.x1
    int y2 = (int) new B2().x2; // p1.B2.x2
    int y3 = (int) new B2().x3; // p1.B1.x3
}
class B1 {
    int x1 = 1;
    int x2 = 2;
    int x3 = 3;
}
class B2 extends B1 {
    // will be removed by Monarch, shadows p1.B1.x1
    int x1 = (int) 1.0;
    // will be removed by Monarch, shadows p1.B1.x2
    double x2 = 2.0;
}
```

Fig. 2: Field shadowing can threaten migration correctness.

references to shadowing methods must occur within the nested method where the shadowing method is declared.

Figure 3 gives an example of Java code in which the method declaration `p1.A.f()` is shadowed. This declaration is shadowed in the context of the class `InnerA`. In this case, the shadowing method declaration is `p1.A.InnerA.f()`.

```
package p1; // Target code base
public class A {
  public int f() { return 1; }
  public int g() { return 1; }
  public class InnerA {
    // shadowing method removed during migration
    public int f() { return (int) 2.0; }

    // depends on p1.A.InnerA.f()
        public int y1 = f();
        public int y2 = g();
        public int y3 = z1;
  }
  public int z1 = 1;
  InnerA myThing = new InnerA();
  // int z2 = myThing.g(); // compile error
  // int z3 = myThing.z1;  // compile error
}
// =========================================
// External to Monarch analysis
package app;
import p1.A;
class App {
        A.InnerA myThing = (new A()).new InnerA();
        int x1 = myThing.f();
        // int x2 = myThing.g(); // compile error
        int y1 = myThing.y1;
        int y2 = myThing.y2;
        // int y3 = myThing.z1; // compile error
}
```

Fig. 3: An example of method shadowing.

In order to assure correct dependency analysis involving method shadowing it is necessary for a target code base to satisfy the *containment property* defined as follows.

Property 1: A target code base C satisfies the **containment property** if whenever a top-level class belongs to C, all its internals (e.g., nested classes) also belong to C.

The containment property is a natural property to assume for a targeted code base. Validating that a target code base satisfies this property is part of the preparation stage of migration (which lies outside the scope of this paper). We explicitly mention containment property here for the sake of completeness, but will tacitly assume this property for the remainder of the paper. As a result, secondary-resolvent issues surrounding method shadows do not extend into the application. Thus, $\mathcal{I}nterlude$'s checks are sufficient to assure that method shadowing poses no additional challenges to migration, and therefore no specific policy need be developed to handle method shadowing (other than requiring the target code base to adhere to the containment assumption).

Policy Constraint 5: In order to be suitable for migration, a target code base C must satisfy the **containment assumption**.

4.3 Method Overriding

```
package p1; // Target code base
public class A extends B {
    // overrides p1.B.f()
    public int f() { return 1; }
    // overrides p1.B.g()
    // will be removed during migration
    public int g() { return (int)1.0; }
}
public class B {
    // overridden by p1.A.f()
    // will be removed during migration
    public int f() { return (int)2.0; }
    // overridden by p1.A.f()
    public int g() { return 2; }
}
// ==========================================
// External to Monarch analysis
package app;
import p1.*;
public class App {
    A myA = new A();
    int x1 = myA.f(); // p1.A.f()
    int x2 = myA.g(); // p1.A.g()
    B myB = new B();
    int x3 = myB.f(); // p1.B.f()
    int x4 = myB.g(); // p1.B.g()
    B myThing = new A();
    int x5 = myThing.f();
    int x6 = myThing.g();
}
```

Fig. 4: An example of method overriding.

Method *overriding* occurs when a type T_2 declares a method m whose signature exactly matches that of a method declared in T_1 where $T_2 <: T_1$. In this case, we say the declaration $T_2.m$ *overrides* $T_1.m$.

Figure 4 shows a small class hierarchy in which method overriding occurs. In this case, `p1.B.f()` and `p1.B.g()` are *overridden methods*, and `p1.A.f()` and `p1.A.g()` are *overriding methods*.

Policy Constraint 6: When removing method $T.m$ all methods overridden by $T.m$ must also be removed.

4.3.1 Interfaces

It should be noted that the undecidability of points-to analysis provides sufficient justification for adopting a migration policy that deletes methods whose declarations are overridden by methods utilizing features unsupported by the SCore. However, this removal policy also solves the problem of method implementation requirements imposed by interfaces. Specifically, if a class implements an interface, then the class must have declarations for all abstract methods declared in the interface. Removal of such an "interface method" from the class would result in a migrated code base that fails to compile.

4.4 Overloading

Method *overloading* occurs when methods have identical names but different signatures. A complete analysis of method overloading must take into account issues such as visibility, widening conversions, auto-boxing, and "closest fit" signature matching (which can get complex when a method signature contains multiple parameters).

From the perspective of migration, the case that must be considered is: "What happens if an application contains a reference to an overloaded method whose declaration resides in the original target code base, but whose declaration is removed as a result of migration?" In such a situation, secondary resolvents are possible. Furthermore, a particularly unfortunate form of overloading can arise involving parameters of type `java.lang.Object`. For example, Figure 5 reveals that the class `java.lang.StringBuilder` contains numerous overloaded declarations of its `append` method. Noteworthy is that (1) all declarations shown have the same arity, (2) the formal parameter of the first declaration is of type `java.lang.Object`, and (3) aside from a reference to the first declaration, a reference (in an application) to any other append declaration will have `append(Object obj)` as a secondary resolvent. The root cause of such secondary resolvents results not from overriding, but from how Java resolves references to overloaded methods.

For references occurring within the target code base, secondary resolvents arising from overloading poses no analysis problems. However, this is not the case for references to overloaded methods occurring within the application, since these are not subjected to analysis. Particularly unappealing is the idea of expanding the removal of a declaration like `append(float f)` to also include the removal of `append(Object obj)` in order to assure the correctness of reference resolution within an application. Fortunately, the properties of $\mathcal{I}nterlude$ can be leveraged to avoid this situation. Specifically, $\mathcal{I}nterlude$ will fail to compile class files containing symbolic references to non-existent methods.

```
public StringBuilder append(Object obj) ...
public StringBuilder append(String str) ...
...
public StringBuilder append(float f) ...
public StringBuilder append(double d) ...
```

Fig. 5: Overloaded declarations of the method `append` in the class `java.lang.StringBuilder`.

Policy Constraint 7: Secondary resolvents resulting from references to overloaded methods/constructors impose no additional constraints on migration because $\mathcal{I}nterlude$ will fail to produce a ROM image if it encounters a reference to a non-existent method or constructor.

4.5 Special Cases

In the following sections discuss removal cases requiring specialized analysis.

4.5.1 Constructor Removal

$\mathcal{M}onarch$ migration exclusively consists of removal of fields, methods, and constructors – and anonymous types. Removal of named types is not supported since this has the potential to decimate a code base – an example of which will be discussed shortly. However, this restriction can pose a threat to the correctness of migration in the case when the constructors of a type are exhaustively removed. In particular, when encountering a class that does not explicitly contain a constructor, the Java compiler will automatically generate a no-argument default constructor for that class.

Policy Constraint 8: Migration may not exhaustively remove all explicitly declared constructors belonging to a type.

We encountered the problem of exhaustive constructor removal in practice. In particular, the core API targeted for migration to the SCore containes the class `java.lang.Throwable`. Every constructor of `Throwable` had a dependency on a native method called `fillInStackTrace`. This native method is not supported on the SCore platform. Thus, the constructors for the class `Throwable` needed to be either exhaustively removed or or some subset needed to be re-implemented. Exhaustive removal of the `Throwable` constructors yielded unacceptable results since it mandated the (manual) removal of the class `Throwable`. This lead to a cascading sequence of removals that decimated the migrated code base. Specifically, since `Throwable` is the supertype of all `Exception` and `Error` classes, and since the `Exception` and `Error` classes were part of the code base targeted for migration, the removal of `Throwable` would also require the removal of the `Exception` and `Error` classes as well as any of their subtypes.

In this case, the dependency problem was resolved during the re-implementation stage (see Section 1).

4.5.2 Initialization Blocks

A *static initialization block* is a mechanism that enables nontrivial initialization of static fields. Such an initialization capability is useful since class initialization methods (i.e., `clinit`) cannot be explicitly defined. In Java 1.1, anonymous classes were introduced and *instance initialization blocks* were introduced to compensate for the fact that constructors may not be defined for anonymous classes.

When a static initialization block is present in a class, it is implicitly associated with the class initialization method for that class. Similarly, all instance initialization blocks in a class are associated with *all* constructors for that class. This makes the removal of initialization blocks problematic. For example, removal of an instance initialization block implies removal of all constructors for the class. Therefore, $\mathcal{M}onarch$ classifies initialization blocks as "must have" functionality whose dependencies must be addressed in the manual re-implementation stage (see Section 1).

In practice, the use of instance initialization blocks is rare and the use of static initialization blocks is infrequent. In the Core API targeted for migration there are 7 static initialization blocks with average LOC = 6, and 0 instance initialization blocks.

> *Policy Constraint 9:* The removal phase of migration classifies initialization blocks as "must have" functionality. Furthermore, $\mathcal{M}onarch$ will migrate all initialization blocks without performing dependency analysis. It is left to the re-implementation phase to assure that all initialization blocks are free from unwanted dependencies.

5. Related Work

Behavior-preserving refactoring [6] and removal-based migration both center around modifications to source code constrained by properties involving primary and secondary resolvents. Many standard refactorings involve either changing a declaration (e.g., renaming) or repositioning a declaration within a subtype hierarchy (e.g., extracting a superclass or converting a local variable to a field). For example, the *rename method* refactoring fully takes into account overriding and considers overloading issues to a limited extent. This is similar, but not identical to what is needed for $\mathcal{M}onarch$ migration. Noteworthy is that method renaming extends across the entire subtype hierarchy renaming method declarations in all sibling, cousin, and descendent classes. This is somewhat different from what is done in $\mathcal{M}onarch$ migration. Specifically, in order to maximize the number of declarations in the migrated code base, removal of a method only impacts ancestors (i.e., supertypes) of the class containing the method declaration flagged for removal and not its sibling, cousin, or descendent classes.

Refactoring has also been used to capture developers' manual modifications of libraries with the intent of replaying the captured actions on client software that uses the modified libraries [7]. In this case, a separation is made between the evolution of an API and modifications to applications using the evolved API. The goal of replay is to bring application programs "up to date" with the evolving APIs they depend on. Henkel and Diwan have developed a semi-automated refactoring capture-and-replay tool called CatchUp! for this purpose. Such replay abilities are similar to those in $\mathcal{M}onarch$ which formulate re-implementations as replayable transformations.

Refactoring with type constraints has been used to remove deprecated code [8] and convert legacy code with unchecked downcasts into type-safe generic code [9]. In a similar vein, a tool called *Rosemari* [10] has been developed to upgrade legacy applications making them compliant with evolving frameworks such as JUnit. In this case, code migration is based on annotation refactorings. For example, JUnit version 3 requires that test cases and test suites adhere to certain naming conventions. In contrast, JUnit version 4 imposes no naming conventions, but does require test cases and test suites to be properly annotated.

Our work differs from refactoring-based work related to Java libraries in that we exclusively modify the libraries not the client software that uses the libraries. In addition, whereas others focus on preserving the API exposed by the libraries, we reduce the functionality of the libraries because of restrictions of the target platform. To the best of our knowledge this has not been done elsewhere. Even Oracle does not provide a migrated version of the standard libraries for use on their Java Card platform [11].

Rayside and Kontogiannis [12] discuss a process to extract Java library subsets for supporting embedded systems applications by removing unused components from the library. They have the capability to produce library subsets having certain properties: (1) a space optimized subset, (2) a partial space optimized subset, and (3) a space reduced subset. The production of a subset is application specific with the space optimized subset being the most aggressive. The space optimized subset is created by removing all fields and methods that are not referenced by a given application. This is slightly different than the migration goals we are pursuing in which we want to universally prohibit access to fields and methods depending on features that are not supported by the target platform (i.e., the SCore). Furthermore, $\mathcal{I}nterlude$, the class loader for the SCore [13], has similar removal capabilities to the space optimized subset produced by Rayside and Kontogiannis. In particular, when processing the class files for a given application the class loader for the SCore removes all methods (but not fields) that are not referenced.

6. Conclusion

This paper conducts an in-depth analysis of the impact that declaration removal can have on Java code bases. This analysis is performed in the context of an API migration effort whose goal is to produce Java source code suitable for execution on the SCore platform.

At this time, the primary code base targeted for migration is a set of Core APIs belonging to the Standard Edition (SE) of the Java Platform. Key constraints imposed on migration are: (1) API removal analysis may not make assumptions about (SCore) applications using the migrated API, and (2) SCore application starts on a desktop environment containing the un-migrated form of the target Core APIs and finishes on a simulation environment containing the migrated version of the Core APIs.

From our experiences we conclude that Core API migration is possible and can be automated to a significant extent. A beta-version of this policy has been implemented in a migration tool called $\mathcal{M}onarch$, capable of fully automatic removal-based migration. The policy is enforced by both the $\mathcal{M}onarch$ migrator and the $\mathcal{I}nterlude$ classloader.

References

[1] T. Lindholm and F. Yellin, Eds., *The Java Virtual Machine (Second Edition)*. Addison-Wesley, 1999.

[2] V. Winter, , C. Reinke, and J. Guerrero, "Using Program Transformation, Annotation, and Reflection to Certify a Java Type Resolution Function," in *Proceedings of the* 15^{th} *IEEE International Symposium on High Assurance Systems Engineering (HASE)*, January 2014.

[3] J. A. McCoy, "An Embedded System For Safe, Secure And Reliable Execution of High Consequence Software," in *Proceedings of the* 5^{th} *IEEE International Symposium on High Assurance Systems Engineering (HASE)*. IEEE, 2000, pp. 107–114.

[4] G. L. Wickstrom, J. Davis, S. E. Morrison, S. Roach, and V. L. Winter, "The SSP: An Example of High-Assurance System Engineering," in *HASE 2004: The* 8^{th} *IEEE International Symposium on High Assurance Systems Engineering*. Tampa, Florida, United States: IEEE, 2004, pp. 167–177.

[5] V. L. Winter, H. Siy, J. McCoy, B. Farkas, G. Wickstrom, D. Demming, J. Perry, and S. Srinivasan, "Incorporating Standard Java Libraries into the Design of Embedded Systems," in *Java in Academia and Research*, K. Cai, Ed. iConcept Press, 2011.

[6] M. Fowler, *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, 1999.

[7] J. Henkel and A. Diwan, "CatchUp! Capturing and Replaying Refactorings to Support API Evolution," in *Proceedings of the 27th International Conference on Software Engineering (ICSE'05)*, St.Louis, Missouri, USA, 2005.

[8] I. Balaban, F. Tip, and R. Fuhrer, "Refactoring Support for Class Library Migration," in *Proceedings of OOPSLA 2005*. San Diego, California, United States: ACM, 2005, pp. 265–279.

[9] A. Donovan, A. Kiezun, M. S. Tschantz, and M. D. Ernst, "Converting Java Programs to Use Generic Libraries," in *Proceedings of OOPSLA 2004*, Vancouver, BC, Canada, 2004, pp. 15 – 34.

[10] W. Tansey and E. Tilevich, "Annotation Refactoring: Inferring Upgrade Transformations for Legacy Applications," *SIGPLAN Not.*, vol. 43, no. 10, pp. 295–312, Oct. 2008. [Online]. Available: http://doi.acm.org/10.1145/1449955.1449788

[11] Oracle Sun Developer Network (SDN), "Java card technology. http://java.sun.com/products/javacard/."

[12] D. Rayside and K. Kontogiannis, "Extracting Java Library Subsets for Deployment on Embedded Systems," *Science of Computer Programming*, vol. 45, no. 2-3, pp. 245–270, November-December 2002.

[13] S. Morrison, "SSP Class Loader Responsibilities," Sandia National Laboratories, Tech. Rep., 2005, internal Report.

Decomposed processes in Cloud BPM: techniques for monitoring and the use of OLC

José Martinez Garro[1], Patricia Bazán[2], and Javier Díaz[2]
[1]Facultad de Informática, UNLP (Universidad Nacional de La Plata), La Plata, Buenos Aires, Argentina
[2]LINTI, Facultad de Informática, UNLP (Universidad Nacional de La Plata), La Plata, Buenos Aires, Argentina

Abstract - *BPM (Business Process Management) in the Cloud has triggered revisions over several concepts like process decomposition, execution and monitoring. Decomposition allows processes to be executed in cloud oriented and embedded environments. This situation takes advantage of both approaches considering topics like sensitive data, high computing performance and system portability. Decomposed processes need to be monitored conserving the original business model's perspective. It can be considered also for monitoring purposes some data generated during process status changes. These data units are useful due to they contain information associated with the process execution. In this paper we present a model for decomposed process monitoring which also considers OLC (Object Life Cycle) data objects in order to provide a wider set of information for measuring and improving purposes. We also make a comparison about features like process execution and monitoring, considering hybrid environments versus embedded ones, including the use of OLC data objects.*

Keywords: OLC, decomposition, monitoring, cloud, BPM.

1 Introduction

The problem of monitoring a business process in a cloud oriented collaborative environment, conserving the process model's original perspective is deepened in the present work. It is made by facing the possibility of saving and analyzing OLC data objects generated through process transitions, in order to embrace as much relevant information as possible for process measuring and improvement. This work begins with a current bibliography analysis considering concepts associated with process decomposition in a hybrid scheme, process monitoring in the cloud and the necessity of binding processes with data objects, in order to obtain relevant information for process measuring and improvement. In second place it is presented a proposal with the architecture of a process monitoring application which considers at the same time OLC data objects related to process transitions. It is also included a comparison about the mentioned trends in process execution and monitoring, considering both hybrid and embedded environments. Finalizing the document we present some conclusions about the current state of the art and future work proposals in these research lines.

1.1 Related work

There are different trends when it comes to BPM in the cloud. A major topic in the research field is process decomposition and the different concepts associated with it: expressions for cost calculation, equations to obtain the best distribution scheme according to the infrastructure and the involved applications, and also methods to improve process performance. There are several articles providing different perspectives on these subjects [2] [3] [6] [11]. Regarding to process monitoring, there are some research lines introducing concepts about process measurement, process performance, business activity monitoring and OLC data objects, generally in embedded environments exclusively [7] [8] [10]. Considering cloud environments in order to cover this gap, we have presented in [17] an architectural proposal for a decomposed process monitoring application in a hybrid environment. In the present paper we go further deep into this architecture, introducing some new features by applying the concept of OLC data objects in order to obtain more information for process monitoring and measuring.

2 Hybrid architectures

Privacy protection is a major concern when the purpose is to put BPM in the cloud. Once private data are outside the organization, there is a certain lack of control over them. Besides, it is necessary to observe product's portability and its versions, and their availability in a cloud system. Other two problems also often considered are performance and efficiency.

The scalability and high availability of computing force are highly exploited by the intensive computational activities inside processes. The non intensive computational tasks, on the other hand, not always take advantage of this context. The performance of one activity running in an embedded environment should be better compared to the cloud, because of the amount of data that is transferred in order to execute the activity [1] [6].

- Possible architectures: in most BPM solutions, the process engine, the activities and data are located in the same side, even in an embedded or cloud approach. There are some papers introducing the PAD model (Process - Activity - Data) of Fig. 1 as a distribution possibility for BPM in the cloud. In this approach the process model, the involved activities and the process data are separately distributed. The PAD model defines four possibilities of distribution:

1) The first pattern is the traditional alternative where all elements are distributed over the final user side.

2) The second pattern is useful when the user already has a BPMS, but the high computing activities are located in the cloud in order to increment their performance.

3) The third pattern is useful for users who still do not have a BPMS, so they can use the cloud system in a “pay per use” way. In this approach the activities with low computing intensity or the ones with sensitive data management can be located on the final user side.

4) The fourth pattern is the cloud based model, where all the elements are located in the cloud.

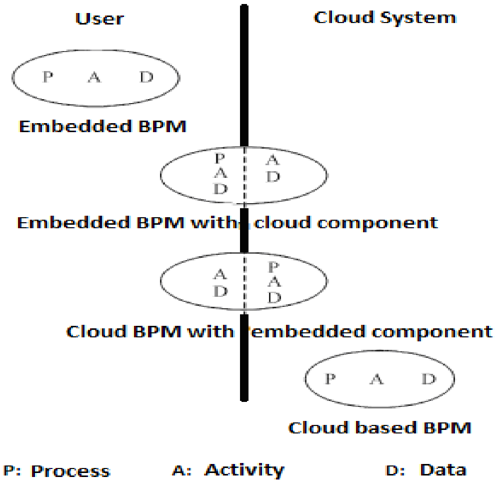


Fig. 1 : PAD Distribution Model [6]

- Business processes often manage flows of two kinds: control and data. Control flows regulate the execution of activities and their sequence, while data flows determine how the information is transferred from one activity to the next one in the process. BPM engines must manage both kinds of flows. A data flow could contain sensitive data, so when a BPMS is deployed in the cloud, the content of those flows should be protected. An example of the proposed architecture could be a scenario where the engine in the cloud only deals with data flows using reference identifiers instead of real data. Sensitive data are saved in the final user side, and non sensitive data are saved in the cloud. This scheme allows sensitive data not to travel indiscriminately through the web.

- Optimal distribution: the cloud system costs have been studied several times previously. There are formulas that allow calculating the optimal distribution of activities, since they can be located in the cloud or in an embedded system. The calculation takes in consideration different items, like time, money and privacy risks. [2] [3] [4] [6].

2.1 Criteria for process subdivision (Decomposition)

It is possible to generalize the distribution and identify a fifth pattern where the process engine, the activities and data are deployed in the cloud and in the final user simultaneously. This solution presents two potential benefits:

1) The process engine manages control and data flows. Once the activity receives data from the process engine and conludes its execution, the outcomes are passed again to the process engine. Considering now a sequence of activities located in the cloud, and a process engine deployed in the final user side: each activity uses data produced by the previous activity as an income. Data are not passed directly from one activity to the other but they are sent to the process engine first. Since data transference is one of the billing factors in this model, this kind of situations could become more expensive when large amounts of data are transmitted between activities. To avoid this problem a process engine can be added to the cloud, in order to regulate the control and data flows between the activities located in it.

2) When the cloud is not accessible, users can execute business processes in a complete way in the embedded system until the former one is available again [5] [11] [12].

In order to run a single business process between two separated engines, it should be divided into two individual processes, being convenient in this case for users to take a distribution list of the process and its activities. The process can be automatically transformed into two units: one in the cloud and the other in the embedded system. The communication between both systems can be described using a choreography language, like BPEL. Business process monitoring is more complicated now, since the process has been divided into two or more parts. As a solution, a monitoring tool can be developed for the original process, through the combination of the individual process monitoring details. This point will be analyzed further down [1] [16] [17]. As we presented previously in [17], there are several approaches implementing process decomposition in a hybrid architecture considering different aspects like data sensibility, application portability and high computing. Once the decomposition criterion was established, it is necessary to synchronize the different parts in runtime, which can be made following different lines. In our case, it was made by using Bonita BPMS and its process connectors, in order to initiate a new piece of the process in a new server once the previous one is finished. The result of this implementation is a set of cloud nodes joined by process connectors in runtime accomplishing in this way the original process model execution [5] [6] [16] [17].

2.2 Hybrid environments and process monitoring

As it has previously seen, major problems about using a partitioned process model are, besides the proper execution, the gathering and monitoring of the different distributed instances (either in an embedded system or in the cloud), and at the same time the accomplishment of viewing them under the optic of the "original process" to which they belong. To face this inconvenient we have designed a solution considering distributed and intercommunicated components forming an architecture, which is described as follows. On the one hand, it is necessary to associate the different process instances initiated by using a *chain*, with the purpose of gathering information about them by accessing the different involved servers. The execution model based on decomposed processes consists of linking each instance flow to the corresponding partitioned processes. Thus, when an instance finishes in a server, it initiates automatically a new instance corresponding to the next process partition, depending on the distribution model. For this purpose, each node in the architecture should be capable of establishing communication with the next node in order to initiate new instances, and gather in this way information about them. Namely, given a new instance which was initiated in a node of the architecture, we should be able to obtain, not only its data but every instance generated by it in another server.

2.2.1 Bonita Open Solution: API and connectors

There are several ways of implementing instance flow linking. In our case we have selected Bonita Open Solution [17] as the BPMS. So, once the original process was partitioned over the servers following criteria like sensitive data storing, data transferring and application portability, we have used the API and the connectors provided by the BPMS in order to create instances and recover their information using Java classes. These classes use the API as libraries, including functions like server authentication, instance launching, instance information gathering and process variable setting. These classes are invoked from the process definition using connectors.

It was also included on each process definition the information needed for the communication with other Bonita servers present in the architecture. Taking advantage of this link, it is possible to launch new instances in those servers by using connectors. Thus, every instance when is finished will execute the connector, which allows initiating a new instance by using the API, linking in this way automatically the process execution flow [6] [16] [17].

2.2.2 Centralized front-end

As it was previously described, a monitoring application must be developed in order to show integrated data related to distributed instances. Facing the execution trace, it is very important for each instance to be able of storing, not only its own information but the one associated with the instances created by it over other servers. In this way, by accessing the initial instance of the process, it is possible to recover the information associated with the next instance, and so on in order to obtain the complete execution trace.

Once recovered the information from the different servers, it must be provided an application in charge of the gathering process and showing the obtained data seamlessly. The most important thing in this aspect is to accomplish monitoring transparency for the users: they should not be forced to distinguish the server where the activity was executed, but they should visualize the different instances and observe them as a unique entity, according to the original process model. The implementation of this feature was made through a cloud based web application. This application was placed there in order to access each involved server, being them cloud or embedded, assuring in this way access to users from every point. For this purpose it is important for the application to have a catalog with the existing servers considering their location information updated. Each of these servers has a copy of a web service (*getInstanceService*), which receives a process definition id and returns information of each instance existing in the server that is associated with the definition sent as a parameter. The information returned includes instance id, current status (executing, completed, suspended), current activity if the instance is not finalized, start and end date. In this way, the application located in the cloud sends to each server a web service invocation with the selected process definition as a parameter, and receives the information about the associated instances. Then, this information is visualized in a web interface, where the user can select a particular instance and observe its details. In order to make this, the application has another web service (*getInstanceActivityService*) used to get from the engines the details of each activity associated to the instance. The returned information includes activity id, participant, start date, current status and end date. Once ended this collection phase, we need to remember that each instance contains also some information about the different instances initiated by it over the servers in the architecture. In this way, the web application will have to concatenate the received information and allow the users to observe the monitoring details in a transparent and integrated way [8] [9] [12].

2.2.3 Application's architecture

We can observe in Fig. 2 the different distributed components identified in the architecture design, as well as the internal relationship between them and the user.

The solution is composed by three main nodes: the cloud, the embedded or traditional systems and the monitoring application. The cloud works as a container for several elements: the BPMS, the monitoring application, the REST

API used by the developers in order to integrate applications with the process engine, and eventually a geolocation service which allows assigning to mobile clients the most convenient version of the service according to their localization [17].

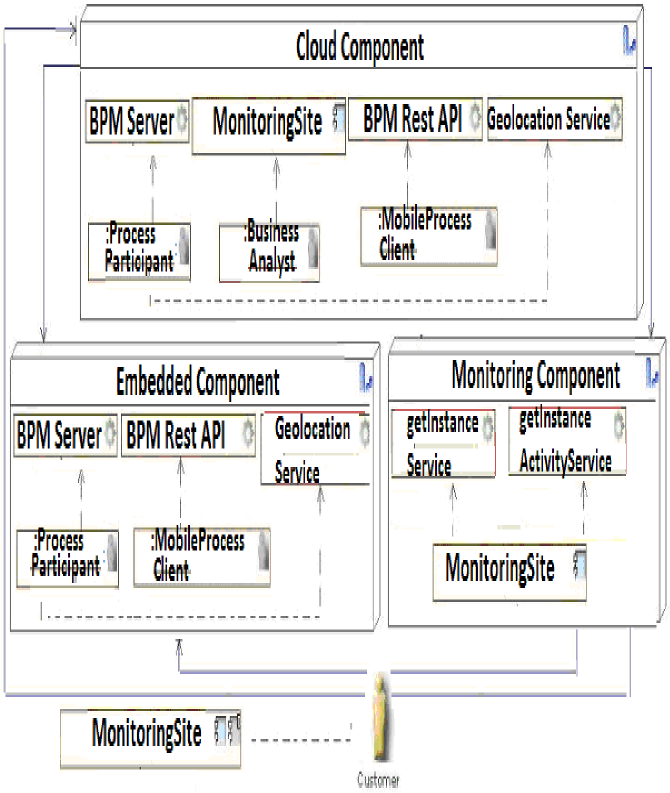


Fig.2 : Application architecture and user location

On the other side we find the embedded type components, namely traditional BPM applications which belong to the organization, and because of different reasons like data sensibility or application portability, they are not located in the cloud. These nodes, functionally speaking, are equivalent to the cloud ones, even when they have access restrictions and lower computing force compared with the first ones. The third component is related to the monitoring function. It is used by the application, and is in charge of returning information about instances and activities which were executed in every node of the distributed architecture. The web services *getInstance* and *getInstanceActivity* were constructed jointly with the monitoring application, and are executed on demand by this one. They are communicated with the process servers through an API (in our case, the one provided by Bonita), and are in charge of returning, in first order, information about the instances initiated on each server, and then, some data about the activities composing these instances.

2.2.4 Component Communication

If we consider every component present in the architecture, we have analyzed the communication between each one of them through an application communication diagram. There we can observe the most important involved applications, their main actors and the interaction of the different distributed software components.

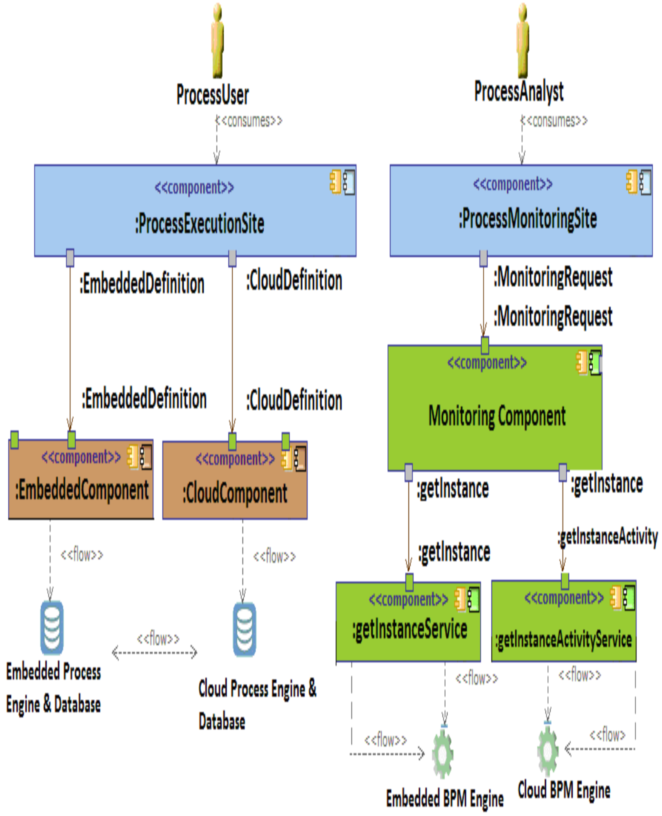


Fig.3 : Application Communication Diagram

We can see at the same time the different user profiles involved in the execution of the components represented in the architecture. While the preponderant role in process execution is the activity's participant, the monitoring site results are important for the business analyst, as well as for the architecture administrators who can optimize services or process components (Fig. 3).

A feature in common between the process execution application and the monitoring one is location transparency. Users should not be necessarily notified about changes in the execution environment, in case we are considering a decomposed process where the activities are located in different servers. This is very useful in order to allow users to have a unified vision of the process, more than a partitioned one, which main existence reason is related to technical issues and not with logical or business aspects generally. We can also visualize in Fig. 3 how both the execution and monitoring components access indistinctly to the cloud or the embedded nodes, in order to gather information about each instance initiated over the distributed servers.

2.3 Enriching Process Monitoring by using OLC objects

2.3.1 Foundations

In the field of BPM, monitoring is used to observe process behavior, and also probably to react to events and predict future process steps during execution. Processes that are automated using information systems like process engines, can be correctly monitored due to the system often offers logging capabilities and thus, the process can be easily recognized. In contrast, in environments where processes must be executed manually in a big portion, for example in a health care dependency, a high number of events are not captured automatically. An event monitoring point is related to certain events captured by an IT system connected to a specific event source, e.g. a database or a barcode scanner, and informs when certain state transitions (for example enabled, started or finished) occur in a process activity. In this case, probabilistic means that it is possible to provide an indicative index about process progress, but it is just an approximation [7] [8] [9]. In this way, an approach using events extracted from data objects creation or modification is introduced in order to increment the number of observable events used for process monitoring and progress prediction. Those are called state transition data events. After recording a data object with a specific status assigned, we can assume its existence. Additionally, this approach can be used to identify incorrect behavior in a similar way of what is done in the data conformance field [12] [13] [14].

The presented approach allows approximations through process execution based on information about data objects. These objects and their life cycle are the foundation for this approach. An OLC data can be represented through a Petri net, where a node describes a data state and a transition represents a data state change from the predecessor to the successor. An OLC specifies all the allowed data state changes of the corresponding data object. Based on this OLC, the transitions that can be monitored with events are selected. We consider an event as a fact that occurs in a particular point in time, in a certain place and with a specific context which is represented in the IT system. The data state transitions that are observable have been connected to the events which provide information about transition triggering. This connection is made by joining the particular OLC transitions to an implementation which extracts information from an event source, including the correlation to a specific data object in runtime. Nevertheless, not all data state objects in the OLC need to be reflected in the process model. Based on the assignment of data objects to activities, it is possible to provide information about process execution in runtime. We assume that an activity is enabled if it can be executed with the control flow specifications, and the incoming data objects are available in the required data state. We also assume that an activity is done as soon as the output data object reaches a certain data state. In design time, data state transitions can be marked as “observable”, referencing the before mentioned runtime monitoring capabilities. In runtime, process progress can be recognized through data state transition events that occur in the event storage. In order to obtain non observable details in process monitoring through data manipulation, the introduced approach should be combined with other techniques [13] [14] [15].

2.3.2 OLC management in Cloud BPM

In terms of implementation, the monitoring application should be capable of gathering process status information, and at the same time, all the information marked as observable in process transitions.

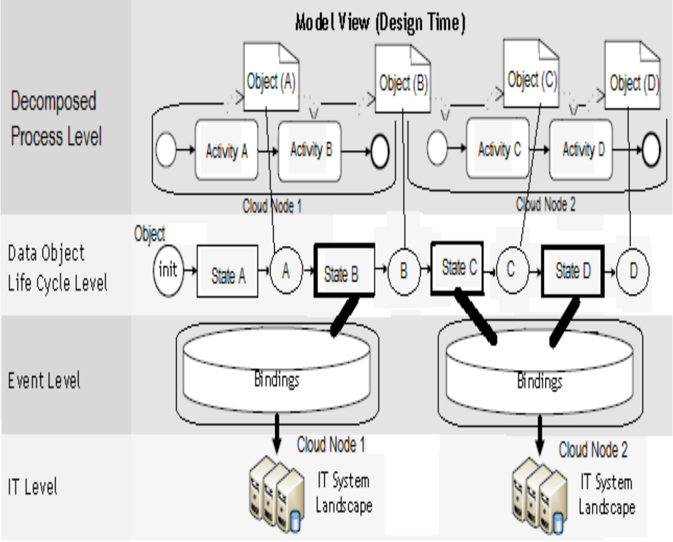


Fig. 4 : Decomposed Model View in design time [8].

In order to make this, it is necessary to bind the information in the process engine database with the information related to the different OLC data objects identified along the process definition. In Fig. 4 we can see how the different states are traversed by events and the information recording process in the environment. It is mandatory in this case to bind process activities and transitions with the respective OLC objects in order to gather all this information. We are considering that the process is already decomposed in the cloud environment, so the OLC objects should be collected from the corresponding nodes. The web application that we have considered before for process monitoring should be altered in order to gather this new information [7].

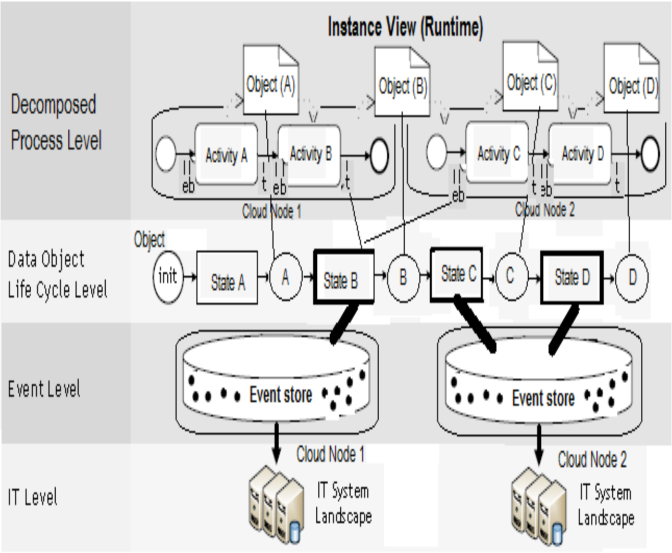


Fig.5 : Decomposed Model View in runtime [8]

In Fig. 5 we can observe how the different transitions insert information in the event storage during runtime. It is important to bind the recorded information with process details, in order to obtain all the events once the monitoring application is gathering the data to be shown [8] [9]. Each service that implements process activities should be responsible for saving with the business data some information about the process execution (instance's identification, timestamps and activity's identification) in the event storage, in order to be able of recovering the OLC objects during the monitoring process. This is the most traditional way to integrate the process logic with business data: using event data persistence as a way of integration in order to recover the associated information [8].

2.3.3 Considering OLC objects in the Monitoring Component

Taking in consideration the previously presented architecture, the monitoring component is formed essentially by two web service definitions: *getInstanceService* and *getInstanceActivityService*. The first one is in charge of gathering information about an instance from each node in the hybrid architecture, and the second one acts once the first one was executed, in order to obtain details about each activity associated with the instance. Now these services will keep their behavior, but should change their definition in order to consider besides the OLC data objects associated with the process, which means not only to search over the process engine's databases for information but to access the different sources in the event storage. All these mechanisms should be duplicated on each distributed node. A really important issue to consider in this context is how to display the OLC data objects in order to show them to the user in a helpful way. Analyze these objects provides an interesting vision about process execution, performance and progress, but they depend severely on the organization's context, so the monitoring application should consider a standard visual way to show them in order to uncouple the visualization mechanism and the business logic [9] [10] [11].

2.4 Process Monitoring Techniques comparison

Process monitoring, mostly applied as BAM (Business Activity Monitoring), is a discipline deeply developed in embedded schemes, where all the information needed is located in one server, and the process engine has not major issues during gathering it. It was also researched the possibility of taking business data objects in order to enrich the monitoring process with information related to the organization's logic besides the proper information collected by the engines. As it was previously presented, the inclusion of BPM in the cloud has triggered the consideration of hybrid schemes, where both the execution and monitoring processes are more intricate compared with the embedded version, due to the connectors and the distributed gathering process required. In table I. we compare hybrid environments with traditional ones in terms of process execution and process monitoring, and several concepts associated with them.

TABLE I. COMPARISON OF THE EXECUTION AND MONITORING FEATURES

	Embedded environments	Hybrid environments
Process execu-tion	Is performed by the individual process engine, which is in charge of guaranteeing the execution flow between the activities and gateways existing in the process model.	The execution on each node is provided by the individual engine located inside of it. It is necessary to link the different instances through process connectors or a choreo-graphy language like BPEL
Process monito-ring	Generally, the embedded systems have their own components for process monitoring. They are in charge of gathering the information from the proper engine and perform measurements about the different defined KPIs.	It is necessary to build a new application in charge of gathering the information from the different nodes in order to accomplish the original view according to the decomposed process model.
Gathe-ring process	This activity is performed by the same unit in charge of displaying the executed instances' progress in the individual engine.	Due data are distributed along the architecture, it is necessary to implement modules in charge of collecting information from the different intervening nodes.

OLC data objects	The information to gather in this regard depends on the organization's logic. It is necessary to configure the events and the storage from where the information is going to be recovered.	Since the events and storages are distributed along the architecture, the agents in charge of this task should consider now the observation of distributed sources.

3 Conclusions and future work

Process decomposition is a very useful mechanism in order to take advantage of the different features offered by embedded and cloud environments, but it inserts complications on how to monitor processes with the original model perspective, and also on how to obtain more relevant information related to the business logic and not necessarily to the process engine (like OLC data objects). Our future work is focused on improving the monitoring application in order to develop a standard visualization mechanism which allows showing the OLC information in an uncoupled way, ergo not to depend on the particular organization's business logic, considering the fact that our processes are distributed over the hybrid architecture. For making this, we have to analyze some patterns about OLC objects and obtain a general BAM based set of indicators, useful for process measuring and progress analysis (Business Activity Monitoring), but now applied on cloud based distributed processes.

4 References

[1] T. Kirkham, S. Winfield, T. Haberecht, J. Müller, G. De Angelis, "The Challenge of Dynamic Services in Business Process Management", University of Nottingham, United Kingdom, Springer, 2011

[2] M. Minor, R. Bergmann, S. Görg, "Adaptive Workflow Management in the Cloud – Towards a Novel Platform as a Service", Business Information Systems II, University of Trier, Germany, 2012

[3] M Mevius, R. Stephan, P. Wiedmann, "Innovative Approach for Agile BPM", eKNOW 2013: The Fifth International Conference on Information, Process, and Knowledge Management, 2013.

[4] H Sakai, K Amasaka. "Creating a Business Process Monitoring System "A-IOMS" for Software Development". Chinese Business Review, ISSN 1537-1506 June 2012, Vol. 11, No. 6, 588-595.

[5] M. Gerhards, V. Sander, A. Belloum, "About the flexible Migration of Workflow Tasks to Clouds -Combining on and off premise Executions of Applications", CLOUD COMPUTING 2012: The Third International Conference on Cloud Computing, GRIDs, and Virtualization, 2012.

[6] E Duipmans, Dr. L Ferreira Pires, "Business Process Management in the cloud: Business Process as a Service (BPaaS)", University of Twente, April, 2012.

[7] JP Friedenstab, C Janieschy, M Matzner and O Mullerz. "Extending BPMN for Business Activity Monitoring". University of Liechtenstein, Hilti Chair of Business Process Management, Vaduz, Liechtenstein. September 2011.

[8] N Herzberg, A Meyer "Improving Process Monitoring and Progress Prediction with Data State Transition Events". Hasso Plattner Institute at the University of Potsdam. May 2013.

[9] M Reichert, J Kolb, R Bobrik, T Bauer. "Enabling Personalized Visualization of Large Business Processes through Parameterizable Views". Hochschule Neu-Ulm, Neu-Ulm, Germany. November 2011.

[10] J Kolar, T Pitner, "Agile BPM in the age of Cloud technologies", Scalable Computing: Practice and Experience, 2012.

[11] A Lehmann and D Fahland , "Information Flow Security for Business Process Models - just one click away", University of Rostock, Germany, 2012.

[12] R Accorsi, T Stocker, G Müller, "On the Exploitation of Process Mining for Security Audits: The Process Discovery Case", Department of Telematics, University of Freiburg, Germany, 2012.

[13] A Frece, G Srdić, M B. Jurič, "BPM and iBPMS in the Cloud", Proceedings of the 1st International Conference on Cloud Assisted ServiceS, Bled, 25 Octubre 2012

[14] S Zugal, J Pinggera and B Weber. "Toward enhanced life-cycle support for declarative processes". JOURNAL OF SOFTWARE: EVOLUTION March 2012

[15] J.Martinez Garro, P.Bazan "Constructing hybrid architectures and dynamic services in Cloud BPM¨ Science and Information Conference 2013 October 7-9, 2013 | London, UK.

[16] J. Martinez Garro, P. Bazan "Constructing and monitoring processes in BPM using hybrid architectures". IJACSA Journal, 2014 January. London, UK.

[17] Bonita Open Solution http://es.bonitasoft.com/. October, 2013.

A comparative study on usage of traditional and agile software development methodologies in software industry of Asia

Syed Faisal Ahmed Bukhari and Hira Khan
Department of Computer Engineering, Sir Syed University of Engineering & Technology,
University Road, Karachi-75300, PAKISTAN

Abstract - *In recent years, agile methodology has been adopted for software development in comparison with traditional models like Waterfall model, Spiral model, Rapid Application Development (RAD) model. In this work, a comprehensive study has been done for comparison between agile and traditional models and their usage in software industry of Asia. This research is conducted to assist software development professionals for selection of suitable development model for small, medium and large size projects considering factors like scope, quality, cost, time and risk. Based on collected data, it is concluded that for small scale projects, software professionals feel more satisfied in using traditional methodology whereas for medium scale and large scale projects, professionals prefer to adopt agile models.*

Keywords Agile methodology, system development life cycle, project scope

1 Introduction

The systems development life cycle (SDLC) is a conceptual model used in the development of project that describes the stages including requirements gathering, designing, implementation / coding, testing and deployment. According to literature (Khurana and Gupta 2012), SDLC methodologies are the process to assure that software meet up established requirements. These methodologies entail the discipline to the development process to formulate the development more efficiently.

These methodologies resolved the problems arising from code and fix strategy. By the time of system growth it becomes increasingly complicated to add new features or to fix any bug (Kamel *et al.* (2010). By the invention of Waterfall model (the basic SDLC model) changing requirements were fixed once but practically requirements just could not be fixed (Royce 1970).

After requirement gathering, development teams work together with each other to create the best possible architecture for the product. The programmers implement design in code and lastly, the complete designed system is tested and dispatched. This process sounds good theoretically but practically it does not always work well if users change their minds after months or even years of requirements accumulation. This results in building replicates as it is very complex to interrupt the momentum of the project to accommodate the change.

It was argued (Beck 1999) for the solution of the above issues that the iterative and incremental techniques break the development into portions. Incremental development intended to reduce development time by various overlapping increments. Iterative development process breaks the project into deliverable iterations of variable length. Similarly, the Spiral Model evades detailing and defining the whole system upfront contrasting iterative development, where the system is built into pieces and prioritized by means of functionality. Spiral and iterative development process models presented a great increase in agility over the Waterfall process, but some practitioners believed that they still did not take action for change in the growing business world.

2 Research objectives

The overall purpose of this research is to conduct a detailed review of both traditional and agile methodologies applied in software industry of Asia. The main focus of this research is to study adoptability of methods and compare them in order to attain professional satisfaction. Because there are many developers, who have implemented the agile methodologies, but on the other hand, there are several developers who are satisfied with traditional methodologies according to their company's trends and needs. However, the research is conducted with the interest of finding empirical data to come up with tabular plans that would be useful for software industry.

The primary aim of this research is to study and contribute the knowledge about adopting the newly introduced agile software development model in contrast of SDLC models. Specifically, the research objectives are summarized in threefold:

1) To compare characteristics, strength and weaknesses of traditional SDLC models and agile models.

2) To gain an understanding of the reasons that why information technology (IT) industry are now inclined on agile models and which software development method (agile/traditional) is more suitable for software development professionals (with respect to scope, quality, cost, time, and risk) on the basis of three different scales?

3) To discuss risks associated with each of these methodologies.

A questionnaire was designed to get the opinion of software developers in Asia to evaluate usage of methodologies for software development according to the size of the project.

2.1 Traditional heavyweight models

Software methodologies like waterfall model, spiral model and RUP model etc. are often called traditional software development methodologies and classified as heavyweight methodologies (Nikiforova 2009). These are based on sequential series of steps that use comprehensive description and heavy documentation for all the sets of requirements and they do not support requirement changes. For example, waterfall model has linear nature and it is easy to follow and implement (Petersen *et al.* 2009). Its disadvantage is that it forces to define requirements thoroughly during the system requirements definition stage which is unrealistic (Chocano 1996).

The spiral model was revealed by Barry Boehm in 1988. In this model, although the developers and customers better recognize and respond to risk at every evolutionary level. The drawback of this model is that if a major risk is not revealed and handled, problems will occur with the possibilities of entire software failure. Another traditional heavyweight model is RUP model which was proposed by Rational Unified Corporation. It is an incremental process where the overall project is divided into iterations (Booch *et al.* 1998). According to researchers (Amlani 2012), this is a complete methodology and all of its documentation is easily available. The drawback as mentioned by programmers is that the process is cost consuming and its lengthy documentation (which is necessary in all iterations) consumes more time (Runeson and Greberg 2004).

2.2 Agile lightweight models

The most clearly focused description of agility is that it is the aptitude to both produce and retort to change with the intention of profit in an unstable business environment (Highsmith 2002). XP, stands for extreme programming, is most extensively used method in agile methodologies (Beck 1999). It focuses on the development rather than executive and managerial sides of software projects. XP projects begin with a release planning phase, pursued by several iterations, each of which terminates with user acceptance testing (Abrahamsson and Ronkainen 2002). The main focus of XP is to get the job done. The main disadvantage of this technique is that as XP is code oriented rather than design oriented therefore it has less documentation. The lack of formalism and design in this model can be problematic for large programs especially when many team members are associated with the project.

Among all of the agile methodologies, Scrum is exceptional because it initiates the idea of empirical process control. Empiricism states that knowledge comes from experience and decision making based on what is known. Like XP, Scrum is also an iterative and incremental approach. This was started as a framework that has been used to manage difficult products. The most common reason for the failure of project using Scrum is that many professionals are still unfamiliar with Scrum, even after taking a Scrum class or reading a few research papers about it. Sometimes organization is not yet setup the Scrum or may be the teams do not know how to employ scrum according to the company`s present constraints or may be the project becomes too complicated to get it under control.

Feature-Driven Development Process (FDD) is one of the agile processes that do no talk and write too much and unlike XP and Scrum, FDD designed to work with a large team for large projects. FDD splits the large team into smaller feature-focused teams. Unlike other agile methodologies, FDD is more appropriate for large projects. The limitation of FDD is that it does not identify what technology to use. Another disadvantage of FDD is class ownership.

3. Research approach

Commonly two approaches of research methods are used, i.e. quantitative and qualitative. In this work quantitative approach is used. For this purpose a self-administered close ended questionnaire is circulated by using http://www.docs.google.com in order to collect the primary data for statistical analysis. The main purpose for using that platform is to gather data from all over Asia.

Finding relevant professionals and software organizations is a very crucial part for this research. To make sure that survey questionnaire is distributed to the professionals using traditional SDLC and agile methodologies throughout the Asia, much efforts has been done which included access local software houses through their web sites and using www.linkedIn.com and www.facebook.com/sqlportal (social networks) to get the maximum number of responses.

3.1 Hypothesis

The aim of this research is to compare the traditional and agile methodologies that the IT industry of Asia prefers for the development of three different scales of projects. For checking whether IT companies are using traditional or agile approach the following hypothesis is used:

H10: Companies do not prefer to work under agile approach rather than traditional approach for small-scale projects.

H1: Companies prefer to work under agile approach rather than traditional approach for small-scale of projects.

H20: Companies do not prefer to work under agile approach rather than traditional approach for medium-scale projects.

H2: Companies prefer to work under agile approach rather than traditional approach for medium-scale projects.

H30: Companies do not prefer to work under agile approach rather than traditional approach for large-scale projects.

H3: Companies prefer to work under agile approach rather than traditional approach for large-scale projects.

H40: For small-scale projects, software professionals do not prefer agile (XP, Scrum, FDD) technique to adopt for a successful completion of a product in terms of product's scope, quality, cost, time and risk.

H4: For small-scale projects, software professionals prefer agile (XP, scrum, FDD) technique to adopt for a successful completion of a project in terms of product's scope, quality, cost, time and risk.

H50: For medium-scale projects, software professionals do not prefer agile (XP, Scrum, FDD) technique to adopt for a successful completion of a project in terms of product's scope, quality, cost, time and risk.

H5: For medium-scale projects, software professionals prefer agile (XP, Scrum, FDD) technique to adopt for a successful completion of a product in terms of product's scope, quality, cost, time and risk.

H60: For large-scale projects, software professionals do not prefer agile (XP, Scrum, FDD) technique to adopt for a successful completion of a product in terms of product's scope, quality, cost, time and risk.

H6: For large-scale projects, software professionals prefer agile (XP, Scrum, FDD) technique to adopt for a successful completion of a project in terms of product's scope, quality, cost, time and risk.

In above hypothesis, Hn0 and Hn (where n = 1 to 6) show null hypothesis and alternate hypothesis respectively.

3.2 Questionnaire formulation

In order to obtain data from IT organizations in Asia, a survey questionnaire was presented to the respondents. This questionnaire was filled by the various employees from IT departments of the companies or software houses. The questionnaire was based on elements of existing project development approach according to three different sizes of the project. It is divided into three parts. The first part contains questions regarding size of project and the methodology used. The second part was related to resources and attributes. The third part was designed to collect data about different characteristics including strengths, weaknesses and risk factors.

4 Results and analysis

As a result of survey, 59 responses were received from different type of software development professional working in all over Asia. After reviewing the results, it was evident that 62.7% of population using agile software process while 37.3% using traditional software development methods (Figure 1). On the basis of this result, hypothesis H1 is rejected while two hypotheses (H2 and H3) stand out as accepted.

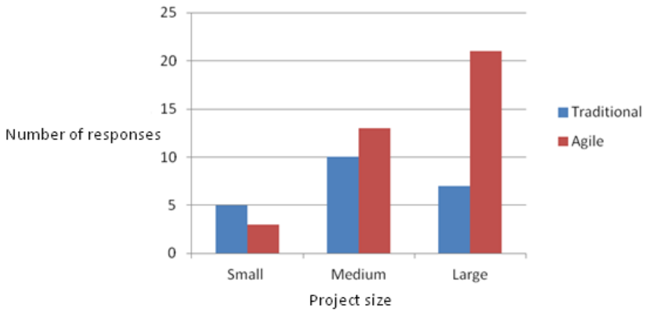


Figure 1: Use of traditional/agile models based on project size

Further, in this research work, five basic attributes (scope, quality, cost, time and risk) of project development were taken to test the preference of software professionals based on each aspect separately. Respondents were required to record their views against six models of project development in terms of five different factors. Out of these six models, three were selected from agile and three from traditional techniques.

Since the test is applied on multiple models in terms of multiple factors, therefore the average values for small (Tsmall), medium (Tmedium) and large-scale (Tlarge) projects are computed by adding up the responses against three models (Waterfall, Spiral and RUP) of traditional technique in terms of all five attributes.

Similarly, the average values for small (Asmall), medium (Amedium) and large-scale (Alarge) projects are computed by adding up the responses against three models (XP, Scrum and FDD) of agile technique in terms of all five attributes.

4.1 Small scale projects analysis

On the basis of above procedure, the average resulted percentage for traditional technique Tsmall and Asmall based on five attributes, i.e. scope (TSs = 50% and ASs = 50%), quality (TQs = 50% and AQs = 50%), cost (TCs = 63% and ACs = 38%), time (TTs = 63% and ATs = 38%) and risk (TRs = 75% and ARs = 25%) are 60% and 40% respectively.

$$Tsmall = (TSs + TQs + TCs + TTs + TRs) / 5 \qquad (i)$$

$$Asmall = (ASs + AQs + ACs + ATs + ARs) / 5 \qquad (ii)$$

As explained by the results and analysis of above five aspects (that are product’s scope, quality, cost, risk and time); agile doesn’t prove to be the most preferred software development technique used by most of the software professionals for small-scale projects. Therefore hypothesis H40 is accepted and H4 is rejected.

4.2 Medium scale projects analysis

Similar to the above approach, the average resulted percentage for traditional technique Tmedium and Amedium based on five attributes, i.e. scope (TSm = 43% and ASm = 57%), quality (TQm = 39% and AQm = 61%), cost (TCm = 35% and ACm = 65%), time (TTm = 35% and ATm = 65%) and risk (TRm = 35% and ARm = 65%) are 37.4% and 62.6% respectively.

$$Tmedium = (TSm + TQm + TCm + TTm + TRm / 5 \qquad (iii)$$

$$Amedium = (ASm + AQm + ACm + ATm + ARm) / 5 \qquad (iv)$$

As explained by the results and analysis of above five aspects (that are product’s scope, quality, cost, risk and time); agile is proved to be the most preferred software development technique used by most of the software professionals for medium-scale projects. Therefore hypothesis H5 is accepted.

4.3 Large scale projects analysis

For large-scale project, the average resulted percentage for traditional technique Tlarge and Alarge based on five attributes, i.e. scope (TSl = 18% and ASl = 82%), quality (TQl = 29% and AQl = 71%), cost (TCl = 21% and ACl = 79%), time (TTl = 21% and ATl = 79%) and risk (TRl = 21% and ARl = 79%) are 22% and 78% respectively.

$$Tlarge = (TSl + TQl + TCl + TTl + TRl) / 5 \qquad (v)$$

$$Alarge = (ASl + AQl + ACl + ATl + ARl) / 5 \qquad (vi)$$

Therefore it is concluded that for large scale projects hypothesis H6 is accepted.

4.4 Hypotheses assessment summary

On the basis of above analysis (shown in Figure 1**)** hypothesis H1 has been rejected while two hypotheses (H2 and H3) stand out accepted. The analysis of five attributes has indicated (Figure 2-6) that hypothesis H4 is rejected while two hypotheses H5 and H6 stand out accepted.

4.5 Selection of methodology based on different priorities

During this research the following results are also obtained based on different priorities of software customers:

4.5.1 Strictly follow a project plan

It is clearly mentioned in Figure 7 and Table 1 that approximately 63% of the respondents consider that fulfillment of the user requirement is more important than strictly following an initial plan, while around 25% were against it. The remaining 12% of the respondents were neutral.

4.5.2 Detailed documentation

Heavy documentation seems to be a negative aspect for most of the respondents as shown in Figure 7. About 59% selected it as a bad or slightly bad attribute in the successful completion of the projects, while 27% respondents are agreed with this attribute and 14% are neutral.

4.5.3 Hiring highly skilled professionals

Approximately 81% respondents are agreed for hiring skillful workers to produce good quality software, only 10% respondents considered it as a bad approach while 9% are neutral.

4.5.4 Autocratic management style

Unlike the results of other research works conducted in different countries, 63% of the respondents of Asia like the democratic management style rather than the self-organizing team, 24% respond neutrally and only 13% respondents think that this type of management does not allow the developers to be more creative.

4.5.5 Flexible project plan

Approximately 60% of the respondents think that there should be flexibility in a project plan in order to welcome change in requirements and try to produce software which holds complete sets of user specifications whereas approx 25% of the respondents did not support any flexibility in a project plan while approx 15% responds neutrally.

4.5.6 Focus on working software rather than detailed documentation

As shown in the graph that contrary of the results of detailed documentation, approx 58% of the respondents considered that basic documentation could be less time consuming and will give a positive impact for the successful completion of a project. Approx 27% dislikes this characteristic while 15% of the respondents are impartial.

4.5.7 Use of a good development tool

Approximately 59% of the respondents are not in favor of just using a good development tools and not pay attention in having a skilled professionals for the coding, testing or other development procedures. 24% of respondents considered that using a good development tools are more appropriate rather than hiring a skilled professionals which is costly too. And remaining 17% are impartial.

4.5.8 On-site customer

Having customer on-site is the main attribute of agile methodologies and according to the results as shown in Table 1, the working professionals are not very much satisfy with this attribute as approx 56% of the respondents selected it as a bad or slightly bad for the success of the project while 36% are in favor and 8% are neutral.

4.5.9 Self-organizing team

Approx 61% of the respondents didn't agree with this attribute because of the reason of results drawn in autocratic management style which is more likely to be followed for the development of the software. Only 27% of the respondents considered self-organized teams as a good attribute and remaining 12% are impartial about this aspect.

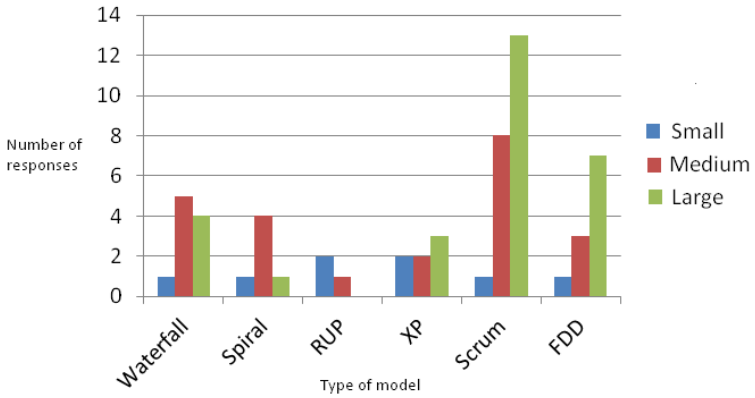


Figure 2: Use of traditional Vs. agile model based on scope

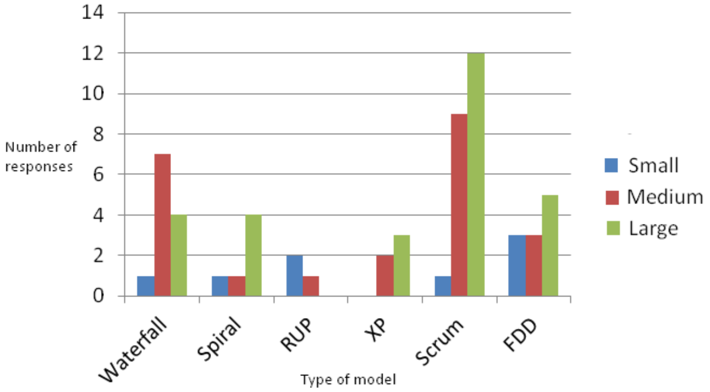


Figure 3: Use of traditional Vs. agile model based on quality

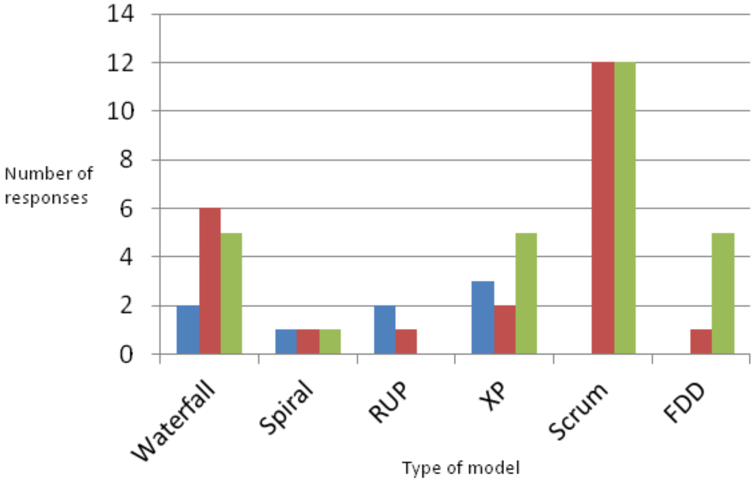


Figure 4: Use of traditional Vs. agile model based on cost

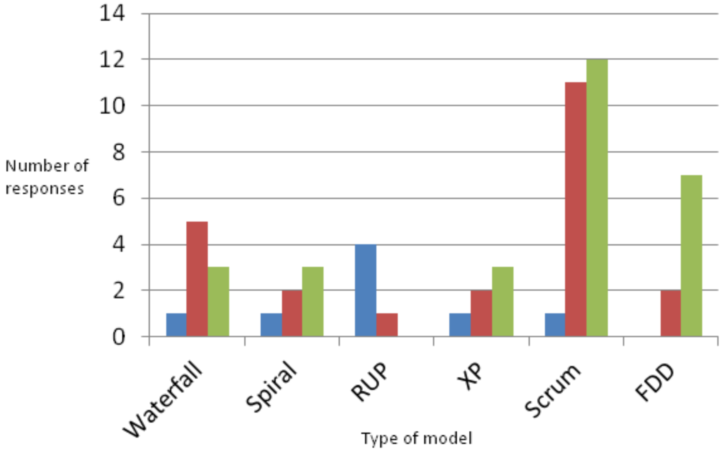


Figure 5: Use of traditional Vs. agile model based on risk

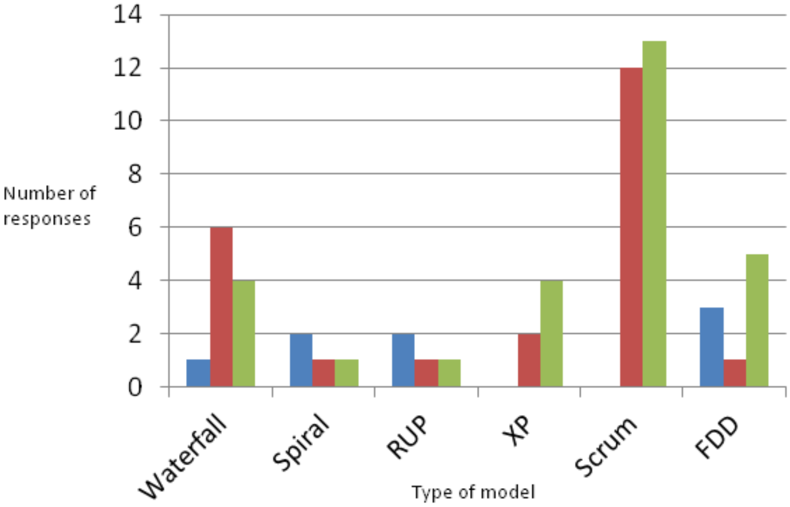


Figure 6: Use of traditional Vs. agile model based on time

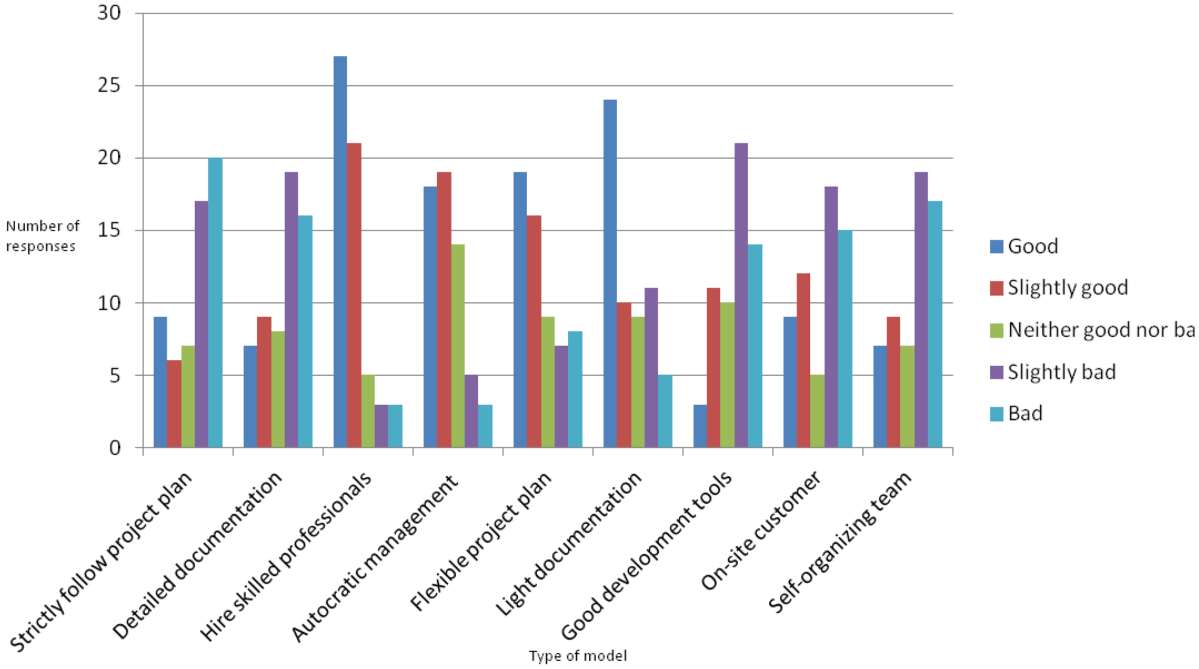


Figure 7: Priorities of software professionals

5 Conclusion

As a result of this research it is suggested that a hybrid methodology should be adopted by the software organizations working in Asia. It comprises of the following steps:

1. Prepare a flexible plan starting through requirements gathering phase which can be repeated at any stage of development according to the needs of customer.

2. Detailed documentation should be avoided at every stage and step perhaps only imperative and significant documentation must be needed.

3. Don't completely emphasize on development tools with novice professionals, who don't have any previous experience and can work on cheap rates. Management must hire highly skilled professionals with novices to deliver a good quality software that meet customer requirements.

4. Self-organizing behavior is not appropriate for the environment of Asia. There should be some team lead decisions for the distribution of responsibilities within a team. The management must allocate the duties to every team member according to their expertise.

5. Customer should always welcome with the set of their requirements but the presence of customer on site is not a good idea for team members.

6 References

[1] Abrahamsson B. S., and Ronkainen J., "Agile software development methods: review and analysis", VTT Publication, Espoo. 107, 2002

[2] Amlani R. D., "Advantages and limitations of different SDLC models", International Journal of Computer Applications & Information Technology 1, 6-11, 2012

[3] Beck, K. "Embrace change with extreme programming", IEEE Computer, 70–77, 1999

[4] Booch G., Robert C. and Newkirk J, "Object oriented analysis and design with applications", 2nd edition, Addison Wesley Longman, 1998

[5] Chocano M. E., Comparative study of iterative prototyping vs. waterfall model applied to small and medium sized software, MS Thesis, Massachusetts Institute of Technology, 1996

[6] Highsmith J., "Agile Software Development Ecosystems", Addison-Wesley Professional, 2002

[7] Khurana G. and Gupta S., "Study and comparison of software development life cycle models", IJREAS, 2(2) 1-9, 2012

[8] Kamel M., Bediwi I. and Al-Rashoud M, "Planned methodologies vs. agile methodologies under the pressure of dynamic market. JKAU: Eng. Sci., 21 (1) 19-35 2010

[9] Nikiforova O., Nikulsins V. and Sukovskis, U. "Integration of MDA framework into the model of traditional software development, Frontiers in Artificial Intelligence and Applications, Databases and Information Systems V, 187 229–239. IOS Press, Amsterdam, 2009

[10] Petersen K. , Wohlin C. and Baca D., The waterfall model in large-scale development, Proc. 10th International Conference on Product-Focused Software Process Improvement, Oulu, 386-400, 2009

[11] Royce W.W., "Managing the development of large software systems: concepts and techniques", Proc. WESCON, 1–9, 1970

[12] Runeson P. and Greberg P., "Extreme programming and rational unified process – contrasts or synonyms?", Lund University, Sweden, 2004

Acknowledgements

The authors would like to thank Higher Education Commission, Pakistan for supporting this research.

The Position of Component Certification in CBSE Activities

Lina khalid Ahmed

Department of Software Engineering, Zarqa University, Amman, Jordan

Abstract - *CBSE (Component Based Software Engineering) is the most important approach to software development because it is based on reuse technology. The successful reuse of component requires a development process tailored to CBSE, so it includes activities that find and compose reusable components. Reuse components can be the reason for building high quality products because they are chosen according to some issues that lead to the concept of component certification. This certification has its position in CBSE main activities. This paper defines the component certification and describes the positioning of it in CBSE activities and how this certification affects the success of this approach.*

Keywords: Component, CBSE, CBSE approach Component certification.

1 Introduction

Development with reuse has become a strategy for new systems. It has been used in response to the demand for lowering software cost, increasing time to market and producing a high quality product.

When a reuse concept is used, the process includes an activity where the abstract concepts begin to create executable reusable components.

Units that are reused may be of different sizes, one of the main units that are reused is component which ranges in size from subsystem to object. The main advantage for reusing components is that fewer software components are needed to be specified, designed, implemented and validated, all of which lead to cost reduction. However, the problem related to reuse strategy through using components is the cost associated with specifying a component that is used through reuse process which is suitable to be used in that specific situation. These additional costs mean that the reductions in the overall development costs through reuse may be less than expected.

The term certification is introduced through CBSE activities and it can be used to help the developer choose a trustworthy system.

This paper introduces the effectiveness of component certification through CBSE activities and its position. It also shows how this certification helps the developers to integrate trustworthy components.

This paper is classified as follows: Section 2 describes the related works and includes all the authors who have worked on this area. Section 3 defines Component Based Software Engineering approach, and component certification respectively. Section 4 describes the result. Section 5 sums up everything in the conclusion.

2 Related works

Many researchers work on component technology that is based on reusability. [2], the authors in this paper describe the relationship between the evaluations performed during certification and their selection. They propose a components-based life cycle for COTS (commercial-off-the-shelf) and software product line development. Moreover, they identify the process characteristics between the two types of evaluation and finally classify the required qualities during certification process. In [3], the authors propose a concept called SCL (Software Certification Laboratories); they suggest that this concept must take part in the certification product role which then offers the consumer's trust. In this method, SCL took all the information from the developer's site and passed it to the consumer's site and then returned back to collect this information from the user and used it to produce the warranties according to these results. In reference [4], a complete component-based business document modeling was produced. This modeling is built upon existing standards that are extended by introducing the concept of generic business document template out of the specific needs of the user's document. The result part of this paper is a complete library of reusable business components that has been developed to easily produce a new business model. Another work is [5], where the objective of this work was to describe a method to measure and certificate the ability of software component to perform the reliability quality.

The result of this technique is possibly certified components as well as the system that contains the components.

This paper describes the importance of component certification through CBSE activities. It describes the major activities of component based approach then defines the process of component certification and its position on these activities.

3 Component Based Software Engineering(CBSE)

The objective of this section is to describe the essential terms that are used in this type of approach then its shows how this approach works through its activity.

3.1 Component and component model definitions

In the software industry, the term "component" is used in many different ways. For example it is applied to user interface components implemented as Active X, to major infrastructure items such as database management system, and to any software artifact that can be reused.

We can define a software component as *a separable piece of executable software that can interoperate with other components within supporting environment and it is accessible only through its interfaces*. In order to combine with other components, it must be able to achieve details of its interfaces [6]. Defining components as separable entities makes it possible to achieve the maintainability benefits offered by components. It also makes easy reuse of components in specific scenarios, such as requirements for a new user interface, or changes the structure of the company [6].

Some essential characteristics of components that are used in CBSE approach are [1]:

- **Standardized**: which means that the component used in a CBSE activities has to match with some standardized component model (which is defined later in this section)

- **Deployable**: the component must work as a stand alone entity on a component platform that implements the component model

- **Documented**: components should be fully documented so that the user can make a decision whether the selected component meets its needs or not.

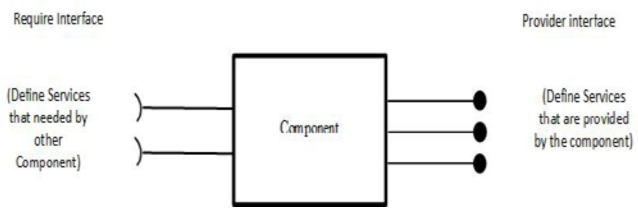

Require Interface
Provider interface
(Define Services that needed by other Component)
Component
(Define Services that are provided by the component)

Figure 1. The component interfaces

Figure 1 shows, the components have two related interfaces. These interfaces reflect the services that the component provides (provider interfaces) and the services that the component requires (Require interfaces) in order to work properly [1].

A component model can be defined as a standard for component implementation, documentation and deployment. These standards are for component developers and providers, for developer to make sure that the components can interoperate, and for the providers of the components who provide middleware to support component operation.

3.2 CBSE approach

CBSE is a process that highlights the design and construction of systems using reusable software components. By that definition, we can assemble components from a catalogue of reusable components in a cost and time effective manner [7].

CBSE has become the most important approach because all software systems became more complex and larger than before. The only way to deal with this complexity was through reuse rather than re implement. The **essential** activities for CBSE approach with reuse are [1]:

- **Component analysis**: the stakeholder of the system defined their requirements in the abstract view rather than in details. Here the complete set of requirements is defined in order to identify the complete set of components. Usually there is no exact match between the selected component and the requirements, but the selected components may be used only to provide some functionality of the requirements.

- **Modify requirements**: requirements are modified and refined according to the available components. If the user is not satisfied from the available requirements, he can switch to other requirements that support his request in this stage. If the modification is impossible, the first stage is repeated to search for other components for an alternative solution

- **System design with reuse:** the architectural design of the system is built. Further refinement for the design and more searches for the component are made through this stage. In this phase the designer takes into account the components that are reused and organizes the framework to accommodate with it.

- **Composition process**: the discovered components are integrated with the component model infrastructure. So the integration process is part of the of the development process. Figure 2 represents the main activities of CBSE approach

Figure 2 shows that there are two additional stages, one at the beginning (requirement specification), and one at the end (system validation). They are not defined in the previous stages because they are the same in all other approaches so what is mentioned is only what makes the CBSE approach different, and that is what is demonstrated in the gray ellipses in the figure.

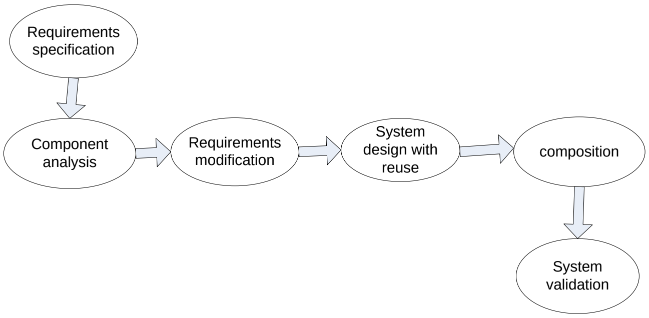


Figure 2. The Basic Activities of CBSE

3.3 Component certification

In order to make a component available for reuse or sale, enough information must be made available about it so that it would be possible to decide whether it is useful or not. The information should treat the component as a black box. This type of information can be maintained and stored in a catalogue. One suggestion for ensuring the quality of the components is a process of third party certification, in which many companies would test the component and certify it for its purpose [8]. This concludes that the component certification is a process to ensure that the software components match to basic standards. Based on this certification, trusted components are integrated. Moreover, the certification stands out as an important area to evaluate the component reliability level, although that task seems to be very difficult because the community of software engineering has expressed many proprieties to evaluate the specific components. [5, 10] a certification body must have a certification system that describes techniques for performing the certification process. A certification process can be defined as a process of verifying properties which are related to a component and this certification can provide the validity to that component [9].

4 Result

Component certification is a technique to make sure that a component follows the standardization. Based on this certification, the assembled components into an application are trusted. High quality products that are based on CBSE approach always need an effective certification. Some issues need to be taken by organizations to make CBSE successful. These issues are:

- Analyzing the domain which identifies and distributes a set of software components that can be used in existing and future software in a particular application domain (the overall goal of this issue is to enable software engineers to share these components).

- Component qualification and composition
They ensure that the chosen component is properly fit to the architectural style that is proposed. After that, when the component is qualified for reuse within an application architecture, conflicts may occur so adaptation

process technique (called component wrapping) is used. Component wrapping customizes the components to remove the conflicts through integration. Component composition assembles all adapted and engineered components to distribute the suitable architecture for an application.

- Analysis and design for reuse
  The requirement model is first analyzed and the elements of this model are compared to the descriptions of the reusable components. If the components match the requirement, it will be reused in the system from the library of reusable components which is called a repository of components. Otherwise the component must be created to fit the requirements in the model and this should create reusable components.

- Classifying and retrieving components
  A 3 C model (concept, content and context) is described to reuse an appropriate component in any system. The concept is a description of what a component does and the content of the component describes how the concept is achieved. The context places a reusable software component within its domain application.

The result of these issues must be certified components that can be stored in a repository of components to be used by the developer through applying CBSE approach. The developer looks for the appropriate components, modifies them as needed and incorporates them in the system. Selecting the most appropriate components makes the CBSE approach have good advantages such as reducing the amount of software to be developed and also reducing the cost and time, all of which leads to a faster delivery time to market.
The position of component certification is explained in figure 3.

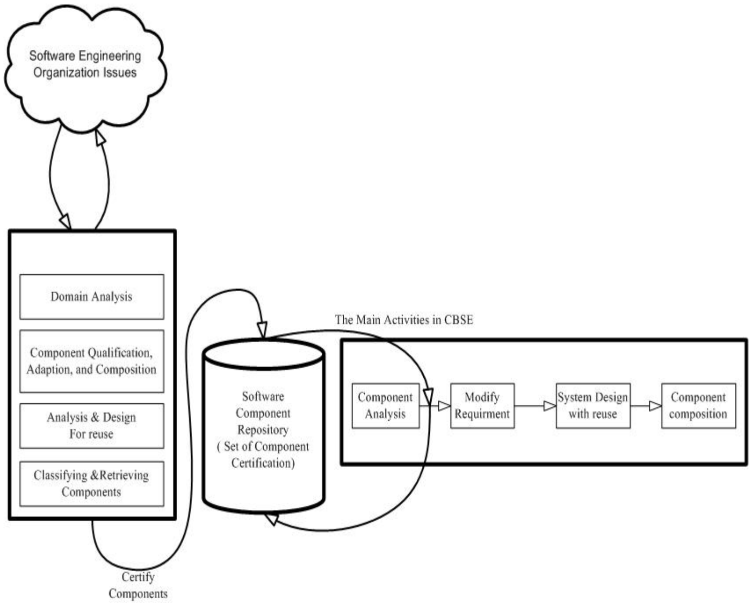

Software Engineering Organization Issues
Domain Analysis
Component Qualification, Adaption, and Composition
Analysis & Design For reuse
Classifying &Retrieving Components
Certify Components
Software Component Repository (Set of Component Certification)
The Main Activities in CBSE
Component Analysis
Modify Requirment
System Design with reuse
Component composition

Figure 3. The position of component certification in CBSE

The role of the component certification appears through the first and second stags of CBSE activities. The requirements are specified and the right component is chosen. That is done according to some considerations that are specified through component certification which is stored in the repository of components.

The most important criteria that should be into account through selecting the components are:

- Functional requirements of the system: the selection of components must be done to satisfy the main user and system requirements

And

- The quality of the product

The quality attributes of the product is very important through selecting components. The final architecture of the system must provide the key attributes of the system, so the components that are selected and the assembled must provide those quality attributes.

Many methods and strategies are used to complete the selection process. One of the main strategies is OTSO (Off-The-Shelf Option). The OTSO method was developed to make the selection of the components according to the customer's requirements. The main principle of this methodology is comparing the costs associated with each alternative component.

5 Conclusion

One of the most important reasons for choosing CBSE approach is the plan for reuse. The idea is to build a system from existing components by assembling and integrating them. What makes this approach very usuful is the existance of the certification. Component certification is a very important part in this approach because it ensures selecting the right component for reuse. All in all, this paper spots on the component certification and its position in CBSE activities.

6 REFERENCES

[1] Somerville I." Software Engineering" . Pearson Education, 2011. ISBN10: 0-13-705346-0.

[2] Land R, Alvaro A, Crnkovic I. "Towards Efficient Software Component Evaluation: An examination of component selection and certification". IEEE computer society, 34 the Euromicro Conference Software engineering and Advanced Applications, 2008.

[3] T. Janner et al., “A Core Component Based Modeling Approach for Achieving E-business Semantic Interoperability". Journal of theoretical &applied E-commerce research, Vol 31, issue 3, PP: 1-16, 2008.

DOI: 10.4067/S0718-18762008000200002.

[4] Tamal Sen and Rajib Mall. “State-Model-Based Regression Test Reduction for Component-Based Software”. ISRN Software Engineering, vol. 2012, Article ID 561502, 9 pages, 2012. DOI:10.5402/2012/561502.

[5] Alvaro A, Almeida E, Meira S. "Software Component Certification: A survey". Conference on software engineering and advanced applications, IEEE, 2005.

[6] K. Whitehead, "Component –based Development: Principles and Planning for Business Systems". Pearson Education, 2002. ISBN 0-201-67528-5.

[7] Pressman R. "Software Engineering: A practitioner's approach". McGraw-Hill, 2010.

[8] Whitehead k. "Component –based Development: Principle and planning for Business Systems ".Person Education,2002. ISBN 0-201-67528-5.

[9] J. Stafford, K. Wallnau , "Is the Party Certification Necessary? " Software Engineering Institute, Carnegie Mellon University, USA.

http:://niap.nist.gov/howabout.html.

[10] Alvaro A, Land R,Alvaro A,Crnkovic I,"Software Component Evaluation:A theoratical study on component selection and certification,"Malardalen university,2007, ISSN14404-3041.

The Size of Software Projects Developed by Mexican Companies

J. Aguilar[1], M. Sánchez[2], C. Fernández[2], E. Rocha[2], D. Martínez[2], and J. Figueroa[2]
[1]Departament of Engineering, Universidad Popular Autónoma del Estado de Puebla, Puebla, Pue., México
[2]Institute of Computer Engineering, Universidad Tecnológica de la Mixteca , Huajuapan, Oax., México

Abstract - *Currently, most software projects around the world are small rather than large. Despite this, there are more methodologies, tools, frameworks, processes, and so on, for developing and managing large software projects than for small ones. Small software projects are important because they generate considerable resources. For example: apps (small mobile applications) generate around $25 billion dollars of revenue. This paper shows our findings regarding the size of the projects built by Mexican software development companies. We surveyed 107 Mexican companies and found that 92% of their developed projects are micro and small, and 8% are medium or large. In addition, according to our research, 84.1% of companies in Mexico are micro or small businesses.*

Keywords: Small projects, Software size classification, Companies size classification.

1 Introduction

There are many examples of tools, methodologies or, processes oriented to medium or large projects such as the following: CMMI[13], COCOMO [14], EPM [15], among others. These tools, methodologies and processes are highly accepted around the world. We need the same global acceptance for small software projects, tools, methodologies and processes, because micro and small software projects generate an important economic impact. For example, apps are mainly small projects, and the app stores run by Apple Inc, Google Inc, and so on, offer more than 700,000 apps each and, generate around $25 billion dollars of global revenue [7].

First of all, we need to identify the size of software projects. There are several factors, among them: i) Project development cost, ii) Number of people required to develop the software, and iii) Amount of software to be produced [6]. For example, a small software project has these factors: i) Cost between USD 7,500.00 and 192,000.00; ii) Team of less than 4 persons; and, iii) Size between 9 KLOCs and 38 KLOCs. This is our personal viewpoint about small software projects, although each author has his or her own viewpoint about software size classification.

In this paper we will show the findings of our research regarding: a) Size of software projects being developed in Mexico, and b) Company size in Mexico. We want to answer the following questions: What is the size of software projects developed by micro and small software companies in Mexico? How is software size to be classified? These questions are important to answer because there are currently many software process models oriented to large and medium projects, but micro and small projects have perhaps been forgotten.

2 Background

This section shows the main elements of our research.

2.1 Software Size Classification

First of all, we are going to show different viewpoints regarding software size classification. Fred Brooks [1] said that there are different kinds of software: i) Programs complete in themselves, ready to be run by the author on the system on which it was developed, ii) A programming product that can be run, tested, repaired and extended by anybody, iii) A Programming System, which is a collection of interacting programs, and iv) Programming Systems Product, which costs nine times as much as the other three kinds. Another author, Watts Humphrey [2], said that there are five stages of Software Product Size: Stage 0, very small program elements, written by programmers alone. Stage 1, small programs, or modules, designed, implemented, and tested by programmers alone; these programs typically range in size from only a few dozen to several hundred LOCs (Lines of Code). Stage 2, larger programs, or components, that typically involve teams of developers who develop and integrate multiple stage-1 modules into larger stage-2 component programs. Stage 3, very large projects that involve multiple teams controlled and directed by a central project management. Finally, Stage-4, massive multi-systems that involve many autonomous or loosely federated projects. The next author, Solvita Berzisa [3], said that project size is described with four attributes where three attributes –team size, budget and duration- are scalar values and such values should be grouped into intervals. The team size values have been divided into two intervals: less than seven and seven and more. Analyzed budget attribute values have been divided into five intervals: less than 10 000 USD, 10,000 to 50,000 USD, 50,000 to

100,000 USD, 100,000 to 500,000 USD, and more that 500,000 USD. Project duration values have been divided into four intervals: less than six months, six months to a year, from a year to two years and more than two years. These authors did their own software size classification.

Besides these viewpoints, The International Software Benchmarking Standards Group (ISBSG) is the global and independent source of data and analysis for the IT industry. There are 5,052 projects included in the ISBSG[1] in June 2009, of which 76.1% are micro and small projects, from 0 to 399 function points, which corresponds to approximately 31,920 LOCs. We applied Backfiring[4] to convert function points to lines of code. Smaller projects are more common in this repository. The projects have been submitted from 24 countries. Projects of different size ranges have different key characteristics [5][8][9]. For example, we can use table 2.1 to find the ideal duration and staffing ranges for a project, based on its estimated effort.

Table 2.1 Time and effort in software projects according to their size.

Project Size	**Effort.** Work hours of effort**.**	**Months Duration**	**Ideal staffing**
Small	8-360	0-3	1 person
Medium	361-3600	3-6	3-7 persons
Large	3601-24 000	6-12	7-24 full-time persons

A software project can be considered to be small if one or more of the following criteria apply:

a) Less than two man-years of development effort is needed.
b) A single development team of five people or less is required.
c) The amount of source code is less than 10 KLOCs.

The size of a particular project can vary greatly depending on the language chosen. For instance a 200,000 line Perl project generally has the functionality of a 1,200,000 line C project. This is from the estimate in Code Complete that states that Perl usually requires 1/6 the number of lines to do the same task as C. However, different size ranges entail different issues that involve major organizational differences. Specifically:

1. Small - reasonable size for one person to produce in a reasonable time.
2. Medium - reasonable size for a small team (max 8; beyond that size, communication issues arise that drop productivity and require reorganization to handle) to develop and entirely understand.
3. Large - A large team (significantly > 20 - note that the average team of 20 is about as productive as a team of 5-8 due to organizational issues).
4. Huge - The project is large enough that specialized tools are probably needed to help with project navigation so that team members can figure out what they need to understand to work on it.

As can be seen, there are a lot of viewpoints regarding software size classification. In this paper we try to take into account all these opinions and offer a proposal.

2.2 Economic impact of small software projects.

Basically, an app is a small software project. For instance, the average size of an Apple app is around 10 KLOCs [12]. For this paper, then, we consider an app as a small software project. Thus, we can see that there are millions of mobile devices sold around the world. For example, in 2012, there were 446 million Android devices, 199 million Apple devices and, 17 million Windows devices sold. Furthermore, there are millions of apps available [11]. The apps market was about USD 25 billion which is expected to be USD 155 billion in 2017 [10]. Obviously, the economic impact of small software projects is very important.

3 Our research

This section describes in detail our research; we only show our results about what kind of software are mainly implemented by Mexican software development companies. Section 3.1 will explain the methodology used for our work. Section 3.2 describes the results of our research, explaining the software size classification, company size classification and the size of Mexican software projects.

3.1 Methodology description

We began this research by doing a systematic search of the literature related to our project. So far, we have not found similar studies at least for Mexican companies. It is important to mention that this study is part of a larger project in which we want to identify good techniques and practices for developing small projects. The problem is that there are not enough tools, processes, methodologies and so on for small software projects. This problem was identified by some of our researchers, because they had been working directly in software development companies.

For this project, we wanted to have a better knowledge of Mexican companies working in software development projects and to gather data about characteristics of their small

[1] The global and independent source of data and analysis for the IT industry. www.isbsg.org

software projects. In order to get this data, we contacted software companies from Mexico and offered a free SCRUM course to members from participating companies. We received answers from 107 Mexican companies. The results of our research are detailed in section 3.2.

3.2 Results

We analyzed data gathered from our survey, where 107 Mexican companies were asked about: a) Their size, and b) The size of the projects they developed. We found that 92% of their projects were micro and small projects, and 8% were medium or large projects. Additionally, we found 84.1% are micro and small companies and they barely use processes, methodologies or tools for developing and managing their small projects.

The survey was mainly focused on the size of projects being developed in Mexico and company size. We wanted to answer the question, what is the size of software projects being developed by micro and small software companies in Mexico?.

3.2.1 Kind of Projects Developed

With our research, we could answer this question: Q1. Since your company was founded, what percentage of projects (micro, small, medium, large) has your company developed? The companies were able to answer this question because we gave them our Project Size Classification Table (Table 3.2.1) in which they saw our classification of the sizes of different projects.

Table 3.2.1 Project Size classification

Project Size	**Features**
Micro	• Size (Lines of source code): from 1,600 to 9,600 • Development time (months): from 1 to3 • Development time (hours): from 160 to 960 • Team size (members): from 1 to 2 • Cost (USD): 1,200 to 4,800*
Small	• Size (Lines of source code): from 9,601 to 38,400 • Development time (months): from 3 to 6 • Development time (hours): from 961 to 3,840 • Team size (members): from 2 to 4 • Cost (USD): 7,501 to 192,000*
Medium	• Size (Lines of source code): from 38,401 to 960,000 • Development time (months): from 7 to 60 • Development time (hours): from 3,841 to 96,000 • Team size (members): from 5 to 10 • Cost (USD): from 192,050 to 4,800,00.00*
Large/Big	• Size (Lines of source code): from 960,000 to ∞ • Development time (months): from 61 to ∞ • Development time (hours): from 96,001 to ∞ • Team size (members): from 11 to ∞ • Cost (USD): from 4,800,00.00 to ∞*
Note	*The cost is different in each country. We took USD 50.00 per man-hour.

Our data analysis showed us than Mexico is a country where mainly micro & small software projects are developed, consisting of 92% of software projects (Fig. 3.2.2.1). Medium & large software projects represent a much smaller percentage-- only 8%. We can see these results in figure 3.2.2.1.

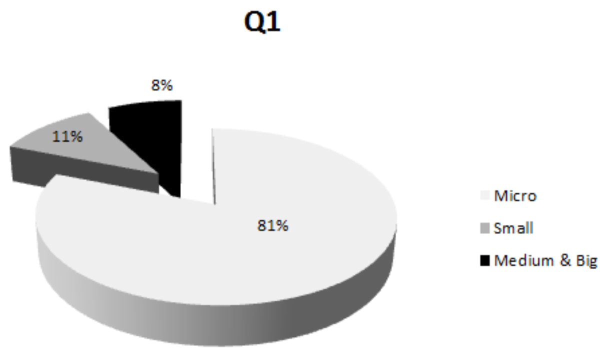


Figure 3.2.2.1 Software projects developed in Mexico

We could also answer another question: Q2: What was the cost of your small projects?

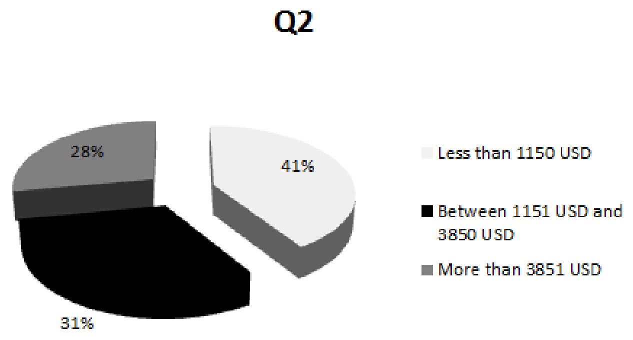


Figure 3.2.2.2 Small software projects cost

The main activity of Mexican companies is to develop small projects, and the price of these small projects was mainly very low. 72% cost less than $3,850 USD ($ 50,000.00 Mexican pesos) (Fig. 3.2.2.2). Thus, we can infer: Mexican software development companies have developed small projects and its costs have been very low. Therefore, we need to give them mainly information to develop and manage micro & small projects.

Additionally, we posed this question: Q3: How long was needed to develop a small project?

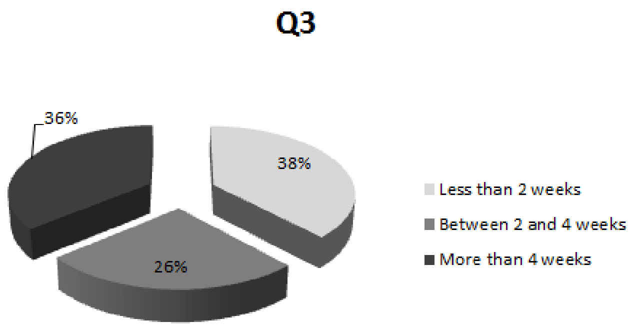


Figure 3.2.2.3 Time to develop small software projects

We can see that Mexican software development companies need only a few weeks to develop a small project. 64% of projects required only one month or less to finish (Fig. 3.2.2.3). The question Q2 showed us the low cost to develop small projects. It may be that the reason for this low cost is the relatively little time necessary to develop this kind of project.

3.2.2 Company Size Classification.

We could answer another question: Q4. How many people work in your company?. The companies had four options: a) 1-5, b) 6-10, c) 11-50, d) 51 or more. When we analyzed the answers, we found that 84.1% were micro & small companies and 15.9% were medium and big (large) (table 3.2.2).

Table 3.2.2 Q2, Mexican Companies: Size classification.

Size	Employees	Number of Mexican companies.	Percentage
Micro	1-10	62	57.9%
Small	11-50	28	26.2%
Medium & Big (Large)	51 or more	17	15.9%
Total	107	107	100%

This table shows us that in Mexico there are more micro and small companies than medium and big or large (Fig. 3.2.2.4). Thus, we need to focus on developing tools, processes, methodologies, model life cycles, and so on, for the micro and small sector.

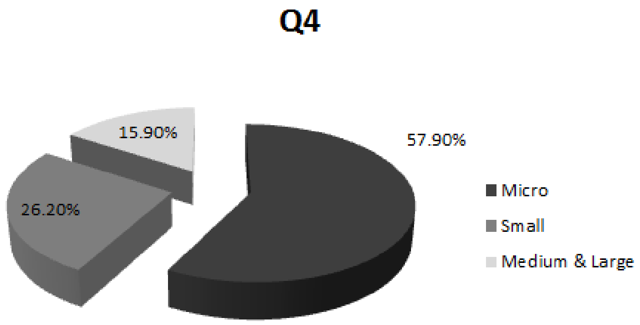


Figure 3.2.2.4 Mexican companies' size

4 Future Work

With our findings, we have established the situation regarding the size of software developed and companies' size in Mexico. Based on these results, we will need to research the following: What kind of tools, methodologies and processes are the companies currently using? Which are most useful for Mexican companies? What must a Mexican company do to successfully adopt a tool, methodology or process to support small projects' administration and development? Perhaps, we will need to make some proposals for new tools, new process or new methodologies focusing mainly on our Mexican context.

5 Conclusion

We have found useful information to develop a strategic plan to help Mexican companies develop quality small software projects. For example, we have found: a) Most of the companies surveyed have developed a considerably high number of small projects. We can thus infer that the main activity of these companies is to develop small software projects. b) The cost to develop small software projects is very low; this could be a problem when the companies need economic resources to buy software or hardware for their operations. c) The time required to develop small projects is only one month or less, although 36% of small projects needed more than one month to be finished. d) Mexico is a country where 84% of software development countries are micro or small. We have not shown what percentage of small projects fail, but in this research we can see: e) Failure percentage in small projects is directly related to software-company size; the reasons for increasing failure in small projects are not clear. Finally we can say: In Mexico, mainly, software development companies need: tools, processes and methodologies to develop small software projects. The small software projects have an important economic impact. It is very important to be aware of the size of companies in our country or region, and to know the size of the projects developed by them, in order to make plans to support them.

6 References

[1] Frederick P. Brooks, Jr. “The Mythical Man-Month“: Essays on Software Engineering, Addison-Wesley, ISBN 0-201-83595-9, 1995.

[2] Watts S. Humphrey, “A Discipline for Software Engineering“, SEI Series in Software Engineering, Addison Wesley, ISBN 0201546108, 2005.

[3] Solvita Berzisa, “Project Management Knowledge Retrieval: Project Classification“, Proceedings of the 8th International Scientific and Practical Conference. Vol II, ISBN 978-9984-44-071-2. 2011.

[4] Capers Jones, “Backfiring: converting lines of code to function points“, Computer, Vol. 28, no. 11, pp 87-88, November 1995.

[5] Stacy Goff, “The Successful Project Profile: A monograph“. Associates, Inc, the ProjectExperts, 2010

[6] ESA BSSC. “Guide to applying the ESA software Engineering Standards to small Software Projects“. (1996).

[7] Jessica E. Lessin and Spencer E. Ante, “Apps Rocket Toward $25 billion in Sales, The Wall Street Journal . http://online.wsj.com/article/ SB10001424127887323293704578334401534217878.html (2013).

[8] Mark C. Paulk. “Using the Software CMM in Small Organizations“. Proceedings of the Pacific Northwest Software Quality Conference and the Eight International Conference on Software Quality , 1998.

[9] Claude Y. Laporte, Frédéric Chevalier and Jean-Claude Maurice “Improving project management for small projects“. ISO Focus+, www.iso.org/isofocus. 2013.

[10] “ Transparency Market Research, Smartphone applications Market –Global Industry Size, Share, Trends, Analysis and Forecasts 2012-2018“.

[11] “ Mobile Statistics”, http://www.mobilestatistics.com, January 2014.

[12] “ Information is Beautiful”, http://www.informationisbeautiful.net/visualizations/million-lines-of-code/ January 2014.

[13] James Persse, “Project Management Success with CMMI“, Prentice Hall Computer, 2007.

[14] Barry W. Boehm, Chris Abts, A. Winsor Brown, Sunita Chulani, Bradford K. Clark, Ellis Horowits, Ray Madachy, Donald Reifer, Bert Steece, “Softwre Cost Estimation With COCOMO II“, Prentice Hall PTR, 2000.

[15] Value Prism Consulting, “Microsoft Office Enterprise Project Management (EPM) Solution Delivers Strong Business Value“, 2009.

SESSION

SOFTWARE ENGINEERING AND APPLICATIONS AND RELATED ISSUES + EDUCATION AND TRAINING

Chair(s)

TBA

A Methodology for Development of Enterprise Architecture of PPDR Organisations

W. Müller, F. Reinert

Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB

76131 Karlsruhe, Fraunhoferstraße 1

GERMANY

Abstract - *The growing number of events affecting public safety and security (PS&S) on a regional scale with potential to grow up to large scale cross border disasters puts an increased pressure on agencies and organization responsible for PS&S. In order to respond timely and in an adequate manner to such events Public Protection and Disaster Relief (PPDR) organisations need to cooperate, align their procedures and activities, share the needed information and be interoperable.*

This paper provides an approach to tackle the above mentioned aspects by defining an Enterprise Architecture (EA) of the organisation and based on this EA define the respective System Architectures. We propose a methodology for the development of EA for PPDR organisations. Our methodology refines architectural artefacts of the OSSAF approach and introduces a lightweight architecture development model relying on capability based planning as the organisational top level approach.

Keywords: *Architecture framework, Public Protection & Disaster Relief, NAF, OSSAF*

1 Introduction

Public Protection and Disaster Relief (PPDR) organisations are confronted with a growing number of events affecting public safety and security. Since these events either expand from a local to a regional and to an international scale or are from beginning affecting multiple countries the pressure on PPDR organisations to be able to cooperate in order to respond timely and adequately to such events increases as well. The need of cooperation demands for aligned procedures and interoperable systems which allows timely information sharing and synchronization of activities. This in turn requires that PPDR organizations come with an Enterprise Architecture on which the respective System Architectures are building. The Open Safety & Security Architecture Framework (OSSAF) provides a framework and approach to coordinate the perspectives of different types of stakeholders within a PS&S organisation. It aims at bridging the silos in the chain of commands and on leveraging interoperability between PPDR organisations. Our work is based on OSSAF and provides the methodology to describe the OSSAF perspectives and views with the adequate models.

2 Related work

The goal of Enterprise Architecture design is to describe the decomposition of an enterprise into manageable parts, the definition of those parts, and the orchestration of the interactions between those parts. Although standards like TOGAF and Zachman have developed, however, there is no common agreement which architecture layers, which artifact types and which dependencies constitute the essence of enterprise architecture.

[7] defines seven architectural layers and a model for interfacing enterprise architectures with other corporate architectures and models. They provide use cases of mappings of corporate architectures to their enterprise architecture layers for companies from the financial and mining sector.

A layered model is also proposed by [10]. The authors propose four layers to model the Enterprise Architecture: A Strategy Layer, an Organizational Layer, an Application Layer, and a Software Component Layer. For each of the layers a meta-model is provided. The modeling concepts were developed for sales and distribution processes in retail banking.

MEMO [11] is a model for enterprise modeling that is based on an extendable set of special purpose modeling languages, e.g. for describing corporate strategies, business processes, resources or information. The languages are defined in meta-models which in turn are specified through a common meta-metamodel. The focus of MEMO is on the definition of these languages and the needed meta-models for their definition.

The Four-Domain-Architecture [8] divides the enterprise into four domains and tailors an architecture model for each. The four domains are Process domain, Information / Knowledge domain, Infrastructure domain, Organization domain. Typical elements for each domain are also provided. The authors also provide proposals how to populate the cells of the Zachman framework with architectural elements.

The Handbook on Enterprise Architecture [9] provides methods, tools and examples of how to architect an enterprise through considering all life cycle aspects of Enterprise Entities in the light of the Generalized Enterprise Reference Architecture and Methodology (GERAM) framework.

None of the papers addressing Enterprise Architectures covers the special needs of PPDR organizations with their need on timely cooperation, alignment of procedures, and interoperability needs across different organizations.

3 Approach

3.1 Enterprise Architecture Frameworks

Supporting the development of dedicated enterprise architecture is the task of Enterprise Architecture Frameworks (EAF). According to [3] more than 50 published frameworks for EA exists, for example ADS, AGATE, EAF, GERAM, MODAF, PERA, TISAF, E2AF, CIMOSA, SABASA, OBASHI, ARIS etc. to name a few. The most well known frameworks are the The Open Group Architecture Framework (TOGAF) [1], Zachman Architecture Framework (ZAF) [4] and the NATO Architecture Framework (NAF) [5].

In general EA frameworks have different characteristics concerning intension and content. Some provide a methodology, others provide templates and meta-models, some provide governance aspects, others also provide tool support and some provide combinations of parts or all of them. The intensions covered range from management support through Government & Agencies, Military, Interoperability and Manufacturing-specific to pure technically oriented frameworks.

3.2 Open Safety & Security Architecture Framework (OSSAF)

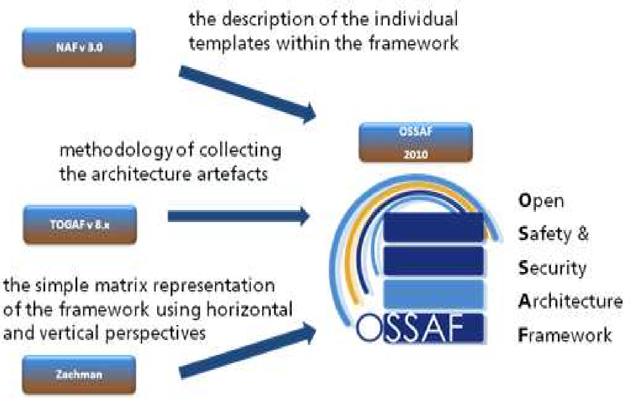
For PPDR organizations, [2] proposes the Open Safety & Security Architecture Framework (OSSAF). The framework incorporates concepts of several mature enterprise architecture frameworks such as the Zachman Architecture Framework (ZAF), the TOGAF framework and the NATO Architecture Framework (NAF) [5]. Reusing fundamental concepts provides a sound founding to OSSAF. There is also an explicit statement within the whitepaper, that a mapping between OSSAF and other existing frameworks is possible. Nevertheless OSSAF builds mainly on (see Figure 1):

1. The methodology of collecting information and artifacts contributing to the architecture from TOGAF.

2. The two dimensional matrix representation of the framework for structuring the different perspectives from Zachman.

The OSSAF whitepaper [2] also mentions that the NAF meta-model and views may be used where suitable for describing the content of the different perspectives, but does not provide details on the application of the NAF views.

Figure 1: Inputs to OSSAF


NAF v 3.0
the description of the individual templates within the framework
OSSAF 2010
methodology of collecting the architecture artefacts
TOGAF v 8.x
Open Safety & Security Architecture Framework
the simple matrix representation of the framework using horizontal and vertical perspectives
OSSAF
Zachman

OSSAF proposes a total of four perspectives and a total of twenty views. In general it depends on the intention of the architecture under development which views are actually instantiated. In other words the views can be tailored to the specific needs of the architecture under consideration.

3.3 EA development methodology for PPDR organizations

The proposed methodology for the development of enterprise architecture of PPDR organizations follows a pragmatic approach, looking at an “enterprise” as the joint undertaking of one or more organizations with PS&S responsibilities that operate across a distributed and often complex environment. This understanding states, that we see an enterprise in this context as nonprofit-oriented organizations or complex structures of organizations (inter-organizational aspect of enterprise definition) such as national PPDR organizations, for example national police or fire-fighter organizations.

We see the enablement of PPDR organizations regarding their agility, interoperability and mutual networking as an evolutionary course of action based on a planned and predictable process which has to deal with highly complex issues. In computer science the divide and conquer principle is often used to handle complex problems. We refer to this principle to support the evolutionary process in order to deal with chunks of smaller complexity. These chunks with reduced complexity are the Capabilities. One can understand a Capability according to [1] as:

”An ability that an organization, person, or system possesses. Capabilities are typically expressed in general and high-level terms and typically require a combination of organization, people, processes, and technology to achieve.

For example, marketing, customer contact, or outbound telemarketing."

The introduction of Capabilities as main planning items leads to the approach of capability based planning. This is of course not a new approach. It is applied for years in the context of the NATO in order to provide interoperability across the whole spectrum of system solutions required for overall portfolio of the NATO enterprise putting extra attention on interoperability from technical up to the semantic level.

As a side note, there may arise some confusion on capabilities in relationship to requirements. In contrast to a capability, a requirement must be understood as a singular documented physical and functional need that identifies a necessary attribute, characteristic, or quality of a system to have value and utility to a customer, organization, internal user, or other stakeholder [6].

Seeing capability based planning as the overarching guideline, our actual approach for the development of an EA proposes scenarios as main input. Preceding to the definition and development of scenarios the first step in the development approach is the definition of Visions and Goals in order to depict an overall strategy including the winning of supporters for the overall architecting approach.

Keeping visions and goals in mind the next step is the development of representative scenarios. One can see the scenario development as the first concrete step towards enterprise architecture and being crucial for our proposed methodology. Therefore this step requires a very close cooperation with operational end-users. The development is an iterative and interactive process with a successive refinement of operational procedures, and requirements. Scenarios derived include as-is scenarios and also to-be scenarios or a mixture of both of them. The to-be scenarios are particular important to identify lacks in capabilities. Since the scenarios are developed in close cooperation with end-users, it is assumed that they reflect user's needs in a sufficient manner. In case of encountering deficits while designing the architecture in that the scenarios don't provide enough input to the development of the perspectives/views the OSSAF Engagement Questionnaire will be used to obtain the required information. This questionnaire is already defined in OSSAF. The further steps include:

- Define scope and principles of the architecture (for example require to use the SOA paradigm for the architecture, kind of architecture e.g. Reference architecture or Target architecture to be developed)
- Refer to or define/adapt a common modeling vocabulary (further addressed in 3.3 Definition of the Modeling Vocabulary)
- Define stakeholders addressed
- Tailor the architecture perspectives and views (architecture artifacts) according to the kind of architecture addressed.
- Map architecture artifacts to the corresponding stakeholders
- Analyze Scenarios/Use Cases on operational Capabilities required. This is a creative process and one has to align actual capabilities with further capabilities addressed in order to identify possible lacks of capabilities. As a starting point serve the six top-level capabilities of OSSAF in order to classify further capabilities derived.
- Analyze Scenarios/Use Cases on functional and non-functional requirements
- Derive Capability taxonomy und dependencies (capability based planning concept)
- Identify and analyze operational Context including: Nodes conducting (operational) Activities, Players, Activities, Information flows, Processes and Constraints (e.g. operational rules))
- Identify Services (operational as well as technical services) and Systems in order to support the capabilities, operational requirements and information exchange needs. Take existing technical Standards and Service catalogues under consideration. That means, re-use existing solutions first before developing new ones, being normally not that mature.
- Design Services (including communication services) and Systems (including communication systems) as well as their interactions (logical definitions of systems and services) and derive conceptual information models form the operational information exchange requirements.
- Define technical implementation of Services and Systems (physical definition) and a data models corresponding to the conceptual information model
- Define the standards that have to be considered or describe emerging standard configurations (products) resulting from the architecture approach.
- Finally validate the architecture with operational-end users and the stakeholders. In general it shall be noticed, that the proposed method is normally conducted in several iterations after defining the scenarios itself, although it may be necessary to refine them during further architecture development.
- After the final architecture validation, develop a potential migration plan which the decision-makers responsible for the organizational enhancements may adopt.

3.4 Definition of the modeling vocabulary

Defining a common vocabulary/meta-model in order to describe the architecture, i.e. its components and relationships is a very beneficial task. It enables the description of the architecture in a consistent and coherent way. Referring to such a meta-model also supports the use of tools for architecture modeling. Since the OSSAF framework already proposes to use NAF views where suitable as templates for describing the OSSAF views and the NAF views defines a vocabulary, our approach is to use NAF as the modeling vocabulary where suitable.

Table 1: Mapping of NAF templates to OSSAF views

OSSAF Views	OSSAF Perspectivec							
	Strategic		Operational		Functional		Technical	
	Vision & Goals	NAV-1 NCV-1	Use Case Scenarios	No proper NAF view	Systems & Services	NSOV-1 NSOV-2 NSOV-3 NSOV-4 NSOV-5 NSV-12	Solution Context	No proper NAF view
	Capability Planning	NCV-2 NCV-4	Operational Concepts	NCV-4 NCV-5 NCV-6 NOV-1	Functional Requirements	NSV-2d NSV-4 NSV-5 NSV-6 NSV-7 NSV-10a	Standards & Protocols	NTV-1
	Funding Model	No proper NAF view	Operational Nodes Model	NOV-2	Systems Connectivity Model	NSV-1 NSV-2a NSV-2b	Device Connectivity Model	NSV-2a NSV-2b NSV-2d
	Laws & Regulations	No proper NAF view	Organization Chart	NOV-4	Systems Interface Model	NSV-1 NSV-2 NSV-3	Product Specification	(NTV-1)
	Local Market Landscape	No proper NAF view	Process Model	NOV-5 NOV-6a NOV-6b NOV-6c			Product Configuration	NTV-3
			Information Exchange Model	NOV-3 NOV-7				

In general NAF provides a mature common meta-model to describe the contents of the corresponding views. Every view contains a section of the overall meta-model in order to describe view-specific contents and relations. In addition to the concepts and relationships the meta-model also defines the semantics of each of these elements [5]. The reader should refer to the NAF documentation [5], especially CHAPTER 5 “NATO Architecture Framework Metamodel (NMM) and Architecture Data Exchange Specification (ADES)” and CHAPTER 4 “Architecture Views and Subviews” in order to get a detailed insight and understanding of NAF.

Being in proposal state, OSSAF actually does not define the contents of the NAF meta-model being part of the corresponding OSSAF views. Only general hints are given. Therefore, at first, a mapping between OSSAF views and NAF views has to be established. Table 1 summarizes the first results of a general mapping of NAF views to OSSAF perspectives. Each column represents a perspective defined by the OSSAF framework. The rows represent the views per perspective, each with a specific semantics defined by OSSAF. To the right of each OSSAF perspective we refer to the corresponding NAF-views which we see suitable for representing the semantics required by OSSAF. For example to describe the “Capability Planning” view of the “Strategic” perspective it is suggested to use the NAF Capability View-2

("NCV-2") and Capability View-4 ("NCV-4") view accordingly. In order to describe the OSSAF "Operational Concepts" view of the "Operational" perspective we refer to several NAF views form the NAF Capability and Operational descriptions. These are the Capability dependencies View ("NCV-4"), the Capability to organizational deployment mapping View ("NCV-5"), the Operational activity to capability mapping View ("NCV-6") and finally form the NAF Operational description the High level operational concept description View ("NOV-1"). Another example for the suggested re-use of NAF views in order to describe the required semantics of the OSSAF is given for the "Systems Interface Model" view of the OSSAF Functional perspective. Here we refer to the NAF Systems descriptions in form of the System Interface description ("NSV-1"), the Systems communications description ("NSV-2") and the System to System matrix ("NSV-3") views defined in NAF in order to describe the corresponding OSSAF view.

As the table shows, a direct mapping between OSSAF and NAF views is not always possible ("No proper NAF view" comment at the corresponding matrix entry in the table). For example the OSSAF "Funding Model" could not directly be mapped to a corresponding NAF view. In such cases suitable representations will be proposed in a follow-on work.

4 Conclusions and further work

The proposed methodological approach provides a starting point to the development of Enterprise Architectures for PPDR organizations. Based on the Enterprise Architecture, specific System Architectures may be derived.

The proposed EA methodology will be used in the SALUS project [12] to define the Enterprise Architecture of PPDR organizations and the System Architecture of the communication network for those organizations. As the need arises it will be further refined. In addition, a tool support for modeling the different NAF views is under development. This tool captures the relevant meta-model parts of NAF as an UML-profile extension which enables modeling in an UML-style. It will be used in SALUS to support the design and development of the above mentioned architectures.

Acknowledgement: The work described in this paper was partly funded by the European Commission within the European Seventh Framework Programme under Grant Agreement 313296, SALUS - Security And InteroperabiLity in Next Generation PPDR CommUnication InfrastructureS

5 References

[1] Website TOGAF, http://www.opengroup.org/togaf/

[2] Open Safety & Security Architecture Framework (OSSAF), http://www.openssaf.org/download

[3] D. Mattes „Enterprise Architecture Frameworks Kompendium“, Springer-Verlag Berlin Heidelberg, 2011

[4] Website Zachman Framework, http://zachman.com/

[5] NATO Architecture Framework Version 3, ANNEX 3 TO AC/322(SC/1-WG/1)N(2007)0004

[6] Website Wikipedia, http://en.wikipedia.org/wiki/Requirement

[7] R. Winter, R. Fischer "Essential Layers, Artifacts, and Dependencies of Enterprise Architecture", Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW'06), IEEE Computer Society, 2006

[8] B. IYER, R. Gottlieb "The Four-Domain-Architecture: An approach to support enterprise architecture design", IBM Systems Journal, Vol 43, No 3, 2004, pp. 587- 597.

[9] P. Bernus, L. Nemes, G. Schmidt (Editors) „Handbook on Enterprise Architecture“, Springer, 2003.

[10] Ch. Braun, R. Winter "A Comprehensive Enterprise Architecture Metamodel and Its Implementation Using a Metamodeling Platform", In: Desel, J., Frank, U. (Eds.): Enterprise Modelling and Information Systems Architectures, Proc. of the Workshop in Klagenfurt, GI-Edition Lecture Notes (LNI), Klagenfurt, 24.10.2005, Gesellschaft für Informatik, Bonn, P-75, 2005, pp. 64-79.

[11] U. Frank, "Multi-Perspective Enterprise Modeling (MEMO) - Conceptual Framework and Modeling Languages", Proceedings of the Hawaii International Conference on System Sciences (HICSS-35), 2002, p. 3021ff.

[12] SALUS: Security and interoperability in next generation PPDR communication infrastructures. http://www.sec-salus.eu/

Positive Train Control: Concepts, Implementations, and Challenges

David J. Coe and Jeffrey H. Kulick
Department of Electrical and Computer Engineering
The University of Alabama in Huntsville, Huntsville, Alabama, USA

Abstract - *For over a hundred years train control has been largely distributed. To prevent collisions, railway track systems were divided into a series blocks ranging in length from a few miles to tens of miles, and a signaling system was developed to constrain movement of trains between blocks. The imprecise nature of position information afforded by the block-oriented system required as a general rule that no two trains were allowed to occupy the same block. The advent of centralized Positive Train Control (PTC) coupled with precision navigation and remotely controllable switches and signals allow for the development of more optimal scheduling algorithms for improved safety and railway throughput. This paper outlines key aspects of PTC deployment, describes some of the new rules and conditions that must be formulated, potential control systems for implementing these rules, and a PTC test bed currently in development at The University of Alabama in Huntsville.*

Keywords: Positive train control, GPS Challenged Navigation, Inertial Navigation, Software Safety Engineering

1 Background

Positive Train Control (PTC) is a congressionally mandated computerized control system for the nation's railroads that will allow centralized management of safety critical conditions such as a detecting, slowing and stopping speeding trains by remote control [1]. The system was mandated following the 2008 Chatsworth train collision in which a railroad operator, distracted while texting, ran a signal leading to a fatal head-on collision between a passenger train and freight train that resulted in 25 fatalities and 130 injuries [2]. Such operational errors that might lead to an accident might be prevented through a centralized safety critical system that could remotely intervene when an unsafe condition was detected.

Deployment of a comprehensive PTC system is an enormous undertaking, costing freight railroads alone an estimated $8 billion [3]. This task includes not only the design and development of suitable software but also the precise mapping of 60,000 miles of railroad right-of-way and associated fielded assets and the deployment of massive amounts of infrastructure including central office hardware, 22,000 instrumented locomotives, 36,000 wayside signaling and network hardware, as well as 4,800 switches and 12,300 signals, both remotely controllable. Locomotives instrumentation must include precise location detection and communication hardware, primarily based on Global Positioning Systems (GPS) and Inertial Measurement Systems (IMUs) for localization and speed detection in GPS challenged areas such as in deep valleys, tunnels, and underground yards such as found in inner city terminals.

The advent of such a massive and costly instrumentation of the nation's railroads, however, provides a great opportunity to deploy computer control systems for not only safety considerations such as collision avoidance but performance enhancing activities such as precise scheduling that will allow reduced separations between trains. To take advantage of these additional capabilities in a safe manner will require the development of railroad control software that incorporates modern vehicle capabilities such as those found in Google Cars and aircraft under RPN (required precise navigation). The critical nature of these systems will also require evaluation and mitigation of security concerns.

1.1 Conventional Locomotive Location Hardware: The Track Circuit

The *track circuit* has traditionally been used to detect occupancy of a track block by a locomotive or other rail rolling stock and to signal the state of occupancy to any train operators in adjacent blocks [4]. Invented by Dr. William Robinson in the 1870s, the classic track circuit shown in Figure 1 utilizes the two track rails as an integral part of an electrical circuit that controls block occupancy signals. A voltage source such as a battery supplies an electric potential between the two rails. The signal relay solenoid coil is also connected across the two rails such that it remains energized (conducting electricity) when there is no train located on that section of the track. With the solenoid energized, the signal relay presents the track-clear signal to oncoming trains. When the axle of a locomotive or roll stock enters the block, the axle electrically shorts the two rails together, de-energizing the solenoid coil resulting in display of the track-occupied signal. The classic track circuit requires adjacent blocks to be electrically isolated from each other. Modern variations of the track circuit allow for the use of non-isolated welded rails.

It is important to note that track circuit occupancy signaling is automatic and decentralized, requiring no human intervention to set the occupancy signals. The track circuit also plays a key role in the event of an accident or during

Copyright © 2014 CSREA Press, ISBN: 1-60132-286-0; Printed in the United States of America

track maintenance since work crews may deliberately create an electrical short between the rails to force display of the track-occupied signal to block entry of a train into a particular block. The simplicity and reliability of the track circuit has made it a key element of modern block signaling systems for over 100 years and to this very day.

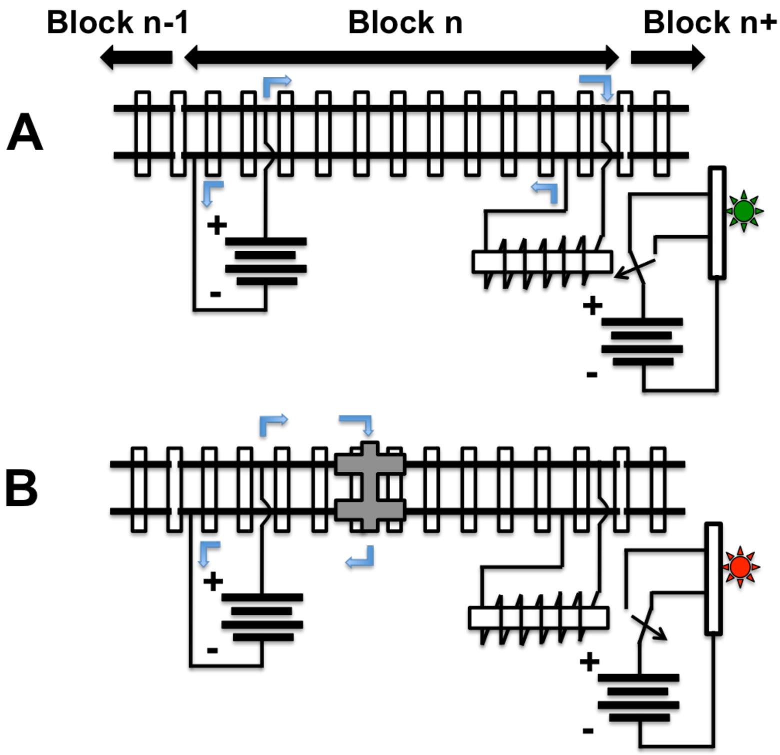


Figure 1 – Track circuit schematic showing direction of current flows for both (A) unoccupied and (B) occupied conditions [4].

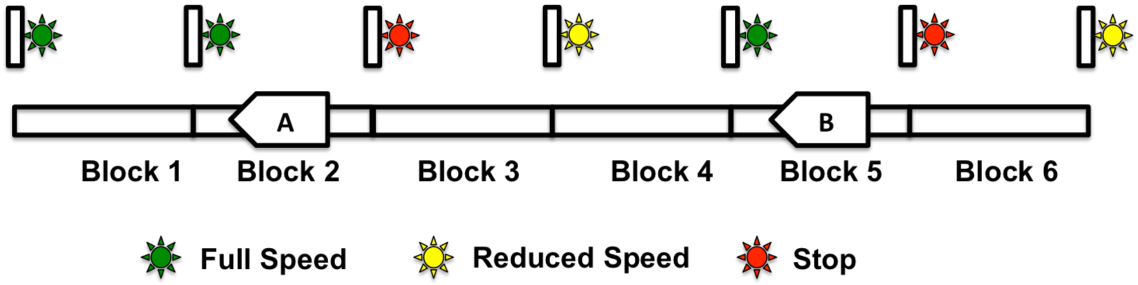


Figure 2 – Automated Block Signaling diagram showing signals visible to train A and train B as they travel from right to left on the same track [5].

1.2 Automated Block Signaling (ABS)

ABS is an automated, decentralized approach that relies on the use track circuit-based occupancy detection to maintain a safe separation between trains. While specific ABS rules vary by railroad, track systems are divided into blocks that may be of varying length, and a system of automated signals convey block vacancy and occupancy information to train operators. Figure 2 above illustrates how block signaling logic may be used to maintain safe train separation [5]. Given that trains A and B occupy blocks 2 and 5, respectively, any trailing trains would see a red signal for those two blocks, indicating that they must stop since blocks 2 and 5 are currently occupied. Looking ahead, train A sees only green signals ahead so it may proceed to the left at full speed from block 2. Train B sees a green signal before block 4, so it may proceed at full speed from block 5 into block 4, but the yellow signal prior to block 3 indicates that train B must reduce speed if it enters block 3. Assuming that train A is traveling at full speed at it travels from block 2 into block 1, the trailing train B should be unable to catch up if traveling at reduced speed through block 3. Should train A stop for some reason, a red signal will be shown at the entry point to its current block to ensure that any trailing train stops before entering the block occupied by train A.

A key issue with ABS is that of track utilization. Suppose that a mile long train currently occupies a twenty-mile long block. Independent of the precise location of the train within the block, all of the track within the block may be unusable since the current ABS system bases its signaling decisions on block-level occupancy detection. Given a more precise measurement of train position, one or more additional trains could safely occupy the twenty-mile long block provided that adequate stopping distance is maintained.

2 Positive Train Control (PTC)

The block diagram in Figure 3 shows the four primary subsystems of the Vital Electronic Train Management System (V-ETMS), which is a modern positive train control system currently being deployed by the large freight railroads in the United States [6] in compliance with the new directive. The Office Segment provides interfaces to the network time server, the dispatcher, and the other backend systems. The Wayside Segment consists of the monitored and remotely controllable switches and signals deployed trackside along with their associated Wayside Interface Units (WIUs) that connect each of these assets to the Communications Segment. The Wayside Segment may forward block status information directly to the Locomotive Segment via in-cab signaling. The Locomotive Segment consists of one or more Train Management Computers that serve as the central manager for in-cab signaling, the GPS receivers, the in-cab video displays, as well as interfaces to the throttle, braking, whistle, and other locomotive systems including the event recorder. The Communications Segment, still under development, will serve as the central interconnection medium for the Office, Wayside and Locomotive segments. Possible means of communication under consideration include WiFi, or WiMax networks, cellular networks, satellite networks, or custom 900MHz or 220MHz data radios.

3 Dealing with the Unexpected Unexpected

A key problem with the computer mediated dynamically changing track configurations supported by PTC is dealing with changes in track properties such as closed, limited speed, track workers present, etc. that occur thousands of time a day across the national railway network. When designing safety critical systems, engineers attempt to anticipate possible fault mechanisms and design the system in such a way as to mitigate the associated risks. Proven techniques have been developed such as the use of redundancy and the use of sensor-signal systems to warn of dangerous conditions are frequently employed as a result. Consider, for example, the design of a train control system. One can design the system for “the expected” normal operating conditions and “the expected unexpected” which are the anticipated failure conditions whose impact can be mitigated since they were anticipated during the design process. Examples of “the expected” and “the expected unexpected” for a train control system appear below.

- “The expected”
  - Normal running, slowing, and stopping by an operator under signal control.
  - A signal correctly indicating the occupancy state of an upcoming block.

- “The expected unexpected”
  - A sensor detects a driver falling asleep and triggers an alarm to wake the driver.
  - An emergency braking system stops a train when the primary braking system fails.

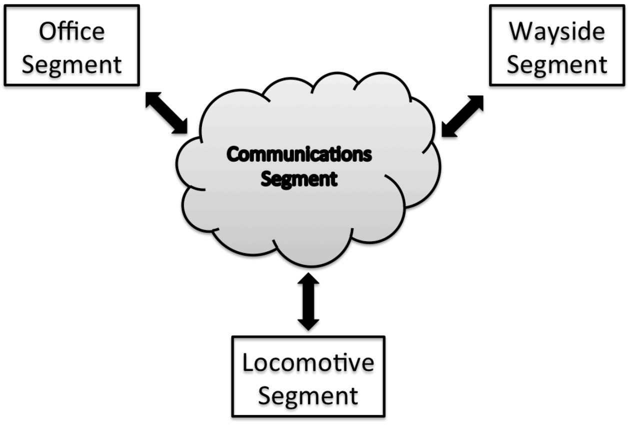


Figure 3 – Block diagram showing major V-ETMS subsystems [6]

The more difficult problem is the design of the system to be safe in spite of what we call the *unexpected unexpecteds*. If one views the system as a state machine, these types of occurrences may place the system into states that the designers may never have envisioned as existing or possible. The real challenge for designers is to craft the control system in such a way that if the system ends up in an unanticipated state, that the existing control logic steers the system back into one of the known states. Examples of such unanticipated events appear below.

- "The unexpected unexpected"
  - A driver becoming distracted by texting who fails to follow signals as a result.
  - A train failing to stop after both the primary and backup braking systems fail.

4 When the Flapping of Butterfly Wings Cause a Hurricane

In a situation where a rule system is static, there are proven methods and techniques, such as the use of a theorem proving or model checking methodologies, for verifying that the rules are consistent with each other, that there are no deadlocks, etc. Rule changes under PTC will be more dynamic in nature to address emerging situations such as railway crossing collisions and the need for repairs to the tracks, signals, or the trains themselves. These emerging events that are local in nature (the butterfly) may directly impact schedules and rail traffic nationally (the hurricane).

Consider, for example, a master schedule that has created a gang of closely following trains along a main railway line. If the lead train must stop for whatever reason, then that emerging local condition must first be detected and signals issued to slow or stop the trailing trains to prevent a collision. The overall system state may now be such that the master schedule is now impossible to execute safely. For the system to recover gracefully, preferably in an automated fashion, adjustments to the master schedule must be identified, verified as safe, and implemented in a timely fashion, even if those changes involve rescheduling and rerouting of some combination of trains scattered across the United States in addition to those trailing trains locally impacted by the emergent event.

The dynamic nature of these events may necessitate the development of new techniques for modeling the rule systems in such a way that the consistency, correctness, completeness, and safety of the modified rules and schedules can be verified in real or near real time to ensure safety and to maintain efficient track utilization. Moreover, the overall impact of certain new capabilities with PTC, such as the shorter time intervals between closely following trains, may have unintended consequences upon highway traffic at crossings such as traffic backup on surface streets due to the now more frequent transit of gangs of trains through the crossing.

5 PTC Test Bed at UAH

The Department of Electrical and Computer Engineering at The University of Alabama in Huntsville has previously developed a model railroad test bed for software safety engineering. As described in earlier publications [7] the test bed consisted of an oval track layout with two passing sidings and multiple detection and control capabilities. In the past detection systems included magnetic and optical sensors, as well as electrical contact sensors using vehicle impedance as a trigger, much like the classic track circuits used on real railroad systems. Graduate and undergraduate students have used the test bed to develop train-scheduling software to manage train operations and safety monitoring systems that will detect and mitigate conditions that may lead to a collision using aviation safety engineering standards such as DO-178C, ARP4761 and ARP 4754. The test bed has also been used as a platform for exploring the use of various sensors for train localization including video-based tracking systems [8] and ultra wideband synthetic aperture radar systems for all weather use [9].

Over the past year, a new model railroad test bed called UAH OnTrack has been developed to explore the software safety engineering issues associated with Positive Train Control [10]. As with V-ETMS and other real world PTC systems, the new test bed incorporates precision navigation in the form a sensor package carried by each train that will provide precise train localization information to the scheduling and safety systems. The original test bed detection systems could only determine a train's position to within a few feet since the previous system mimicked conventional block-oriented detection and signaling systems. Trains operating on the new PTC test bed will incorporate a localization sensor package consisting of a MEMS-based IMU and a tie counting system that affords train localization to within 0.5 cm. A photograph of the sensor package mounted in a gondola car is shown below in Figure 4.

6 Future Work

Upon completion of the current test bed, shown in Figure 5, work will proceed in developing modeling and verification techniques for the dynamically changing rules and track configurations that will be experienced through Positive Train Control and precise scheduling. This may include verification of message passing protocols to ensure that safety messages are properly received and instantiated in the railway model to verification that recovery rules for unwinding and recovering from unsafe states are themselves not introducing yet even more unsafe configurations.

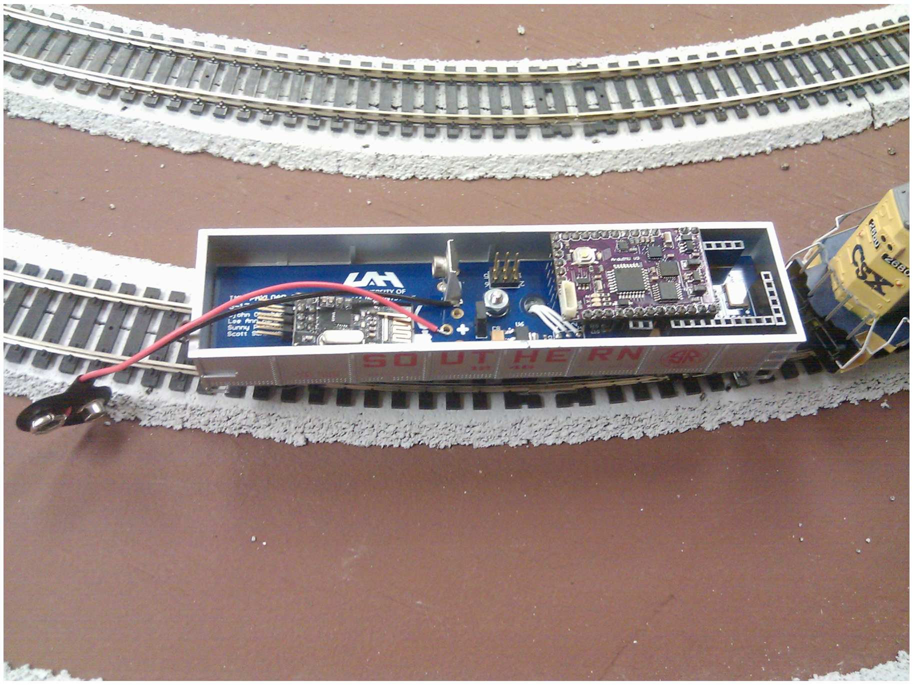


Figure 4 – Photograph of IMU/tie counter sensor package mounted in a gondola car for testing.

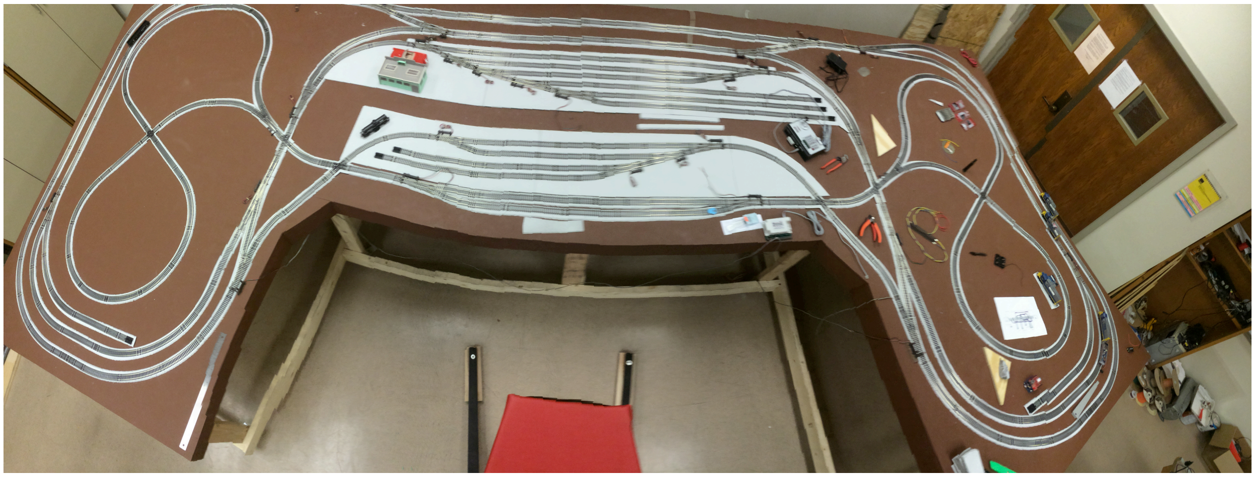


Figure 5 – Panoramic photograph of PTC test bed under construction.

7 References

[1] Rail Safety Improvement Act of 2008, URL http://www.fra.dot.gov/eLib/Details/L03588

[2] Kevin Clerici, "Officials push plan to avoid rail collisions", *Ventura County* Star, September 16, 2008, URL http://www.vcstar.com/news/2008/sep/16/tragedy-of-metrolink-train-111-looking-ahead-to/

[3] Association of American Railroads, "Positive Train Control", April 2013, pp. 1-4, URL https://www.aar.org/keyissues/Documents/Background-Papers/Positive-Train-Control.pdf

[4] American Railway Association, *The Invention of the Track* Circuit, New York, 1922

[5] Mark D. Bej, "Railway Signalling and Operations: Automated Block System", URL http://broadway.pennsyrr.com/Rail/Signal/abs1.html

[6] Wabtec Railway Electronics, "Vital Electronic Train Management System (V-ETMS) Concept of Operations v1.0", March 24, 2010.

Copyright © 2014 CSREA Press, ISBN: 1-60132-286-0; Printed in the United States of America

[7] David J. Coe, Joshua S. Hogue, and Jeffrey H. Kulick, "Software Safety Engineering Education," *2011 International Conference on Software Engineering Research and Practice (SERP'11), WORLDCOMP 2011*, July 18-21, 2011, Las Vegas, NV.

[8] Travis Cleveland, David J. Coe, and Jeffrey H. Kulick, "Video Processing for Motion Tracking of Safety Critical Systems," *2013 International Conference on Software Engineering Research and Practice (SERP'13), WORLDCOMP 2013,* July 22-25, 2013, Las Vegas, NV.

[9] Thu Truong, Michael Jones, George Bekken, "Senior Project: UWB SAR, Ultra Wideband Synthetic Aperture Radar", Senior Design Project, University of Alabama in Huntsville, November 2012, URL http://www.timedomain.com/UAH%20Senior%20Project%20-%20Final%20Presentation%20V3.pdf

[10] Scott M. Schiavone, , SJohn Chambers, Sunny J. Patel, Lee Ann Hanback, David J. Coe, Jason Winningham, B. Earl Wells Jr, Jeffrey H. Kulick, "UAH OnTrack: Precision Navigation System for Research on The Software Safety Issues of Positive Train Control", *2014 International Conference on Software Engineering Research and Practice (SERP'14), WORLDCOMP 2014.*

Proposal of Avatar Generation System based on design according to generation of comic character

Reiko KUWABARA

Graduate School of Engineering, Toyo University
Kujirai2100, Kawagoe-City, Saitama, Japan
s36c01400047@toyo.jp

Takayuki FUJIMOTO

Graduate School of Engineering, Toyo University
Kujirai2100, Kawagoe-City, Saitama, Japan
me@fujimotokyo.com

***Abstract*— Character design of animation and comic has been sensitively reflects the fashion of the era. For example, in the 1970s, when mini skirt is prevalent in society, there were many characters of comics that had been serialized in that time wear a mini skirt. In this research, I investigated features of the comics which were popular in each generation to the 2000s from the 1960s in Japan, and analyzed the correlation between them and fashion or design which were popular at that generation. Based on the results of research and analysis, I suggested the entertainment applications that generate an avatar that reflects the age and generation. When users make an Avatar by choosing the fashion and hairstyle of choice, it determines whether the user's sense is closer to which generation.**

I. INTRODUCTION

Character design of animation and comic has been sensitively reflects the fashion of the era. In this research, I investigated features of the comics which were popular in each generation to the 2000s from the 1960s in Japan, and analyzed the correlation between them and fashion or design which were popular at that generation. Based on the results of research and analysis, I suggested the entertainment applications that when users make an Avatar by choosing the fashion and hairstyle of choice, it determines whether the user's sense is closer to which generation.

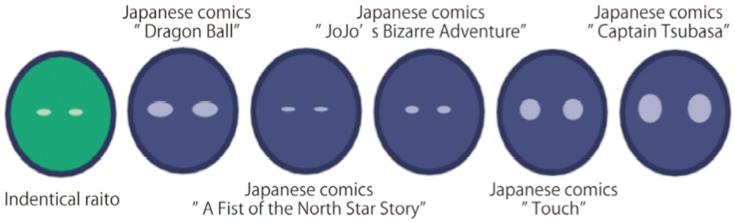
II. RESEARCH ON DESIGN STRUCTURE OF THE COMIC CHARACTERS OF THE 2000S-1960S

In this research, at first, I investigated the design structure of the character of the popular comics in Japan to the 2000s from the 1960s. From there, I classified the shape of the part of the character, and compared percentage of body proportion and the human anatomy with real people by generation.

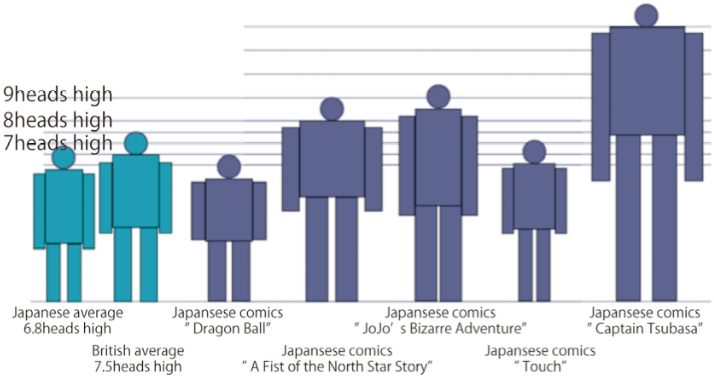
A. Case 1: Character structure of popular comics of the 1980s

I will show below the design structure of the character of popular comics of the 1980s.

Comic characters in the 1980s were seen a variety of design. Some have head body design is close to a real person, others have lower design. Although some have head and body design is extremely high. However, as a common point, most characters are drawn to the long legs extremely. They have also a feature that eye is drawn to the slightly bigger than the actual person is large.


Japanese comics " Dragon Ball"
Japanese comics " JoJo' s Bizarre Adventure"
Japanese comics " Captain Tsubasa"
Indentical raito
Japanese comics " A Fist of the North Star Story"
Japanese comics " Touch"

(Figure 1 The ratio of the face and eyes of Japanese popular comic characters of the 1980s)



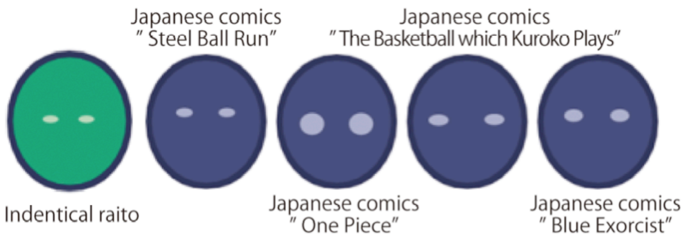
(Figure 2 Head and body and physique of Japanese popular comic characters of the 1980s)

B. Case 2: Design structure of the comic characters of five years

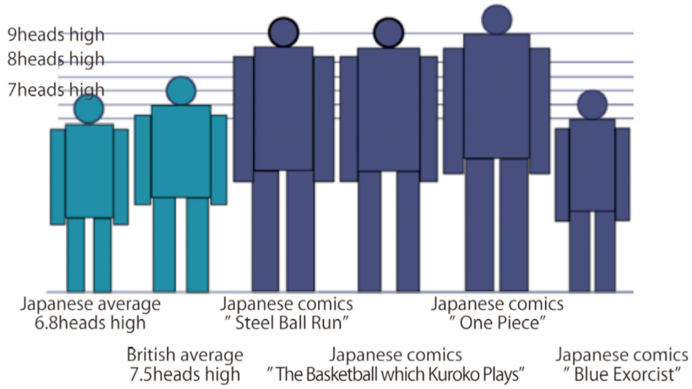
I will show below the design structure of the comic characters in the recent five years from 2013.

Copyright © 2014 CSREA Press, ISBN: 1-60132-286-0; Printed in the United States of America

Many comic characters have feature which they are 7.5 or more heads tall. It is higher than the real people. In addition, there is a tendency that the foot is long. In the 1980s, the eyes were drawn extremely large, but recently, comic characters that size of the eye is drawn close to the structure of real people have appeared.



(Figure 3 Ratio of eye and face of Japanese popular comic characters of the past five years)



(Figure 4 Head and body and physique of Japanese popular comic characters of the past five years)

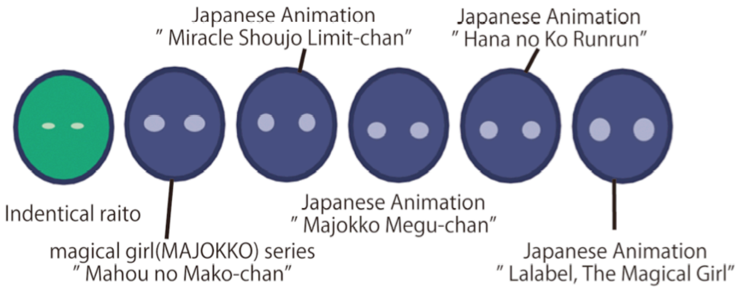
III. Case 3: Design structures of "Majokko" series

A. *Features of character design of Majokko series*

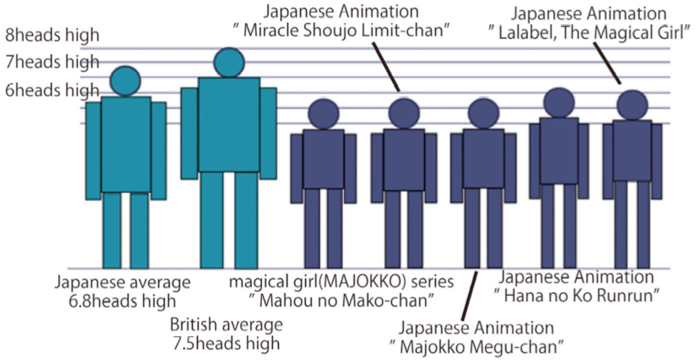
"Majokko(magical girl = mahō shōjo)" animation series is Girl's animation series that produced in Japan in the 1960s and follow to today. Even outside of Japan, they are viewed widely and has produced a number of similar various works. The magical girl (mahō shōjo) series reflects greatly girl's fashion in many cases. Characters of magical girl (mahō shōjo) series are drawn in a lowered head and body than real people. They are almost from 4.5 to 6.2 heads tall. Eye is extremely larger than the characters of comics for boys. That is, the eye occupies more than one-seventh of the face in many cases. In addition, there are tendency that the width of the eye are narrower than the characters of comics for boys, and the position of the eye is placed lower.

B. *Correlation of the epidemic in the real world and comic characters design*

By comparing magical girl (mahō shōjo) animation aired and the female idol that was in vogue at that time, I found that they are largely approximate. For example, in magical girl (mahō shōjo) anime of the 1970s, many of the characters wear mini skirt. Socially, in the 1970s, miniskirt was boom. Also, even if I look magic girl (magical girl = mahō shōjo) anime in recent years, I found a common point that group. For example, there are heroine of more than one person in "Magical Girl Madoka ☆ Magika" was popular in the 2011 and Idol which was popular in those days was the group named AKB48. I considered Majokko series is strongly influenced by idle at the time.



(Figure 5 The ratio of the face and eye of "Majokko" series)



(Figure 6 Body and head and physique of Majokko series)

IV. Avatar generation System based on classification of epidemic animation according to generation

A. *Abstract of System*

I will show below an abstract of the smart phone application that I propose in this research. By selecting a body part of a design symbolizing the comic and anime which are popular in the age and generation stored in the application, you combine and produce an avatar original. Further, the tendency of the generation of the selection part of the avatar generated avatar you have produced is determined " whether the sense which generation " generally. Choice is a 12 the following items.

①Shape of eye ②Size of eye ③Shape of eyebrows
④Shape of nose ⑤Shape of mouth
⑥Size of mouth ⑦Hairstyle
⑧Shape of face and outline ⑨Scale of head and body
⑩Clothes ⑪Shoes ⑫Other, accessories

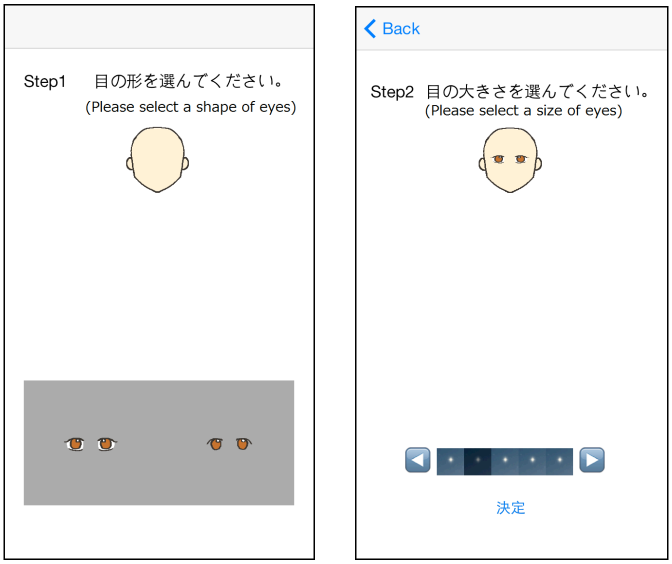
The determination result have six patterns, that is, 1960s , 1970s , 1980s , 1990s , 2000s , and all age type from generation trend of the selected part.

B. Development environment

Development environment is Xcode (Version 5.0.1) of Mac OS (10.8.5). I used the Objective-C and developed as for iOS application. It uses the iOS SDK 7.0 and runs on iOS7.0. Because it is the application that diagnoses from the chosen alternative and generates avatar, I created a program to reflect on the next page the selected image, and point to be added to a particular category by the selected image. At the end, it has the sum of the points that have been added and calculates how much each category accounts for the total, and display them.

C. Execution example of the application

I will show below an example of executing the application proposed in this thesis.

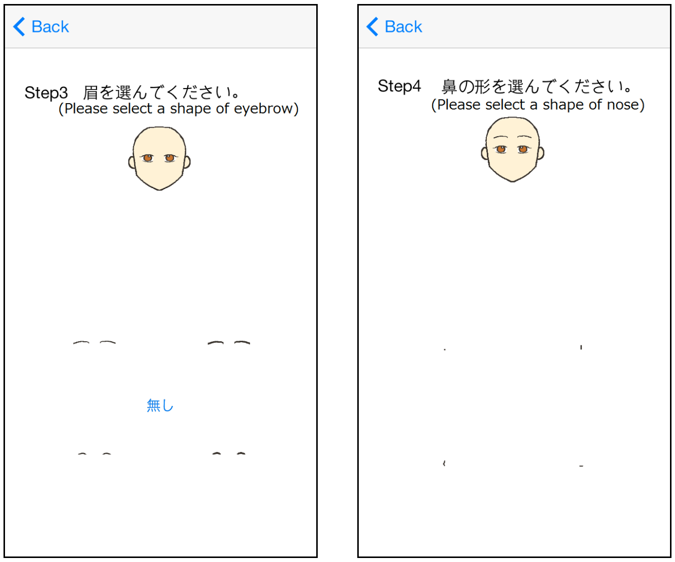

Step1　目の形を選んでください。
(Please select a shape of eyes)

Step2 目の大きさを選んでください。
(Please select a size of eyes)

(Figure 7　Step1 and Step2 screen)

1) Select the shape of the eye: First, you decide the shape of the eye. Form of a specific to age has not been found at this stage in the form of the eye itself, it is assumed that a point is not added in any generation at the stage of the selected shape of the eye. However, Application does the process of reflecting the image of the eye chosen in the next screen.

2) Select the size of the eye: Then you decide the size of the eye. Select what pieces of the size of the eye are equal to a vertical length of the face at this stage. There are five choices, which are 5 to 1, 6 to 1, 7 to 1, 8 to 1, 9 to 1. 8 to 1 is selected in the following example so that 100 points are added to the 2000's. Process of switching an image of the eye in the moment to adjust the size, and processing to the next screen by pressing the ENTER button is done.

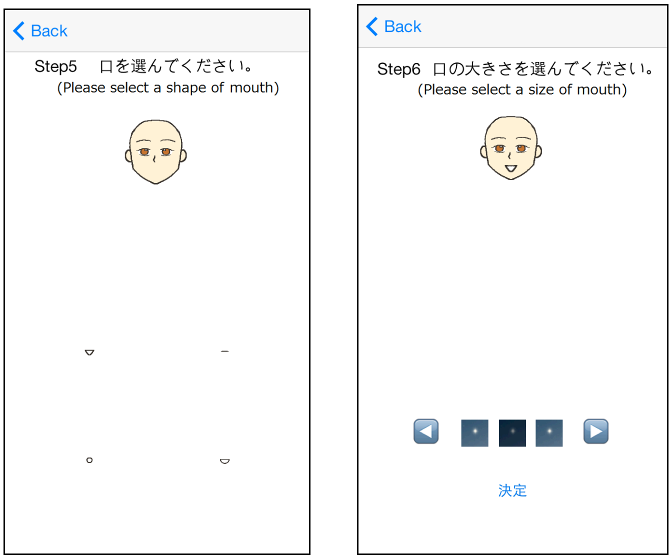

Step3　眉を選んでください。
(Please select a shape of eyebrow)

Step4　鼻の形を選んでください。
(Please select a shape of nose)

(Figure 8　Step3 and Step4 screen)

3) Selection of the shape of the eyebrow: Then you decide eyebrows. Choice is five elongated eyebrow, thin and long eyebrows, thin and short eyebrows, thick and thin eyebrows, no eyebrows. Form of a specific to age has not been found at this stage in the form of the eyebrows themselves, it is assumed that a point is not added in any generation at the stage of the selected eyebrows. Process of reflecting the image of the eyebrows chosen in the next screen is done.

4) Selection of the shape of the nose: Then you decide the nose. Choice is a four, which are point nose, vertical line nose, horizontal line nose, key nose. Form of a specific to age has not been found at this stage in the form of the nose itself, it is assumed that a point is not added in any generation at the stage of the selected shape of the nose. Application does the process of reflecting the image of the nose chosen in the next screen.

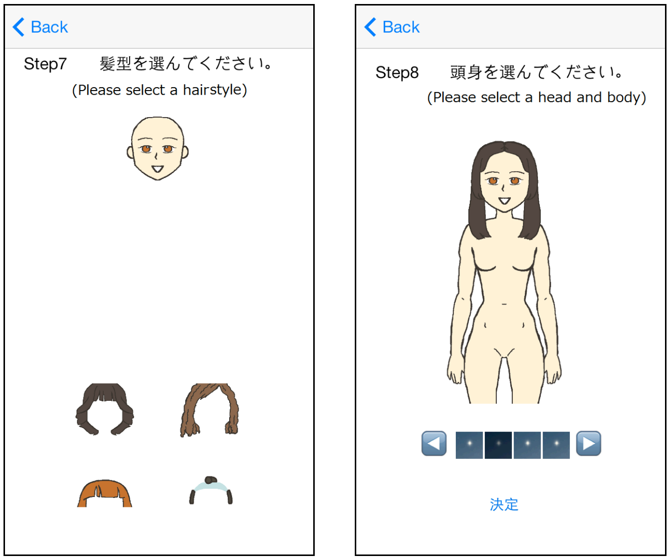

Step5　口を選んでください。
(Please select a shape of mouth)

Step6 口の大きさを選んでください。
(Please select a size of mouth)

(Figure 9　Step5 and Step6 screen)

5) Selection of the shape of the mouth: Then you decide the shape of the mouth. Choice is a four reverse triangular, linear, point type, the semi-circular. Form of a specific to age has not been found at this stage in the form of the mouth itself, it is assumed that a point is not added in any generation at the stage of the selected shape of the mouth. Process of reflecting the image of the mouth chosen in the next screen is done.

6) Selection of the size of the mouth: You decide the size of the mouth. The choices are three stages, which are large, medium and small. Size of a specific to age has not been found at this stage in the size of the mouth itself, it is assumed that a point is not added in any generation at the stage of the selected size of the mouth. Application does the process of switching an image of the mouth in the moment to adjust the size, and processing to the next screen by pressing the ENTER button.



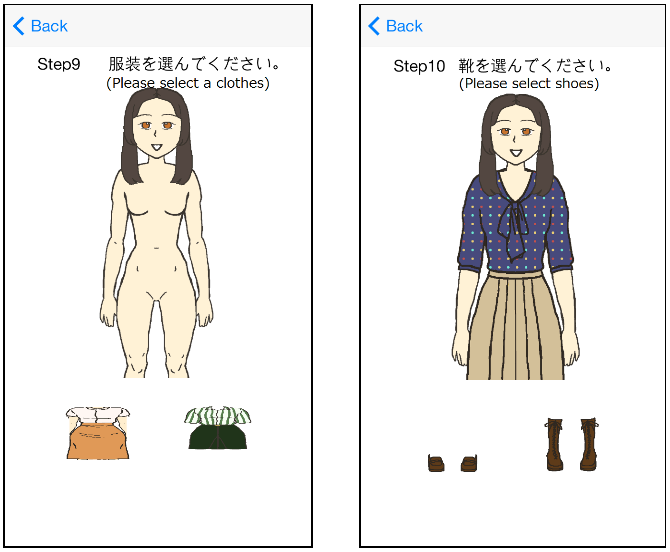
Step7 髪型を選んでください。(Please select a hairstyle)

Step8 頭身を選んでください。(Please select a head and body)

(Figure 10 Step7 and Step8 screen)

7) Selection of hairstyle: Then you decide the hair. Based on the survey results, choices were prepared parts of hairstyle was popular in the 2000s-1960s. In the following example, because a princess cut has been selected, 100 points are added to the 1970s. Basically, 100 points are added to ages hairstyle you chose is very popular, but the 10000 point is added to the Edo era type if you select choice of jokes, a topknot. Also process of reflecting the image of the mouth chosen in the next screen is done.

8) Selection of head and body: You decide to head and body. Choices and four stage, which are 5 heads tall, 6 heads tall, 7 heads tall, 8 heads tall. In the following example, because 7 heads tall has been selected, 25 points are added to the 1990s and 25 points are added to the 1970s and 50 points are added to the 2000s. Because it is the size adjustment, the process of switching an image of the head and body in the moment to adjust the size, and processing to the next screen by pressing the ENTER button is done.



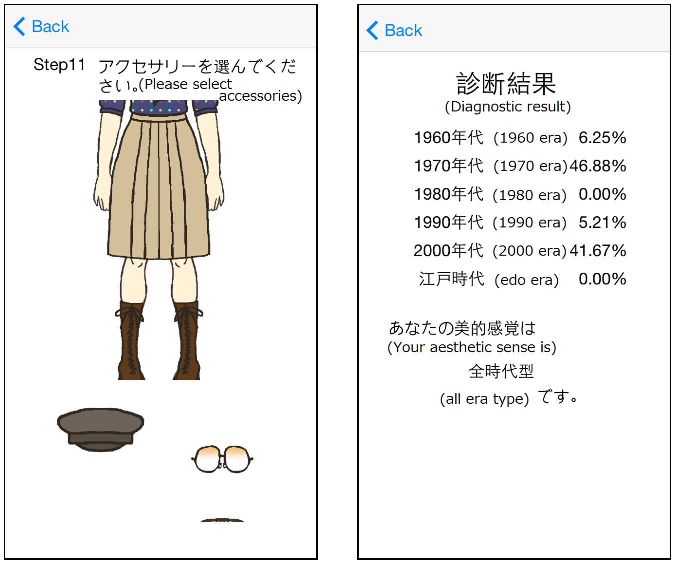
Step9 服装を選んでください。(Please select a clothes)

Step10 靴を選んでください。(Please select shoes)

(Figure 11 Step9 and Step10 screen)

9) Selection of clothes: You decide the clothes. Providing part of clothes that were popular to the 2000s from the 1960s, 100 points are added to the age clothes you chose was very popular. Points are added to the 2000's in the following example. Also process of reflecting the image of the mouth chosen in the next screen is done.

10) Selection of shoes: You decide the shoes. Similar to the stage you choose the clothes and hairstyle at step 7 and 9, providing part of the shoes that was popular in the 2000s to 1960s, points are added to the age of the shoes you have chosen is very popular choice. It is assumed that the 50 point addition here. However, some are likely to be worn in all ages, such as sneakers, it is assumed that 10 points is incremented by each age if you select it. Also process of reflecting the image of the mouth chosen in the next screen is done. Platform boot is selected in the following example, points are added to the 1970's.



Step11 アクセサリーを選んでください。(Please select accessories)

診断結果
(Diagnostic result)

1960年代 (1960 era) 6.25%
1970年代 (1970 era) 46.88%
1980年代 (1980 era) 0.00%
1990年代 (1990 era) 5.21%
2000年代 (2000 era) 41.67%
江戸時代 (edo era) 0.00%

あなたの美的感覚は
(Your aesthetic sense is)
全時代型
(all era type) です。

(Figure 12 Step11 and Step12 screen)

11) Selection of accessories: You choose the other accessories, which are Hat, glasses, and earrings. Similar to step 7, 9 and 10, points are added to the ages accessories you chose was very popular. 30 points are added here. And process of reflecting the image of the mouth chosen in the next screen is done.

12) Generation of avatar and generation decision: You press the diagnostic button, total the points that have been added by the option you selected in each step, and calculate what percentage of the total each category to the 2000s from the 1960s occupy, and display them. Then, it is determined to be the age of aesthetics of your age accounted for 60% or more. It determined that half of two age when two age accounted by 50%.It determined that all age types, when each age was less than 30%.

V. Challenges for the future

In this research, I have developed a generational avatar generation application based on the character design of comics and animations that were popular in Japan to the 2000s from the 1960s.

Although Comics and anime in Japan are prevalent worldwide and accepted, in principle, as a general rule, the subject of this application is limited to Japanese. In the future, I want to aim for applications targeted as well as comics and animation outside of Japan.

References

[1] Private comic history http://p.booklog.jp/book/26062

[2] The golden ratio of beauty face http://www.dental-bigan.com/golden.html.

[3] Comics whole volume dot-com annual ranking http://www.mangazenkan.com/.

[4] By age comic rankings http://www.discas.net/netdvd/stJComicNendai.do

[5] Magical girl anime that we longed to their costume ranking http://ranking.goo.ne.jp/ranking/026/5N7DmzTP3aUj/

[6] Idols of glory to look back at the debut year http://www.otokichi.com/main/newotokichi/idoldebutehistoryjp.htm.

[7] Girl comic portal site of adult http://www.girls-comics.com/

[8] Age epidemic http://nendai-ryuukou.com/

Synchronous Gestures for Co-located Collaboration on Multiple Mobile Devices

Yuguang Zeng[1], **Jingyuan Zhang**[1]
[1]Department of Computer Science, The University of Alabama, Tuscaloosa, AL 35487 USA

Abstract— *Synchronous gestures have an intention to solve the sharing problems between mobile devices. It is always tedious for users to share their files with nearby users without wireless access points. We propose synchronous gestures to provide an interaction interface to connect users' mobile devices . Those gestures include Pinch, which can be used to connect devices and share files between two devices, and DragAndPress, which enables one-to-many communication. We identify the general requirements of those gestures and implement a prototype system allowing users to share digital objects. Our approach is secure, has no bezel issues, and require no centralized servers or additional equipments.*

Keywords: mobile devices; synchronous gestures; spontaneous device sharing; co-located collaboration

1. Introduction

Mobile devices have been becoming ubiquitous and it is not uncommon for people to gather together to exchange documents or photos through their mobile devices. Currently, even for nearby devices, users have to make efforts to share by email attachments, on-line storage services and other Web applications. With wireless technology, direct connection between surrounding devices is possible and more effective. Users still need to configure their devices to make them connected with each other. How to make mobile devices with limited I/O capabilities to achieve that in an efficient way? This is the spontaneous device sharing problems, defined as how purposeful connections can establish dynamically between two or more devices without knowing each other's network address in advance. This is not only research issues for interaction design and system implementation, but an emerging problem desiring practical solution for daily life.

A new interaction technique known as synchronous gestures are proposed by researches to address spontaneous device sharing problem. With the technique, users make gestures to connect devices rather enter network address. Currently, proposed gestures include shaking two closed devices [1], bumping a pair of devices together [2], pressing a button on each device at the same time [3], and stitching a pen across surfaces of multiple devices [4]. The synchronous gestures are still under investigation.

We propose two synchronous gestures to let users communicate in a group with ease. Pinch gestures establish ad hoc connections between two nearby devices. When there is an item under the Pinch strokes, the item is shared on the other device involving the pinch gesture. DragAndPress enables a file to be shared among multiple devices. When one user drags a file out of screen side and other can press on their own screens if they want to receive the file. We build a prototype to show proof of concept and demonstrate the usability of our gestures.

2. Related Work

Previous researches have proposed systems for collaborating with mobile devices. However, some work requires special hardwares such as radio-frequency identification(RFID) tags [5], cameras [6], or sensors [7]. Some systems involve using nearby devices, and need manual configuration, including network configuration [8] and display configuration [9].

Synchronous gestures describe patterns of distributed user activities. Those activities have to happen at the same time or in a short consecutive time. There are literatures describing research on synchronous gestures. "Smart-Its Friends" [1] need users to hold together and shake a pair of accelerometer-augmented handled devices to connect them. The technique enables devices to receive movement data from other device and compared received data to its recent own movement data. If a similar pattern is recognized, a connection will be established between them. SyncTap [3]chooses synchronous buttons on devices at first. Two devices are connected when a user presses synchronous buttons down at the same time. The user needs to repeat synchronous operations if there is a collisions of overlapping actions at the same time. Bumping devices together can be used to create a shared display that spans two or more devices [2]. Signal patterns of accelerometers from two devices are compared to determine whether the bumping is intentional. Stitching [4] asks users to draw a continuous line across the screens of different devices. By analyzing the path between two devices, the system determines whether the path is a stitch. A server is needed to recognize stitch gestures and then informs devices network address if the server found a match. In addition, devices need to know the particular network address of the server.

There are some existing wireless standards related to the spontaneous device sharing problem. Bluetooth standardized as IEEE 802.15.1 [10] supports device discovery within radio range, but explicitly requires users to press button to add nearby Bluetooth devices. Infrared Data Association(IrDA) [11] is used to transfer data between two devices and requires devices keep still during transmission. Wi-Fi Direct [12] enables devices to connect each other without an access point and to transfer data at a typical Wi-Fi speed as high as 250Mbps [13], which makes it possible for users to share large data such as videos, photos and documents.

3. Procedure for Pinch

The Fig. 1 shows the procedure of using pinch gesture to connect two devices. When two devices are laid together, pinch gesture can be performed on them. Each device captures the performed gesture and sends the gesture to nearby devices. After sending their own gestures, devices are waiting to receive gestures sent by nearby devices. On getting gestures from other devices, the gesture recognition algorithm described below is used to determine whether received gestures and its own gesture form a pinch gesture. If they do, a connection will be established between those devices.

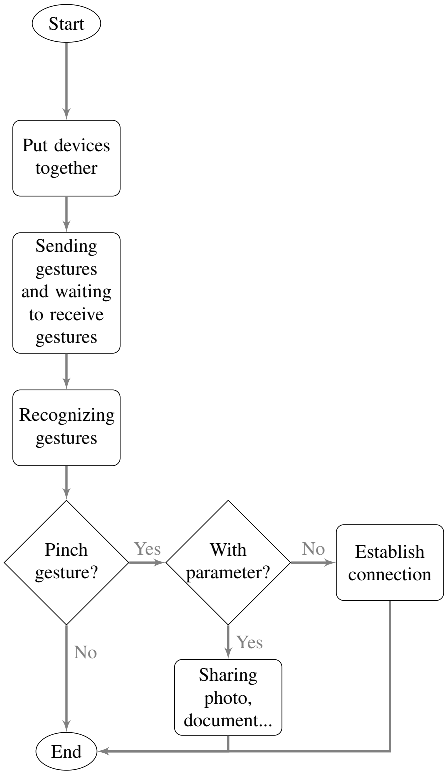


Fig. 1: Procedure for Pinch

4. The requirement of Pinch

Pinch is an explicit user command that enable multiple devices to be connected. Pinch is expected to provide a flexible and potentially extensible facility to support a number of different ways of combining devices, rather than supporting only a single operation or a very limited set of options as in previous systems like [1], [14] . Considering the expectation, Pinch address those design problem:

- **Connection:** How do devices connect to each other? A pinch gesture is performed on those devices which are to establish connection, and the system gives users visual feedback indicated a connection is established.
- **Coexistence:** How the system distinguish gesture for connection from gestures? A connection window shows up to perform Pinch gesture and disappears when a connection is established. Users invoke that window to appear if they need to connect other devices.
- **Proxemics:** How is physical space shared between users? It is a requirement of interaction techniques for impromptu association between devices to maintain social distance while users work closely. Pinch only requires users put their devices together to make connection. After establishing connection, devices can be moved to any place during users' collaboration activities.

5. The Mechanics of Pinch

With those design questions above, Pinch is developed as a new synchronous gestures. We now discuss general concept of Pinch on those design questions.

5.1 Establishment of a Connection

Before getting connected to other devices, devices needs to discover nearby devices, using services provided by underlying networks. We employ discovery mechanism from Wi-Fi Direct in the paper.

To establish a connection, users put their devices together and perform pinch gestures on those devices. The established connection is bidirectional [1], [3]. We focus on interaction techniques to form a purposeful connection between devices.

In our prototype system, each device automatically finds its nearby devices and sends them its gesture strokes. Comparing gesture strokes and time between its own and other devices, devices add as neighbors those having a match. Users do not need to enter any network address during the procedure.

5.2 Recognition of Pinch Gestures

Devices can recognize Pinch gesture by looking at the gesture strokes from nearby devices. *Timeframe* is defined as the time interval during which fingers start pressing on the screen, and end up leaving the screen as in Fig. 2. The participating device expects gesture strokes from other devices during timeframe.

The specific criteria are summarized in spatial and temporal aspects:

- Those gesture strokes must end near the adjacent screen edges and last longer than a timeout(150 milliseconds).

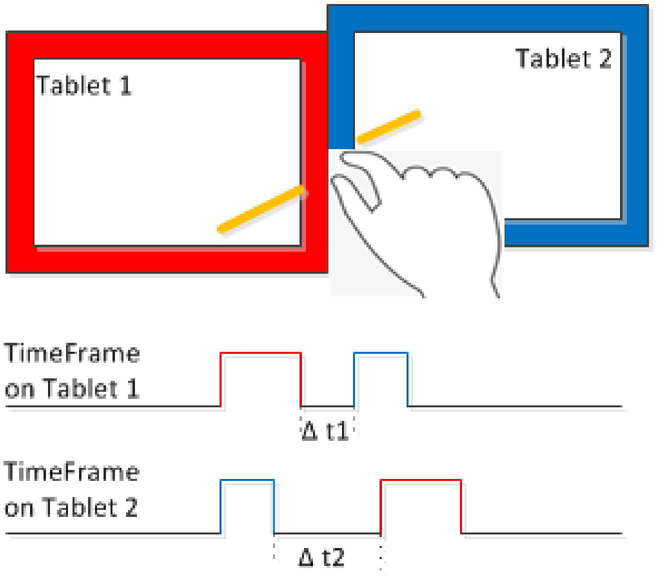


Fig. 2: Conceptual diagram of recognizing pinch gesture

- The direction of those two strokes are opposite.
- The slopes of those strokes are similar(must match within ±5°).
- Timeframes of gesture strokes are concurrent. After sending out their gesture strokes, devices must receive the gesture strokes from other devices within network delay(20 milliseconds).

These criteria are suffice to recognize distributed pinch gestures done in purpose. False positive was not a problem because incidental finger motions from other devices rarely satisfy these criteria.

5.3 Coexistence of Traditional Interactions

Pinch aims to establish connection between devices. However, all the gestures near the edge make the system assume the users need to connect another device. We divide gestures into two categories. One is for connection and usually happens near the screen edge. No parameters is required for those gestures. The other is for operations for files and requires parameters to continue. The prototype system below will give better design for distinguish those activities.

5.4 Pinch with Parameters

When an item is under a pinch gesture, the action of pinch gesture is re-defined as sharing the item to other device involving the gesture. That provides a one-to-one communication for users and that is useful in a group communication.

5.5 Sharing Physical Space

Users need flexible ways to share physical space in collaboration with others according to social conventions, task type, and individual preferences. Pinch gesture is designed for two kinds of distance: intimate and personal. Pinch gestures require devices put together for connection, which only happens in friends, colleagues working together. After connection, users can choose any space orientation that they feel like and protect their privacy information, and continue the communication or collaboration.

6. DragAndPress

DragAndPress is to provide one-to-many communication. The requirement of DragAndPress is to notify other group member when a user want to share a file to other. That notification can be in either oral or message. When a file is dragged out of the screen on the user, only those device pressed by users can receive the files.

7. Implementation

7.1 Developement Environment

We choose Android platform and Android devices, Google Nexus 7 from Google. Android 4.0(API level 14) or later devices support Wi-Fi peer-to-peer(P2P) . Android Wi-Fi P2P complies with the Wi-Fi Alliance's Wi-Fi Direct™certification program. Those Android APIs enables developers to write applications that allows devices to discover and connect to other devices supporting Wi-Fi P2P, and then to communicate over a speedy connection.

7.2 Implementation Issues

7.2.1 One-to-one communication

Although Android Wi-Fi P2P framework provides APIs to discover and communicate with other devices, there is the following limitation on the framework. The communication mode is many-to-one when a group is formed. A device creating the group acts as a group owner. Group members can only communicate with the group owner, and cannot communicate with other group members. Based on the Android Wi-Fi P2P, we implement a message exchange protocol , which enables devices to communicate with others directly. The following is the procedure:

(i) Acting as group owner, a device create a group, and invite other devices to join the group.

(ii) After nearby devices join the group, they will send their own IP address as message content to the group owner.

(iii) The group owner collects the incoming message and store them as a peer list. The list is wrapped in a message package and is sent to every device on the list.

7.2.2 Join pinch group

Our pinch gesture perform on two devices. However, it is not convenient and efficient to get every device connected($n!$, if n is the number of device) . How spread quickly the transitivity character of pinch gestures? We employ the following method. When a pinch gesture is recognized, those devices add each other in their pinched list and send the list to group members. The friend of my friend is my friend. That greatly improves the efficiency(n, if n is the number of devices).

7.2.3 Gesture recognition

Recognition algorithm of gesture is based on the characters of the pinch gestures in Fig. 2. The algorithm is described as the following:

Algorithm 1 Algorithm for gesture recognition

```
Input: Two gesture strokes, $g1, g2$; Two timestamps, $t1, t2$;
Output: Decide whether two gesture strokes form a pinch gesture, $isPinch$;
// Compare difference of two timestamps $|t1 - t2|$ with time stamp difference threshold $T_{threshold}$;
if $|t1 - t2| > T_{threshold}$ then
    $isPinch = false$;
    return $isPinch$
end if

Extracting the set of sampling point arrays $sg1$ and $sg2$ from $g1$ and $g2$, respectively;
Calculate the cosine similarity, $similarity$, of $sg1, sg2$;
if $similarity > Sim_{threshold}$ then $//Sim_{threshold} = 0.8$, $by\ default$
    $isPinch = false$
    return $isPinch$
end if

Compare the direction of $g1$ and $g2$ on $x$ and $y$ axis;
if they are the same direction on $x$ and $y$ axis then
    $isPinch = false$
    return $isPinch$
end if

return $true$
```

7.3 Application

We implement a prototype system, PinchToShare, to test the usability of our synchronous gestures. The application enables users to share their files between themselves. Just a pinch gesture is performed on their devices and there is no any extra network configure. Users start PinchToShare as Figure 3a, and wait for a moment for discovering nearby devices. When the green board appears as Fig. 3b, users can use Pinch to connect their devices. If the gestures is recognized as pinch, the application border will turn magenta. After that, users can share their files by using DragAndPress. In Fig. 4a, user1 shares the image on left top to user2 by sharing gesture. The receiving file show in a different color on user2's device as Fig. 4b. With our gestures, users can easily get their devices connected and share files to each other.

8. Discussion

8.1 Security and Privacy

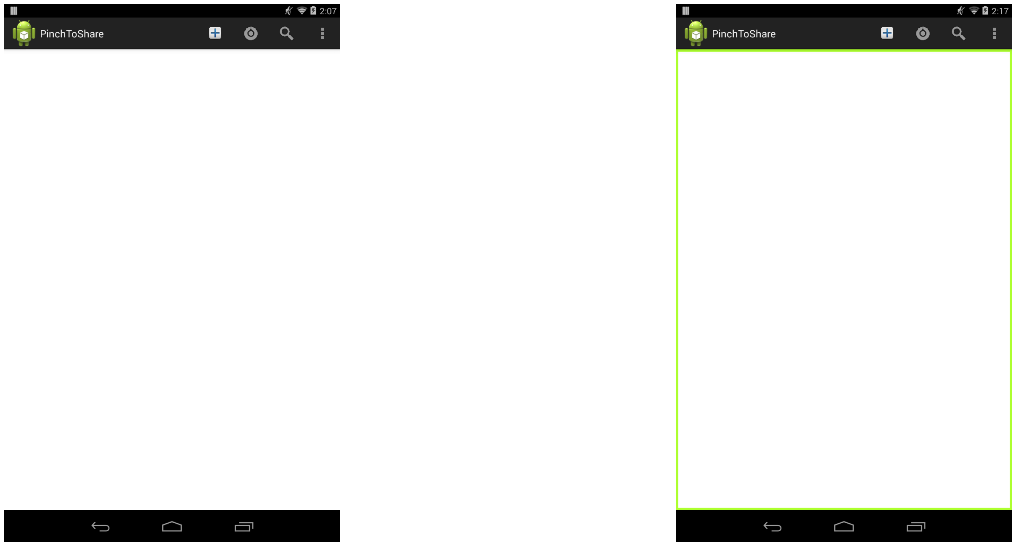
Security was considered when mobile devices were used in collaboration. In order to connect nearby devices, pinch gestures have their own security measures. Devices are put together only under the permission from users. With permissions, one user can touch other's devices. The physical nature of pinch gestures indicate the inherent social protocols, which invokes verbal communication on the beginning of collaboration.

8.2 Cooperative Pinch

While pinch gesture is performed by a single user, we discover pinch gesture can be performed in a cooperative way: users start draw strokes on their screens at the same time. The strokes need to meet the characters of pinch gestures. With cooperative pinch, we can develop a natural metaphor to support one-to-many connections. With such connection, a user sends digital objects such as documents to a group of users.

9. Conclusion and Future Works

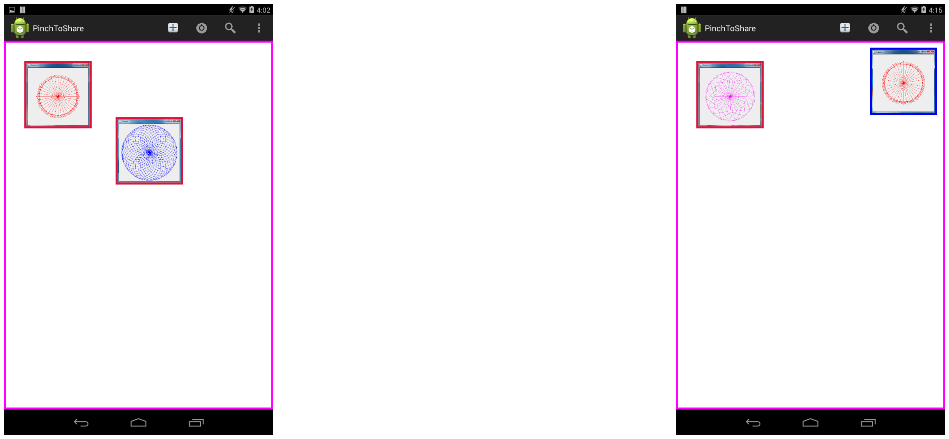
Synchronous gestures are proposed to improve the collaboration in a dynamic peer-to-peer setting. Pinch provides a convenient interaction technique for users to bind mobile devices in ad hoc. DragAndPress provides one-to-many communication. To demonstrate concepts of our gestures, a prototype system is implemented. We also note the future exploration based on our gestures. We recognize the pinch



(a) Initial Main Window

(b) After finishing discovering nearby devices

Fig. 3: Initial Window



(a) User 1's device

(b) User 2's device

Fig. 4: Sharing file

gesture by the behavior character, which was inaccurate on some time. Learning algorithm will be investigated to discover user patterns when performing pinch gestures. Exploring learning users' patterns improve not only accuracy of gesture reorganization, but the security for connection. We hope more novel interaction techniques emerge to foster communication and collaboration and greatly improve user experience of multiple mobile devices.

References

[1] L. E. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl, and H.-W. Gellersen, "Smart-its friends: A technique for users to easily establish connections between smart artefacts," in *Proceedings of the 3rd international conference on Ubiquitous Computing*, ser. UbiComp '01. London, UK, UK: Springer-Verlag, 2001, pp. 116–122. [Online]. Available: http://dl.acm.org/citation.cfm?id=647987.741340

[2] K. Hinckley, "Synchronous gestures for multiple persons and computers," in *Proceedings of the 16th annual ACM symposium on User interface software and technology*, ser. UIST '03. New York, NY, USA: ACM, 2003, pp. 149–158. [Online]. Available: http://doi.acm.org/10.1145/964696.964713

[3] J. Rekimoto, Y. Ayatsuka, and M. Kohno, "Synctap: An interaction technique for mobile networking," in *Human-Computer Interaction with Mobile Devices and Services, 5th International Symposium, Mobile HCI 2003, Udine, Italy, September 8-11, 2003, Proceedings*, ser. Lecture Notes in Computer Science, L. Chittaro, Ed., vol. 2795. Springer, 2003, pp. 104–115.

[4] K. Hinckley, G. Ramos, F. Guimbretiere, P. Baudisch, and M. Smith, "Stitching: pen gestures that span multiple displays," in *Proceedings of the working conference on Advanced visual interfaces*, ser. AVI '04. New York, NY, USA: ACM, 2004, pp. 23–31. [Online]. Available: http://doi.acm.org/10.1145/989863.989866

[5] C. Swindells, K. M. Inkpen, J. C. Dill, and M. Tory, "That one there! pointing to establish device identity," in *Proceedings of the 15th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '02. New York, NY, USA: ACM, 2002, pp. 151–160. [Online]. Available: http://doi.acm.org/10.1145/571985.572007

[6] S. Ransiri and S. Nanayakkara, "Smartfinger: An augmented finger as a seamless 'channel' between digital and physical objects," in *Proceedings of the 4th Augmented Human International Conference*, ser. AH '13. New York, NY, USA: ACM, 2013, pp. 5–8. [Online]. Available: http://doi.acm.org/10.1145/2459236.2459238

[7] D.-Y. Huang, C.-P. Lin, Y.-P. Hung, T.-W. Chang, N.-H. Yu, M.-L. Tsai, and M. Y. Chen, "Magmobile: Enhancing social interactions with rapid view-stitching games of mobile devices," in *Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia*, ser. MUM '12. New York, NY, USA: ACM, 2012, pp. 61:1–61:4. [Online]. Available: http://doi.acm.org/10.1145/2406367.2406440

[8] B. A. Myers, "Using handhelds and pcs together," *Commun. ACM*, vol. 44, no. 11, pp. 34–41, Nov. 2001. [Online]. Available: http://doi.acm.org/10.1145/384150.384159

[9] B. Johanson, G. Hutchins, T. Winograd, and M. Stone, "Pointright: Experience with flexible input redirection in interactive workspaces," in *Proceedings of the 15th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '02. New York, NY, USA: ACM, 2002, pp. 227–234. [Online]. Available: http://doi.acm.org/10.1145/571985.572019

[10] "Bluetooth." [Online]. Available: www.bluetooth.org

[11] "Irda." [Online]. Available: www.irda.org

[12] "Wi-fi alliance." [Online]. Available: http://www.wi-fi.org

[13] "How fast is wi-fi direct?" [Online]. Available: http://www.wi-fi.org/knowledge-center/faq/how-fast-wi-fi-direct

[14] J. Rekimoto, "Pick-and-drop: a direct manipulation technique for multiple computer environments," in *Proceedings of the 10th annual ACM symposium on User interface software and technology*, ser. UIST '97. New York, NY, USA: ACM, 1997, pp. 31–39. [Online]. Available: http://doi.acm.org/10.1145/263407.263505

Using Formal Concept Analysis to study social coding in GitHub

Otmar M. Pereira Junior[1], **Luis E. Zárate**[1], **Humberto T. Marques-Neto**[1], **and Mark A. J. Song**[1]
[1]Department of Computer Science, Pontifical Catholic University of Minas Gerais
Belo Horizonte, Minas Gerais Brazil

Abstract— *Currently, the world watches the expansion of the social networks phenomenon in the form of several different networks: Facebook, Twitter, Instagram, GitHub and many others. Such sites join together a user base that can reach millions of people connected from places everywhere in the planet. These impressive marks motivate questions about the characteristics of the members of these networks, due to opportunities present in them that can generate economical gains, augment the influence and popularity of an organization or product.*

This paper shows some of the main attributes of the contributors of the 20 most popular projects in GitHub using Formal Concept Analysis to uncover the most relevant characteristics. Despite the global success of the projects, we see geographical clusters of developers behind the most popular projects, what can serve as motivation to engage developers of other regions and increase the cultural variety in popular open source projects.

Keywords: Social Coding, Formal Concept Analysis

1. Introduction

The recent expansion of the social networks reached several different domains, ranging from music preferences to professional networking. Among these platforms, one can find social networks devoted to the development of open-source software. One of the first instances of this kind of social network is SourceForge, still very popular.

More recently, in 2008, GitHub was founded bringing several social features, what made it experiment a great success since then and take the leadership of its category after only three years of existence, having even more affiliated users than the popular SourceForge [1].

Such growth made Github object of several works in the community, as observed in [16], [17], that describes the behavior of GitHub members on the Internet. On the other hand, the authors of this paper could not find any work that tries to explain the causes of the formation of the developer networks of the main projects in GitHub in the same sense of the papers of Yu [18], Singh [12] and Gao [19], that describe the communities formed in SourceForge.

Previous works, like [21], show that some companies are already paying attention to platforms like GitHub to more accurately evaluate technical skills and accomplishments of IT professionals. Nonetheless, GitHub does not provide tools to reduce the efforts of Human Resources professionals willing to understand the typical profile of developers that have skills similar to the ones of a popular project of a given technology. We believe that GitHub data can be valuable indicator of developers possessing certain skills and proven experience.

This paper aims to study the developer networks around the most popular software repositories in GitHub by applying Formal Concept Analysis (FCA) to discover the most relevant common attributes of the developers contributing to GitHub's most successful open source projects using public information of the developers available via GitHub API.

2. Social Coding

The evolution of information and communications technology in the second half of twentieth century, specially in the last 15 years, made the contemporary society highly connected through complex networks, much like in the form described in [15]. As part of this process of increasing interconnection between people all around the world, social networks linking individuals with common interests of several different kinds arose: Facebook reuniting friends, professional networking in LinkedIn, photos and video sharing in YouTube and Instagram and open source software development in SourceForge, GitHub, Google Code and many others.

Among the categories of social networks, the category that brings together software developers is very popular and has SourceForge as one of its main examples since 1999. Still in the current days, SourceForge has over three million registered members. Many popular open source software are hosted in SourceForge, that continues to be a friendlier place to end users than GitHub. While we see many applications redirecting users to download installation binaries in SourceForge, we cannot observe a similar pattern in GitHub, that seems to be more targeted to developers. This may be one of the reasons that GitHub does not provide via its API a downloads counter of the project binaries and even the Downloads API has been deprecated.

Other social networks, such as Facebook, MySpace and Orkut have popularized Web 2.0 by giving a special emphasis on social aspects of the applications used by people on the Internet. In addition to the characteristics found in SourceForge, GitHub put in the spotlight social features of its developers' networks such as user feeds, following users and projects, collaboration to project's wiki pages and visualization of graphs showing the connections of members.

Another facet of collaborative development that has been promoted by GitHub is the use of distributed version control systems instead of centralized ones, like Subversion and CVS. Developed by the famous and acclaimed Linus Torvalds to help in the development of Linux kernel, Git was one of the first distributed version control systems, where different versions of the source code are not in a single server. Each development machine contains all the previous versions of the software, instead of only the workspace version, as is the case when working with centralized version control systems. This paradigm reduces significantly common issues in concurrent software development, like resolving conflicts when merging files before a release. GitHub promotes Git as its main version control system and some of its features to promote social interactions, like forking repositories and sending pull requests to other developers.

This paper studies a network formed by developers that have been tagged as contributor to at least one of the 20 projects in GitHub that had received most stars as of September 2013 in GitHub. We aim to unveil the main characteristics shared between the people that work on open source software projects by using Formal Concept Analyis (FCA) to discover association rules and implication rules between the attributes exposed by the GitHub API [7], that exposes some geographical and personal data about each developer.

2.1 Selected Dataset

In the quest for understanding the popularity of projects in GitHub, we concentrated our efforts on the data of the 20 projects that had been starred the most by GitHub members as of September 2013. The selection of the projects was based upon data available in GitHub Archive [3] and can be seen in Table 1.

The criteria of received stars has been chosen due to the belief in it as an indicator of the project popularity and interest upon the project. According to GitHub API documentation [8], stars are used to bookmark a repository and show an approximate level of interest, what makes it a good indication of project popularity. It is similar to the metric employed by [12] to measure the commercial success of an open-source project in SourceForge platform. Moreover, the authors consider this metric the most appropriate among the ones available in GitHub API to evaluate the success of project.

The gathering of the top projects' ranking was conducted using the tool Google Big Query [2], that imposes some restrictions on the volume of data processed by the submitted queries. Such restriction directed our efforts on the analysis of the 20 most popular projects. Because of that, in this work its expected to observe trends applicable only to the most prominent projects, and attributes that set these projects apart of the less popular ones should be object of other papers devoted to analysis of much large scale, including hundreds or thousands of projects. In spite of the limited number of projects analyzed, Table 1 shows some indications of the *rich gets richer* effect described by [15]: very few projects concentrate most of the characteristics and in general, these very few instances are the ones that continue to grow. We see that in a universe of millions of repositories, the *bootstrap* repository has almost ten times more stars received than the 20^{th} repository of the ranking.

Project name	Stars received
bootstrap	60525
jquery	25750
node	25541
html-boilerplate	22924
rails	19987
d3	19685
Font-Awesome	18594
impress.js	18256
angular.js	16310
backbone	16142
chosen	13927
jQuery-File-Upload	13777
three.js	13249
jekyll	12906
brackets	12900
gitignore	12467
gitlabhq	11027
meteor	10263
textmate	9221
select	8750

Table 1: List of the projects and stars given by GigHub users as of September 2013.

2.2 Network construction

After selecting the projects to analyze data from, further data was retrieved to build contributors' network of the projects. The co-participation of two developers in a project serves as link to this network that was built as an undirected graph. The information about the contribution to a project is available via GitHub API [7] and in order to consolidate the data, an Extract-Transform-and-Load process has been implemented in the tool Pentaho [9]. The list of user attributes retrieved from GitHub API is shown in the section 3.1 and the data gathering process is depicted in Figure 1.

After having the 20 project's contributors data collected, another Pentaho transformation has been implemented to resolve geographic information - continent and country - for each developer using the Microsoft Bing Maps API [10]. Since the location information for each developer is filled in GitHub in natural language form in many different ways - some inform just their country or province while many others give only their city name - a more powerful geographical resolution API was put in place to improve the information about the location of each member of the network.

Finally, after collecting all the attributes about each developer, a Java program has been implemented to generate a

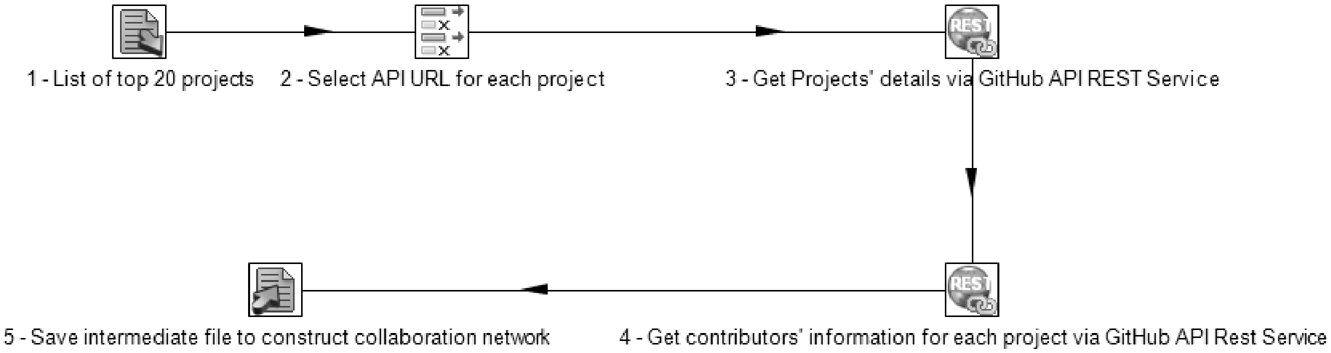

1 - List of top 20 projects
2 - Select API URL for each project
3 - Get Projects' details via GitHub API REST Service
4 - Get contributors' information for each project via GitHub API Rest Service
5 - Save intermediate file to construct collaboration network

Fig. 1: Pentaho transformation implemented to retrieve developer data in GitHub.

file in the format of the PAJEK software. This format is very popular and can be imported by a great number of network analysis tools.

In the the network constructed after applying all these steps, one can observer the small-world phenomenon, originally studied by Stanely Milgram [11]: the graph diameter is only 5, what means that every developer of the 5,385 contributors in this network knows each other through partaking in the same project with at most four other developers. One of the factors that contribute to the low diameter of this graph is certainly the significant number of partaking between pairs of developers: 3,857,471 were found, making this network a fairly dense one.

3. Formal Concept Analysis

Formal Concept Analysis (FCA) is a recent knowledge area that has been in expansion since the 1980s and interesting results have been obtained by its application to hierarchize and discover concepts that were not immediately evident in the original form of the given problem.

The studies in this area involve the definition of a formal context, relating the elements being analyzed and the attributes of these objects.

A formal context is represented by a triple (G,M,I), where G is the set of the objects, M is the set of attributes corresponding to some of the elements' properties under analysis and $I \subseteq G \times M$ is an incidence relation that indicates which objects contain which attributes. In some cases, the attributes have a single valued domain, where the attribute is whether true or false. It is also possible to model attributes corresponding to domains of higher cardinality, like colors or size - small, medium, large. In this work, for the sake of simplicity, we model a formal context where all the attributes are of the former kind.

A concept of a formal context is a pair (A,B), where A, where $A \subseteq G$, is defined as the extent of the concept and consists of all the objects pertaining to the concept. The set B, where $B \subseteq M$, is the intent of the concept is formed by all the common attributes shared between the objects of the extent set.

The construction of a formal context as defined before enables one to build a conceptual lattice that acts as a hierarchy of concepts and enables the discovery of new concepts - unexpected combinations of objects and attributes - and the relations between the objects and attributes.

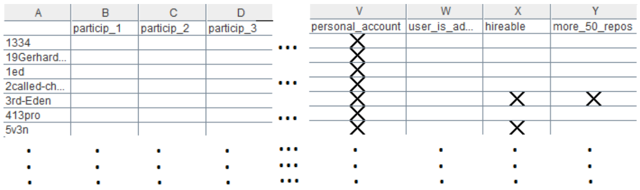
Some algorithms can be applied to find interesting patterns of attributes that appear together in association rules and implication sets that are not trivial neither redundant. In general, algorithms studied by FCA academics and available in some tools are used to mine logical dependencies between the attributes and groups of instances that given a subset of attributes are frequently found together.

We recommend readers interested in more details about Formal Concept Analysis to refer to [4] for a thorough explanation of the mathematical foundations around the Formal Concept Analysis.

3.1 Formal Context definition

In this work, the set of objects G has been defined as the set of developers who contributed to the 20 selected projects. These developers have several properties describing them in their profile and 36 of them were used to construct the set of binary attributes, M, which elements are shown in the following list with a description of each one of the selected properties:

1) Indicator of whether or not the registered user account is personal. When an account is not registered as a personal one, it means that it has been registered in the name of an organization.
2) Flag that indicates if the user administrates a project in GitHub.
3) Profile information indicating member availability for hiring.
4) The contributor has more than 50 public repositories in GitHub.
5) The user has already shared more than 50 gists - code snippets used in discussion with other members of



A	B	C	D		V	W	X	Y
	particip_1	particip_2	particip_3		personal_account	user_is_ad...	hireable	more_50_repos
1334				...	X			
19Gerhard...					X			
1ed					X			
2called-ch...				...	X			
3rd-Eden					X		X	X
413pro				...	X			
5v3n					X		X	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Fig. 2: Part of the formal context imported into Conexp. Project contributors are represented by their logins in each row and columns indicate the presence of a given property of that developer.

GitHub.

6) The biographical information on the user profile is very long. As inspection of some profiles having this characteristic showed that the biographical information, when very long, is used by developers to publish their curriculum vitae.
7) The member follows more than 1,000 other GitHub users.
8) The developer has more than 1,000 followers.
9) The user account was created more than three years ago.
10) Member location is in North America.
11) Person location is South America.
12) Developer is based in Europe.
13) User's location is Africa.
14) Asia is the informed location by the user.
15) The developer has informed his or her location as somewhere in Oceania.
16) The last 20 attributes indicate the participation in project *i*, where *i* indicates the participation of the member in the project indexed by *i* in the list of projects sorted alphabetically.
    For example: a developer who contributed to the first project in the list, angular.js, will have the Participated$_1$ attribute set to true, while someone who has not contributed to the last project in the list, textmate.js, will present the attribute Participated$_{20}$ indicating false for such participation.

In order to represent and analyze the formal context, an specific tool designed to build concept lattices from formal contexts and calculate implication sets, association rules and new concepts. Conexp [5] software was the tool elected to help on the analysis of the formal context.

All the attributes mentioned above were retrieved from the network built previously and imported into Conexp [5]. Figure 2 shows part of the formal context defined in this section. The rows correspond to the login of GitHub members and the columns are marked with an X when an attribute of the list 3.1 is true and is left blank otherwise. More details about this tool can be found in its documentation page [6].

4. Results

The analysis of the formal context was conducted using Conexp software. In this section, the most relevant rules will be discussed. Conexp computed a total of 1,822 concepts, 758 implication sets and 872 association rules involving different combinations of the attributes found on the input data. The most prominent combinations are discussed next.

Implication sets contain logical implications involving the attributes of the given formal context. Although it is possible to analyze all the combinations of the finite set of attributes and objects of the formal context, clearly, many of them would be trivial or redundant. It is thus necessary to restrict the set of implications that should be analyzed. Conexp was used to compute the minimal set of rules and the most relevant characteristics of these implications - frequent attributes - are discussed below.

- The implication *User account is personal → user account created more than three years ago* is true for 5,365 of 5,385 developers. This means that in general, companies and other institutions do not contribute to the popular open source projects in GitHub. Furthermore, the developer is an old member of GitHub. Popular projects in GitHub seem to be an space where newcomers may not be welcome: in the vast majority of the cases analyzed, the attribute that indicates the user as having an user account registered more than three years ago prevailed.
- Results regarding user's geographical information: 421 rules involve North American members; 342 developers based in Europe; 204 have developers with undetermined location; 187 relate South American developers; 133 rules involve Asian members of GitHub, 115 from the Oceanic continent and African developers are part of just 34 implication rules. Despite the fact of being a platform spread across the world, there is a great deal of concentration of users in North America and Europe. Surprisingly, in the Asian continent we find less contributors than in South America, a continent having approximately 10 times less inhabitants.
- The implication *available for hiring → User account is personal* AND *user account created more than three years ago* that is true for 1,277 members indicate that long time contributors are also paying attention to new professional opportunities in GitHub.

5. Conclusions

In this paper we have seen that the Formal Concept Analysis can be a valuable tool to analyze very large scale social networks, like GitHub, which user base reaches marks around millions of users and projects. The theoretical and practical framework in Formal Concept Analysis field enables the application of techniques to study social network characteristics. This was the case with GitHub, where relevant knowledge related to the geographical location and affiliation time to the network was gathered by analyzing the output of the FCA algorithms.

It was possible to realize that despite the possibility of connecting people around the world, some clusters are geared towards the most developed locations, namely North America and Europe, contrasting with a very low number of users in the African continent. This brings the question for future works that investigate the causes of such clustering: do they appear due to social-economic factors or is the social network built around people from the same region? The prevalence of longtime user accounts in GitHub as contributors of the most popular projects is also a factor to be taken into account, given the possible importance users seem to give to the past contributions of the developers to the community.

Besides all these questions around attributes of the members, there is a difficulty that is present when trying to analyze social networks: the manipulation of an ultra-large volume of data that create several technical challenges to manipulate the information and execute algorithms to analyze input data.

Due to the huge amount of information, there is no tool capable of accessing very recent GitHub data, and this is aggravated by the addition of a higher volume of information in GitHub than the amount of data that can be collected by a single machine during the same period of time - GitHubArchive reports peaks of 50,000 events per hour in GitHub, while the GitHub API imposes a restriction of 5,000 requests per hour for authenticated users that are querying its database. Initiatives like an open platform available to the mining of ultra-large scale software repositories, like [14] are very valuable to future investigations around the characteristics of software development social networks.

Acknowledgements

The authors acknowledge the financial support received from the Foundation for Research Support of Minas Gerais State, FAPEMIG, through Project CEX PPM 107/12, and the National Council for Scientific and Technological Development, CNPq, Brazil.

References

[1] K. Finley (2011). "Github has surpassed sourceforge and google code in popularity" [Online]. Available: https://www. readwriteweb. com/hack/2011/06/github-has-passed-sourceforge.php

[2] K. Sato, "An Inside Look at Google BigQuery, White paper." *Google Inc*, 2012.

[3] I. Grigorik (2013). "Github archive" [Online]. Available: http://www.githubarchive.org/.

[4] B. Ganter, R. Wille, and C. Franzke, *Formal concept analysis: mathematical foundations*. Springer-Verlag New York, Inc., 1997.

[5] S. Yevtushenko (2005). "Conexp software," [Online]. Available: http://conexp.sourceforge.net/.

[6] Conexp (2006), "Conexp documentation page" [Online]. Available: http://conexp.sourceforge.net/users /documentation/.

[7] GitHub (2013). "Github API v3," November 2013 [Online].Available: http://developer.github.com/v3/.

[8] GitHub (2013). "Starring | github api," [Online]. Available: https://developer.github.com/v3/ activity/starring/.

[9] Pentaho Corporation (2014). "Pentaho data integration," [Online]. Available: http://www.pentaho.org.

[10] Microsoft Corporation (2014), "Bing maps api," [Online]. Available: http://www.microsoft.com/maps/.

[11] S. Milgram, "The small world problem," *Psychology today*, vol. 2, no. 1, pp. 60–67, 1967.

[12] P. V. Singh, "The small-world effect: The influence of macro-level properties of developer collaboration networks on open-source project success," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 20, no. 2, p. 6, 2010.

[13] S. M. Dias and N. J. Vieria, "Um arcabouço para desenvolvimento de algoritmos da análise formal de conceitos," *Revista de Informática Teórica e Aplicada*, vol. 18, no. 1, pp. 31–57, 2011.

[14] R. Dyer, H. A. Nguyen, H. Rajan, and T. N. Nguyen, "Boa: a language and infrastructure for analyzing ultra-large-scale software repositories," in *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013, pp. 422–431.

[15] D. Easley and J. Kleinberg, *Networks, crowds, and markets*. Cambridge Univ Press, 2010, vol. 8.

[16] B. Vasilescu, V. Filkov, and A. Serebrenik, "Stackoverflow and github: associations between software development and crowdsourced knowledge," in *Social Computing (SocialCom), 2013 International Conference on*. IEEE, 2013, pp. 188–195.

[17] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, "Social coding in github: transparency and collaboration in an open software repository," in *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*. ACM, 2012, pp. 1277–1286.

[18] L. Yu and S. Ramaswamy, "A study of sourceforge users and user network," in *2013 AAAI Fall Symposium Series*, 2013.

[19] Y. Gao and G. Madey, "Network analysis of the sourceforge. net community," in *Open Source Development, Adoption and Innovation*. Springer, 2007, pp. 187–200.

[20] C. Carpineto and G. Romano, *Concept data analysis: Theory and applications*. John Wiley & Sons, 2004.

[21] Marlow, Jennifer, and Laura Dabbish. "Activity traces and signals in software developer recruitment and hiring." Proceedings of the 2013 conference on Computer supported cooperative work. ACM, 2013.

DEVELOPMENT OF A MATHEMATICAL MODEL FOR DESIGNING RELIABLE INFORMATION SYSTEMS AND ITS PROPERTIES

The 2014 International Conference on Software Engineering Research and Practice (SERP'14), Las Vegas, USA, July 21-24, 2014

Seilkhan Boranbayev
L.N. Gumilyov Eurasian National University
5 Munaitpasov Street
Astana, 010008, Kazakhstan
sboranba@yandex.kz

Askar Boranbayev
Nazarbayev University
Astana, Kazakhstan
aboranbayev@nu.edu.kz

Sanzhar Altayev
L.N. Gumilyov Eurasian National University
5 Munaitpasov Street
Astana, 010008, Kazakhstan

Abstract—It is necessary to reduce dependence of results quality from development of information systems on such subjective factors as qualification and experience of performers, to lower risk of unsuccessful completion of the project. There is an urgent need today for science-based technological methods of information systems to plan the parameters of a software project, to guarantee the required quality of results. Thus, the creation of scientific methods and technologies for the information systems design is an important scientific and technical problem. This article is devoted to development of methods of design and distribution of resources for development of information systems. Implementation of the developed methods and technologies in projects for automation of enterprises of various activity spheres will streamline the process of modeling and collect in the process of performing a formal information to plan the next stages of the project, provide functional completeness and logical integrity of their results.

Keywords— *model, method, design, information system, technology, network****.***

I. INTRODUCTION

Presently there is an increase in quantity and complexity of projects on complex automation of the enterprises. High dimensionality and complexity of the automation object determines the iterative nature of the design methods and the need for their commercial nature means having a deep formalization of the technology implementation of all phases of the project. Existing methods are definitely solve the problem of the development of software, however, have no sufficiently industrial properties.

High complexity of the automation object determines the nature of the work from the earliest stages consisting in inspection, modeling and the analysis domain. At the same time the development of information systems has its own characteristics, which should be reflected in special events to maintain the logical integrity of the results throughout the project.

II. MATHEMATICAL MODEL OF INFORMATION SYSTEM DESIGN

We will use oriented bonded network reflecting the progress of the project and intended for the analysis of the project's logical structure as a model of information system design. The network has a single input and a single output vertex. Each vertex is the work in the project.

We will call a simple network as source network portion having a single input and a single output vertex [1] and consisting a linear sequence of project work. All works included in a simple network, performed sequentially from the first to the last. We use the notation of [1-5].

We will consider an initial network as set of final number of simple networks. Execution sequence of simple networks displays the project's progress.

We will call a network of project progress as oriented bonded network *S=(M, R, m$_0$)* with a single initial vertex *m$_0$* without incoming arcs, in which the final set of vertices *M={m$_q$}*, $q=\overline{0,N}$ is a simple network, and the final set of edges *R={r$_j$}*, $j=\overline{1,J}$ is a logical connections between simple networks [2]. Vertices *m$_0$* and *m$_t$* can be entered artificially.

We will call a way as sequence of network's vertices *(m$_0$, m$_1$, m$_2$,..., m$_q$,..., m$_t$)*, that for any value *q*, $0 \le q \le t-1$, couple *(m$_q$, m$_{q+1}$)* is an edge $r_j \in R$. If *m$_t$= m$_0$*, a way is called as a contour.

Vertex m_q is called the predecessor of vertex m_γ, if there is a way from vertex m_q to vertex m_γ. If (m_q, m_γ) is an edge, vertex m_q is called an immediate predecessor of vertex m_γ.

If vertex m_q is a predecessor of vertex m_γ, vertex m_γ is called the follower of vertex m_q. If (m_q, m_γ) is an edge, vertex m_γ is called the immediate follower of vertex m_q.

Simple network SP_q is called the predecessor of simple network SP_γ, if there is a way from simple network SP_q to simple network SP_γ. If between them there are no other simple networks, simple network SP_q is called the direct predecessor of simple network SP_γ.

If simple network SP_q is a predecessor of simple network SP_γ, simple network SP_γ is called the follower of simple network SP_q. If between them there are no other simple networks, simple network SP_γ is called the direct follower of simple network SP_q.

Vertex m_q is called the ancestor of vertex m_γ, if each way from initial vertex m_0 to vertex m_γ contains vertex m_q. If vertex m_q is the ancestor for vertex m_γ and for vertex m_γ there are no other ancestors on ways from vertex m_q to vertex m_γ, vertex m_q is called the direct ancestor of vertex m_γ.

Lemma 1. Initial vertex m_0 of a network *S=(M, R, m$_0$)* is the ancestor of each vertex of the network $m_q \in (M - \{m_0\})$.

Corroboration. The corroboration follows from network structure.

Lemma 2. If vertex m_q is the ancestor for vertex m_γ and vertex m_γ is the ancestor for vertex m_k, then vertex m_q is also the ancestor for vertex m_k.

Corroboration. Because vertex m_q is the ancestor for vertex m_γ, that each way from initial vertex m_0, passing through vertex m_γ, including any way *(m$_0$,..., m$_\gamma$,..., m$_k$)*, contains vertex m_q. Therefore, vertex m_q is also the ancestor for vertex m_k.

Lemma 3. If vertex m_q is the ancestor for vertex m_γ, then vertex m_γ can't be the ancestor for vertex m_q.

Corroboration. The corroboration follows from the definition of an ancestor.

Lemma 4. If vertices m_q и m_γ are ancestors of vertex m_k, then vertex m_q is the ancestor for vertex m_γ or vertex m_γ is the ancestor for vertex m_q.

Corroboration. Any way from initial vertex m_0 to vertex m_k contains vertices m_q and m_γ. Let's say that vertex m_γ isn't the ancestor of vertex m_q. We will prove that vertex m_q is the ancestor of vertex m_γ. Really, if vertex m_q isn't the ancestor of vertex m_γ, it means that some way from initial vertex m_0 passing through vertex m_γ *(m$_0$,..., m$_\gamma$,..., m$_k$)*, doesn't contain vertex m_q. It contradicts that vertex m_q is the ancestor of vertex m_k.

Lemma 5. For each vertex $m_k \in M, k \ne 0$ there is only one direct ancestor.

Corroboration. Let's say that vertex m_k has two direct ancestors: m_q and m_γ. Then, by Lemma 4, vertex m_q is the direct ancestor of vertex m_γ or vertex m_γ is the direct ancestor of vertex m_q. Let for definiteness vertex m_q is the direct ancestor of vertex m_γ. Then, by Lemma 3, vertex m_γ can’t be the direct ancestor of vertex m_q. However, if vertex m_q is the direct ancestor of vertex m_γ, and vertex m_γ is the direct ancestor of vertex m_k, than vertex m_q can't be the direct ancestor for vertex m_k. This contradiction proves that vertex m_k has the only one direct ancestor.

It is possible to connect a great number of predecessors and followers (both direct and usual), and also a great number of ancestors and direct ancestors with each vertex $m_q \in M$ of a network *S=(M, R, m$_0$)*.

We will designate MNP_q^+ – set of direct predecessors for vertex m_q; MNP_q – set of predecessors for vertex m_q; MNS_q^+ – set of direct followers for vertex m_q; MNS_q – set of followers for vertex m_q; MP_q – set of ancestors for vertex m_q; MP_q^+ – direct ancestor of vertex m_q.

It is obvious that $MNP_q^+ \subset MNP_q$, $MNS_q^+ \subset MNS_q$, $MP_q^+ \subset MP_q$.

$$MNP_q = \{m_\gamma \in M \mid m_q \in MNS_\gamma\} \qquad (1)$$

$$MNS_q = \{m_\gamma \in M \mid m_q \in MNP_\gamma\} \qquad (2)$$

$$MP_q = \{m_\gamma \in M^* \mid M^* \subset M, M^* = \bigcap_i Q_q^i\} \qquad (3)$$

where Q_q^i – *i*-th way from initial vertex m_0 to vertex m_q.

Figure 1 shows an example of a network.

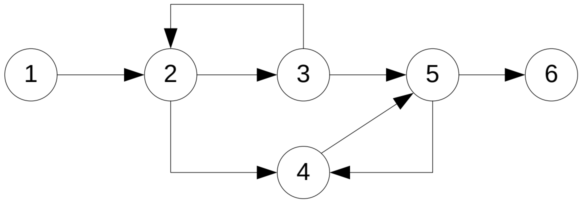


Figure 1 - Example of a network

m_0=1, M={1,2,3,4,5,6}, R={(1,2), (2,3), (2,4), (3,2), (3,5), (4,5), (5,4), (5,6)}

III. PROPERTIES OF MATHEMATICAL MODEL OF INFORMATION SYSTEMS DESIGN

Let a network S=(M, R, m_0) of the project and some vertex g.

We will call a subnet *T(g)* as a network portion with entrance vertex *g* satisfies the following conditions:

1) Vertex $g \in T(g)$ is the only one input vertex of subnet *T(g)*, that is any way from input vertex m_0 of a network *S* to vertex *g* doesn't contain any vertex from a set *T(g)-{g}*; we will call a vertex *g* as subnet root;

2) Each vertex of a subset *T(g)-{g}* is a follower of vertex *g*, that is $(T(g)-\{g\}) \subset MNS_g$;

3) If in a subnet *T(g)* there are closed ways, all of them contain vertex *g*, that is the subset of vertices *T(g)-{g}* doesn't contain cycles.

Figure 2 shows some examples of subnets. For network *S*, shown in figure 3, Figure 4 shows subnet.

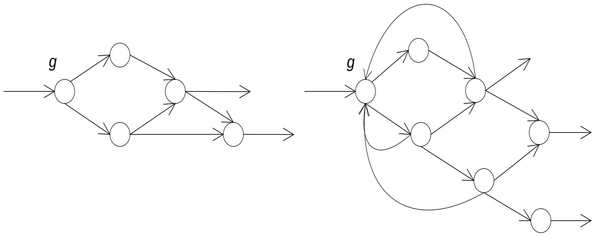


Figure 2 - Subnets

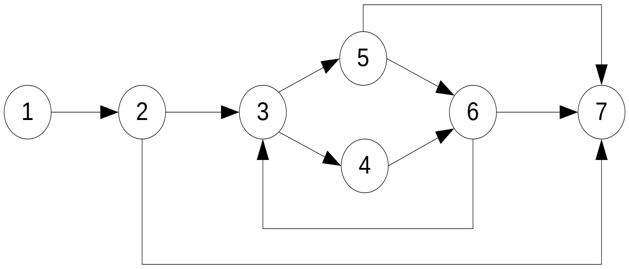


Figure 3 - Network S

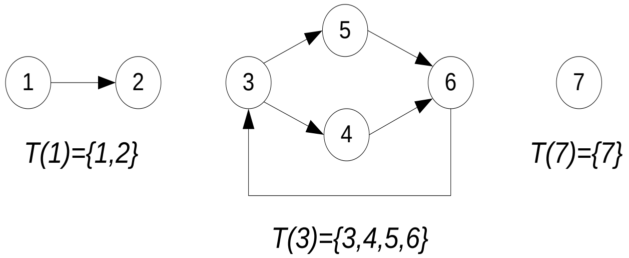


Figure 4 – Subnets of network S

Theorem 1. The root *g* of a subnet *T(g)* is the ancestor for all vertices of a subset *T(g) - {g}*.

Corroboration. Let's say that there is some vertex $m_q \in T(g)$ for which root of a subnet isn't the ancestor, that is $g \notin MP_q$. Then there has to be at least one way *(m_0,…, m_q)* which doesn't contain vertex *g*. It means that subnet has not the only one input vertex. It contradicts subnet definition.

Theorem 2. Each vertex of a network *S=(M, R, m_0)* is a part of only one subnet of this network.

Corroboration. We will assume that the network *S=(M, R, m_0)* exists a vertex m_q, which is part of two different subnets *T(g_1)* and *T(g_2)*, $g_1 \neq g_2$, $T(g_1) \notin T(g_2)$. Then, by Theorem 1, the roots g_1 and g_2 are the ancestors of vertex m_q, that is $g_1 \in MP_q$ and $g_2 \in MP_q$. It means that each way *(g_1,…, m_q)* from a root of a subnet *T(g_1)* to vertex m_q contains a root of a subnet *T(g_2)*, that is $g_2 \in T(g_1)$, or each way *(g_2,…, m_q)* from a root of a subnet *T(g_2)* to vertex m_q contains a root of a subnet *T(g_1)*, that is $g_1 \in T(g_2)$. However, in the first case existence of a subnet *T(g2)* is denied, and in the second case existence of a subnet *T(g1)* is denied. It proves impossibility of accessory of vertex m_q simultaneously to two or more subnets.

Theorem 3. Let *T(g)* – a subnet of network *S=(M, R, m_0)*. Some vertex $m_q \in M$, $m_q \neq g$ can be in subnet *T(g)*, if only all her direct predecessors are in this subnet.

Corroboration. Let's say that it not so. Then subnet *T(g)* would haven't a single input vertex, which contradicts subnet definition.

Consequence 1. If all direct predecessors of some vertex $m_q \in M$ of a network *S* aren't a part of one subnet, then vertex m_q is a root for other subnet of this network.

We will call technologically admissible vertices sequence of a subnet *T(g)* as an ordered vertices sequence *(m_1,..., m_j,..., m_q)*, *m_1=g*, satisfies the following condition: if vertices of a subnet process in this sequence, then for each vertex m_j, $1 < j \leq q$, all her predecessors in the subnets reached along ways from a root of a subnet, not containing cycles, will be processed up to vertex m_j.

Network *S=(M, R, m_0)* can "break up" on a final set of subnets *I={T(g_1), T(g_2),..., T(g_j),... }*. If each subnet present as a single vertex, then for subnets can set the same relations as for vertices of network *S*.

Let the final set of a network subnets *S=(M, R, m_0)* be *I*. We will call a network $S^2 = \left(M^2, R^2, m_0^2\right)$ as an integrated second rank network for a network *S*, if the following conditions are satisfied.

1) The set of vertices M^2 of a network S^2 is a set of subnets of network *S*, that is $M^2 = I$, and each vertex of network S^2 is only one subnet of network *S*, and each subnet of a network *S* in the network S^2 represented only one vertex. We will consider that a subnet *T(g_j)* in the network S^2 represented vertex m_j^2 .

2) The set of edges R^2 is the logical communications between subnets of network *S*. That is $\left(m_j^2, m_i^2\right) = r_\mu^2 \in R^2$, if there is a vertex $m_q \in T\left(g_j\right)$, that $\left(m_q, m_\gamma\right) = r_f \in R$, where $m_\gamma = g_i$.

If in a subnet *T(g_j)* there are some vertices satisfying this condition, that edge $\left(m_j^2, m_i^2\right) = r_\mu^2 \in R^2$ represents all edges coming out vertices of a subnet *T(g_j)* and entering a root of a subnet *T(g_i)*. For example, two vertices 5 and 6 there are in a subnet *T(3)* of network shown in figure 3, which leave an edge in a root of a subnet *T(7)*.

3) Vertex m_0^2 represents that subnet of a network *S*, for which the root is the input vertex m_0 of this network.

Later network *S=(M, R, m_0)* we will call also a network of rank 1 $S^1 = \left(M^1, R^1, m_0^1\right)$, and its subnet – subnets of rank 1 *I^1={T^1(g_1), T^1(g_2),..., T^1(g_j),... }*.

Similarly, for a network *S=(M, R, m_0)* it is possible to determine integral networks of ranks 3, 4, etc.

Generally we will call a network $S^n = \left(M^n, R^n, m_0^n\right)$ as an integral network of rank *n* ($n \geq 2$) for a network *S=(M, R, m_0)*, if the following conditions are satisfied.

1) Set of vertices M^n represents a set of subnets I^{n-1} of an integral network S^{n-1} of rank *(n-1)*, that is $M^n = I^{n-1}$.

2) Set of edges R^n represents a set of logical communications between subnets of rank *(n-1)*, $\left(m_j^n, m_i^n\right) = r_\mu^n \in R^n$, if there is a vertex $m_q^{n-1} \in T^{n-1}\left(g_j\right)$, that $\left(m^{n-1}{}_q, m_\gamma^{n-1}\right) = r_f^{n-1} \in R^{n-1}$, where m_γ^{n-1} – root of subnet T^{n-1}*(g_i)*.

3) Vertex m_0^n represents a subnet of network $S^{n-1} = \left(M^{n-1}, R^{n-1}, m_0^{n-1}\right)$ for which the root is the vertex m_0^{n-1}.

From a network *S=(M, R, m_0)* it is possible to receive integral networks until the integral network of some rank *m* will be received.

Figures 5.1-5.4 show network S=(M, R, m_0) and sequence of its integrated networks.

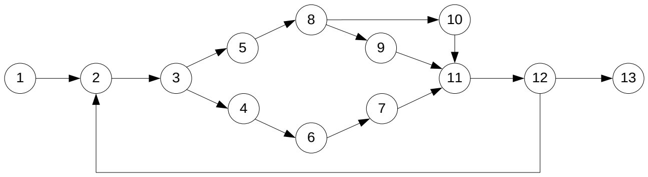


Figure 5.1 - Network S^1 = S

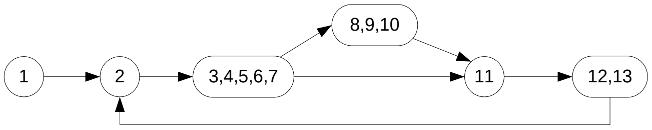


Figure 5.2 - Integral network of rank 2 S^2

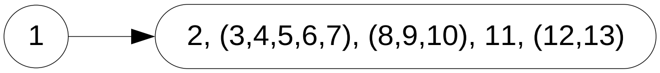


Figure 5.3 - Integral network of rank 3 S^3

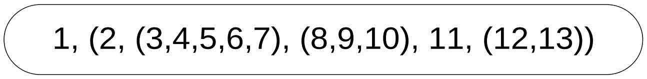


Figure 5.4 - Integral network of rank 4 S^4

It is necessary to notice, that not all networks $S=(M, R, m_0)$ can be reduced to a single vertex in the process of building integrated networks.

Figures 6.1 - 6.2 show example: integral network S^2 of rank 2 there is only for network $S=(M, R, m_0)$, and it contains some vertices, not a single vertex.

Each subnet S2 of integral network contains only one vertex. Therefore if we will construct the integral network S^3 of rank 3 we will receive the same network S^2.

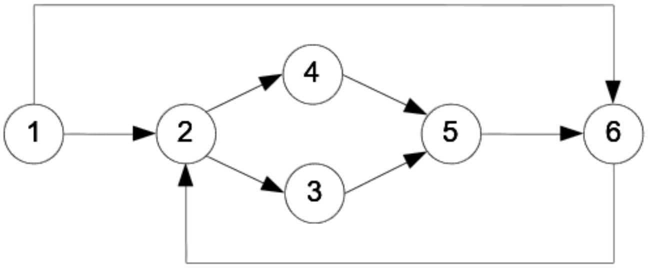


Figure 6.1 — Network $S^1 = S$

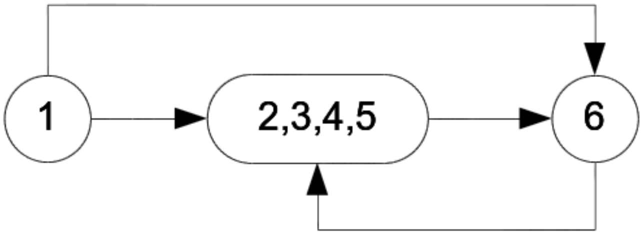


Figure 6.2 - Integral network S^2

The network model of works stages allows to integrate works for more intelligent perception of all necessary works at information system design. Presentation of works stages of the project as a network model allows to estimate scales of information system, is a basis for further planning and distribution of resources at realization of information system at the subsequent stages.

REFERENCES

[1] Boranbayev S.N. Methods for developing information systems. Astana: Master PO, 2012. -256 p.

[2] Boranbayev S.N. The theory of information systems. Astana: Elorda, 2006. -212 p.

[3] Boranbayev S.N. Mathematical Model for the Development and Performance of Sustainable Economic Programs // International Journal of Ecology and Development, Vol. 6, No. W07, 2007, p.15-20.

[4] Boranbayev A.S., Boranbayev S.N. Computer-Based Automation In Medicine And Proposed Methodology And Component-Based Architecture of a Computer-Assisted Posting and Processing of Medical Billing Claims. Proceedings of the 2010 International Conference on Internet Computing - ICOMP'10, Las Vegas, Nevada, USA, July 12-15, 2010, p.116-122.

[5] Boranbayev A.S., Boranbayev S.N. Defining Optimal Approaches And Methodologies for Concatenating Necessary Features of Various Java-Based Frameworks and Efficient Technologies for the Effective Development of Enterprise Information Systems. Proceedings of the 2010 International Conference on e-Learning, e-Business, Enterprise Information Systems, and e-Government - EEE'10, Las Vegas, Nevada, USA, July 12-15, 2010, p.385-390.

Proposal on Feeding Support Application Software based on Research into the Effect of a "Fasting State" on "Mental Alertness"

Motoichi Adachi
Dep. of Information System
Toyo University
Kawagoe, JAPAN
m-adachi@amy.hi-ho.ne.jp

Takayuki Fujimoto
Dep. of Information System
Toyo University
Kawagoe, JAPAN
me@fujimotokyo.com

Abstract

This research aims to find academic evidence to support the following phenomenon: Most people seem to feel *sluggish in a fed state* and *feel mentally alert in a fasting state*. Based on prior research that shows that Brain-derived neurotrophic factor (BDNF) which is found to improve brain function is secreted in a fasting state, this research reveals: how much fasting results in alert state ; how much time it takes to enter an alert state; how long an alert state lasts. Also, we propose the development of smartphone application software, that can notify the user of how *clear* their state of mind is at any given time using the mathematical calculation, not by relying just on feeling as before.

1. Introduction

Everyone has a time in which one's mind is *mentally alert* or *sluggish*. For instance, there is no doubt that one is obviously in a sluggish state when they have a fever due to a cold. On the other hand, there are moments when one feels like they can make good progress with any kind of task thanks to an alert state.

In such moments, what kinds of phenomenon generates brain function? With regards to this question, a study was previously carried out by The National Cardiovascular Centre Research Institute: Department of Molecular pathogenesis: Laboratory of Biomolecular Research. The findings of this study show (1): "brain derived neurotrophic factor(BDNF) plays a beneficial role in improving brain function by increasing neurite outgrowth and synapse formation, ... Approximately a 30% restriction on eating (diet control) increases Brain-derived neurotrophic factor(BDNF) to become brain nourishment. In a related study, professor Shuzo Kumagai from Kyushu University claims that (2): "It seems that dietary restriction has the effect on neurogenesis and neuroprotection by increasing neurotrophic factor. ..... Dietary restriction has a great influence on the brain as well as on the increase of Brain-derived neurotrophic factor(BDNF) expression". Thus, it is becoming clear as a result of the scientific research, not just with experiential understanding, that one is *alert when BDNF increases*, namely, in a *fasting state*(3)(4). However, there are two difficulties with these preceding studies.

1. It is quite difficult to know if a person's level of BDNF is increasing or decreasing in their daily life. In short, the medical approach, which is to have blood drawn, is not a practical method for a daily use. This is not *an effective technique to provide a clear indication of the extent of a person's alert state* which would decrease the burden of everyday life.

2. They claim through the use of brain science that *Brain-derived neurotrophic factor increases at a fasting state of approximately 30%* and one becomes alert. However, their studies do not clarify, on a general or daily perceptive level *how much fasting results in alert state; how clear that state is, or how long this alert state lasts*.

2. Research on the Relationship between the Fasting State and Being Alert

2.1. The Survey Methodology

To fulfill this research we first carried out a questionnaire to clarify the common feelings associated with being *alert* or *sluggish*. The experiment was conducted on a group of 113 randomly chosen people aged in their teens to their fifties. Here is the survey overview: In this survey, respondents recorded the *fasting state* and *alertness* every 30 minutes from just after waking up to just before going to sleep for 2 days by using the survey sheet indicated in Figure 1.

The method of implementation is;

（Step.1） The survey participant records their age, gender, and the quantity of physical activity and exercise they do. Reference values of the quantity of physical activity and exercise are; *low* for *almost all day non-standing work*; *moderate* for *a lot of walking and physical activity*; *high* for *active in exercise*. This is implemented in line with the *Exercise and Physical Activity Guide for Health Promotion* by Ministry of Health, Labour and Welfare of Japan.

(**Step.2**) The survey participants start recording just after waking up. On the survey sheet, a 24 hour day is divided by 30 minutes. Every 30 minutes just after waking up, they record *fed/ fasting* state and *alert/ sluggish*.

There are three formats to record their perceived level. ○ × form is applied to record the more intuitional judgment as indicated as below.

<Fed Level>	<Fasting Level>
・Fed a little : ○	・Fasting a little : ×
・Fed : ○○	・Fasting : × ×
・Fed very much : ○○ ○	・Fasting very much : × × ×
・Neither : Blank	・Neither : Blank

<Alert Level>	<Sluggish Level>
・A little alert : ○	・A little sluggish : ×
・Alert : ○○	・Sluggish : × ×
・Very alert : ○○○	・Very sluggish : × × ×
・Neither : Blank	・Neither : Blank

(**Step.3**) The participants accurately record the time they eat meals or snacks.

(**Step.4**) Repeat this up to just before going to sleep.

2.2. Survey Results Overview

In this paper, we have undertaken the survey indicated above, on 113 people chosen at random. A summary of results is shown in Fig.2

Total	Alert3	Alert2	Alert1	Neither	Sluggish1	Sluggish2	Sluggish3
Fasting3	19	37	34	19	18	20	22
Fasting2	27	93	88	35	49	34	29
Fasting1	24	132	136	73	93	47	37
Neither	119	284	345	257	143	94	83
Fed1	24	76	151	102	113	82	30
Fed2	18	45	56	51	93	94	30
Fed3	6	27	29	24	48	33	34

(Fig. A. Survey Result1 n=113)

The *level of fasting / fed* are on the vertical axis and the horizontal axis represents the *alert/ sluggish level*. 1, 2, 3 indicates the number of "○" or "×". Fig. A is a matrix in which all the participants are counted based on the recorded points for each 30 minutes.

For example, the top left cell of the matrix shows that there are 19 respondents who were *alert* ○○○ *(very alert)* when they were *fasting* × × × *(very fasting)*.

The same manner is adapted to the bottom left cell, which means that there were 34 respondents who were *sluggish* ○○○ *(very sluggish)* when they were *fed* ××× *(very fed)*.

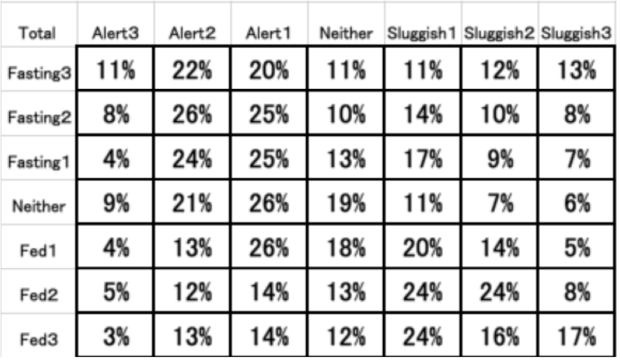
Full○									×	×														
Fasting×					×	×	×	×	×	×									×	×				
					×	×	×	×	×	×	○	○	○	○			×	×	×	×	○			
Alert○					×	×	×	×	×	×	×	×												
Sluggish×					×	×	×	×	×	×	×	×							×	×	×	○	○	○
					×	×	×	×	×	×	×	×	×	×	○	○	○	○	×	×	×	○	○	○
Time	5:00	5:30	6:00	6:30	7:00	7:30	8:00	8:30	9:00	9:30	10:00	10:30	11:00	11:30	12:00	12:30	13:00	13:30	14:00	14:30	15:00	15:30	16:00	16:30

Full○									○															
Fasting×								×	○	○														
			×	×	×	×	×	×	○	○	○													
Alert○							○	○	×	×	×	×	×	×										
Sluggish×	×	×			○	○	○	○	×	×	×	×	×	×										
	×	×	○	○	○	○	○	○	×	×	×	×	×	×										
Time	17:00	17:30	18:00	18:30	19:00	19:30	20:00	20:30	21:00	21:30	22:00	22:30	23:00	23:30	24:00	24:30	1:00	1:30	2:00	2:30	3:00	3:30	4:00	4:30

(Fig. 1. Survey Sheet)

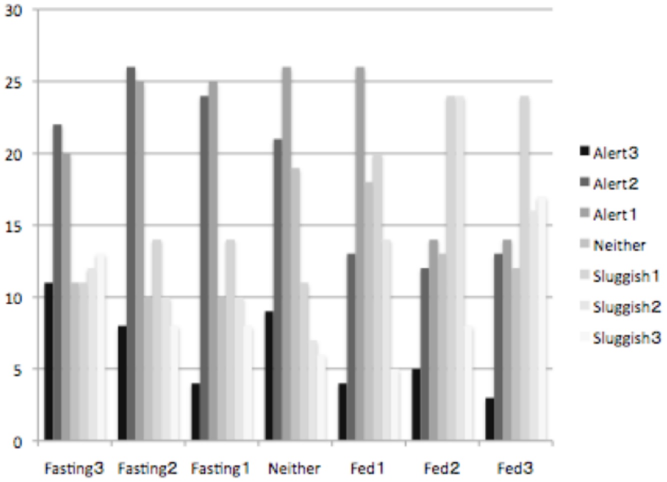
2.3. Analysis of Survey Results

Survey result matrix on *fasting* and *mental alertness* indicated in Fig. A is shown in terms of percentages in (Fig.2) for better understanding and is graphically represented (Fig.3).



Total	Alert3	Alert2	Alert1	Neither	Sluggish1	Sluggish2	Sluggish3
Fasting3	11%	22%	20%	11%	11%	12%	13%
Fasting2	8%	26%	25%	10%	14%	10%	8%
Fasting1	4%	24%	25%	13%	17%	9%	7%
Neither	9%	21%	26%	19%	11%	7%	6%
Fed1	4%	13%	26%	18%	20%	14%	5%
Fed2	5%	12%	14%	13%	24%	24%	8%
Fed3	3%	13%	14%	12%	24%	16%	17%

(Fig.2. Survey Result Ratio)

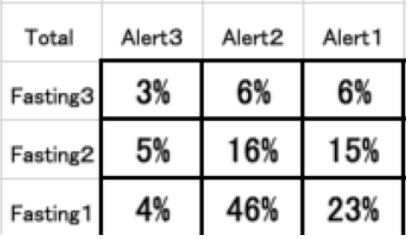


(Fig.3. Survey Result Graph)

The dark colored bars represent a *mentally alert* state and the light colored bars on the graphs represent a *sluggish* state in Fig.3. It is clear from the graphs that in most cases the participants are "*alert*" in the *fasting state* and as they become full it is clear that they tend to become less alert.

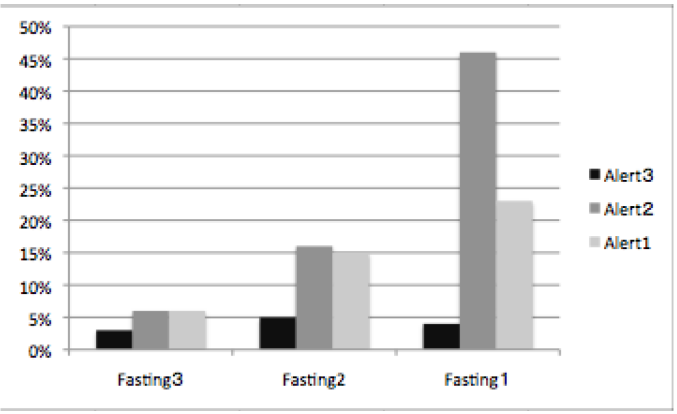
Fig.4 and Fig.5 shows a direct correlation between a *fasting state* and *alertness*.

They explain that alertness specifically tends to appear in the *little fasting state* out of the range of the *fasting state* level.



Total	Alert3	Alert2	Alert1
Fasting3	3%	6%	6%
Fasting2	5%	16%	15%
Fasting1	4%	46%	23%

(Fig.4. Correlation between *Fasting state* and *Mental Alertness*)



(Fig.5. Graphs Showing the Correlation between *Fasting state* and *Mental Alertness*)

3. Lapse Time between Fed to Fasting State and Alertness Duration

3.1. Lapse time from fed to fasting state

Regarding the lapse time between the fed to fasting state, at what stage does the state of being *alert* occur? The survey results so far reveal that *alertness* tends to appear in *a little fasting state*. Accordingly, the lapse time from *fed* to *alert state* is shown in Fig.6.

The average lapse time recorded by the participants is indicated below: from each fed state there are three levels to the beginning of alert state

- Fed1 (Fed a little) : Average 85 min.
- Fed2 (Fed) : Average 151 min.
- Fed3 (Fed very much) : Average 126 min.

	30min	1h	1h30	2h	2h30	3h	3h30	4h	4h30	5h	5h30	6h	6h30	7h	7h30	8h	10h	average
Fed1	6	5	4	6	2	1	0	0	0	0	0	0	0	0	0	0	0	85min
Fed2	4	6	8	13	7	3	3	2	2	2	1	1	1	0	0	1	0	151min
Fed3	3	3	15	5	7	5	1	3	0	0	1	0	1	0	0	0	0	126min

(Fig.6. Lapse Time from *Fed* to *Alert State*)

	30min	1h	1h30	2h	2h30	3h	3h30	4h	4h30	5h	5h30	6h	6h30	7h	7h30	8h	10h	average
Fed1	2	5	5	2	1	2	3	2	0	1	1	1	0	0	0	0	0	146min
Fed2	3	13	5	4	7	3	4	3	6	1	2	0	0	1	0	1	0	159min
Fed3	4	11	6	3	4	5	1	0	0	1	4	1	1	0	1	1	1	165min

(Fig.7. Duration of *being alert*)

It takes a remarkably short time to reach to *the beginning of alertness* in a low level of fed state. Both *Fed* and *Fed very much* states require more than 2 hours to reach *the beginning of alertness*.

3.3. Alert State Duration

The average duration that the state of being *alert* is maintained is indicated below.

- Fed1（Fed a little） : Average 146 min.
- Fed2（Fed） : Average 159 min.
- Fed3（Fed very much） : Average 165 min.

As demonstrated above, the duration of time from *the beginning of alert state* to the end of that state is uniformly around 150 min. regardless of the fed status.

4. Proposal for Smart Phone Application Software to Notify the Alert State

4.1. Proposal for Application Software

According to the survey, it is clear that *fed a little* state tends to be equivalent to the *alert* state and it is reasonable to say that the length of time during which the alert state is maintained can be identified to a certain extent. Based on the analysis of the survey results, we propose the smartphone application software which enables the user to make the most of this state of *being alert*.

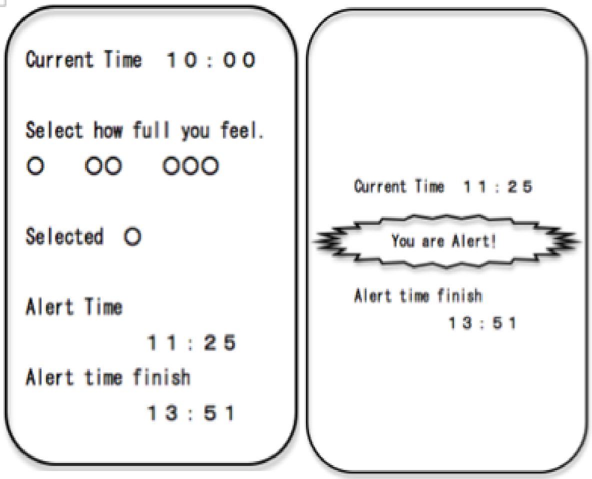
4.2. Application Configuration

In this paper we propose a smartphone application which has two main functions. The first is a function that displays the time in which the users are supposed to be in an *alert* state by calculating the meal-intake time and the fed level. The second function provides guidance related to meal-intake time and portion control so that the user can become *alert* at a specified time. Nowadays a large number of people carry their smartphone on them all the time, just as they do a wallet. It is effective to use this application on the smartphone, a device closely connected to our daily lives.

4.3. Alert Time Display Function

This function is utilized in conjunction with the built-in clock of the device. At first, the user inputs the *fed* level by using 1, 2, 3 every time they have had meal. 1 indicates the lowest fed level and 3 indicates the highest fed level.

The sign to notify that they have become alert will be displayed on the screen of the smartphone when it gets to the time, which is set in accordance with the length of time (Fed1: 85 min./ Fed2: 151 min./ Fed3: 126 min. later)from fed state to the beginning of the *alert* state presented in this research. *The approximate time the alert state ends* will also be displayed by adding the times (Fed1: 146 min. / Fed2: 159 min. / Fed3: 165 min. later) to the alert state beginning time.



Current Time 10：00

Select how full you feel.

○ ○○ ○○○

Selected ○

Alert Time

11：25

Alert time finish

13：51

Current Time 11：25

You are Alert!

Alert time finish

13：51

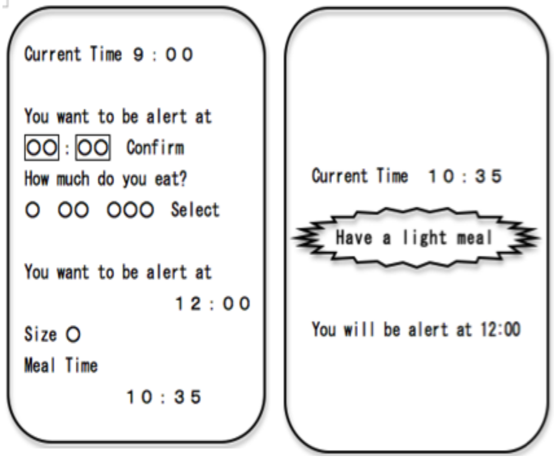
（Fig.8.Application Demo Image 1）

In short, this application aims to clearly indicate, by means of notifications, when a person is entering a state of being *alert* or *sluggish*. It allows the user to manage when they become alert. Up to now this state could only have been recognized intuitively by the individual.

4.4. Meal Time Guide Function to Create an Alert State

This function enables the user to enter an *alert state* by providing guidance on the meal times. This function aims to create the alert state intentionally by controlling the time and intake size of the meal in accordance with the targeted time, for instance, when there is *a particular time that you want to become alert* such as an important meeting. In the following example the user wants to become alert at noon.

If the meal size is *Fed1*, *alert state* will appear 85 min. after eating. Therefore it is ideal to intake the meal around 10:35AM, 85 min. before noon. In case of *Fed2*, *alert state* will appear 150 min. after eating. By using back calculation, this function suggests the user to intake the meal at 9:40AM, 150 min. before 12:00 noon by informing the user of the time.



Current Time 9：00

You want to be alert at

OO：OO Confirm

How much do you eat?

○ ○○ ○○○ Select

You want to be alert at

12：00

Size ○

Meal Time

10：35

Current Time 10：35

Have a light meal

You will be alert at 12:00

（Fig.9. Application Demo Image 2）

5. Further Research

As for becoming alert, it goes without saying that in our daily lives there is a wide variety of factors other than fasting or fed states which affect this. For example, washing your face with cold water, exciting events, light exercise, taking a deep breath, etc. Indeed, the following cases are identified in the questionnaire; the case in which one becomes alert suddenly regardless of the fed of fasting state; the case in which even though alertness appears once, it disappears soon; the case in which alert time lasts longer than usual. There is room for further research in this area.

Furthermore, In relation to the research into the perception of each state, *alert* and *sluggish*, *the definitions used to indicate each state* are fundamentally vague. We would like to clarify what is meant by *alert* or *sluggish* and to explain the differences in the level of each state. Likewise, we would like to explain the definition of *fed* and *fasting* state as well as their degree of variation more clearly.

However, the present study has highlighted certain flaws in the prior studies especially in relation to its usage in daily life, and with regards to the relevance of fasting/fed state and alertness. Therefore, an application that can objectively notify users when they are in an *alert* state, something that could previously only be identified by an individual through their senses, is indeed profoundly significant.

(Note） The reference values for the quantity of the physical activity follow the *Exercise and Physical Activity Guide for Health Promotion* by Ministry of Health, Labour and Welfare of Japan.

REFERENCES

(1)National Cerebral and Cardiovascular Center Research Institute: Department of Molecular Pathogenesis/ Pathophysiological Clarification and Therapeutic development for diseases related to cerebral nerve/surgery
http://www.ncvc.go.jp/res/divisions/etiology/et_005/index.html#4-2

(2) March,2007, RESEARCH IN EXERCISE EPIDEMIOLOGY Vol. 9, Japanese Association of Exercise Epidemiology Journal

(3)Hyperphagia, Severe Obesity, Impaired Cognitive Function, and Hyperactivity Associated With Functional Loss of One Copy of the Brain-Derived Neurotrophic Factor (BDNF) Gene
Juliette Gray1, Giles S.H. Yeo1, James J. Cox2, Jenny Morton3, Anna-Lynne R. Adlam4, Julia M. Keogh1, Jack A. Yanovski5, Areeg El Gharbawy5, Joan C. Han5, Y.C. Loraine Tung1, John R. Hodges4, F. Lucy Raymond2, Stephen O'Rahilly1 and I. Sadaf Farooqi1
Diabetes December 2006 vol. 55 no. 12

(4)Brain-derived neurotrophic factor (BDNF) and food intake regulation
Bruno Lebrun, Bruno Bariohay, Emmanuel Moyse, André Jean
Autonomic Neuroscience: Basic and Clinical Volume 126, Complete , Pages 30-38, 30 June 2006

Game-Based Learning Development Process: an exploratory analysis of game development companies in Brazil

João Coelho Neto[12], Sheila Reinehr[1], and Andreia Malucelli[1]
[1] Polytechnic School, Graduate Program in Computer Science - PPGIa, Pontifical Catholic University of Paraná State (PUCPR), Curitiba, Paraná, Brazil.
[2]Center of Human Sciences and Education, Mathematic Department, Northern Paraná Public University (UENP), Cornélio Procópio, Paraná, Brazil.

Abstract— *The development industry of the electronic games has a considerable comprisement, in both productivity and in the possibility to work with interdisciplinary matters in the process of cognitive and computational development. Considering the educational factors that the electronic games can provide, this paper has as the aim to identify how the educational electronic games are being developed by companies of Brazilian games development and if there is some specific process for the educational area. The aim of this study was to identify which development process used by these companies are, and if there is some specific development process for the educational area to the light of cognitive abilities. The methodological approach used was a qualitative research classified as an exploratory research. During the research it was identified that these companies use development processes adapted, there is not a specific process for the educational electronic games with cognitive and educational comprisement.*

Keywords- Game development; Process Development; Software Engineering; Cognitive Abilities.

1 INTRODUCTION

The electronic game industry in the late twentieth century had a remarkable growth. The electronic game area is coming to a new era of virtual reality and dynamism in its design, since this reality allows to explore processes and objects [1], [2].

Thus, the industry of development of electronic games has a considerable comprisement, both in the area of productivity, as regarding the possibilities of working in interdisciplinary issues of computational and cognitive development besides becoming a popular strategy for learning to be implemented in different educational contexts [3];[4].

Given all this interdisciplinarity that the development of an electronic game has, the making of electronic games is characterized by a process of software development, however, for this development some additional procedures are needed, such as: graphics, animations, script, and videos for the education area, educational and cognitive factors must be related with these procedures [5].

Thus, the development of electronic games is laborious, since it requires the effort of a team of qualified and willing to work together multidisciplinary professionals [6].

So the development process of electronic games is not an easy task, since the designs of these games involve multidisciplinary issues. For these reasons, developers adopt several practices of software engineering for the development of electronic games.

These practices, if any, are very varied and unsteady because they are the results of the mixture of an artistic creation with the software production. The high numbers of challenges and educational and technological innovations to be projected beside the cognitive process to be developed are immeasurable and represent obstacles to overcome.

However, for the development of an electronic game, especially in the area of education, several concepts are discussed, such as: art, sound, gameplay, control systems, artificial intelligence, human factors, and several other artifices intertwined with a traditional development of a software, it generates a scenario which increases more the complexity to build a game, especially in educational area[7].

In order these electronic games can be developed to enable learning, several educational and cognitive factors must be worked in the team, that is why the need for a multidisciplinary team.

These factors should be showed to all the participants, especially for the professionals in the area of computing, in order these approaches are internalized and thus they can be communicated in the development of an electronic game for the teaching and learning process.

Since the area of educational electronic games has grown rapidly due to its diversity and possibility of integration between technology and educational method because they can be defined as, educational games are those created to teach while entertain [8], [9], [10].

Based on its context, the use of educational games in the classroom allow the students opportunities to attract their attention, because they often love playing, and these types of games attract their attention [11] .

Thus, to nurture the use of interactive games in school settings, it is believed that this action will result in advantages for the educational processes of teaching and learning. It is considered that this methodological approach has as the claim the interaction, ingenuity, dynamism towards the consolidation and acquisition of knowledge, concepts and content [12].

Besides the advantages mentioned by Fernandes and Santos Junior [12], a game can fix the attention of the student, facilitating the absorption of knowledge on the attractive animations and other multimedia resources [13].

Considering the educational and cognitive possibilities that electronic games can provide, this article aims to identify how they are being developed by educational electronic games development companies, and if there is a specific procedure for the educational area.

Since these possibilities related to the knowledge of educational theories aim to think of the existing ideas on the cognitive processes, that is, how we are able to get information and change it into knowledge, theories such as behaviorism, cognitivism, constructivism and connectivism reflect the ideas in evolution of the cognitive psychologists over the last century [14] , because of this there was also the need to analyze these theories in the context of the development of educational electronic games.

Thus, as a contribution, there is the identification of this approach in the light of the understanding of the educational and cognitive abilities for the development of educational electronic games, enabling a focused analysis of information on the subject, which can be observed by companies that develop electronic games to the process of teaching and learning.

This work was divided into four sections: the first section aims to present the context of the use of processes and the development of several educational games and the importance of the use of these games in actions that enable teaching and learning, the second section, the methodological approach is described, in the third section, the analysis of the results is discussed, based on results reported by the semi-structured questionnaire, and the fourth and last section ,the conclusions of this subject are appreciated.

2 Research Method

This is a qualitative research, classified as Exploratory Research, this type of research was chosen since such a classification is designed to provide an overview of the studied fact, and a questionnaire was developed to help collect information [15], [16].

A semi-structured questionnaire was used to collect information, this questionnaire consisted of 12 questions and is presented in Table 1 of this paper.

Before being answered, the questionnaire was evaluated by eight experts in the areas of: Computing (Software Engineering and Games) and Education and afterwards applied to 27 professionals as a pilot test.

After the evaluation of the pilot test, the questionnaire developed by Google Docs was sent by e-mail to 38 companies that develop several and educational electronic games, these companies were registered in the Abragames website [17], as Brazilian companies of development of electronics games, only 12 questionnaires returned within 30 days.

The body of informants in this study was composed by twelve participants, composed of companies, and these were identified in the analysis of data of this research by codes: E1, E2, E3,…, E12. For the excerpts identified in the analysis, only a few were selected, whereas these showed consistent theoretical justifications in their notes, constituting the "corpus" of the research.

The results were analyzed by applying the questionnaire to the research participants. This questionnaire aimed to identify whether these professionals are using some development process that has some concern with the educational area, and in case they use it, which they are, allowing the design of the analyzed field.

TABLE I. Questionnaire to collect Information

QUESTION	QUESTION OBJECTIVE
Q1 - Participant Data	Identify the professional experience of the participant, if the participant is autonomous or not and in case he/she works in a company, identify the position.
Q2 - Information about the company	Identify the name, city, type of business, capital formation, market performance, company size and if it develops some courseware. This data was identified only for the participants who are not autonomous.
Q3 - What kind of game do you develop?	Identify if several electronic games and/or educational are developed
Q4 - Do you use any estimation method for game development?	Identify if the participant uses some method of estimation.
Q5 – Do you define strategies in the development process, that is, to define the activities that will be performed for the development of the game.	Identify if the participant defines strategies in the game development process.
Q6 - Do you use some kind of test during the development process of the game?	Identify the types of tests used in the development process.

Q7 - Do you use any environment for the process of games development?	Identify the environment of game development used in the market.
Q8 – Have you ever heard of specific processes for the development of educational electronic games?	Identify if the participant has any knowledge regarding to processes for the educational area. If yes, identify the model or simply a report of the steps known.
Q9 - If you know a specific development process for educational digital games, would you use it?	Identify if there is interest from the participants in using a specific process for developing educational electronic games.
Q10 – Should the process of developing educational electronic games take into account in its initial phase an educational theory?	Identify the importance for the participants of the association of a pedagogical concept in the initial process of developing educational electronic games.
Q11 - Do you use any process in the development of software in the development of the game?	Identify if the participant uses some process of development of software for game development.
Q11.1 - If YES, in which life cycle model is it based?	Identify the model of the life cycle that the participant uses, if any used.
Q11.2 – Have there been changes in the development process in the last three years?	Identify if there was an improvement after inserting some model of game development in the last three years.
Q12. Regarding to the development of educational electronic games	Identify the participant's knowledge about the pedagogical concepts.
Q12.1. Do you develop educational electronic games composed of questions and answers?	Identify which type of conception you use in the process of development of educational electronic games.
Q12.2 – Do you develop educational electronic games that use a range of information to enable the construction of the knowledge?	Identify which type of design you use in the process of development of educational electronic games.
Q12.3. Do you define at the beginning of the development process of educational games, the educational theory that will be used?	Identify if some kind of pedagogical conception is defined at the initial stages of the development.
Q12.4. If the previous question was "yes", which theory do you use for the development of educational electronic games?	Identify if some kind of pedagogical conception is defined in the initial stage of the development, if YES, identify the theory used based on the options available in the questionnaire.
Q12.5. Are there professionals in the area of education involved in the development process?	Identify if there are professionals in educational area that help in the identification of the pedagogical conception.
Q12.6. If "YES" in the previous question, in which phases?	Identify in which stage of the development, the professional of the educational area helps.

To identify the results of this scenario, the next section presents the analysis of the data obtained by the questionnaire sent to the companies of this survey.

3 Data Analisys

The data was divided into three major modules: the first module discusses the profile of the companies and the types of games they develop, the second module identifies the activities and development of several educational electronic games that these companies perform, the third module identifies if the companies who develop educational electronic games have some understanding of pedagogical concepts, such as, if these companies have professionals related to the educational area.

3.1. Module I - Profile of the Companies and Types of Games

All the participants who work in companies involved have professional experience in traditional software development and several electronic games.

In traditional software development, participants had 2 to 3 years experience; 2 participants 4-8 years, and 8 participants over 8 years of experience.

Regarding to the development experience of several electronic games, it was found that 2 participants had up to 3 years experience; 6 participants had 4-8 years experience, and 4 participants with more than 8 years.

Interestingly, the number of participants with over 8 years according to Abragames [17] informs that this area of games development is still relatively new comparing to the length of experience in traditional software development, in this case, it was observed in the participants a reasonable number of professionals with experience in the area of electronic games.

The data shows that there are two participants who also work as autonomous in the area of software development and electronic games, and the work performed by the participants in these companies were: programmer, with a higher incidence, it was obtained 6 participants, and the rest of the positions, with 1 participant each, these positions are: project manager, business analyst, developer, consultant, partner and director.

The companies involved in this research are mostly private companies, totaling 11 companies and one company was identified as a Non-Governmental Organization (NGO), being 5 companies with up to 10 employees; 4 companies 11 to 50 employees and 3 companies with over 350 employees.

Regarding to the constitution of the corporate capital, there are 11 companies with the constitution of National Capital, and only one company, with the constitution of Joint capital, it was also identified during the research that the market performance of these companies is global (3), national (7) and regional (2).

Based on this information, we identified that the majority of respondents work in companies with a market of national presence, and a significant percentage of 25% in companies with a Global market, since many foreign companies are operating in Brazil according to Abragames data [17].

However, it was interesting to report the localized of the companies analyzed in order to present the variation of locations reached by this research: 8 companies are located in the southern region, 3 located in the southeastern region and 1 located in the northeastern region.

The 12 companies develop several electronic games, and on observing the data, it was found that for the type of games that they develop there are 18 results, however, this option the companies participants had more than one alternative choose, it was obtained: 7 companies develop several electronic games and 11 companies develop educational electronic games.

3.2. Module II - Identification of Activities in the Electronic Game Development.

This section aimed to identify, even for companies that do not use a process of software development, the use of independent activities that may contribute to the development of electronic games.

3.2.1 Estimative Method

It was found that 6 companies use some estimating method for the development of electronic games and 6 other companies do not use estimation methods. Still, during the analysis, it was identified that the methods used are point by function (2), lines of code (1) man-hour (1) components (1) and PokerPlan (1).

3.2.2. Stages in the game development

It was possible to observe that 9 companies use some stage in the development process of electronic games and 3 companies do not use stages. Some excerpts show some of the stages used:

> *During the pre-production, we created a Game Design Document (GDD) and a prototype to validate the basic mechanic. In the production phase, we used SCRUM development methodology to define the activities (E2).*
>
> *Through the SCRUM (E3)*
>
> *We do the development planning, we list some architecture, design patterns, we do the documentation and develop by components (E11).*
>
> *[...] I do not follow strictly a single methodology (E12).*

From the excerpts presented on, it can be verified the diversity of stages that companies are using, identifying that they do not use a specific pattern, however, they use a variety of actions that suit the needs of the company.

3.2.3. Testing in the game development

It was observed that 9 companies use some kind of test in the game development process and 3 companies do not use tests. Some excerpts show some of the tests used:

> *Performance Tests, Usability Tests, Functionality, Tests (E2).*
>
> *Programming and testing after the development, the general test is performed with the team members (E3).*
>
> *The unit basic test, integration and system not very elaborated. I analyze if it is performing properly and if it returns what it should (E12).*

From the excerpts presented on, it can be identified a variety of tests used by the companies, checking that there is no specific pattern, however, they use this diversity in an attempt to adapt to the needs of the company.

3.2.4. Environment in the game development

Two companies use environments and 10 companies do not use environments for the process of games development. The 2 companies that use different environments cited the Unity 3D and FlexBuilder environment.

3.2.5. Specific processes for the educational electronic games development.

Firms were asked about the knowledge of specific processes for the development of educational electronic games. All respondents said they are unaware of specific processes for the development of educational electronic games.

3.2.6. The use of specific processes for educational electronic games development.

This stage had as the aim to evaluate in case the companies knew specific development process for educational electronic games, if these would use it.

Nine companies responded that they would use a specific process to educational electronic games and 3 would not use. It was observed that the three companies would not use, do not use methods or estimates and steps during the development of the educational computer game, ie, there is a lack of standardization of shares in its development.

3.2.7. Initial phase of development - educational concept

It was found that 9 companies find important in the early stage of the development process of the educational computer game to approach an educational design and 3 companies do not find it relevant. However, the 3 companies that do not consider it relevant are companies that develop several educational electronic games, with the greater emphasis on the development of several electronic games.

3.2.8. The use of a development process

During the data analysis process it was questioned if the companies use some type of software development process in the development of electronic games and, if yes, which models they based on.

It was possible to observe that 9 companies use some software development process in the development of electronic games, and 3 companies use their own processes.

The processes cited were: classical, cascade or linear - sequential (2); iterative model (1), Agile methodologies (5), RUP (1) own models (3), being:

> *Initially developed a prototype for the project's feasibility (E3).*
>
> *A mixture of several agile methodologies that help in the flow of the development using only the useful techniques and discarding the bureaucracy. We used for example a little Scrum, code review, etc. (E6).*
>
> *Simplified Agile + Heroes (E9).*

During the collection of this information, it was identified some procedures or adaptations of models made during the development of electronic games.

3.2.9. Changes in the development process in the last three years

It was observed that 10 companies indicated that there were changes in the development process in the last three years and 2 companies indicated that they didn’t have changes.

The main benefits with the use of a development process were: improvement in development time, improve the quality of the final product, improvement in productivity, relative cost and risk reduction.

3.3. Module III - Pedagogical Identification in the Educational Electronic Game Development.

In further analyzes it will be approached issues related to the development of educational electronic games process. These analyzes will be based on the understanding of companies in relation to the type of instructional design that is used in order to identify if these companies use some design and if they use, which is each one’s perception to educational electronic game developed.

Thus, if the development team knows these conceptions, they can use them in order to enable actions that compose develop expectations, or even cognitive abilities, which will be reflected in the educational electronic game.

The conceptions identified in this research were: instructional, which aims to make games consisted of questions and answers, creating a scenario of issues and alternatives, and the Constructivist or Constructionist which aims to make games formed by a set of information, creating a scenario of question which favor information about the algorithm to be developed, thus enabling the generation of knowledge by the means.

These conceptions if they are worked with the development team can be beneficial to the process of teaching and learning since the games provide recreational and specialized pedagogical objectives for the development of reasoning and learning situations, these are the main factors of the use of these instruments in the educational area [18].

The next questions are related to the understanding of the type of educational electronic games that are developed by companies.

3.3.1. Development of educational digital games consisted of questions and answers.

Five companies answered that develop games consisted of questions and answers and 7 companies replied that they do not develop.

3.3.2. Development of educational digital games using a set of information.

Seven companies develop educational electronic games using questions and answers and 5 companies do not use this format.

These questions aimed to identify if the professionals who are part of these companies understand the types of games developed with pedagogical concepts informed by the companies themselves. Thus, we attempted to identify if the development teams understands which pedagogical conceptions each type of game corresponds because each conception has an action on the cognitive development process of the learner.

3.3.3. Definition of the educational theory at the beginning of the development process of educational digital games.

It was identified that: 4 companies use some theory at the beginning of the development process and 8 companies do not use, among the concepts used it was identified as: constructivism (2) and instructionism (2).

With that, the next question aimed to identify if there are education professionals in the development of the educational electronic game helping these professionals of the area in the development of these educational issues.

But when analyzing along with the other items in this section, the E1 develops games formed by questions and answers and a set of information, but it identifies instructionism even having professionals of the area on his staff.

The E2 develops educational electronic games that consist of questions and answers and that the instructionism was pointed as a theory for the development, however, lack job in education involved in the development process in stage of creation of educational content.

The E3 develops educational electronic games which are formed by questions and answers and using a set of information to enable the construction of knowledge, define the conception in the early of the process, the instructionism theory was pointed as for the development of educational electronic games, and they have professional in education area involved in the development process at the stage of creation of pedagogical content.

The E4 develops several educational electronic games that consist of questions and answers it uses a set of information to enable the construction of the knowledge, it does not define conception at the beginning of the process, it does not have professionals in the area of education involved in the development process.

With that it identifies a disparity of information, because the professionals of these companies that participated in the interview indicate that they develop both types of games, 2 out of the 4 companies have professionals in education area working together, anyway, identifying contrary conceptions, indicating only one.

As they develop the two types, they should have selected for the instructional conception for the game formed by answer and question and Constructivist or Constructionist for the game forming a set of information.

3.3.4. Professionals in education area involved in the development process.

When asked about the involvement of professionals in the area of education during the development process of educational electronic games, 6 companies answered they do not have professionals in the area of education and 6 companies answered that they have these professionals.

In case of involvement of professionals in the area of education, companies were asked about the benefits with this addition, the results indicated benefits in: requirements collect with 4 answers; systems analysis with 1answer; tests with 4 answers, statement with 3 answers.

The amount of results differs from the number of companies as the participant could choose more than one option.

4 Final Conclusion

The research investigated about the development of educational electronic games in companies that develop several electronic games focused on the process of teaching and learning in Brazil.

It was identified during the survey that these companies use development processes adapted, without a specific process for the area of educational electronic games and even if they approach some cognitive or educational issue during the development stages.

Moreover, during the analysis, it was observed some discrepancies in answers among the companies, since that even developing educational electronic games, some do not answer all the questions relating to the definition of conception at the beginning of the development and a confusion among which conceptions are used for each type of educational electronic game developed even having education professionals on their teams.

The identification of the lack of educational actions during the development of an educational electronic game is worrisome because these approaches are of great importance for the development of pedagogical actions that will be externalized for the final game.

Thus, if the developers understand the pedagogical practices or even identify the cognitive abilities that can be used during the development of the educational electronic game, these actions can be inserted, thus helping to develop a game that facilitates the process of teaching and learning of the content proposed.

Therefore, as a consequence of the result of this research, it has been developed an Educational Electronic Games Development Process that enable the development teams to contact with educational and cognitive approaches in order to help them in generating inquiries that can be formalized in cognitive actions in the final product.

5 Acknowledgment

The authors would like to thank the workers at the companies who took the time to complete the questionnaire. We would especially like to thank the Fundação Araucária - Foundation for Scientific and Technological Development of the State of Paraná, Brazil, fostering agency, for financial support in the development of this research.

6 References

[1] João Ricardo Bittencourt; Lucia Giraffa. “A utilização dos Role-Playing Games Digitais no Processo de Ensino-Aprendizagem”. Technical Reports Series: PPGCC/FACIN, Number 03, September, 2003.http://www3.pucrs.br/pucrs/files/uni/poa/facin/pos/relatoriostec/tr031.pdf.

[2] Márcio Sarroglia Pinho. “Realidade Virtual como Ferramenta de Informática na Educação”. In: VII Simpósio Brasileiro de Informática na Educação (SBIE). Belo Horizonte, MG: SBC. pp.1-9, 2006.

[3] Alessandra de Souza; Ivette Kafur. “O fator emocional no desenvolvimento de jogos”. In: SBC Proceedings of XI SBGames – Art & Design Track – FullPapers – Brasília – DF, pp. 130-133, 2012.

[4] Vigdis Vangsnes; Nils Tore Gram Økland; Rune Krumsvik. “Computers games in pre-school settings: Didactical challenges when commercial educational computer games are implemented in kindergartens”. Computers & Education, 58, pp.1138-1148, 2012.

[5] Tiago Keller Ferreira. “Um processo para produção de Game Concept com base em Planejamento Estratégico”. 74f. Dissertação (Mestrado em Engenharia de Produção) – Universidade Federal de Santa Maria. Orientador: Dr. Marcos Cordeiro d´Ornellas. Santa Maria – RS, 2010.

[6] Bruce R. Maxim; Benjamin Ridgway. “Use of Interdisciplinary Teams in Game Development”. 37th. ASEE/IEEE Frontiers in Education Conference, Milwaukee, WI, pp. T2H1-T2H5, 2007.

[7] Fábio de Souza Petrillo. “Práticas ágeis no Processo de Desenvolvimento de Jogos Eletrônicos”. 168 f. Dissertação (Mestrado em Computação) – Universidade Federal do Rio Grande do Sul. Orientador: Dr. Marcelo Soares Pimenta. Porto Alegre – RS, 2008.

[8] Alexandre Souza Perucia; Antônio Córdova de Berthêm; Guilherme L. Bertschinger; Roberto Riberto Castro Menezes. “Desenvolvimento de Jogos Eletrônicos: Teoria e Prática”. 2.ed. São Paulo: Novatec Editora, 2007.

[9] Pablo Lavín-Mera; Pablo Moreno-Ger; Baltasar Fernández-Manjón. “Development of Educational Videogames in m-Learning Contexts”. In: Second IEEE International Conference on Digital Game and Intelligent Toy Enhanced Learning, IEEE. doi:10.1109/DIGITEL. 21. pp. 44-51, 2008.

[10] Jeannie Novak. “Desenvolvimento de games”. Trad. Pedro Cesar Conti; Rev. Téc. Paulo Marcos Figueiredo de Andrade. São Paulo: Cengage Learning, 2010.

[11] Enrique Barra Arias; Daniel Gallego Vico; Sandra Aguirre Herrera; Juan Quemada Vives. “A web tool to create educational content with gaming visualization”. Frontiers in Education Conference (FIE), Seattle, WA. pp. 1-6, 2012.

[12] Rúbia Juliana Gomes Fernandes; Guataçara Santos Junior. “The SIMS: Jogo Computacional como Ferramenta Pedagógica na construção do Conhecimento Matemático”. Revista Eletrônica TECCEN, Vassouras, v. 5, n. 1, pp. 21-36, jan/abr, 2012.

[13] Alex Machado; Paôla P. Cazetta; Priscyla C. dos Santos; Ana Mara O. Figueiredo; Leandro dos S. Sant’ana; Sebastião de Freitas Dutra; Esteban Clua. “Uma proposta de Jogo Digital 3D com questões didáticas”. In: XXII Simpósio Brasileiro de Informática na Educação (SBIE). Aracajú – SE, pp.620-629, 2011.

[14] David Gouveia; Duarte Lopes; Carlos Vaz de Carvalho. “Serious Gaming for Experiential Learning”. 41st ASEE/IEEE Frontiers in Education Conference, Rapid City, SD, , pp. T2G-1 – T2G-6, 2011.

[15] Antonio Carlos Gil. “Como elaborar Projetos de Pesquisa”. 5ª. Edição – São Paulo, Brazil: Editora Atlas, 2010.

[16] Marina de Andrade Marconi; Eva Maria Lakatos. “Fundamentos de Metodologia Científica”. 7ª. Edição – São Paulo, Brazil: Atlas, 2010.

[17] Abragames “A indústria brasileira de jogos eletrônicos: um mapeamento do crescimento do setor nos últimos 4 anos”. 2008. http://www.abragames.org/docs/Abragames-Pesquisa2008.pdf.

[18] Rafael Rieder; Elisângela Mara Zanelatto; Jacques Duílio Brancher. “Observação e Análise da Aplicação de jogos educacionais bidimensionais em um ambiente aberto”. In: IX Taller International de Software Educativo (TISE), Santiago – Chile, pp. 61-66, 2004.

Easel: Purely Functional Game Programming

Bryant Nelson, Joshua Archer, Nelson Rushton
(bryant.nelson | josh.archer | nelson.rushton) @ ttu.edu
Dept. of Computer Science, Texas Tech University
Box 43104 Lubbock, TX 79409-3104

Abstract – *In response to a growing interest in functional programming and its use in game development we've developed the Easel Framework which describes an engine for creating real time games by defining pure functions. This paper describes the framework and an implementation of this framework in SequenceL.*

Keywords: SequenceL, Easel, Game Programming, Functional Programming, Parallel Programming

1 Introduction

In his keynote address at Quakecon 2013, John Carmack, founder of Id Software and creator of the computer games Doom and Quake, shared his views on functional programming within the realm of video game development [1]. Carmack expressed that the use of pure functions simplifies the code base for very large projects by, among other things, ensuring that various parts of the software do not interfere with each other. The benefit of the modularity inherent in functional programming is something that has been known for some time [2]. This paper describes a game programming framework which allows games to be written in the pure functional language SequenceL [3], and an implementation of that framework in C#. The framework is called Easel and can be used to test Carmack's hypothesis in its purest form, by writing games without producing any new procedural code, or code with side effects.

There have been previous attempts at making a purely functional game programming framework. A fairly popular example is the ELM language developed by Evan Czaplicki [4]. In the case of ELM, an entirely new programming language was developed in an attempt to facilitate the creation of responsive GUIs using a functional language. There are also examples of using Haskell to program games [5].

These previous attempts at functional game engines try to handle everything, from the I/O and rendering to the game logic, in a functional language. Our opinion is that this leads to unintuitive engines, and complex game programs. Easel is an attempt to distill the logic of real time games down to its simplest form using a functional language, and then handle the rendering of these games in a procedural language.

2 Easel Description

The goal of Easel is to enable the creation of real-time games by defining pure functions. The Easel Framework is a system description for a game engine which has two key parts, a functional language used to write games for the engine, called the game implementation language, and a program which runs games written in that language, called the rendering backend.

2.1 Overview

The Easel framework requires a built-in data model, consisting of the following types, to be defined in the game implementation language:

- *Point* -- a structure of the form `(x: int, y:int)`
- *Color* -- a structure of the form
  `(red: int, blue: int, green: int)`
  with 0 ≤ red, blue, green ≤ 255
- *ImageType* -- one of the following strings:
  “segment”, “circle”, “text”, “disc”, “triangle”, or “graphic”
- *Image* – a structure of the form:
  ```
  (kind:ImageType, iColor:Color,
  vert1:Point, vert2:Point,
  vert3:Point, center:Point,
  radius:int, height:int, width:int,
  message:string, src:string)
  ```

  In practice only a subset of the fields will be used to define a specific kind of image, and there are six kinds of images:
  - ○ *Segment* --
    ```
    (kind:‶segment″,vert1:Point,
    vert2:Point, iColor:Color)
    ```
  - ○ *Circle* --
    ```
    (kind:‶circle″,center:Point,
    radius:int, iColor: Color)
    ```
  - ○ *Disc* --
    ```
     (kind: ‶disc″, center:Point,
    radius:int, iColor:Color)
    ```
  - ○ *FilledTriangle* --
    ```
    (kind:‶triangle″,vert1:Point,
    vert2:Point, vert3:Point,
    iColor: Color)
    ```

  - ○ *ImgFile* -- `(kind:"graphic",source :string, center:Point, height: int, width: int)`
- *Sprite* – a sequence of images
- *Click* -- a structure of the form `(clicked: bool, clPoint: Point)`
  If clicked is false then this interpreted that there was no mouse click in the given frame.
  If clicked is true, then point is the mouse click for the frame.
- *Input* -- a structure of the form `(iClick: Click, keys: String)`
  This is interpreted as the input vector for a given frame, consisting of a possible mouse click and a sequence ascii of codes of pressed keys.
- *Sound* -- a string which is "ding", "bang", "boing", "clap", or "click", or the name of a .wav or .mp3 file.

To create a game, the game implementation language is used to define the following type and functions:

- `State` -- a data type whose instances are possible states of the game
- `initialState()` -- the starting state of the game
- `images(`*`S: State`*`)` -- is a sequence whose members are the images to be displayed in the program window when the game is in state *S*.
- `sounds(`*`I: Input, S: State`*`)` -- a sequence of sounds played when input *I* is accepted in state *S*.
- `newState(`*`I: Input, S: State`*`)` -- the new state resulting from accepting input *I* in state *S*.

The rendering backend is responsible for executing the game, retrieving input from players, and displaying the images and sounds from the game. The overall algorithm for the rendering backend is presented in Figure 1.

```
S := initialState()
while True:
    display images(S)
    retrieve userInput
    play sounds(userInput, S)
    S := newState(userInput, S)
```

Figure 1: Algorithm `PlayGame`

The `PlayGame` algorithm is very similar to the standard game loop that is often encoded by hand when writing a game. The Easel engine, however, removes the need to write a main game loop. This abstraction allows the game programmer to focus on the logic of the game and not have to worry about the details of handling input from the user and rendering graphics to the screen.

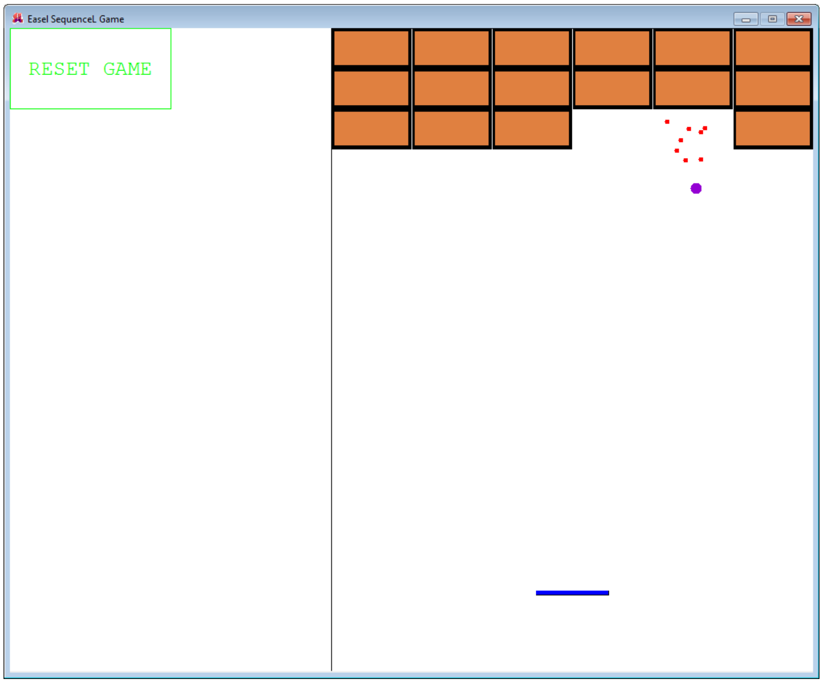


Figure 2: C# Rendering GUI Playing Breakout

2.2 Implementation

An Easel framework has been implemented, consisting of a rendering backend and graphical frontend written in C#, which runs games implemented using the Easel Framework in SequenceL. This implementation is referred to as Easel$_{SL}$.

SequenceL is a small, statically typed, general purpose, functional programming language [3]. The key reason for which SequenceL was chosen as the game implementation language is the fact that it is purely functional. SequenceL compiles to C++ code, allowing it to be easily interfaced with a graphical front end, which is a requirement for a game engine. Additionally, programs written in SequenceL are automatically compiled to highly parallel C++ [6]. All of these reasons contributed to the choice of SequenceL as the game implementation language.

Figure 3 shows a simple example of the SequenceL function definitions needed to encode a very simple game in the Easel$_{SL}$ game engine. The "game" simply displays the current time.

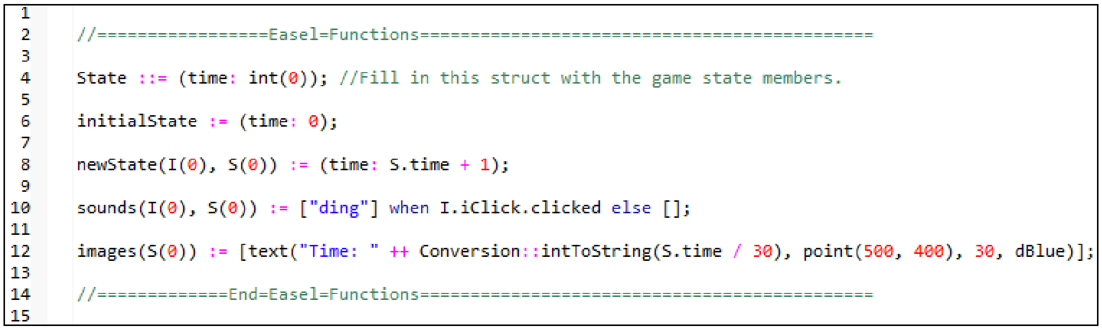
2.3 Rendering Backend

C# was chosen to implement the rendering backend, due to its extensive libraries and ease of graphical development.

An obvious drawback of this choice is that the framework is restricted to the Windows operating system.

The GDI+ libraries were used to render the graphics from the games. These libraries provide access to the standard Windows graphics API. They are not high-performance, but they have performed adequately thus far.

When a user runs the game engine, they are queried for the location of a game file written in SequenceL. The SequenceL compiler is then called to compile the game source file into C++ code. The Visual Studio C++ compiler is then called to compile that C++ code into a C++ DLL. The game engine is then able to access the Easel functions and execute the game.



```

//================Easel=Functions=============================================

State ::= (time: int(0)); //Fill in this struct with the game state members.

initialState := (time: 0);

newState(I(0), S(0)) := (time: S.time + 1);

sounds(I(0), S(0)) := ["ding"] when I.iClick.clicked else [];

images(S(0)) := [text("Time: " ++ Conversion::intToString(S.time / 30), point(500, 400), 30, dBlue)];

//============End=Easel=Functions=============================================

```

Figure 3: Simple Easel Game in SequenceL

The game engine runs the `PlayGame` algorithm until the player interrupts it by exiting the application.

3 Conclusions

Several simple games have now been written using the Easel$_{SL}$ engine. These games range from Tic-Tac-Toe to Breakout. The engine was in fact used in an undergraduate Concepts of Programming Languages course at Texas Tech University to provide students with hands-on experience using a functional language for game programming.

It has become apparent that there are some inherent limitations in the design of Easel. One such limitation is that s the state gets large, as in most games of considerable size, the new state becomes too large to efficiently pass by value. In addition, all game actions that affect a single intuitive state variable must be located in one place together, which can be unintuitive in complex games.

4 Future Work

Work is currently under way to extend the current game engine to directly support 3D rendering. The project has also inspired research into what is currently being called Concrete State Machine Language (CSML). Future work includes the use of abstract state machines calling SequenceL functions to address the limitations discussed in the previous section.

5 References

[1] John Carmack's keynote at Quakecon 2013 part 4. 2013, http://youtu.be/1PhArSujR_A.

[2] J. Hughes, "Why Functional Programming Matters," in The Computer Journal - Special issue on Lazy functional programming archive Volume 32 Issue 2, April 1989, pp. 98-107.

[3] B. Nemanich, D. Cooke, and J. N. Rushton, "SequenceL: transparency and multi-core parallelisms," in Proceedings of the 5th ACM SIGPLAN workshop on Declarative aspects of multicore programming, 2010, pp. 45–52.

[4] E. Czaplick, "Elm: Concurrent FRP for Functional GUI", Master's Thesis, Harvard School of Engineering and Applied Sciences, www.seas.harvard.edu/sites/default/files/files/archived/Czaplicki.pdf, March, 2012.

[5] M. H. Cheong, "Functional Programming and 3D Games," Master's Thesis, The University of New South Wales School of Computer Science and Engineering, www.cse.unsw.edu.au/~pls/thesis/munc-thesis.pdf, 2005.

[6] B. Nelson and J. N. Rushton, "Fully Automatic Parallel Programming," presented at the Worldcomp 2013, at The 2013 International Conference on Foundations of Computer Science, 2013.

Efficient and Effective Training in New Languages to Developers

M. Barjaktarovic
Department of Computer Science, Hawai'i Pacific University, Honolulu, Hawai'i, U.S.A.

***Abstract** - Learning to program is similar to learning a foreign language. There are many programs for "fast learning" of various foreign languages. How can such an efficient outcome be accomplished in programming? In industry, workers are expected to learn new languages, systems, and environments. In college, students often arrive with less than desirable level of skills in mathematics, reading comprehension, and a structured approach to studying and solving problems, in general. In this paper we describe a beginners' class in programming that efficiently and effectively trains learners to design and implement correct, documented, and tested code. After about 10 hours of in-class training, students can look up and use math API functions and solve quadratic equations. After about 15 hours of in-class training, students can look up and use random number function API and design, implement and test a simple game such as "heads or tails" or "paper, rock, scissors."*

Keywords: training, software, programming, language, test, development.

1 Introduction

Graduates with good programming skills are in high demand. Industry continuously demands higher quality and more competence in graduates as well as current state-of-the-art skills, such as test driven development, agile development, design patterns, and modern languages and environments such as Ruby or Hadoop. Industry expects employees to keep their skills current, typically on their own. Graduate school also expects students to arrive with sufficient training. For example, the new Georgia Tech online master's degree, produced in collaboration with AT&T, emphasizes the real-life skills of "systematic functional testing approach" which depends on working with functional specification and proceeding to identify and design test cases to drive the software development [10].

The issue then is how to train both experienced and newcomers in the new technologies, efficiently and effectively. Any programmers should be able to learn new technologies rather quickly if they have a solid foundation in programming concepts, deeply rooted in mathematics. Given that younger generations in the USA do not feel strongly about studying STEM fields and thus are less likely to choose and excel in programming, how do we produce adequate numbers of competent technical work force, capable of using mathematics and computer science tools to solve real-world problems. These questions are very important when assuring a country's economic prosperity and security; for example, competing in the global market and digitally protecting and defending the country's resources [3][7][9]. Efforts to increase understanding of effective undergraduate STEM education produced analysis of effective practices, directions for future research and suitable evaluation criteria [4][5][6][8].

The results of the 2012 PISA study show that American teenagers are scoring quite low in mathematics and reading skills. The U.S. ranking is 23-29 out of 34 developed countries worldwide [2]. In addition, college students tend to study less than desirable 2 hours per week of study outside of class per credit hour of instruction [1]. The issue that many faculty face in institutions in higher-level learning becomes: how to fill in the lack of prerequisite knowledge that students have when they enter college and how to prepare students to keep on progressing in their academic work. The same issue faces industry in terms of skills updates of current employees.

What steps are necessary to establish a solid foundation in the concepts and to keep learning and improving? As many mathematics and programming instructors know, there are particular issues when training students to program, due largely to students' lack of a solid mathematical foundation, formalized thinking, reading and comprehension skills, and structured work and study skills, all required for success in programming. Teaching substitution skills to students of all levels is discussed in [12]. More advanced technical skills of modular design, more advanced features of programming and extensive test development are described in [11] and [13].

In this paper we describe a beginners' class in programming, intended for students with no programming experience and/or anyone learning a new language/environment, that produces rather effective and efficient results. This class can be offered in academia, as in-house training, or as an independent course for only learning to program. After about 10 hours of in-class training, students can look up and use math application programming interface (API) functions and solve quadratic equation. After about 15 hours of in-class training, students can look up and use random number function API and design, implement and test a simple game such as "heads or tails" or "paper, rock, scissors."

The class is based on our own notes as a textbook. We examined many "learn to program books" and have not seen any using our approach, based on traditional practices in engineering. The language used is Java, but concepts apply to many other languages. We show students examples of the same code done in different languages to illustrate that the same concepts are used although the syntax may be different.

2 Prerequisites

There are certain skills necessary for success in programming, yet likely to be lacking in students and thus preventing them from learning. The most commonly lacking skills are:

1. substitution into formulas, which leads to not understanding how to use basic rules of programming language syntax and especially how to use API to call various functions.
2. ability to understand written instructions, i.e. reading and comprehension skills. Lack of these skills leads to inability to set up the problem, which then leads to inability to program it. This lack is amplified by the lack of mathematical understanding. Confusion is often related to mathematical and technical concepts, such as "ask the user to enter the input and then check if the input is in the right range" or "tabulate the results."
3. ability to think in a formalized, structured way, understanding steps. In other words, ability to think algorithmically and to design correct code.

These skills should be mastered in discrete mathematics course typically required of computer science freshman; however, students might have done poorly in the class and/or have not taken the class (for example, they are learning in a community college setting, this is in-house training for a new language, etc.). The programming instructor is well advised to somehow fill in the missing knowledge and further polish it, in a time efficient manner. The investment in mastering these skills will lead to a more productive and satisfactory programming experience. We find it most efficient to teach these skills through programming exercises.

3 Skills

Currently, programming is experiencing a renaissance in terms of mathematics, as unit testing and test-driven development has brought back flow charts, algorithms, and first order logic into programming. Without reliance on formalism of mathematics, programming can seem a mysterious activity that requires memorizing many seemingly meaningless and unrelated steps. Students are not able to see the similarity of concepts between various languages and then quickly shift from one language to another. In our experience, students learn rather fast when they are taught the "story" aka the reasoning behind the programming (involving hardware, computer architecture, and Boolean nature of digital components) and taught to program with awareness of the underlying hardware mechanisms. In addition, programming becomes a very effective tool to sharpen mathematical skills, because every step and every assumption has to be made explicit. We use a proven-to-be effective, traditional, basic engineering approach of "testable boxes," which requires some skills in abstract thinking and formalized approach.

The rigor and objectivity of mathematics requires some training. We observe in our classrooms that students unexperienced in mathematics do not like "being wrong." One aspect of programming is that compiler will report (in writing) anything "wrong" and will refuse to continue. A typical response from a new programmer (young or not) is to say: "The computer doesn't like me!" and to get rather frustrated. Indeed, programming is an objective activity, it is essentially an interaction with raw hardware without any artificial intelligence. We tell students that they are talking to a robot that has been programmed in a very limited way and understands only certain things, as specified by Java syntax. Students become enthusiastic about "interacting with a robot" since that viewpoint makes programming more of a game and most students love to play computer games.

Test-driven development is emphasized by enforcing a unit testing approach: designing suitable test cases and test stubs, providing expected results, running the code to see if it does produce the same results, and commenting on the process.

System analysis skills, working with functional specification and following instructions is emphasized by providing problems with simple yet particular instructions to be followed. For example, the exact format of pretty print, or the units and values required for calculation.

4 Starting with Hardware

The sequence of lessons followed can be described as:

1. Review of digital literacy concepts (hardware, operating system, computer architecture, programming languages, compilers, integrated development environments (IDEs). (1.5 hours)
2. Introduction to basic computing concepts (memory layout, variables, storing of variables, declaration, initialization, assignment statements). (1.5 hours)
3. Introduction to the first program: "Hello world!" to get students used to compiling and running code and IDE. (1.5 hours)
4. Introduction to basic programming concepts: assignments, using common functions, basic I/O, conditionals, loops, random numbers, arrays and strings. (the rest of semester, and covered in varied level of detail depending on students' abilities, needs, and number of credit hours counted towards the class, i.e. allocated time.)

In the following sections, we will describe each programming unit individually.

4.1 Learning by Mimicking Patterns

This portion of class is an introduction to variable naming conventions, code documentation, conversion of units, and appropriate print statements aka pretty print. Students are given a simple program that accomplishes something tangible in terms of computation; a simple problem that they have solved on paper in some math class. There are several purposes of this program: 1) Introduce different types of statements; get the students to intuitively grasp the meaning of various statements in code and learn the technical terminology of higher-level imperative languages such as initialization, declaration, assignment. 2) Introduce basic debugging and testing skills and test-driven development concepts. Students debug the code using the print statements as well as the debugger. 3) Introduce the concept of what the program knows and what must be kept in mind of the programmer, such as meaning of variables and their units.

For this purpose, we like to use a program that calculates the surface area and volume of a sphere, because these formulas are rather simple, familiar, yet illustrative enough. The code given to students always has an intentional bug in calculations. For example, we calculate the surface area of a sphere with radius to the third power, instead of second. Since students are just learning to program, they typically will miss that the formula is wrong until we test the program. Also, they expect code to be "correct" and this intentional twist is teaching them to look at code with a critical eye, read it carefully, and test it. They learn that a program that compiles does not necessarily produce correct results; they learn that it is very necessary to come up with test cases and test in an organized manner. (3 hours, with extensive review of all concepts learned so far.)

4.2 Memory and Variables, Terminology

The first program introduces the basic terminology and syntax and the intuitive feel for it. We proceed to explain the concepts in a more formalized way. Students are introduced to the concept of data types, variable storage, variable overwriting, integer truncation and casting, as it pertains to different types of basic statements (declaration, initialization, assignment, read-in, print). Memory layout and the way that variables are stored is discussed, in a rather simplistic but visually illustrative manner that students can relate to. In this portion of the class, students get used to the idea of reading code from top to bottom without any ability to return back to the earlier lines of code. Also, they learn to appreciate different data types and the storage required for them. At this point, features of different languages are discussed, such as automatic initialization vs. not, memory content of unused and/or uninitialized cells, type checking, and automatic casting. (1.5 hours)

4.3 Common Functions

We continue to work with the concepts from the first program in a more formal way. Students are introduced to commonly used mathematical functions and their API. This strategy is often not used for teaching programming to new beginners because it is rather mathematical and does require knowledge of mathematical functions, input arguments, output arguments, and substitution. However, this portion of class is meant as a speedy course in discrete mathematics, intended to address all such likely weak areas. The investment in this foundation pays off as students are able to independently use any functions from its API, such as random number generator and string operations. In addition, this portion of the course is used as a speedy course in reading skills and independent research, as students are asked to look up Java API and write a test stub for a particular function. Typically, many students have problems with these topics because their substitution aka pattern matching skills need to be more developed. (3 hours)

4.4 Conditionals; Defensive Programming

Students are introduced to conditional statements. This portion of class is an opportunity to introduce flow charts, logic, and algorithms. Conditional statements are taught first with pseudocode, in order to grasp underlying mathematical concepts behind syntax of any programming language.

Students typically quickly grasp logic when it is explained in rather formal notation, such as unary and binary operators, starting from a Boolean variable and making more complex Boolean statements using Boolean operators. Again, students are given intentionally buggy code and find errors in it; in this case, the most common errors are wrong data types, e.g. not using Boolean values for the "if" condition. Eventually, students work with actual Java syntax. Students are shown C/C++, Ruby, Python and Fortran syntax, and demonstrated the similarities in actual syntax and their equivalence in semantics. Students often have difficulties with switch statements. Depending on the class, it might or might not be necessary that all students fully grasp switch statements and thus not necessary to spend extra time on it.

This portion of class introduces skills for independent study and test-driven development by designing simple test programs to learn new concepts. Students run given Java code that uses "if" and nested "if" statements with print statements that help trace the program. Following the given pattern, students are asked to create and add their own test statements.

Software security is an important feature of good programs. Students are introduced to defensive programming by using conditionals to test if user input is valid for a particular function that is called. For example, is the user entering a positive number when calling a log function; is a square_root function called with an argument greater than or equal to 0. (3 hours)

At this point, students can independently write code to solve a quadratic equation with non-complex roots.

4.5 Random Numbers

At this point in the course, which is about 1.5 months of instruction 3 hours per week at a rather slow and methodical pace, students can write their own program from scratch, without any sample code shown by the instructor. During class, on their own, individually, they look up random() method from Java Math class API, and use it to generate a HEAD or TAILS message to the user. Most students move on and complete a “paper, rock, scissors” game which requires nested “if” statements and avoiding integer division. At this point, students are quite enthusiastic about their programming abilities and many will complete the game in its full form as “human versus computer.” (1.5 hours)

4.6 Strings and Arrays

Students are introduced to strings first by intuitive feel and then by a more formalized approach. Students are given sample code using Java String class method substring(. , .). They are asked to use this code as a starting pattern, apply their knowledge of how to use function declarations from API, and write their own test stubs for methods length() and concat(.,.).

In order to understand strings and various array specific terminology used in String API (such as beginning index, end index, sequence, or character array), students are introduced to the concept of array, array index, and value at the index. Since students are familiar with the concept of memory and storage, they can grasp the visual representation of array as a “row in memory” rather than “one cell” used for a single integer variable. (3 hours)

4.7 Loops as “if jump”

Students are introduced to “while,” “for” and “do-while” loops. A while loop is simply an “if-jump” structure (for example, as in assembly language syntax). When explained in this way (assuming that students truly understand the “if” statement and the concept of executing code line by line, top down, as covered in earlier sections), “while” loop can be learned in literally 10 minutes. From this point of view, it is easy to then introduce a “for” loop. Students also learn the concept of loop counters. The next step is to introduce loops in which the counter is used as a variable for some calculation inside the loop, and loops which do not depend on counters but some conditions. (3 hours)

4.8 Team Project

Students pair up and do a project of their choice. There is a list of suggested projects, however students are free to pick their own with the consent of instructor. The project will require some conditional statements, looping, strings, random numbers, and will most likely be in a form of a game. Students are very enthusiastic about writing their own game. More advanced students at this point can write code for matrix manipulation, for example to display an “avatar” in a random slot of a grid and to move it around. The class is allowed to spend about 2 last weeks of the semester coding during class time. (about 6 hrs)

4.9 Arrays and Subprograms

If time permits (e.g. the students came in better prepared) and/or depending on the needs of students (e.g. the entire class consists of only engineering students and/or is the only programming class in their curriculum), the next section would introduce arrays, including 2-D arrays, and writing subprograms. If it is necessary to make time to cover these topics, some of the previous topics can be skipped. However, if the entire class consists of engineering students, it is quite likely that everything can be covered.

5 Outcomes

We have taught programming at a large research university with generally better prepared students, at a smaller teaching university with a diverse student population, and at evening programs for adults changing their occupation to be more technical. The outcomes that we present here can be accomplished in 1 credit hour (1.5 hours of meeting per week in computer lab, for 15 weeks) with better prepared students; and in 3 credit hours (3 hours of meeting per week in computer lab, for 13 weeks) with less prepared students.

We provide the examples of tests used for learning Java, demonstrating what students can learn rather quickly. The tests can be easily translated into any other higher-level language.

5.1 Checkpoint #1

At the end of the first month (4 weeks of instruction for 3 hours per week in computer lab), a diverse group of new students, many not well prepared, can solve the following problems on their first test. “SOP” stands for “System.out.println” and is used just in this paper for the sake of saving space. Also, code is shown without any blank spaces for the same reason.

1. Complete the code below with the following:
   a. Declare an integer variable called k.
   b. Initialize k to 3.
   c. Add a user-friendly interface when inputting radius (with units)
   d. Add declaration and assignment statement for area
   e. Add declaration and assignment statement for volume
   f. Replace the 2 existing print statements with 1 print statement to print the values of area and volume on one line
   g. Replace the 2 existing print statements with 3 print statements that print a header and the values, showing

radius, area and volume with units. It should look approximately like this:

```
Radius    Area     Volume
cm        cm^2     cm^3
1         2.4      4.5
```

h. Write a message to the user on the screen, saying "Good bye!"

i. The data type of radius is ________.

```
//This code calculates area and volume of
sphere, given its radius
//Input: radius (from the screen)
//Output: area and volume of a sphere (to the
screen)
import java.util.Scanner;
public class Sphere {
public static void main(String argv[]) {
  Scanner input = new Scanner( System.in );
  double radius = input.nextDouble();
  SOP("Sphere volume is : " + (4.0/3.0) *
Math.PI * Math.pow(radius,2));
  SOP("Sphere Surface Area is : " +  4.0 *
Math.PI * Math.pow(radius,3));
}
```

The most common mistakes in this problem are not following the instructions carefully and printing two statements instead of one, wrong use of print statements, and declaring incorrectly.

2. Trace the following code. (If it could not run, say so).

```
int x = 5;        SOP(x)   //would print  _____
int y = 6.8;      SOP(y);                 _____
double  z = 3;    SOP(z);                 _____
x = x+1;          SOP(x);                 _____
x = 8;            SOP(x);                 _____
```

About 98% of all students get this problem correctly.

5.1.1 Results

This exam was taken by 19 new students, most in their second semester of attending college, many unprepared for college and one with a learning disability. The grades are shown below.

	Exam 1
Average grade	83.6
Std dev in grade	24.0
Max grade	100 out of 100
Min grade	17 out of 100
# of grades == 100	6
# of grades 90-100	7
# of grades 58-78	4
# of grades <36	2

5.2 Checkpoint #2

After an additional month of classes (total of 8 weeks, 3 hours of class per week in computer lab) new students can solve the following problems. Since new students might be still catching up with their mathematical skills, descriptions of Boolean operators and the algorithm for finding the solutions to quadratic equation are provided on the exam.

1. Trace the following pseudocode and fill in the blanks with what gets printed. If nothing gets printed, indicate so. Put check marks next to statements that get executed. Suggestion: also keep track of variables on the side.

```
A = 7
B = 10
IF (A > 5)
      B = 3
END IF
PRINT  A, B ________        ________

A = 10
IF (A > 5 AND A < 8)
      B = 4
ELSE
      B = 5
END IF
PRINT  A, B ________        ________
A = 1
IF (A > 5 OR A < 8)
      B = 10
      IF (A > 5)
           B = 20
      END IF
ELSE
      B = 30
END IF
      PRINT  A, B _______   _______
```

97% of all students solved this problem correctly and did not need to put check marks next to lines that "fired."

2. You have looked up Java API for Math class and you found the method called "random." This declaration and documentation is in the API:

// Returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0.

```
  static double random();
```

Fill in the following code skeleton to be able to call this method and test if it works correctly.

a. What is the data type of input argument to random()?

b. What is the data type of output?

c. Call the random method and store the result.

d. Print the result (PRINT or SOP is fine)

```
public class TestRandom {
   public static void main(String argv[]) {

   }
}
```

Only a few students could not do this problem. Most points were taken if students forgot to answer questions a. and b., or to declare the data type of the result of random().

3. a. Fill in code below to make a simple game. Ask the user to enter a number between 0 and 1. First check if the user entered a number in the right range. If not, inform the user and quit the program. If yes, inform them if they got head or tails. If the number is less than 0.5, print HEAD! to the screen. Otherwise, print TAIL! If you need Java logical operators, their descriptions are on page 1.

b. Provide three test cases (two “straightforward” and one “tricky”) and expected results.

c. **Extra credit:** instead of asking user to enter the initial number, generate the number using random() method.

```
import java.util.Scanner;
public class HeadsTails {
  public static void main(String argv[]) {
   Scanner input = new Scanner(System.in );

   x = input.next_________();

  }
}
```

About 95% of students completed part a. correctly. Five students lost 10 points since they did not answer part b. at all, either because they did not read the instructions, or because they did not understand how to generate test cases.

4. a. Write the code to calculate the solution(s) to quadratic equation. Ask the user to enter a, b and c, calculate x, and print out the result(s).

Use Math class sqrt(.) method. This is its declaration: static double sqrt(double x);

b. Describe how you would design the test cases and which test cases you would use.

Review:

Quadratic equation is of the form:

$ax^2 + bx + c = 0$ where a,b, and c are real constants, and x is any number

If b^2 – 4*a*c is less than 0, the solutions are complex and we do not know how to calculate them at this point. Inform the user.

If b^2 – 4*a*c is greater or equal to 0, and if a is not equal to 0, then there are two solutions for x (one solution for x has + and the other has – in front of the square root):

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

```
//This code provides solution to quadratic
equation ax^2 +bx + c = 0
//Input: a,b,c (from the screen)
//Output: x  (there can be one or two
values) (to the screen)
import java.util.Scanner;
public class Quadratic {
  public static void main(String argv[]) {
   Scanner input = new Scanner(System.in);

  }
}
```

This problem led to the most mistakes. Five students lost 10 points since they did not answer part b. at all, either because they did not read the instructions, or because they did not understand how to generate test cases. Among the incorrect solutions provided, the two most common problems are: 1. wrong algorithm; x is calculated before b^2 – 4*a*c <0 is checked. 2. students with less mathematical prowess were unsure how to translate the formulas given into code.

5.2.1 Results

This exam was taken by 17 beginner students. Two students with lowest grades on exam 1 dropped out, because they were missing class and/or not submitting homework. The grades are shown below.

	Exam 2
Average grade	88.4
Std dev in grade	12.5
Max grade	105 out of 100+5 pts extra credit
Min grade	66
# of grades ≥ 100	4
# of grades 90-100	7
# of grades 80-90	1
# of grades 66-78	5
# of grades <66	0

5.3 Student Feedback

It is known that students tend to study less than recommended 2 hrs per credit hour [1]. There are many reasons for this, including a lack of study skills, the need to work long hours which take away from study time, or a lack of motivation. 17 students filled midterm class evaluation after the second exam.

Table 1 – Student midterm feedback: number of students reporting their most difficult and easiest topics

Topics:	if	variables	strings	functions	trace
Most difficult	3	0	3	4	0
Easiest	5	4	0	1	1

Table 2 – Student midterm feedback: study habits.
Legend: 0 –none; 1-little; 2-some; 3-a lot

% of class attended	# hrs studied outside of class	# hrs I think I should study	% of class for which I was prepared	How much I asked questions	How much I paid attention in class	How much I learn in class
99	3	5-6	90	3	3	3
100	2	6	90	0	3	2
95	1-2	1-2	90	1	2	3
98	2	3	90	1	3	3
100	2	6	95	3	2.5	2.5
99	5	10	100	1	3	3
99	2-4	4-5	10	3	3	2
100	8	8	85	2	2	2
100	6	6	100	1	3	3
100	2-3	3	90	0	3	3
98	5	5	90	1	3	2
75	8	20	65	1	1	3
100	6-10	4-6	95	3	3	3
Average:	4.6	7.4	83.8	1.5	2.7	2.7

6 Conclusions

This paper presents our approach and results in teaching programming to beginners without any previous programming experience. This approach can be used in many environments such as academia, in-house training, independent programming courses, for both new programmers and experienced programmers learning a new language/system. The approach is based on a traditional engineering practice of testable units and understanding the underlying mechanisms. After about 10 hours of in-class training, students can look up and use math API functions and solve quadratic equation. After about 15 hours of in-class training, students can look up and use random number function API and design, implement and test a simple game such as “heads or tails” or “paper, rock, scissors.”

7 References

[1] A. C. McCormick, "It's About Time: What to Make of Reported Declines in How Much College Students Study (published in Liberaral Education vol.97, no.1)," 2014. [Online].Available: https://www.aacu.org/liberaleducation/le-wi11/LEWI11_McCormick.cfm.

[2] OECD, "Results for PISA 2012: United States," December 2013. [Online]. Available: http://www.oecd.org/pisa/keyfindings/PISA-2012-results-US.pdf.

[3] US News, "Report: Shortage of cyber experts may hinder government," 2014. [Online]. Available: http://abcnews.go.com/Technology/story?id=8147774.

[4] National Science Foundation (NSF), "NSF13-126 Common Guidelines for Education Research and Development," September 2013. [Online]. Available: http://www.nsf.gov/publications/pub_summ.jsp?ods_key=nsf13126.

[5] National Science Foundation (NSF), "NSF 13-127 FAQs for Common Guidelines for Education Research and Development," September 2013. [Online]. Available: http://www.nsf.gov/publications/pub_summ.jsp?ods_key=nsf13127.

[6] National Intitiative for Cybersecurity Education (NICE), "Home Page," 2014. [Online]. Available: http://csrc.nist.gov/nice/index.htm.

[7] PCAST, "Engage to Excel: Producing One Million Additional College Graduates with Degrees in STEM," February 2012. [Online]. Available: President’s Council of Advisors.

[8] National Research Council of National Academies, "Discipline Based Education Research: Understanding and Improving Learning in STEM" 2012. [Online]. Available: http://www.nap.edu/openbook.php?record_id=13362.

[9] The Joint Task Force on Computing Curricula ACM and IEEE Computer Society, "Computer Science Curricula 2013: Curriculum Guidelines forUndergraduate Degree Programs in Computer Science," December 2013. [Online]. Available: http://www.acm.org/education/CS2013-final-report.pdf.

[10] Georgia Institute of Technology, "Georgia Tech Online Master in Computer Science," 2014. [Online]. Available: http://www.omscs.gatech.edu/courses/.

[11] M. Barjaktarovic, "Training Effective Developers," in *The 11th International Conference on Software Engineering Research and Practice (SERP'12) (pp.375-380)*, Las Vegas, 2012.

[12] M. Barjaktarovic, "Teaching Mathematics and Programming Foundations Early in Curriculum Using Real-Life Multicultural Examples.," in *The International Conference on Frontiers of Education: Computer Science and Computer Engineering (FECS'12) (pp.78-84).*, Las Vegas, 2012.

[13] M. Barjaktarovic, "Teaching Design and Testing in Computer Science Curriculum," in *International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS'12) (pp. 106-112).* , Las Vegas, 2012.

An Experiment Comparing Easel with Pygame

Josh Archer, Bryant Nelson, and Nelson Rushton (nelson.rushton@gmail.com)
Computer Science, Texas Tech University, Lubbock, Texas, USA

Abstract – *A framework for developing games using a functional language was designed and implemented at Texas Tech. After it was created, a study took place. It consisted of taking a group of developers with the same task, splitting them in half and giving one group our system, and the other group a well known system. This paper describes the experiment, results, and future work.*

1 Introduction

Easel [Nelson 2014] is a framework for creating real time games by defining pure functions. It was designed principally for the purpose of game programming for math education. An easel game is created by defining the following types and functions in the functional programming language SequenceL [Cooke 2008]:

- *State -- a structure type whose instances are possible states of the game*
- *initialState() -- the starting state of the game*
- *images(S) -- If S is a state, images(S) is a sequence whose members are the images to be displayed in the game window when the game is in state S.*
- *sounds(I,S) -- a sequence of sounds played when input I is accepted in state S.*
- *newState(I,S) -- the new state resulting from accepting input I in state S.*

Given a file containing definitions for the types and functions above (and any helpers necessary), the game algorithm runs as follows until interrupted (typically, by the user closing the game window). Note that the PlayGame algorithm is implemented as a fixed C# program to be linked with SequenceL code written by the student/developer.

Algorithm PlayGame:

State variables:

```
  S: State, C: Click, K: list<char>, I: Input,
lastFrameTime: time-in-seconds
```

Procedure:

```
S := initialState()
while True:
set lastFrameTime to the current time
flip screen display to images(S)
```

If the left mouse button has clicked downward since the last frame, while the mouse was positioned in the game window, store the mouse position in C; otherwise set C equal to (clicked:false).

```
Set K equal to the list of depressed keys
I := (C,K)
play all of the sounds in sounds(I,S)
S := newState(I,S)
Pause until currentTime >= lastFrameTime + 0.0
```

This paper describes an exploratory study designed to test the ability of new game programmers (who are not new programmers) to develop simple games using Easel. The experiment is described in Section 2. The observed results are reported in Section 3, and Section 4 describes new hypotheses and future work.

2 Experimental Design

A group of 35 undergraduate students was divided into two groups alphabetically by last name, and two game design projects were assigned to each group. The first project that was assigned was called *Box Spin*, a simple game where the player can rotate and scale a box drawn on the screen. The second game was *Collision Course*, in which there is a disc in the center of the screen, and darts created by the player move at a constant rate towards the disc until they hit it. The specifications for the games follow, they take place on a 1000x800 pixel screen, with a constant framerate of 30 FPS

The specification for Box Spin is as follows: There is a square box in the center of the screen. The box can rotate left or right, and the box can grow or shrink. The box always remains centered, and the four edges of the box must be visible at any time. The state of the game consists of the length of the box's sides, and the box's orientation angle. The box begins with its size as 10x10 pixels, and its sides parallel to the *x* and *y* axes. The player can press any of the following keys to interact with the game: 'W', 'A', 'S', 'D', 'X. In each frame, the player can perform the following actions:

1. If 'A' is pressed and 'D' is not pressed, box rotates left by 3 degrees.
2. If 'D' is pressed and 'A' is not pressed, box rotates right by 3 degrees.

3. If ‘W’ is pressed and ‘S’ is not pressed and the size of the box is less than or equal to 500x500 pixels, then the sides of the box grow by 4 pixels.
4. If ‘S’ is pressed and ‘W’ is not pressed and the size of the box is greater than or equal to 10x10 pixels, then the sides of the box shrink by 4 pixels.
5. If ‘X’ is pressed then the game returns to the initial state.

The Collision Course game is defined as follows. There is a disc in the center of the screen. Darts (smaller discs) can appear wherever the player clicks on the screen. Darts will always move directly toward the disc in the center at a constant velocity until they reach the center, after which they disappear. The state of the game consists of a collection of darts and their positions, and whether the game is paused or not. The initial state of the game is an empty collection of darts, and unpaused. The player can press ‘X’, or ‘P’ to interact with the game. The player can click at any location on the game window. In each frame,

1. If the player clicks on the game window at point (*x*, *y*) outside of the disc, then a dart is created and centered at (*x, y*).
2. If the player presses ‘X’ then the game returns to its initial state.
3. If the player presses ‘P’ and the game is paused then, the game is unpaused.
4. If the player presses ‘P’ and the game is unpaused then, the game is paused.
5. If the game is not paused, then every dart moves directly toward the center of the screen by a distance of 3 pixels.
6. Any dart that reaches the center (will pass through (500,400) in the next frame) disappears.

In Phase I of the study, each student in the class was assigned to write Box Spin, with students in Group I using Easel and SequenceL, and students in Group II using Pygame and Python. The entire class was given the same specification for Box Spin, by which their submissions would be graded for success or failure. In addition to the spec, the class was given a lecture covering the math needed to implement the game. They were encouraged to come ask any questions needed during office hours.

In Phase II the students wrote *Collision Course*, and switched the languages, with Group I now using Pygame and Group II now using Easel. Once again the entire class was given a specification, a lecture on the math needed, and available office hours for help.

During both phases, students received links to the documentation for Python Pygame, SequenceL, and Easel. The documentation for SequenceL and Easel can be found at http://goo.gl/1UcEty. One of the Pygame tutorials students received was http://goo.gl/Ul8wZ4, which discusses the architecture of game loops and how to set the frame rate in a real time game. It is worth noting that most of the students had not used SequenceL before, while most had used Python since it is the CS1 language at Texas Tech.

3 A Priori Hypotheses

Going into the experiment, we had a few patterns that we would look for. Once such pattern would be that, contrary to intuition, the abundance of documentation and examples for Python/Pygame would actually cause difficulties for the students developing in that language. The idea behind this hypothesis is that given a plethora of information written by numerous authors on numerous subject, the developer would be overloaded with information that was not directly relevant to their task: learning to use the language properly for development.

The other hypothesis was that the nature of a functional language would greatly increase the ease of developing a game. We decided to look very closely at the flow of the programs that the subjects would create, seeing if ones built in SequenceL/Easel seemed to allow the developer to implement the specification as closely as possible with minimal translation from spec to product.

4 Observed Results

For project 1, the success rate for students who used Easel was 6 successes out of 18 attempts, and the success rate for students using PyGame was 3 successes out of 17 attempts. A “success” is defined here as writing a game that functions according to its specification. For project 2, the success rate for Easel was 7 out of 17, and the success rate for Pygame was 2 out of 18.

For both projects, incorrect submissions in PyGame were due to framerate most of the time. There were more runnable PyGame submissions than Easel ones. All but two runnable submissions (i.e., submitted programs that did not crash on opening) in Easel were correct.

5 New Hypotheses and Future Work

The most frequent errors in the Pygame programs involved handling the frame rate. We thus hypothesize that this is a stumbling block for new game programmers, and that the fact that it is handled automatically in Easel was a significant reason for the higher success rates for students using Easel. In the Box Spin game, frame rate errors explain all of the difference in success rates. They were, however, not a significant factor in Collision Course.

The large number of tutorials, and large amount of sample code available for Python and Pygame seemed to actually hurt the students’ success rates when using these tools. It seemed that students searched repeatedly for a library function or example that would solve their problems for them, ultimately without success. With Easel, on the other hand, students knew they would have to solve the kernel of the problem themselves, and so they rolled up their sleeves, got to it, and ultimately succeeded at a higher rate.

Ideally, we would like to conduct an experiment in which the students do each project in one observable session. We, as observers, have no real way to gauge exactly what the individual students' issues were with the games since they took the work home. This would also allow us to keep accurate track of the time students spent on each game.

6 References

[Cooke 2008] Daniel E. Cooke, J. Nelson Rushton, Brad Nemanich, Robert G. Watson, Per Andersen: Normalize, transpose, and distribute: An automatic approach for handling nonscalars. ACM Trans. Program. Lang. Syst. 30(2) (2008)

[Nelson 2014] Bryant Nelson, Josh Archer, and Nelson Rushton. Easel: Purely Functional Game Programming. Submitted to SERP 2014.

Electronic Private Library Portal

Shahriar Movafaghi
Department of Computer Information Technology
Southern New Hampshire University

ABSTRACT

The Electronic Private Library (EPL) consists of several different categories of media such as books, mail, photographs, audio, and video. Often these entities are located on different platforms. They may be accessible from personal computers, mobile devices, or located in cloud files. For the purpose of creating easy access to all user information, this paper explores the use of a portal to navigate through various applications and gather all items to one EPL.

Data is often accessible through various platforms, creating duplicates when aggregated. First, this paper explores the migration of data, and addresses the best way to handle duplicate data in the EPL. Second, the recommended process to becoming a wholly digitalized entity is explained. Third, should the user desire to create different versions of any given document, the issues associated with the complexity of the data manipulation and eBook versioning are addressed.

Finally, the means associated with creating an EPL is explored. This discussion focuses on several issues such as taxonomy, cost/benefits, archiving, customization and personalization of the portal. Information security is also discussed considering its paramount importance when creating an EPL.

1. INTRODUCTION

This paper explores how an individual can create an EPL consisting of several different categories, such as books, mail, photographs, audio, and video. In the industry, a digital firm is defined as an [1]: organization where nearly all significant business processes and relationships with customers, suppliers and employees are digitally enabled, and key corporate assets are managed through digital means. In any industry, the main reason for becoming a digital firm is to increase productivity and efficiency; ultimately helping the digital firm reach its goal of becoming more profitable. However, there is no correlation between the funds that a firm spends on information technology and productivity [2]. Similarly, one may assume there is no correlation between the amount an individual spends on an EPL and the overall monetary savings. Some of the data in an EPL may have sentimental value, such as photographs and video. These items may not be measurable solely in monetary values.

Personal libraries can include almost anything, although books may be the most typical component [3]. Some individuals disassociate themselves from electronic apparatus, to the extent that the user will ask their assistant to print their emails and then type the user's written response. Some individuals and students prefer print books rather than eBooks. However, print book circulation declined by 23% between 2005 and 2009 [4], suggesting the trend is towards a greater use of eBooks.

2. E-Documentations and eBooks

The creation of an EPL and the establishment of a paperless environment can be an overwhelming task for an individual. Therefore, we recommend gradual implementation. To begin with, the user should receive all transaction statements through primary personal email, forwarding all necessary emails from secondary accounts (personal or work related). This would include items such as financial institution statements, bank statements, credit card statements, tax statements, and store purchase receipts. If you are responding to an email using the secondary account, make sure to BCC (Blind Carbon Copy) the primary email account. The primary personal email account is the repository of all emails, therefore it can be an easy access source to search for all categories of documents.

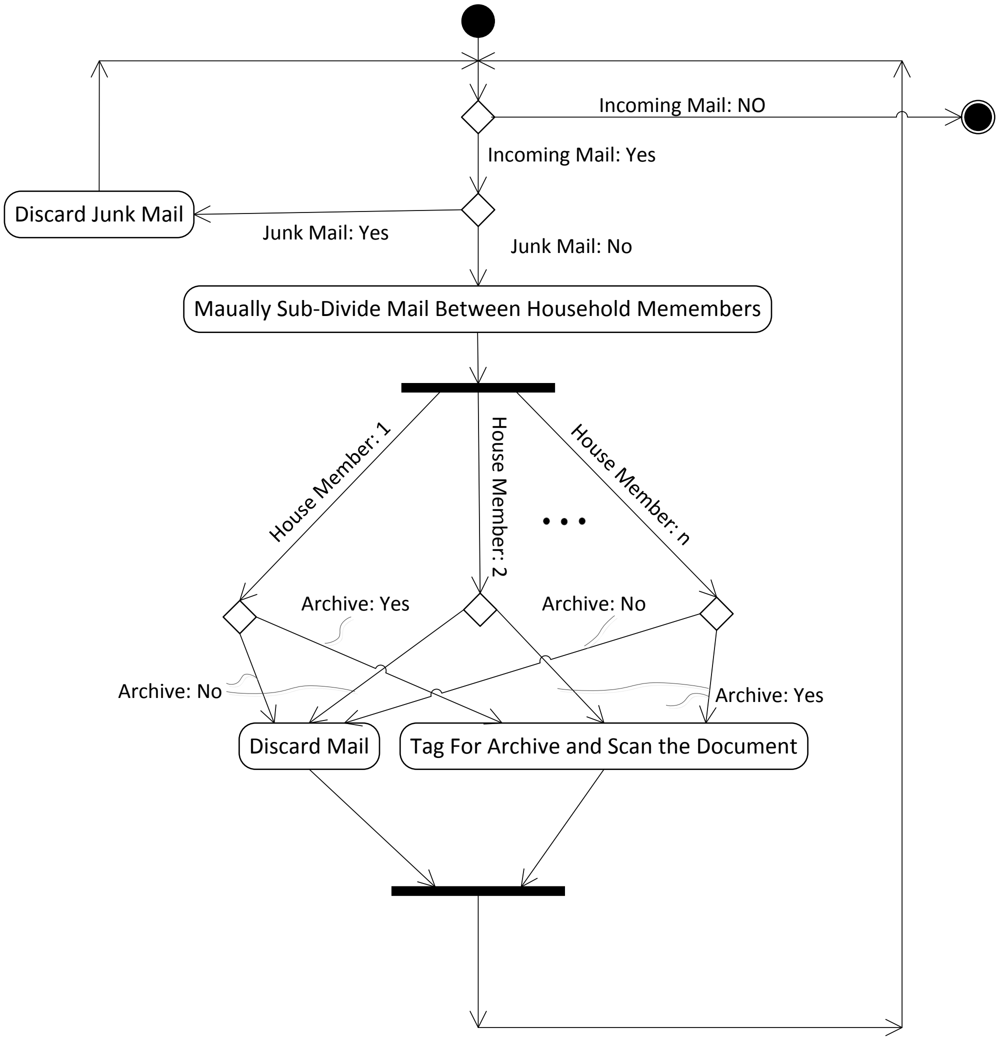


Figure 1 - Activity Diagram for Distribution of Incoming Mail for House Members

Figure 1 is the activity diagram for the distribution of incoming mail for house members. The primary household member is responsible to identify documentation that is important enough to scan and store digitally. We recommend selecting one platform to store your sensitive data such
as Microsoft Windows or Mac OS, which is distributed by Apple, Inc. One can store non-sensitive data in the clouds. Cloud computing is based on virtual servers. A virtual machine (VM) is a software implementation in a computing environment upon which an operating system (OS) or program can be installed and run. The virtual machine typically emulates a physical computing environment, but requests for CPU, memory, hard disk, network and other hardware resources are managed by a virtualization layer which translates these requests to the underlying physical hardware [5.a]. Virtual server technology can host various operating systems and applications using several features such as load balancing and failover [6].

With all its benefits, cloud computing also brings concerns about the security and privacy of information as a result of its size, structure, and geographical dispersion. Such concerns involve the following [7]:

- Leakage and unauthorized access of data among virtual machines running on the same server
- Failure of a cloud provider to properly handle and protect sensitive information
- Release of critical and sensitive data to law enforcement or government agencies without the approval and/or knowledge of the client
- Ability to meet compliance and regulatory requirements
- System crashes and failures that make the cloud service unavailable for extended periods of time
- Hackers breaking into client applications hosted on the cloud and acquiring and distributing sensitive information
- The robustness of the security protections instituted by the cloud provider
- The degree of interoperability available so that a client can easily move applications among different cloud providers and avoid "lock-in"

E-Documents can be stored using file systems or a documentation management system that uses a database system. When creating an EPL with a file system, store all documentations in one root directory which is subdivided to different categories. This makes it easier to back up the root directory by removing the need to remember different disk volumes and directories to back-up. Consistent use of standard file names is helpful and should be employed, for example name a file with a date stamp and file description (yyyy-mm-dd – description).

We recommend several levels of security for each file category. The user can set the level of security for different objects. You may consider four levels of security. The highest level of security would cover direct financial information for logging into your bank accounts or credit card providers. The second level would likely cover indirect financial, such as logging into an online store that will take credit card information, the third level would cover emails for personal use and social media and the lowest level would cover usernames/passwords to play games, watch video and/or animation.

Each level of security has many requirements that a user can establish. For example, the highest levels of security require a complex password that includes both upper and lower case characters, numbers, and special symbols. In addition to the password, the security login may require the user to select a preselected photo from a list of items as well as use of a numeric pin. The second level may only include the complex password. The third level of security may have a password that only includes alphanumeric and numeric passwords, and the fourth level may only contain alphanumeric characters.

The use of electronic reading devices has proliferated in the last few years. These reading devices appear to be particularly popular with young readers. For example, a generation of students familiar with computers, cell phones, iPods, and other high-tech devices is more likely to embrace electronic book technology for both educational and recreational reading. Educators and school librarians enthusiastically support the use of these devices as a means of encouraging students to read and explore information [8]. Some students and professionals use different applications such as Note Taker HD [5.b], GoodReader [5.c], and Adobe Reader [5.d]. These applications allow for commenting by using features such as highlighting text, sticky notes, and shapes. Regardless of different applications used, the document should be transferable to another platform without losing the comments.

We recommend using eBook tools that are portable to other devices when considering the formulation of an EPL. Figure 2 shows the typical architecture of a portable application. The primary responsibilities of the presentation layer are to display information to the user and to interpret commands from the user into actions upon the domain and data source [9]. Creating layers improves maintenance and cost, however performance declines.

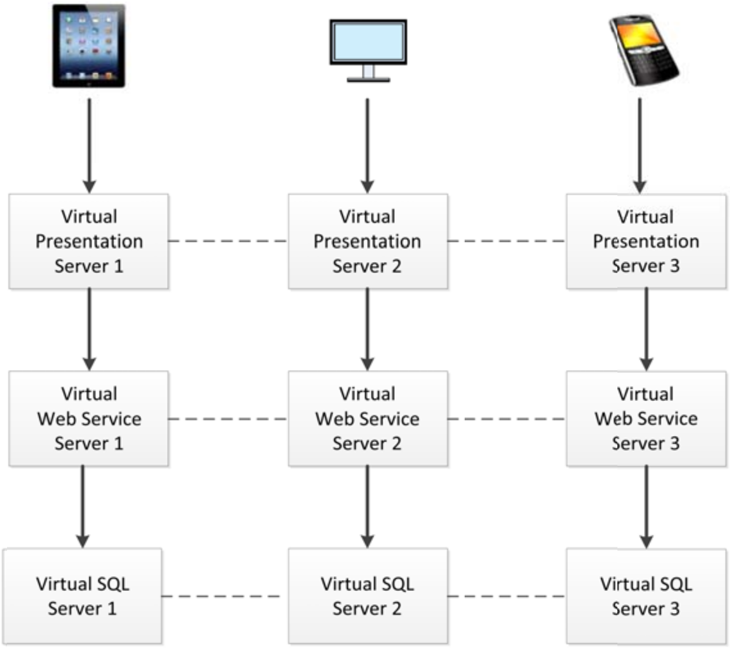

Virtual Presentation Server 1
Virtual Presentation Server 2
Virtual Presentation Server 3
Virtual Web Service Server 1
Virtual Web Service Server 2
Virtual Web Service Server 3
Virtual SQL Server 1
Virtual SQL Server 2
Virtual SQL Server 3

Figure 2 – Typical Architecture of the Portable Applications

A Web Service is a business component that provides a useful, reusable functionality to clients or consumers. A Web Service can be thought of as a component with truly global accessibility—if you have the appropriate access rights, you can make use of a Web service from anywhere in the world as long as your computer is connected to the Internet [10]. SQL server is used to access the data from the database.

3. Digital Photographs

Digital Photographs are files with formats such as: JPEG, PNG, GIF, TIFF, and BMP. The highest quality digital photo format is TIFF because TIFF files save all the characteristic of an image including layers. However, the size of the file is very large [5.e]. TIFF is a widely used image format that permits multiple representations of the same image [11]. Both PNG and BMP are lossless, meaning the quality in both of the formats will not degrade no matter what the image. PNG is supported by a majority of editing software available for images. BMP file format tends to be a large file size and not all programs will support BMP. The JPEG format is the smallest file format type available.

Most editing software supports the JPEG format, however, images may experience reduced quality upon conversion to the JPEG format. The size as well as the quality of the converted file to JPEG can be affected by the software a user choses to utilize to convert the file. The format commonly used for graphics displayed on web pages is GIF. The compression process for GIF files is lossless, similar to PNG and BMP, however, saving images that contain many colors using the GIF format which has a limited palette of colors, will result in decreased quality of photos. GIF files, unlike JPG files, support backgrounds that are transparent allowing GIF files to blend with the colors on the backgrounds of websites. The market share of the three leading formats on websites are from highest to lowest JPG, GIF and PNG, respectively [12]. We recommend using TIFF file format whenever possible in order to best aggregate digital photographs in an EPL. However, it is most convenient to convert your images to JPG for posting on the web and/or sending through email since the size is smaller than the TIFF format and there is minimal quality difference.

Petrelli and Whittaker [13] characterize and compare physical and digital mementos in the home, *"Physical mementos are highly valued, heterogeneous and support different types of recollection. Contrary to expectations, they are not purely representational, and can involve appropriating common objects and more idiosyncratic forms. In contrast, digital mementos were initially perceived as less valuable, although participants later reconsidered this. Overall digital mementos were more limited in function and expression than their physical counterparts, largely involving representational photos and videos."* We recommend taking a photo of items in the home that will remind you of an event that happened and then add an explanation of that event (Figure 3).

Rodden and Wood discuss the findings of a study that investigated how people manage their collection of digital photographs [14]. In any case, we recommend that users store photographs utilizing cloud computing services. Since the requirements for e-documentation, photos, audio and videos involve personal preference; we investigate three major cloud-computing services. Cloud computing companies were assessed according to a rating system based on the following criteria: reliability, price, security, company stability, application integration/migration, user friendliness, ease of accessibility, performance, support and tools. The three cloud computing companies that will be reviewed in this document include: Google Drive, Microsoft SkyDrive and Amazon Cloud Drive. Cloud computing needs can vary from user to user, so it is up to the user to select criteria that best satisfies the user needs.

Frame Caption: I feel that my life is like a puzzle: All the pieces fall into the place eventually and the amazing thing is that none of the pieces are missing. July 1999, Shahriar Movafaghi		*"On Christmas 1997, my daughter Olivia, who was only seven years old, received a 1000 piece puzzle from her brothers, Matthew and Stephen. I started working on the puzzle next to the Christmas tree. As time passed I had to move it to several other rooms until it was done. I was amazed that over a year and a half, I kept finding stray puzzle pieces in different locations."*

Figure 3- 1000 Pieces Puzzle Photograph

Google Drive allows users to have control over all "file types in a single place, including images" [5.f]. Users can "sync or upload files types in the following formats: .jpeg, .png, .gif, .tiff, and .bmp" [5.f]. Google Drive also allows users to share images with other users through email.

Microsoft SkyDrive allows users to store digital photos in the formats of JPG, JPEG, GIF, BMP, TIF, and TIFF. These images are displayed as thumbnails and can be viewed by other users on SkyDrive as well as in a slide show online, depending on the permission rights.

Amazon Cloud Drive grants users 5 GB of storage for free, which is roughly enough space to store around 2,000 photos. Compatible photo formats that can be uploaded to the Cloud Drive include: PNG, JPG, JPEG, GIF, BMP, TIF, and TIFF.

Amazon Cloud Drive grants users 5 GB of storage for free, which is roughly enough space to store around 2,000 photos. Compatible photo formats that can be uploaded to the Cloud Drive include: PNG, JPG, JPEG, GIF, BMP, TIF, and TIFF.

4. Digital Audio

There are many different types of digital audio files with various features and benefits that appeal to users depending on their needs. There are three basic types of digital audio files: uncompressed, or "common" systems, such as the WAV format; formats that use a compression technique, but lose absolutely none of the data in the compression, known as lossless compression; and formats that do lose some of the original data, but retain fairly high quality, known as lossy compression [5.g].

One of the most commonly utilized digital audio formats is WAV. WAV files are typically large in size due to the fact they are not at all compressed. WAV files are best used in situations where there is no concern for space.

Another well-known digital audio format is the MP3. MP3 files are very good to use when sharing online and can be used when space is a factor. MP3 can be compressed to a much more compact size than WAV formatted files. Although the quality is reduced due to compression, the difference is very small and is unnoticeable to most users.

Advanced Audio Coding, more commonly referred to as AAC is a popular digital audio format used on the Internet. It is a newer compression system, and is generally agreed to have a higher-quality sound at the same compression levels as MP3. AAC is also able to accept digital rights management (DRM) systems, which limit how the files can be used or transported [5.g].

A digital audio format that is not very well-known is Vorbis (.ogg extension). It is similar to MP3 and AAC and can be used as an alternative to MP3. The quality of Vorbis to MP3 is very comparable.

Google Drive permits users to “sync and upload” audio files such as: “Vorbis Audio codec, AAC audio codec, PCM audio, MP2 audio, MP3 audio, and WAV.

Microsoft SkyDrive allows users to store digital audio using a separate application called Sky Player. In order for users to access Sky Player they must sign into their SkyDrive account. Sky Player will catalog a user’s entire collection automatically. Sky Player gives users the ability to edit meta-data information as well as download songs to a phone, and create playlists [5.h].

Amazon Cloud Drive allows you to store photos, documents, and videos. In order to store digital audio files users must use Amazon Cloud Player. Cloud Drive and Cloud Player are separate services, each with their own subscription offerings. Cloud Player is used to store and play MP3s, and Cloud Drive is a user’s hard drive in the cloud [5.i]. If a user wishes to store audio files other than in the MP3 or AAC format, such as a wav format, it is recommended that the user use the Amazon Cloud Player Converter. Amazon Cloud Converter allows users to convert wav to MP3 or AAC that are supported by the Amazon Cloud Player [5.j].

5. Digital Video

There is a wide variety of digital video formats. The Moving Picture Experts Group often referred to as MPG is one of the most popular formats in the world. The quality of MPEG-1 is very low. The next advancement of MPEG is MPEG-2 followed by MPEG-4, or MP4 or M4v. MPEG-4 has the highest quality and contains many of the features of the MPEG-1 and MPEG-2 and other related standards. In addition, it adds new features such as (extended) VRML support for 3D rendering, object-oriented composite files (including audio, video and VRML objects), support for externally-specified Digital Rights Management and various types of interactivity [5.k].

Another digital video format is Audio Video Interleave, or AVI. AVI is simplistic in reference to its operation of storage and has the capabilities to be played in a wide range of available media players. The MOV file extension was created by Apple as a means to store and play video files [5.k]. The MOV format is used often due to the impressive capabilities for compression. The majority of videos created using digital cameras are typically saved in the MOV format as a default setting. MOV format is also associated with .qt and QuickTime. Advanced System Format, ASF, is comparable to AVI in relation to file compression. Streaming media is the most common use for the ASF format. Windows Media Video, or WMV, is another digital video format that can run with WMP and RealPlayer. Over the years the WMV format has grown to include support for high definition 720 and 1080 video [5.l]. There are several other digital video formats that are less popular than the ones mentioned above, including: Advanced System Format (ASF), Advanced Video Coding, High Definition (AVCHD); as well as Flash Video (FLV or SWF).

Google Drive gives users the ability to regulate file types in a single place, including video files. Google drive allows users to sync or upload video files up to 10GB in size, and sync or upload video files in the formats of: WebM files (Vp8 video codec; .MPEG4, 3GPP and MOV files – h264 and mpeg4 video codecs; AAC audio codec, .AVI [MJPEG video codec, .MPEGPS] MPEG2 video codec; .WMV, and .FLV (Adobe – FLV1 video codec).

Microsoft SkyDrive can play most MP4 (.mp4), Windows Media Video (.wmv), QuickTime movie (.mov), and Apple video files (.m4v) directly from SkyDrive which is found in most web browsers [5.m]. In order to play videos stored on SkyDrive in a web browser, Silverlight is usually a required installation.

The first half of the Cloud Player equation is the browser-based Amazon Cloud Player. This Web-based music player lets you play songs, create and manage playlists, and upload audio files. Unfortunately, you cannot upload files directly into Cloud Player. You are required to download Amazon MP3 Uploader to accomplish that task, which automatically scans your hard drive and uploads files into Amazon Cloud Drive [5.n]. Amazon Cloud Drive is compatible with AVI, MOV and WMV formats.

Table 1 shows the typical scoring model used to evaluate the three vendors, namely Google, Microsoft and Amazon based on each company’s cloud computing services. The criteria is defined by the user and can vary depending on different needs and expectations. The criteria weight is determined by what is most important to the user. However, reliability always gets a weight of 10 because if it does not regularly work the other criteria are irrelevant. The user should choose the top three vendors he or she believes will best meet their requirements, and after assessing these vendors, a score can be derived. There are several other resources available including the Internet, product reviews, and other related literature and documentation that can be researched. Once the assessment is complete a score will be generated and the highest grand total score will assist the user to make an informed choice of a qualified cloud computing option that will fit the needs of the user.

			Amazon		Google		Microsoft	
No	Criteria	Weight	%	Score	%	Score	%	Score
1	Reliability	10	80	800	95	950	51	510
2	Price	8	70	560	97	776	50	400
3	Security	5	60	300	93	465	52	260
4	Company Stability	7	98	686	98	686	98	686
5	Application Integration/Migration	6	70	420	97	582	52	312
6	User Friendliness	3	88	264	93	279	75	225
7	Ease of Accessibility	4	60	240	92	368	70	280
8	Performance	9	76	684	97	873	50	450
9	Support	2	80	160	89	178	85	170
10	Tools	1	87	87	95	95	67	67
Grand Total				4201		5252		3360

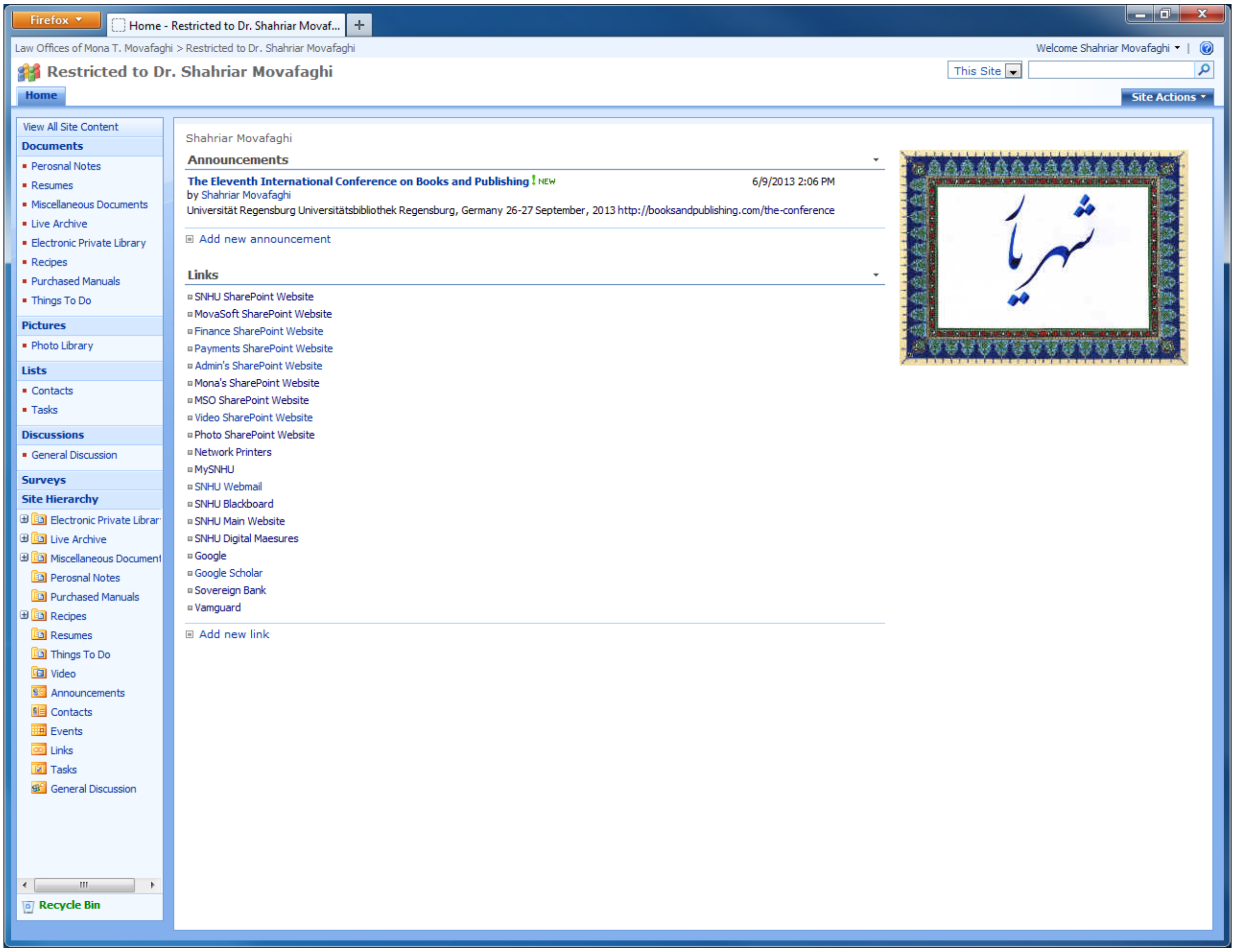
Table 1 – Typical Scoring Model for Selecting an Application

6. Library Portal

As was mentioned in the previous section, the user can store documentation, audio, and video using file systems or a documentation management system that uses a database system. eBooks can be stored in PDF formats, or on mobile devices that can be accessible from other mobile devices or the web.

There are several tools on the market that can be used to create the EPL portal. We recommend that the tool used for the EPL portal have a user-friendly customization feature. Various definitions exist for the term customization and the term personalization in e-commerce. For our purposes, customization occurs when the user has the ability to change the user interface they are viewing using mechanisms that are already built into the system. Personalization occurs when the system modifies the user interface pages on its own without direct input from the user [15]. Personalization is expensive and is used for more professional websites such the Health SmartLibrary [16].

Figure 5 shows a typical EPL portal using Microsoft SharePoint [17]. Microsoft SharePoint can be deployed on the intranet which has a highest security level, or deployed to cloud computing. SharePoint has an easy-to-use customization capability. The image on the right is the name Shahriar written in Farsi. The user can create several SharePoint websites with different categories for ease of access. You can also store and retrieve different version of a document using SharePoint.



Shahriar Movafaghi

Announcements

The Eleventh International Conference on Books and Publishing ! NEW 6/9/2013 2:06 PM
by Shahriar Movafaghi
Universität Regensburg Universitätsbibliothek Regensburg, Germany 26-27 September, 2013 http://booksandpublishing.com/the-conference

Add new announcement

Links

- SNHU SharePoint Website
- MovaSoft SharePoint Website
- Finance SharePoint Website
- Payments SharePoint Website
- Admin's SharePoint Website
- Mona's SharePoint Website
- MSO SharePoint Website
- Video SharePoint Website
- Photo SharePoint Website
- Network Printers
- MySNHU
- SNHU Webmail
- SNHU Blackboard
- SNHU Main Website
- SNHU Digital Maesures
- Google
- Google Scholar
- Sovereign Bank
- Vamguard

Add new link

Figure 4 - Typical Private Library Portal Using Microsoft SharePoint

7. Conclusion

The creation of an EPL and establishment of a paperless environment may take weeks, months or years to complete depending on resources and the complexity of integration of the applications used. In this paper we explore how an individual can create a private library consisting of several different categories such as books, mail, photographs, audio, and video files. The recommendation of how to store and retrieve each data type is discussed. The cloud computing capabilities for three vendors was analyzed and the reviews and recommendations are noted based on a scoring model sample. Finally, the typical EPL portal using Microsoft SharePoint is explained.

Copyright © 2014 CSREA Press, ISBN: 1-60132-286-0; Printed in the United States of America

References

[1] Laudon, K., and Laudon J., Management Information Systems (12th Edition). Prentice-Hall, 2012. ISBN-13: 978-0-13-214285-6.

[2] Brynjolfsson, E., VII Pillars Of Productivity, Optimize 2005.

[3] The Chronicle of Higher Education, six academics' comment on what is in their EPLs, ISSN 0009-5982, 09/2005, Volume 52, Issue 6, p. B.16.

[4] Rose-Wilesa, L., Are Print Books Dead? An Investigation of Book Circulation at a Mid-Sized Academic Library, Technical Services Quarterly, Volume 30, Issue 2, 2013, Pages 129-152.

[5] URLs for different websites

a. Definition of Virtual machine (VM) http://searchservervirtualization.techtarget.com/definition/virtual-machine

b. Note Taker HD http://www.notetakerhd.com/index.html

c. GoodReader http://www.goodiware.com/goodreader.html

d. Adobe Reader http://www.adobe.com/downloads/

e. TIFF, PNG, JPEG or BMP: Which image format offers the best quality **Not for web use**? http://uk.answers.yahoo.com/question/index?qid=20100210043542AALKiUO

f. Images in Google Drive https://support.google.com/drive/answer/2423575?hl=en

g. What are Different Types of Digital Audio Files? www.wisegeek.org/what-are-different-types-of-digital-audio-files.htm

h. Sky Player is an Elegant Way to Play Your SkyDrive Music http://winsource.com/2012/02/16/sky-player-is-an-elegant-way-to-play-your-skydrive-music/

i. What's the difference between Cloud Drive and Cloud Player? http://www.amazon.com/gp/help/customer/display.html?nodeId=200143320

j. WAV to amazon cloud player, hot to convert and upload WAV to amazon cloud player http://dream-cometure.over-blog.com/article-wav-to-amazon-cloud-player-how-to-convert-and-upload-wav-to-amazon-cloud-player-107810477.html

k. The Ultimate Guide to Digital Video Formats http://www.cepro.com/article/the_ultimate_guide_to_digital_video_formats/

l. Video Formats Explained http://www.videomaker.com/article/15362-video-formats-explained

m. Windows - Supported video formats http://windows.microsoft.com/en-us/skydrive/video-formats-supported-faq

n. Amazon Cloud Player http://www.pcmag.com/article2/0,2817,2382832,00.asp

[6] Movafaghi, S., Hojjati, S., Pournaghshband, H., Chan.T. , Collins, J.S, “Impact of Virtualization Technology in the IT Classroom”, WORLDCOMP'12 - The 2012 World Congress in Comp, Las Vegas, Nevada.

[7] Krutz. R., Vines, D., Cloud Security: A Comprehensive Guide to Secure Cloud Computing. ISBN: 9780470589878, John Wiley & Sons © 2010.

[8] Theresa, C., Privacy and E-Books, Knowledge, Quest 40. 3 (Jan/Feb 2012): 62-65.

[9] Fowler, M., Mee, R., Hieatt, E., Foemmel, M., Rice, D., Patterns of Enterprise Application Architecture, Addison-Wesley, 2012. ISBN-13: 978-0-321-12742-6.

[10] Sharp, J. Visual C# 2010. Microsoft Press, 2010. ISBN: 978-0-13-7356-2670-6

[11] Kanade, A., Alur, R., Rajamani, S., Representation Dependence Testing using Program Inversion, 2010, 10 Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering, Pages 277-286.

[12] Kemerer, C., Liu, C., Smith, M, Strategies for Tomorrow's 'Winners-Take-Some' Digital Goods Markets , Communications of the ACM, May 2013, Vol. 56, No. 5

[13] Petrelli, D., and Whittaker, S., Family memories in the home: contrasting physical and digital mementos, Personal and Ubiquitous Computing, ISSN 1617-4909, 02/2010, Volume 14, Issue 2, pp. 153 – 169.

[14] Rodden,K., Wood,K, How Do People Manage Their Digital Photographs?, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2003.

[15] Movafaghi, S., Chan, T., Pournaghband, H., Collins, J., The Customization and Personalization for Library Portal, The International Journal of the Book, Volume 5, Issue 3, pp.7-14, 2008.

[16] Shedlock, J., Frisque M., Hunt S., Walton L., Handler, J., Gillam M., Case study: the Health SmartLibrary experiences in web personalization and customization at the Galter Health Sciences Library, Northwestern University, Journal of the Medical Library Association : JMLA, ISSN 1536-5050, 04/2010, Volume 98, Issue 2, pp. 98 – 104.

[17] Microsoft Office System and Servers Team, Getting started with Microsoft SharePoint Foundation 2010, Microsoft Corporation, Published: June 2011.

SESSION

POSTERS

Chair(s)

TBA

Does Domain Knowledge Increase Creativity during Requirements Development: An Empirical Study

Sonu Sharma[1], Gursimran Walia[2], Kenneth Magel[3]
Microsoft Dynamics[1], Computer Science Department[2, 3]
Microsoft Corporation[1], North Dakota State University[2, 3]
Sonu.sharma@gmail.com[1], gursimran.walia@ndsu.edu[2], kenneth.magel@ndsu.edu[3]

Abstract— **To design and build a system/software, we need to understand the business of the organization. So, understanding the business is very important for requirement analysis of such system and requires an effective and efficient use of domain knowledge. In this paper we discuss the results of the empirical study that was carried out to understand the influence of domain knowledge on the creativity during requirements development.**

I. Introduction

Creativity refers to "the ability to produce new and original ideas and things" [1]. To that end, software development is the process of creating software solutions that are original and useful to end users. Researchers have begun to realize the importance of creativity in software development, especially during the later stages (e.g., design, implementation) of the software process [7]. Majority of creativity research has focused on developing computer-based tools for promoting creativity during the design phase of software development [7].

Generally, requirements engineering is not considered as creative process because of the notion that requirements are gathered from stakeholder and written in a particular notation and that all the creativity happens in the design phase. However, the invention of new software systems and products reveal that the process of discovering what the system will and would not do (i.e., requirements) require creativity. The design of each innovative product has a requirements stage where the stakeholders create, invent, ideas which form the requirements for the software system to be developed [9].

There are multiple factors (e.g., process, technology, domain etc.) that can affect the creativity during different stages (e.g., gathering, analysis, specification, management etc) of the requirements engineering process. We are trying to investigate the effect of "*domain knowledge"* on the creativity during the requirements gathering process. To accomplish that, we compared the creative solutions (i.e., requirement gathered) prepared by domain experts in their familiar domain and unfamiliar domain. This paper reports the results from an empirical study that evaluates the creativity of requirements developers by giving them two problems; one from the domain they are familiar with, and the other one from an unfamiliar domain. They were asked to come up with set of requirements based on problem statement and then another set of requirements based on detailed use case. Then, the set of requirements for each domain were plotted in a detailed concept map and each requirement was given a score based on a scoring technique. The cumulative score of all requirements for each participant is considered as their creativity score.

II. Research Approach: Quantifying Creativity

In this section we will describe our approach to quantify creativity in requirement phase. In order to understand the creativity scoring we used, we need to first understand concept map and its use in computing the creativity score. A concept map is a type of graphic organizer used to help organize and represent knowledge of a subject. Concept maps begin with a main idea (or concept) and then branch out to show how that main idea can be broken down into specific topic. Concept maps consist of nodes (points/vertices) and links (arcs/edges). Each node of the concept map represents a requirement..

Creativity Calculation: The requirement gets more specific as the level increase; this means Creativity is directly proportional to the depth of concept map. To keep the scoring simple we kept value of each node at same level equal to level value. Top level has value 1 and the level value increment by 1 at each level. Creativity score for each participant in each domain (familiar and unfamiliar) is calculated in two Phases. In order to describe C precisely, we introduce the following definitions and notations;

Let M = (R, E) represent a mental map tree, with $r \in R$ representing the set of requirements and $e_{ij} \in E$, represents the hierarchical relationship between r_i and r_j. Let $R_s \subseteq R$ define the set of nodes covered by requirements by a participant.

$C = \sum_{R_s} depth(r)$; Where depth(r) represents the depth of node r. Note that depth of root node is 1. Let R_{PD} be the set of requirements identified using problem description. Let R_{UC} be the set of requirements identified using use cases.

For Phase I (gathering requirements by reading problem description), the score is computed as; $C_{PD} = \sum_{R_{PD}} depth(r)$;

For Phase II (gathering requirements by using detailed use cases), the score is computed as; $C_{UC} = \sum_{R_{UC}} depth(r)$;

To keep the scoring simple we kept value of each node at same level equal to level value. Top level has value 1 and the level value increment by 1 at each level. Creativity score for each participant in each domain (familiar and unfamiliar) is calculated in two Phases.

III. Experiment Design

The main goal of this study is to compare the creativity of solution of domain experts in gathering requirements. The research question is focused with a part of study to evaluate the creativity of domain experts in software problem solving in

Copyright © 2014 CSREA Press, ISBN: 1-60132-286-0; Printed in the United States of America

their own domain and then compare the creativity of their solution in other domain.

We decided to utilize Concept Maps to judge the creativity in finding solution. A very detailed concept map showing requirement flow was created by experts in both the system used for case study in this paper. To quantify creativity, each node of the detailed concept map is given a score based on the level they are in. Each participant's sets of requirements are plotted into the detailed concept. The cumulative score of each participant set of requirement on the concept map is considered their creativity score. The higher a score the higher is considered the creativity of requirements. Even if two participants have same number of requirements their creativity score can differ. The lower a node in concept map a requirement fit in higher is its creativity score.

Nine Computer Science graduate students enrolled in the Software Design course at NDSU were given a problem description on ATM machine (familiar domain) and Communication process software (unfamiliar domain) and were asked to write functional requirements. Each individual was given a set of requirements from familiar domain and an unfamiliar domain and were asked to write the functional requirements. Each participant was required to list all different use cases and the requirements related to each of those use cases. Then, they analyzed each use case to come up with additional set of requirements.

The correctness of the solution/requirements was evaluated and each participant's requirements were placed in a detailed concept map to find the nodes of concept map covered by each participant's set of requirements. Once the detailed concept map was ready for each system, a score was provided to each node of concept map. The topmost node had value 1. And for each level the value of the node was increased by 1. Each node of the concept map represents a requirement for the system. The lower node in the concept map represents more specific requirements that their parents. The scoring of requirement for each participant was divided into two Phases. In Phase I, all the requirements gathered from problem description were plotted into each node of the concept map. And a score was calculated as summation ∑ (of each covered node value). In Phase II, the requirements gathered from detailed Use Case text were also plotted into the concept map and the score was calculated as summation ∑ (each node value covered by requirement created by use case).

IV. Research Results

This section provides results regarding the comparison of the creativity of requirements engineers' w.r.t the familiarity of the domain. We analyzed the creativity score for familiar and unfamiliar domain. Looking at the creativity score for each phase (I and II) and in each domain (familiar and unfamiliar), the requirements that user chose on Unfamiliar domain have higher creative score, thus they cover more complex scenarios.

When we consider the detail Use Case approach, the number of requirement and the creative score is much higher in familiar domain as compared to the unfamiliar domain. For detailed use case, the number of requirements gathered for familiar domain is much higher than those for unfamiliar domain. The same pattern is seen for creativity score.

The creativity score for requirements gathered on problem descriptions shows that out of 9, 6 participants scored higher in creativity of finding requirements. So, using problem description technique, 66.66% of participants had higher creative score in the domain they were not familiar with. But, with inclusion of detailed use case technique 77.77% of participants had higher creative score in familiar domain.

A one-way ANOVA was conducted to compare if there was a significant difference between familiar problem description creativity score (FAM_PD) and unfamiliar problem description creativity score (UNFAM_PD). Similar analysis was conducted to compare the creativity score as a result of utilizing the use cases between familiar (FAM_UC) and unfamiliar domains (UNFAM_UC). The analysis for FAM_PD and UNFAM_PD revealed that people in unfamiliar group exhibited statistically higher significant score (M=20.58, SD=6.36) when compared to scores obtained in familiar domain (M=12.57, SD=5.19) at p=0.01. These results show that domain experts come up with more creative requirements by reading problem description in unfamiliar domain.

Phase II (gathering requirements using use cases). One-way ANOVA between FAM_UC and UNFAM_UC revealed that people in familiar group exhibited statistically higher significant score (M=8.86, SD=4.22) when compared to scores obtained in unfamiliar domain (M=2.77, SD=1.85) at p=0.001. This result shows that the expertise in a domain helped stakeholders to consider more scenarios to come up with more creative requirements. This result shows that domain experts come up with more creative requirements in familiar domain using detailed use cases.

V. Discussion of Results

This study provides evidence that the creativity of an individual varies between the domains he/she is familiar with. The results show the importance of detailed use case in requirement phase. It can help a software organization be more creative in gathering requirements. This study shows that domain experts do a better job of finding requirements in their domain compare to unfamiliar domain. This study showed that using detailed use case approach did help in finding lot of new requirements for both domains. The number of requirements gathered in familiar domain using use case template approach was almost the double of unfamiliar domain.

References

[1] N. Bonnardel. Creativity in Design Activities: The Role of Analogies in a Constrained Cognitive Environment. 1999.

[2] U. Farooq, J.M. Carroll, C.H. Ganoe. Supporting Creativity in Distributed Scientific Communities. 2005.

[3] M.Bailey, C. Coats, K. Hamilton. Understanding Knowledge Management Practices for Early Design Activity and Its Implications for Reuse. 2009.

[4] L. Gabora. Cognitive Mechanisms Underlying the Creative Process.

[5] B. Rolfe. On the Production of Creative Subjectivity. 2009.

[6] The role of domain knowledge representation in requirements elicitation

[7] Neil Maiden, Suzanne Robertson & Alexis Gizikis Centre for HCI Design, City University, London Atlantic Systems Guild, London Provoking Creativity: Imagine What Your Requirements Could be Like.

Program Dependence Graph for Python's Dynamic Reference

Jangwu Jo[1], Hye-Yeon Lim[2] and Dae-Seong Kang[2]
[1]Department of Computer Engineering
[2]Department of Electronic Engineering
Dong-A University, Hadan-dong, Saha-gu, Busan Korea

Abstract – *A program dependence graph (PDG) is a graphic representation of the data and control dependencies within a procedure. PDGs have been used to detect software plagiarism. The current methods of constructing PDGs do not include data dependence incurred by Python's dynamic reference. This paper describes the situation when the conventional PDGs do not include data dependence incurred by Python's dynamic reference, and proposes an algorithm to add data dependence edges of Python's dynamic reference to PDGs.*

Keywords: Program dependence graph, Python, Dynamic reference, Program plagiarism detection

1 Introduction

Python is a general purpose, dynamic programming language, and is widely used in various fields, such as web applications, game development and the creation of desktop and mobile applications[1][2]. Like other dynamic languages, programs in Python are executed through interpreters. This means that source codes of Python can be open to its users. The easiness to access source code enables software plagiarism to be convenient. The research to detect plagiarism of Python software is needed.

The current techniques for plagiarism detection fall into four categories: String-based, AST(Abstract Syntax Tree)-based, Token-based, and PDG(Program Dependence Graph) based[3]. The first three techniques are known to be generally fragile to tricky plagiarism disguises, such as statement reordering and code insertion, but the last technique, PDG-based, is shown to be strong to above disguises[4][5].

Although PDG-based plagiarism detection is state of the art, it has shortcoming to detect plagiarism of Python programs. Due to the dynamic feature of Python, conventional methods of constructing PDGs cannot reflect the data dependence relation that is induced by Python's dynamic reference.

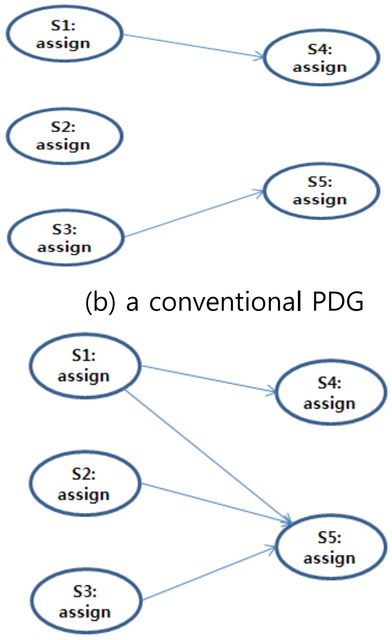
This paper describes the situation when the conventional PDGs do not include data dependence incurred by Python's dynamic reference, and proposes an algorithm to add data dependence edges of Python's dynamic reference to PDGs.

2 PDG and Dynamic reference of Python

Program Dependence Graph(PDG) is a graphic representation of the source code of a procedure[6]. The data and control dependencies between statements are represented by edges between program nodes in PDGs.

```
S0:      alice = { }
S1:      alice['x'] = 1
S2:      alice['y'] = 2
S3:      str = raw_input()
S4:      a = alice['x']
S5:      b = alice[str]
```

(a) Example code in Python



(b) a conventional PDG

(c) a PDG reflecting dynamic reference of python

Figure 1 An illustrative Example of Python's dynamic reference and its PDG

Figure 1 provides an example to illustrate the conventional PDG and PDG with Python's dynamic reference. Figure 1(b) depicts the PDG of the example code in Python in Figure 1(a). The edges in solid lines in Figure 1(b) represent data dependencies. But Figure 1(b) does not represent data dependencies caused by dynamic reference of Python. Even though the value of 'str' in assignment S5 cannot be known before execution, there may be data dependencies from assignment S1 and S2 to S5, since the value of '**str**', the value from **raw_input()** in S3, may be '**x**' or '**y**'. Figure 1(c) shows the added edges from Python's dynamic reference, as well as the edges in Figure 1(b).

3 Data dependencies from Python's dynamic reference

Figure 2. outlines how to add data dependencies of Python's dynamic reference. At line 1, every $\boldsymbol{\alpha[\beta]}$**,** that is not used as definition and both $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are variable, needs to be checked if there may be dependency into the node of $\boldsymbol{\alpha[\beta]}$**.** Then at lines 2, we checked if we can know the value of $\boldsymbol{\beta}$ must be string constants $\boldsymbol{s}$. If true, then at line 4, we compute the live definitions of $\boldsymbol{\alpha[s]}$ and we connect them to the use of $\boldsymbol{\alpha[s]}$**,** that is**,** $\boldsymbol{\alpha[\beta]}$**.** If we cannot guarantee that the value of $\boldsymbol{\beta}$ is constant, at lines 6, live definitions of $\boldsymbol{\alpha[\gamma]}$, γ is any string, need to be connected to node of $\boldsymbol{\alpha[\beta]}$.

```
1 : for each $\alpha$ [$\beta$ ] in program, both $\alpha$ and $\beta$ are
    variable, and $\alpha$ [$\beta$ ] is not used as a definition
    of an assignment statement.
2 : if $\beta$ evaluates to string constants $s$
3 : Then
4 :     make edges from nodes that define $\alpha$ [$s$],
        where, $\alpha$ [$s$] is live to a node of current
        $\alpha$ [$\beta$ ]
5 : else
6 :     make edges from nodes that define $\alpha$ [ɣ ],
        where $\alpha$ [ɣ ] is live to node of current
        $\alpha$ [$\beta$ ] and ɣ is any string
```

Figure 2 Algorithm to build data dependencies from Python's dynamic reference

If we apply the algorithm in Figure 2. to the example code in Python in Figure 1(a), we can get the PDG in Figure 1(c). The dynamic reference, **alice['str']** at line 5 in Figure 1(c) needs to be checked if the value of **'str'** must be string constant or not. Because we cannot guarantee the value of **'str'** to be string constant, live definitions of **alice['str']** are **alice['x']** at line 1 and **alice['y']** at line 2. So the new two edges, node S1 to S5 and node S2 to S5, are added to the PDG reflecting dynamic reference of Python in Figure 1(c).

4 Conclusion

This paper describes Python's dynamic reference, which makes PDG lack of data dependence incurred by Python's dynamic reference. This paper also proposes how to add data dependence edges of Python's dynamic reference to PDG.

5 Acknowledgement

This work was supported **by M2M terminal and solution development project** through **IT cooperative study research center foundation support project** (Dong-A University Media Device Lab) funded by the Korea Telecommunications Operators Association.

6 References

[1] Python, http://en.wikipedia.org/wiki/Python_(programming_language)

[2] TIOBE Programming Community index, www.tiobe.com/tiobe_index/index.htm

[3] V ítor T. Martins, et. Al., "Plagiarism Detection: A Tool Survey and Comparison", *In Proc. of 3rd Symposium on Languages, Applications and Technologies (SLATE'14)*, pp. 143–158

[4] J. Krinke, ″ Identifying similar code with program dependence graphs,″ *In Proc. of WCRE′01 IEEE*, 2001

[5] C. Liu, et al. ″GPLAG: Detection of Software Plagiarism by Program Dependence Graph Analysis″, *In Proc. of Int Conf on Knowledge discovery and data mining*, pp 872-881, 2006

[6] J. Ferrante, et.al. ″The Program Dependence Graph and its use in optimization″, *ACM Trans. on Program. Lang. and Syst.*, 9(3)319-349, 1987

Can we make and use Carbon-Copy relationship to recommend developers to fix bug?*

Jungil Kim and Eunjoo Lee [2]
School of Computer Science and Engineering, Kyungpook National University, Daegu, Korea

Abstract - *Bug triage is a task to find appropriate developers to fix a new reported bug. As it is a time consuming task to manually perform the bug triage, an automated developer recommendation is helpful to efficiently support bug triage. Previous studies mainly leverage text in bug reports such as description, summary and comments. In this paper, we leverage carbon-copy list (cc list) in bug reports. At first, CC-DSN (cc-relation among developers), which is a developer social network graph, is created with cc list. After that, we tried to recommend appropriate developers using the CC-DSN and several social network metrics like Indegree, Outdegree, Closeness, Centrality, etc. The experimental results show that our approach outperformed the performance of DREX with 3%, 9% of precision and 3%, 6% of recall respectively in JDT and Firefox.*

Keywords: Bug triage; repository mining; software maintenance; Developer recommendation; Issue tracking

1 Introduction

Large open source projects, such as Eclipse and Mozilla, mostly adopt a Bug Tracking System (BTS) like BugZilla to keep track of bug history. When a user encounters a new bug, he/she writes detail of the bug according to form of bug report and submits it to the project's BTS. A project manager then assigns the bug report to a developer who is able to handle the bug [1]. This process is often called as bug triage. It is hard to manually assign all of the submitted bug reports since at least 300 bug reports are commonly submitted in large open source project every day [2]. Therefore, it is significant to automatically recommend developers for each bug report [2]. John et al. proposed a technique for automated developer assignment using text of description and summary of bug report. Wenjin et al. proposed a bug recommendation system, DREX [3], which recommends appropriated developers with K-Nearest-Neighbor Search and expertise ranking. In DREX, the expert of developer is determined with the number of comments which each developer added during bug resolution [3]. In Bugzilla, carbon-copy list (cc list) is defined as follows.

"Users who may not have a direct role to play on this bug, but who are interested in its progress."[1]

From cc list, it is possible to infer the users who are interested in resolving the bug. Therefore, not only summary and description, but cc list may also be significant factor to determine contributors of a specific bug, in addition to description, summary and comments,.

In this paper, we construct CC-DSN graph, which presents cc relation among developers by mining cc list of bug reports and perform developer recommendation using social network metrics such as *indegree, outdegree, degree, betweenness centrality and closeness centrality*. In the experiment on open source projects, we show the effectiveness of our approach by comparing with DREX.

2 Construction of DSN using cc-relationship

A DSN usually represents communication network among developers. In DSN, a node indicates a developer and an edge refers communication relationship between nodes [4, 5]. A communication relationship can be established by exchanging comments between developers in a bug report [3]. Wenjin et al. has proposed constructing DSN based on comments remained in a bug report by developers. In [3], each edge is made from a developer, where is at the latter place in a comment sequence of a bug report, to all its previous developers in comment sequence.

In this paper, CC-DSN is proposed which represents cc relationship among developers based on cc list of a bug report. A developer can be added to cc list of a bug report if the developer is interested in processing the resolution of bug. After added to cc list, the developer will receive all of the change information until being resolved. Therefore, developers who were added to same bug report's cc list may have more knowledge about resolution of the bug than others. In the CC-DSN, nodes represent developers who have been added to cc list and edges between nodes can be established if developers have been added to same bug report's cc list. The weight of edge is degree of relationship between developers.

3 Experiments of recommending developers

The research question (RQ) in this paper is as follows.

RQ: Can we use the CC-DSN to recommend appropriate developers to fix bug?

[1] https://bugs.eclipse.org/bugs/page.cgi?id=fields.html#bug_status

[2] Corresponding author

* This study was supported by the BK21 Plus project (SW Human Resource Development Program for Supporting Smart Life) funded by the Ministry of Education, School of Computer Science and Engineering, Kyungpook National University, Korea (2012R1A1A3011005).

Table 1 The Experiment data

	Date	# Bug Report	# Fold Size
CDT	2009 - 2013	3179	317
JDT	2009 - 2013	3286	328
FireFox	2009 - 2013	9931	993
Thunderbird	2009 - 2013	2903	290

Table 2 The result of precision and recall of DREX and CC-DSN

	DREX		CC-DSN									
	SF		IND		OUTD		DGRE		BW		CN	
Proj.	P	R	P	R	P	R	P	R	P	R	P	R
CDT	57	29	46	23	44	22	46	23	43	22	33	16
JDT	54	32	56	33	56	34	58	35	59	35	21	13
FireFox	21	19	28	23	26	23	30	25	26	23	5	4
Thunderbird	41	32	34	25	39	30	39	30	37	28	10	6

(SF : Simple Frequency, IND : Indegree, OUTD : Outdegree, DGRE : Degree, BW : Betweenness Centrality, CN : Closeness Centrality)

To answer to the above RQ, we conducted an experiment on four open source projects (CDT, JDT, Firefox and Thunderbird). At first, we constructed CC-DSN for each component by mining bug reports in each project, to recommend appropriate developers. The social network metrics is selected for ranking developers. When a new bug is reported in a specific component, we find developers who have the top most value of the metrics and suggest them. The precision and recall are used to compare with DREX.

3.1 Experiment data

We collected bug reports from 2009 to 2013 in all of projects represented as Table 1. As the experiment in [3], we only considered fixed bug repots as experiment data set. The number of fixed bug reports extracted from the open source projects, CDT, JDT, Firefox and Thunderbird are 3179, 3286, 9931 and 2903, respectively.

3.2 Setup DREX

DREX employs K-NN search algorithm and ranking methods to perform developer recommendation. Before adopting DREX, *k* value and ranking method should be determined. The ranking method can be some of social network metrics or simple frequency value. At the experiment performed in [3], the best performance of DREX was shown when the value of k was 15 and the simple frequency was used. Therefore, we set the value of k to 15 and selected the simple frequency metric as ranking method for DREX.

3.3 Evaluation

We used 10-fold validation, a folding-based training and validation approach, to achieve higher prediction accuracy. All bug reports in each project were divided into 10 fold with the fold size. In i-th fold phase, fold-i is used as training set and next fold (i+1) is used as validation set. After the i-th validation performed, the validated fold is also added to train set to validate next fold data.

The performance of DREX and our approach were evaluated with following precision and recall formula.

$$\text{Recall} = \frac{|\mathrm{P_i} \cap D_i|}{|D_i|} \quad (1)$$

$$\text{Precision} = \frac{|\mathrm{P_i} \cap D_i|}{|P_i|} \quad (2)$$

Where, D_i and $\mathrm{P_i}$ mean that set of developers which contribute the bug report i to be fixed and Set of recommended developers respectively.

3.4 Results

The results of DREX and our approach are presented in Table 2. In CDT and Thunderbird, DREX outperformed our approach by 10%, 3% with the precisions and by 6%, 2% with recall respectively. However, except BW and CN, our approach outperformed DREX by 3 %, 9% with precision and by 3%, 6% with recall respectively in JDT and Firefox.

4 Conclusion and future work

According to the result of experiment in section 3.D, we should answer to RQ as follow.

“CC-DSN partially shows improved results than DREX. Therefore, we confirm that CC-DSN is also applicable to recommend suitable developers.”

We believe that the cc-relationship proposed in this paper is essential factor to improve developer recommendation system.

5 References

[1] J, Gaeul, S. Kim, and T. Zimmermann. "Improving bug triage with bug tossing graphs." In Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, 2009, pp.111-120.

[2] J Anvik, L Hiew and GC Murphy. "Who should fix this bug?."Proceedings of the 28th international conference on Software engineering, 2006, PP.361-370.

[3] W. Zhang, Y. Yang and Q. Wang, "Drex: Developer recommendation with k-nearest-neighbor search and expertise ranking." In Proceeding of 18th Asia Pacific Software Engineering Conference, 2011, pp.389-396.

[4] Q. Hong, S. Kim, S.C. Cheung and C. Bird, “Understanding a developer social network and its evolution”, In Proceeding of 27th IEEE International Conference on Software Maintenance, 2011, pp.323-332.

[5] M.S. Zanetti, I. Scholtes, C. J. Tessone, F. Schweitzer, "Categorizing bugs with social networks: A case study on four open source software communities.”, In Proceedings of the International Conference on Software IEEE Engineering, 2013, pp. 1032-1041.

SESSION

LATE BREAKING PAPERS AND POSITION PAPERS: SOFTWARE ENGINEERING RESEARCH AND PRACTICE

Chair(s)

Prof. Hamid R. Arabnia

Formal Specification-Driven Development

Richard Rutledge
College of Computing
Georgia Institute of Technology
Atlanta, GA
rrutledge@gatech.edu

Sheryl Duggins, Dan Lo, Frank Tsui
Department of CS and Software Engineering
Southern Polytechnic State University
Marietta, GA
{ sduggins, clo, ftsui }@spsu.edu

Abstract— **Since the inception of software engineering we have focused on product quality and the process that was needed to develop safety critical products. The definitions of quality and associated attributes grew as software engineering matured. One of the early "silver bullets" was the hope that incorporating formalism in the process would bring us marked quality improvement, but the drawback of extensive training and competency detracted from that approach. Nevertheless, recently we have made significant progress in this area, especially in the form of improved tools and training. This paper examines some of the recent improvements made in processes such as Test-Driven Development (TDD), Behavior-Driven Development (BDD), and associated automation tools. It proposes a related methodology, a Formal Specification-Driven Development (FSDD) process that embraces these recent improvements. The paper recognizes that in the area of knowledge transfer, humans will always make errors, however, FSDD will lessen those errors through improved unambiguity, correctness, completeness, and consistency.**

Keywords-component; formal methods; test-driven development; formal specification; behavior-driven development;

I. INTRODUCTION

In search of productivity and quality gains, some software engineers have modified the traditional software development process model. Early work stressed evolutionary, incremental improvement. Efforts such as the Software Engineering Institute (SEI) Capability Maturity Model (CMM) and Software Process Improvement and Capability Determination (SPICE/ISO 15504) encouraged increasingly heavyweight process models. In counter-point, proponents of agile software development often promote revolutionary, lightweight process models. These models may prune sub-processes and/or fundamentally reorder the remaining sub-processes. This paper will examine some of the advantages and shortcomings of one such approach, Test-Driven Development (TDD) [1] and a derivative, Behavior-Driven Development (BDD) [2]. The paper will then introduce a new artifact to the traditional process model and present an argument that this traditional-derived model delivers the advantages of TDD and BDD without their shortcomings. The new artifact is a formal design specification expressed in a behavioral specification language. Hence, the process model proposed herein is referred to as Formal Specification-Driven Development (FSDD).

The scope of this paper will be restricted to a qualitative argument presenting the advantages of a FSDD approach. It should be viewed as a proposed methodology leading to future research to establish the quantitative efficacy of FSDD. Attaining the quantitative efficacy of a process model is difficult due to a number of factors. Consider TDD: although introduced in 2003, TDD still lacks such quantifiable justification. Tsui defines a set of 5 items or criteria that should be included in a process definition: i) activities, ii) control, iii) artifacts, iv) resources and v) tools [3]. Here, we will examine the merits of FSDD mainly through the perspective of Tsui's item iii) and item v), the artifact and the tool.

Analogously to TDD/BDD, the practical application of FSDD requires considerable automation and tool support. Since TDD requires the frequent execution of all test cases by the developer, the cases must be run quickly and efficiently. Hence, the test suite must be fully automated. Similarly, FSDD requires the formal design specification to be utilized in both the implementation and test phases. Although it is possible to perform these steps manually, specification without automation would probably be impractical and would abrogate many of the advantages cited in this paper.

This paper will demonstrate why new methods of software development are needed and provide the underlying motivation. It will establish the viability of the tool and automation support presumed above. It will describe TDD and BDD and introduce FSDD and work through a process example using FSDD. Finally the paper will suggest future work in this area.

II. MOTIVATION

Computer software plays an essential and critical role in managing the infrastructure of a modern society. It is integral to the operation of nuclear power plants, household toasters, and all manner of environments in between. When software fails, airplanes do not fly and cars do not drive. Yet the challenge of building reliable software remains even as the size of software projects scales ever larger. Even the ubiquitous cell phone contains about five million lines of code [4]. Under research funded by the Department of Homeland Security, Chelf at Coverity examined 32 well established, open-source software projects encompassing 17.5 million Lines of Code (LOC) [5]. They calculated an average defect density of .434 per thousand lines of code. In addition to the risk to safety, software defects are also expensive. In 2002 the National Institute of Standards and Technology (NIST) estimated that software defects cost the US economy over $60 billion each year. Further, NIST also determined this figure could be reduced by $22 billion if these defects could have been found more efficiently.

In response to the current state of software quality, the Verified Software Initiative (VSI) was founded to "gain deep theoretical insights into the nature of correct software construction, to radically advance the power of automated tools for the construction and verification of software, and to benchmark the capabilities of such tools through convincing

Copyright © 2014 CSREA Press, ISBN: 1-60132-286-0; Printed in the United States of America

experiments [6].” The VSI is a 15-year collaborative research project with ambitious goals toward an ultimate objective of defect free software. These goals include both the establishment of the theoretical foundations of software verification and the development of a sufficient toolset to validate the approach with real-world software.

While formal specification languages, such as Z, could be employed to demonstrate program correctness, they are still unwieldy, impractical, and cost prohibitive for the vast majority of projects. In current industry practice, an implementer is given a module design specification consisting of Unified Modeling Language (UML) and natural language. The implementer then develops both the source code and the unit tests for validation. However, this process can be substantially flawed. If the implementer misinterprets the design specification, that error will be reflected in both the code and the unit tests, and is likely to remain undetected at least until module integration, if not even later. The VSI effort is one promising attempt to resolve this problem.

A formal specification language is a mathematically precise notation for stating the properties of a software component. A behavioral specification language is a type of formal language intended to express the behavior of a module. A behavioral specification language similar to Java Modeling Language (JML), Spec#, Vienna Development Method Specification Language (VDM-SL), or the Object Constraint Language (OCL) can be employed to mitigate this process flaw. With sufficient tool support, a behavioral design specification can be used for both dynamic, run-time analysis and static analysis of an implementation. Much of the current implementer written unit tests could be replaced with automated specification driven testing, combining the advantages of Bertrand Meyer’s Design by Contract [7] with Test Driven Development. Additional tools could support steps in the Software Development Life Cycle (SDLC) beyond implementation. For example, specifications could be scanned for boundary value conditions [8] to aid the quality assurance team in test case generation.

An ideal behavioral specification language would be rigorous, expressive, assessable, executable and abstract. However, note that aspects of these attributes conflict. Otherwise, designers would long since have adopted Z or a derivative as the specification language of choice. An ideal specification language must discern an appropriate blend.

III. TRADITIONAL SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

A. Description

Traditional software development process methodologies recognize a common sub-set of distinct activities. These include, among others, requirements analysis, software design, implementation, and test. Methodologies differ in how these activities are organized and performed. At the completion of each element of an activity partition, an artifact must be produced to transfer project specific knowledge to the next agent in the selected process. Thus the artifact can be seen as a project-specific bridge between knowledge domains. Both the provider and the consumer of an artifact must be able to comprehend it, and the extent to which they each understand the artifact identically has a significant effect on the fidelity of the next activity performed. Since this is an iterative cycle, such transfer comprehension errors accumulate. The nature of the artifacts and methods of knowledge transfer will be examined more closely below.

B. Issues

Formal inspections and reviews of artifacts in software engineering to detect and remove defects is a natural part of the software process for large and complex projects today. However, an error in knowledge transfer is unlikely to be detected early in the process so the knowledge transfer error may be propagated indefinitely.

The possibility of error exists whenever an artifact is transferred to another agent. In order to minimize knowledge transfer errors, artifacts should possess a subset of the characteristics Pfleeger has identified for one particular artifact, a software requirement [9]. This list includes correct, consistent, unambiguous, and complete. Additionally, these characteristics must apply equally in both the producer and consumer knowledge domains. Considering the traditional SDLC, note that agent re-interpretation of artifacts occurs throughout the model. Thus a unit test can only detect an activity error and will be oblivious to knowledge transfer errors.

IV. BEHAVIORAL/TEST DRIVEN DEVELOPMENT

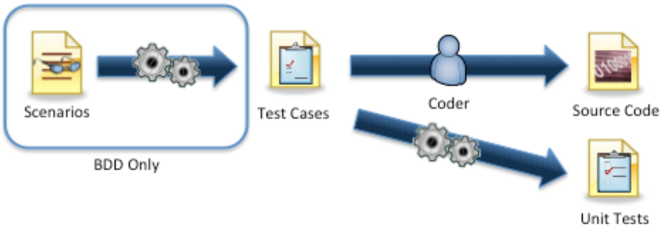


Figure 1. TDD/BDD Artifact Production

A. Description

1. Test-Driven Development (TDD)

Programmers construct unit tests to exercise the proposed unit of source code before releasing as completed functionality. In 2000, Kent Beck introduced eXtreme Programming (XP), an agile development methodology [10]. One of the key practices of XP requires the programmer to build the unit tests first, before any source code is written. Beck later generalized this practice as Test-Driven Development (TDD) [1] and summarized the practice as repeated iterations of the following:

1. Write a new test case.
2. Run all the test cases and see the new one fail.
3. Write just enough code to make the test pass.
4. Re-run the test cases and see them all pass.
5. Refactor code to remove duplication.

For practical usage, the frequent application of this procedure implies the full automation of unit test performance. Since passing all unit tests is synonymous with code completion, the collective set of unit tests **is** the design specification. As such, the unit tests serve both verification and validation. Thus TDD enhances unambiguity and completeness as compared to a traditional model.

One clear advantage to TDD is that it provides an unambiguous design specification. Thus the source code either conforms to its specification, or it does not. In 2004, Erdogmus et al conducted an empirical study of TDD (referred to as Test-

First) [11] and found a small productivity gain. This gain was attributed to better task understanding, better task focus, faster learning, and lower rework effort.

2. *Behavior-Driven Development (BDD)*

While teaching agile practices to industry, Dan North noticed a few recurrent problems with TDD [12]. "Programmers wanted to know where to start, what to test and what not to test, how much to test in one go, what to call their tests, and how to understand why a test fails". Some of these questions arose because TDD inverted their standard practice. They knew what they wanted to code, and then tested what they wrote. TDD requires them to first decide what they want to test. Other questions resulted because Beck described TDD in general terms rather than a procedure. North responded with Behavior-Driven Development (BDD). It is a process implementation of the TDD approach. Hence it possesses the TDD advantages, while addressing what North perceived as its shortcomings. In order for unit testing to drive the implementation process, the tests must constitute a behavioral specification. Supporting the terminology shift, North derived his automation test tool, JBehavior, from Java's JUnit.

North's next significant insight from the phraseology shift was that the semantics of the unit tests (behaviors) was now accessible outside the programming staff. When BDD's naming conventions are followed, a straightforward processing of the unit tests produces a set of English sentences accessible to an analyst. The final step to current BDD practice is provision for a "ubiquitous language" that can be read and written by analysts, and read by frameworks such as JBehavior. These goals are realized by projects such as rSpec and Cucumber for the Ruby language [2]. By extending applicability back into the analysis domain, BDD yields additional completeness and some consistency improvements over TDD.

B. *Issues*

Several issues with TDD/BDD can be qualitatively considered and have been quantitatively analyzed. This paper will discuss the specific first (BDD), and then proceed to the general (TDD). Since the collection of scenarios **is** the specification, the analyst is constrained to a language provided for by the developer and the analyst may only make pre-arranged lexical changes to the scenario specification. These limitations are in stark contrast to the goal of a ubiquitous language. Thus the process is exposed to the same knowledge transfer errors as before and the observed process improvement with regard to consistency is severely restricted.

Without the expansion into the analysis and test realms, BDD devolves into TDD with a specific vocabulary. BDD activities are limited to the developer who specified behaviors (unit tests) before writing any implementation code. In 2005, Erdogmus et al performed one of the first empirical studies of TDD [11]. In a study involving thirty-five third-year students, he found no impact on quality and a statistically insignificant improvement in productivity. Erdogmus explained this lack of clear results in terms of the limited scope of the projects under test and the inexperience of the participants. However, another explanation is that TDD does not address knowledge transfer errors, and thus is not a major contributor to product quality. In 2010, Kollanus reviewed forty empirical studies of TDD [13]. She summarized the results of the findings as follows:

1. Weak evidence of better external quality with TDD
2. Very little evidence of better internal quality with TDD
3. Moderate evidence of decreased productivity with TDD

Summarizing many of the concerns with scaling TDD to significant projects, John McGregor states, "Design coordinates interacting entities. Choosing test cases that will adequately fill this role is difficult though not impossible. It is true that a test case is an unambiguous requirement, but is it the correct requirement?" He goes on to suggest that unlike more robust design techniques, it is not clear how TDD ensures "the correctness, completeness, and consistency of the design [14]."

So the quantitative studies of TDD are decidedly ambivalent. And the qualitative discussion argues strongly that both TDD and BDD, when applied properly to a narrow band of projects and teams, provide quality gains in terms of unambiguity, completeness, and (partially) consistency. Note that since neither process model mitigates the potential for knowledge-transfer errors, correctness is unaffected.

V. Formal Specification-Driven Development

The objective of this paper is not merely to argue that formalism is good, but rather to demonstrate how a specific application of formalism within the software development process, together with adequate tool support, can yield the benefits of TDD and further make significant gains in both correctness and consistency attributes.

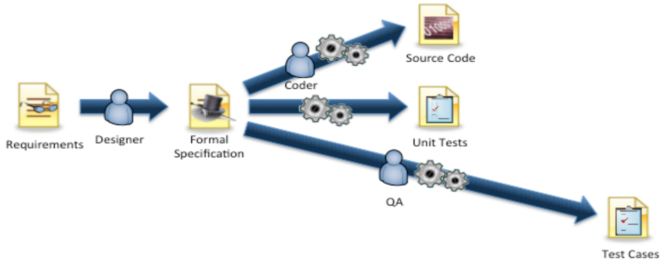


Figure 2. FSDD Artifact Production

A. *Description*

Unlike TDD and BDD, Formal Specification-Driven Development (FSDD) does not affect the underlying process model. Rather, it modifies the principle software design artifact to change the type and nature of knowledge transfer to follow. The design produces a formal specification in an organizationally selected behavioral specification language (JML, Dafny, Spec#, etc.), which becomes the artifact of record for both implementation and test agents. The introduction of a formal specification is a key step towards making significant progress improving both correctness and consistency properties. Although no additional steps have been inserted into the traditional SDLC, a new artifact has been introduced and it will likely require more rigor than prior artifacts. One may reasonably conclude that the design effort is likely to increase. However, the following sections will present a qualitative case that the additional effort would be more than offset by a reduction in re-work due to knowledge transfer errors and downstream activity efficiencies.

The artifacts produced by FSDD are illustrated in Figure 2. Arrows indicate FSDD processes and are labeled to indicate manual and automated steps. Arrows also indicate the artifact flow for input and output of each process. Automation support for artifact production reduces the risk of introducing knowledge transfer errors. The production of a formal design specification allows tool chain support in the development of source code, unit tests, and test cases.

B. Tool Requirements

Tool support is a critical component of FSDD. Even though FSDD can be manually implemented as an intellectual exercise, employing FSDD without automation is impractical at best. The following tools are anticipated in order to apply FSDD in non-trivial circumstances.

1. Design Specifiation Analyzer

The designer uses the design specification analyzer to verify the consistency and completeness of a formal design specification. Referenced entities must be defined and consistent in the specification language typing system. Although this tool does not directly reduce knowledge transfer errors, it does support quality assurance of the design phase artifacts.

2. Implementation-Stub Generator

A subset of the behavioral specification will include an interface specification. This includes such information as classes, methods, pure functions, arguments, and return types. This tool realizes behavior-less stubs in the implementation language of these entities and maps specification fundamental types (Integer) into implementation types (int). The intent of the implementation-stub generator is not behavioral code generation. It saves the coder considerable typing, and more importantly, insures a minimal, 'lexical-level' compliance with the specification. The specification must include all public access to a class, although the coder is free to elaborate private entities.

3. Specification-Aware Compiler

An FSDD-enabled compiler is expected to accept the specification as input in addition to programmer-generated source code. The compiler ensures that the source code is 'lexically' compliant to the specification as generated by the stub generator. Non-compliance will be flagged as an error. The addition of public entities not contained in the specification will also be flagged as an error. The source code will be statically analyzed and predicates that can be proven will be discharged. Unproven predicates will be injected as program logic in the target program to be validated dynamically during program execution.

4. Unit-Test Framework

The unit-test framework will generate automated unit tests from the formal specification. This tool provides the 'driven' in FSDD. The goal of the programmer is to elaborate the generated stubs with source code to create the behavior that successfully completes the unit tests and thus is compliant with the formal specification. The combination of tools and activities works together to enhance correctness.

5. Test Case Analyzer

The test case analysis tool is an aide to Quality Assurance (QA) engineers. It scans the formal specification and reports detected boundary values for subsequent use in acceptance testing against the requirements specification.

C. Issues

1. Formal Specifications Are Hard To Write

FSDD requires the software designer to learn to read and write a new, more abstract language. Additionally, designers with an implementation background will have to adopt a new approach. They must learn to think in terms of 'what' an entity does, rather than 'how' an entity does it. The requirement for this new skill is restricted to software designers. No other labor category within the process must be able to write a formal specification and designers should account for a small percentage of the complete development team.

2. Formal Specifications Are Hard To Read

Formal specification is a challenging subject. Many (perhaps most) programmers lack the training and experience to read and comprehend a formal design specification. In order to evaluate the worst-case ramifications, consider the case in which the entire coding team is unable to read the formal design specification. The specification is a sealed, black box artifact that is only comprehended by the support tools. The coding team can still use the implementation-stub generator to create the initial source files. They would still use designer-specified unit tests to validate their work. But, they would have to rely on the design support artifacts such as UML diagrams and narrative description to convey behavior. This situation does admit the potential for a knowledge transfer error. But the error is contained since coders do not write the unit tests. In those instances in which code completed behavior does not pass unit testing, the coder must coordinate with the designer to resolve the issue.

Although the above scenario is less than ideal due to the additional coordination, the outcome is still superior to the non-FSDD outcome. Without the formal specification, the knowledge transfer error is likely to remain undetected until this particular coding element is integrated with the rest of the developing system. Ideally, the coders can read and comprehend the specification. Then they need coordinate with the designer only when the specification or design documentation is in error.

3. FSDD, Validation, and Non-Functional Requirements

Formal methods and automation can assist primarily with verification. Is the artifact built the right way? Validation is quite a bit more difficult. Is this the right artifact? Answering these questions generally requires more than formal logic. Specifically, they require a qualitative judgment from a knowledgeable subject matter expert. Also, verification of non-functional requirements is challenged by the initial selection of specification language. Naturally, a behavioral specification language is specifically designed to express **behavior**, not performance characteristics. Behavioral specifications are selected as the initial focus of FSDD due to their scope at addressing knowledge transfer issues. But FSDD is not behavioral specific. The specification language can be augmented with additional logics such as temporal logic to specify performance, which is beyond the scope of this paper.

D. Advantages

The utilization of FSDD with full tool support provides numerous advantages to the development process. Similar to TDD, the goal of the programmer is to satisfy the automated tests. However, the tests are not generated by the programmer, but rather are generated by automation from the specification. The programmer continues to add implementation until full compliance is achieved as validated by both static and dynamic analysis. The module/class is then ready for integration.

1. Static Analysis: Implementation

Using the techniques reviewed above, a specification aware compiler will warn the programmer when the implementation does not satisfy its specification.

The compiler will first statically analyze the implementation for compliancy to its specification with three possible outcomes: proven non-compliant, indeterminate, and proven compliant. If non-compliant, the compiler would emit a warning message. After the programmer corrects the mistake, the compiler may be able to prove the new implementation is correct. In this case, unit testing is not required.

2. Specification Derived Unit Tests

The creation of unit tests from the specification is straightforward. The pre-conditions specify the input domain of the specification element. Analysis can readily provide test case data for input partitioning and boundary testing [8]. With test cases selected, unit testing consists of evoking the element with each case and checking the post-conditions and invariants. Any failure constitutes a failure of the unit test.

3. Static Analysis: Test

The test case analyzer can also provide material benefit to functional testing in the Quality Assurance (QA) phase of software development. With a formal behavioral specification, tooling can extract key boundary values to use in functional, black box testing without requiring input domain partitioning and boundary testing [8].

4. Dynamic Analysis: Test

After development of the test cases, integration and system testing will then be performed with instrumented builds of the implementation. Remember that an instrumented build is one in which the unproven pre-conditions, post-conditions, and invariants are asserted (validated dynamically). This approach will detect failures that might otherwise go undetected, and establish responsibility down to the lowest specified element. Kosmatov noted that software testing comprises about 50% of the total cost of software development [15]. In FSDD, the model is provided to the QA engineer without additional effort.

5. Resource Optimization (Personnel)

TDD presumes a uniformly high level of capability amongst the design team [16]. This follows from the assumption that programmers develop their own tests before coding any implementation. Since the test is the specification, each programmer shares equal responsibility for design. This process does not scale beyond small (1 – 5) developer teams for two principle reasons. First is the difficulty with attracting and keeping a large group of top-tier software developers. The second reason is fiscal justification. TDD requires uniform capability amongst the development team and would require more highly skilled software engineers than would FSDD, which does not require uniformity and shifts complex tasks onto a smaller design team resulting in reduced costs.

6. Specification Compliance Verification

During the development process, a significant percentage of total effort is expended on verification activities. According to Tian, "Software verification activities check the conformance of a software system to its specifications [8]." FSDD enforces conformity to the specification throughout the code development phase and provides designers assurance that the produced source code adheres to their intent. However, the FSDD automation cannot replace all review activities. Manual checks of new or modified code are still required to verify compliance to organizational standards.

E. Summary of Impact on Quality Attributes

Table 1 provides a summary of the results of the qualitative analysis and discussion of the process methodologies with regards to the attributes of unambiguity, completeness, consistency and correctness.

VI. PROCESS EXAMPLE

This section will provide a simplified hypothetical walkthrough of an application of Formal Specification-Driven Design (FSDD). The example application will be a simple element of a banking application, the realization of a bank account. The walkthrough will begin with its specification, continue through its implementation, and conclude with some aspects of functional testing.

Table 1: Attribute Impact Summary

Methodology	**Impact**			
	Unambig	***Complete***	***Consistent***	***Correct***
Traditional	Baseline			
TDD	Improved	Improved	Minor	Improved
BDD	Improved	Improved	Minor	Improved
FSDD	Improved	Improved	Improved	Improved

After reviewing the Software Requirements Specification (SRS), the designer completes a design specification in an organizationally appropriate specification language. The language used in this walkthrough does not represent any specific language, but was chosen for maximum simplicity and comprehension. The specification for this design is provided below.

```
class BankAcount
{ // Model Variables
  constant Decimal MAX_BALANCE = 999999999.99;
  constant Decimal MAX_TRANSACT = 99999999.99;

  // Class Predicates
  pre-condition: GetBalance() == 0;
  pre-condition: IsLocked() == false;
  post-condition: GetBalance() == 0;
  post-condition: IsLocked() == false;
  invariant: GetBalance() in [0 .. MAX_BALANCE]; …}
```

The above specification fragment declares one model variable. It exists only in the model and is used to specify behavior. The **constant** keyword declares the model variable to be un-modifiable. **Decimal** and **Boolean** are specification language fundamental data types. For each implementation language, these must be mapped to implementation data types. In this example, classes can have three differing predicates: pre-conditions, post-conditions, and invariants. The pre-condition for a class must evaluate to true at the conclusion of object construction. The post-condition for a class must evaluate to true before object destruction. The invariant for a class must evaluate to true after class construction, before and after any specification methods, and before object destruction. Class method specifications continue below.

```
// Class Methods
void Credit(Decimal[0 .. MAX_TRANSACT] amount)
{ pre-condition: not IsLocked();
  post-condition: post GetBalance() == pre GetBalance()
  + amount;}
```

```
void Debit(Decimal[0 .. MAX_TRANSACT] amount)
{ pre-condition: not IsLocked();
  post-condition: post GetBalance() == pre GetBalance()
  - amount;}

void Lock()
{ pre-condition: not IsLocked();
  post-condition: IsLocked();}

void Unlock()
{ pre-condition: IsLocked();
  post-condition: NOT IsLocked();}

pure Boolean IsLocked();
pure Decimal GetBalance();
```

Each of these methods must exist in a compliant implementation. Additionally, the two **pure** methods, IsLocked() and GetBalance(), must have no side effects. This enhances the ability of the static analyzer to reason about them. After examination with the design specification analyzer, this design specification is committed to source version control. At some later point, the complete (or partially-complete) design is delivered for coding. In this example, the system is to be built in C++. The programmer assigned to implement the BankAccount class begins with the stub generator, which creates two files, BankAccount.h and empty methods with signatures in BankAccount.cpp:

BankAccount.h:

```
class BankAccount
{public:
 const BCD_Type MAX_BALANCE(999999999.99);
 const BCD_Type MAX_TRANSACT(99999999.99);
 void Credit(BCD_Type amount);
 void Debit(BCD_Type amount);
 void Lock();
 void Unlock();
 bool IsLocked() const;
 BCD_Type GetBalance() const;};
```

If coders were to modify any method names, parameters, or attributes they will receive an error message from the compiler, because the source is no longer compliant with its specification. The coder compiles the source file and is rewarded with compiler error messages. The formal methods have not come into play yet. Any traditional C++ compiler would fault the same conditions.

The programmer would then make the modifications to fix the compiler errors to include missing return types. At this point, the FSDD-enabled compiler is able to improve the standard process. Since the compiler has access to the expected behavior, compiling this code would now yield a series of "missing functional implementation" warning messages for each of the empty methods.

The workflow proceeds much as it would under TDD/BDD. The object is to make the errors/warnings go away. So they start to implement the missing behavior. They add behavior, get compiler messages, each time fleshing out more of the code. The static analysis phase of the compiler knows from class pre-conditions that a specific method should return a value after construction. But, if it is missing a functional implementation, the compiler is able to reason that the specified behavior is unlikely to be implemented. Similarly, post-conditions are analyzed and may be shown not to hold in all execution paths. The coder then corrects the errors and continues the implementation until the source code compiles without errors or warnings.

The coder is now ready for unit testing. The coder uses the unit-testing framework to generate the test harness and test cases. The framework examines the specification parameters, pre-conditions and invariants to partition input/class state for selection of test cases. For example, Credit() takes an input Decimal type ranging from 0 to 99999999.99. One reasonable automated partitioning might be minimum, minimum + 1, average, maximum – 1, and maximum or {0, 1, 49999999.99, 99999998.99, 99999999.99}. The framework's test harness would invoke Credit() with each of these parameters and verify post-conditions and invariants. The process is repeated for every specified entity in the source file. If the unit testing reports an error in any method, the coder will again correct the code until the implementation of BankAccount compiles without warnings or errors and all unit tests pass. This signals task completion. Both the coder and the designer have initial confidence that the implementation complies with the design specification. A senior programmer will still need to perform a code review to validate that the implementation uses appropriate data structures, employs suitable algorithms, and adheres to coding standards.

The BankAccount module is now ready for integration by Quality Assurance (QA) engineers. They use the test case analyzer to reveal input partitioning values much as with unit testing above. It aids them in the creation of the integration test cases, which are then run against an instrumented build of the system to dynamically verify specification predicates. A manually created rendition of an instrumented BankAccount follows. Instrumentation code has been highlighted in yellow. If a predicate within an assert() evaluates as false, then a runtime exception is thrown which halts the application and displays identifying information. The instrumented build would be used for integration and system testing. If the performance of the instrumented system was acceptable for deployment, then it could also be used in acceptance testing and delivered as the final product. Otherwise, instrumentation would have to be removed prior to performance and acceptance testing. Also note that static analysis could absolve the need for some of the instrumentation. If an invariant or method post-condition could be proven to hold, then those predicates would not need to be asserted. Similarly, if a method pre-condition could be proven to hold at all call-sites, then its assertion could also be removed.

```
BankAccount::BankAccount()
{ balance = BCD_Type(0);
  locked = false;
  // class pre-conditions
  assert(GetBalance() == 0);
  assert(IsLocked() == false);
  // pre-condition 1 => invariant}

BankAccount::~BankAccount()
{ // post-condition 1 => invariant
  // class post-conditions
  assert(GetBalance() == 0);
  assert(IsLocked() == false);}

void BankAccount::Credit(BCD_Type amount)
{ // pre-conditions
  assert(amount >= 0) && (amount <= MAX_TRANSACT));
  assert(IsLocked() == false);
  // invariant
  assert((GetBalance() >= 0) &&
      (GetBalance() <= MAX_BALANCE));
  // post-conditions require two-state
  BCD_Type preBalance = GetBalance();
  balance.Add(amount);
  // invariant
  assert((GetBalance() >= 0) &&
     (GetBalance() <= MAX_BALANCE));
  // post-conditions
  assert(GetBalance() == preBalance + amount);}

void BankAccount::Debit(BCD_Type amount)
{ // pre-conditions
  assert(amount >= 0) && (amount <= MAX_TRANSACT));
  assert(IsLocked() == false);
  // invariant
  assert((GetBalance() >= 0) &&
     (GetBalance() <= MAX_BALANCE));
  // post-conditions require two-state
  BCD_Type preBalance = GetBalance();
  balance.Subtract(amount);
  // invariant
  assert((GetBalance() >= 0) &&
     (GetBalance() <= MAX_BALANCE));
  // post-conditions
  assert(GetBalance() == preBalance - amount);}

void BankAccount::Lock()
{ // pre-conditions
  assert(IsLocked() == false);
  // invariant
  assert((GetBalance() >= 0) &&
     (GetBalance() <= MAX_BALANCE));
  locked = true;
  // invariant
  assert((GetBalance() >= 0) &&
     (GetBalance() <= MAX_BALANCE));
  // post-conditions
  assert(IsLocked() == true);}

void BankAccount::Unlock()
{ // pre-conditions
  assert(IsLocked() == true);
  // invariant
  assert((GetBalance() >= 0) &&
     (GetBalance() <= MAX_BALANCE));
  locked = false;
  // invariant
  assert((GetBalance() >= 0) &&
     (GetBalance() <= MAX_BALANCE));
  // post-conditions
  assert(IsLocked() == false);}

bool BankAccount::IsLocked() const
{ // invariant
  assert((GetBalance() >= 0) &&
     (GetBalance() <= MAX_BALANCE));
  return locked;
  // invariant: not required on exit from pure method}

BCD_Type BankAccount::GetBalance() const
{ // invariant
  assert((GetBalance() >= 0) &&
     (GetBalance() <= MAX_BALANCE) );
  return balance;
  // invariant: not required on exit from pure method
```

VI. Conclusions and Future Work

As consumers, we routinely and knowingly purchase defective software. We proceed to install it on our computers and entrust our sensitive and valuable data to it. We don't really trust the system manipulating our data, so we back it up. Sometimes, we also don't trust the backup system and back it up as well. When colleagues lose important work due to a software application freezing or becoming unresponsive, we advise them to save more often. Not only do we accept that software is defective, we expect it. As users, we demand new features when the existing ones only mostly work. Corporations staff entire departments to be on call and assist employees when software fails. If architects designed with equivalent defect densities, buildings would only mostly stay up, people would only enter when necessary, would never wander far from an exit, and would keep emergency services on speed dial while inside.

Reliably producing quality software requires discipline and rigor. And verification technologies are not a panacea for all software development challenges. But formal specifications and FSDD mitigate many of the current difficulties. Future work in this area extends from basic research to tool implementation. In order to gain industry traction, a single, general-purpose behavioral specification language needs to emerge with the needed language features. There is still much work to be done in static analysis. Also, the efficiency of current static analysis techniques needs to improve sufficiently to allow completion during an ordinary compilation operation. Current strategies require much more time than compilation itself. Existing tools must also be adapted to fit the descriptions outlined in the previous section. In order to take advantage of modern, multi-core systems, behavioral specification languages must be extended to allow reasoning about concurrency. Finally, in order to spur widespread adoption by the software industry, objective, empirical, quantitative studies of FSDD and formal specification in general must be conducted to establish its business value.

Rererences

[1] K. Beck, *Test-Driven Development: By Example*. Boston: Addison-Wesley, 2003.

[2] M. Wynne, *The Cucumber Book: Behaviour-Driven Development for Testers and Developers*. Dallas, Tex: Pragmatic Bookshelf, 2012.

[3] F. Tsui, "Process: Definition and Communication.," in *Encyclopedia of Software Engineering*, 2010, pp. 715–728.

[4] "The Verified Software Initiative," 2008. [Online]. Available: http://qpq.csl.sri.com/vsr/vsi.pdf/view. [Accessed: 14-Feb-2013].

[5] Chelf, Ben, "Measuring Software Quality: A Study of Open Source Software," Coverity, Inc., 2006.

[6] C. A. R. Hoare, J. Misra, G. T. Leavens, and N. Shankar, "The Verified Software Initiative: A Manifesto," *ACM Comput Surv*, vol. 41, no. 4, pp. 22:1–22:8, Oct. 2009.

[7] B. Meyer, *Object-Oriented Software Construction (2nd Ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1997.

[8] J. Tian, *Software Quality Engineering: Testing, Quality Assurance, and Quantifiable Improvement*. Hoboken, N.J: Wiley, 2005.

[9] S. L. Pfleeger, *Software Engineering: Theory and Practice*, 4th ed. Upper Saddle River [N.J.]: Prentice Hall, 2010.

[10] K. Beck, *Extreme Programming Explained: Embrace Change*. Reading, MA: Addison-Wesley, 2000.

[11] H. Erdogmus, M. Morisio, and M. Torchiano, "On the Effectiveness of the Test-First Approach to Programming," *Softw. Eng. IEEE Trans. On*, vol. 31, no. 3, pp. 226–237, 2005.

[12] "Introducing BDD," *Dan North & Associates*. .

[13] S. Kollanus, "Test-Driven Development - Still a Promising Approach?," in *Quality of Information and Communications Technology (QUATIC), 2010 Seventh International Conference on the*, 2010, pp. 403–408.

[14] S. Fraser, D. Astels, K. Beck, B. Boehm, J. McGregor, J. Newkirk, and C. Poole, "Discipline and Practices of TDD: (Test Driven Development)," in *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, New York, NY, USA, 2003, pp. 268–270.

[15] N. Kosmatov, "Constraint-Based Techniques for Software Testing," in *Artificial Intelligence Applications for Improved Software Engineering Development*, F. Meziane and S. Vadera, Eds. IGI Global, 2009.

[16] Boehm and R. Turner, *Balancing Agility and Discipline: A Guide for the Perplexed*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003.

Automatic High Performance Structural Optimisation for Agent-based Models

A.V. Husselmann[1] **and K.A. Hawick**[2]
[1]Computer Science, Massey University, North Shore 102-904, Auckland, New Zealand
[2]Computer Science, University of Hull, Robert Blackburn, Hull, HU6 7RX, United Kingdom
email: [1]a.v.husselmann@massey.ac.nz, [2]k.a.hawick@hull.ac.uk
Tel: +64 9 414 0800 Fax: +64 9 441 8181

Abstract—*The problem of structural optimisation for agent-based models is one which holds great promise. Being able to optimise a set of behaviours has potential to improve productivity greatly, and at the very least, generate inspiration. This problem consists of three smaller problems which must be mitigated: ease of use, performance, and also the use of combinatorial optimisation.*

In this article, these three problems are managed by introducing a domain-specific language (DSL) operating over graphical processing units (and also single-threading) for performance, and a suitable optimiser for this architecture. We carry out a number of experiments to demonstrate and evaluate the performance and effectiveness of this approach. We conclude that such a methodology is indeed useful and performs adequately but is currently limited by the lack of debugging support and visual programming tools.

Keywords: CUDA, parallel, optimisation, domain-specific languages, agent-based models, karva.

1. Introduction

Agent-based Modelling (ABM) is an elegant and interdisciplinary simulation methodology. It is particularly applicable in areas where a system is comprised of individual components which are *situated* in some manner, *communicate or interact* in a restricted manner, and have some level of *autonomy* [1]. The practice of ABM has already reached into several diverse disciplines and studies including cancer immunology [2], social science [3], synthetic biology [4], land change modelling [5] and even criminology [6].

There is a lack of agreement over the precise definitions of ABM, but as noted by Macal and North, the set of the most common attributes of agent-based models in the literature are [1]:

1) Identity - Agents must be identifiable, discrete individuals.
2) Situation - Agents are situated in some fashion.
3) Goal-oriented - Agents have goals to achieve.
4) Autonomy - Agents are autonomous and may operate independently.
5) Learning - Agents could potentially learn and adapt.

A well-known agent-based model is the Predator-prey model [7], [8]. In essence, this model emulates a simple "food chain" wherein individual predators pursue prey. Prey attempt to flee, but if predators surround the prey, it is eaten. Should a predator not catch prey, it dies of starvation. Both prey and predators breed. This causes an interesting effect on population numbers. In general, this model is useful for ecological inquiry, and also gives a useful benchmarking tool for machine learning algorithms such as Genetic Programming [9]. Despite its simplicity, interesting discoveries have been made with this model. Jim and Giles showed that if predators communicate with simple evolved languages, they are able to catch prey more easily [10].

Such models normally have parameters that require calibration for the purpose of model validation and verification. Instead of laborious trial-and-error tuning, the calibration process is typically reinterpreted as an optimisation problem [11], [12], [13]. Several attempts have been documented in the past, mostly involving genetic algorithms [12], [14], and other evolutionary algorithms [13]. A limitation of such systems is that they only make provision for automatically calibrating *scalar* parameters.

These efforts can be very useful, but cannot extensively modify the structure of the model itself without considerable difficulty in representation. Should a model be structurally suboptimal, it may well be that a properly configured parameter optimisation effort could be entirely fruitless. The work of Epstein in 1999 perhaps earmarked evolutionary algorithms (EAs) for structural optimisation with agent-based models [15]. Epstein noted that, in comparison, parameter optimisation deals with a much smaller search space than that of the search for a set of behaviours, or rules.

Like parameter optimisation, there have been a number of attempts to optimise structure in agent-based models. In 2012, van Berkel proposed the use of Grammatical Evolution (GE) [16] for the purpose of generating NetLogo [17] programs based on a set of predetermined "building blocks" [18], [19]. EAs such as GE do require a cost function to indicate the relative fitness difference between individual candidates. In his work, van Berkel conceded issues regarding the process of selecting these functions. Earlier, in 2010, Learning classifier systems, Q-learning and neural networks were investigated for the same purpose [20]. Among these algorithms, none were clearly better than the others. Another attempt involved the Ant Hill problem, and was done by the evolution of finite state machines using EAs [21].

It is not generally easy to write an agent-based model in C++ and implement an evolutionary algorithm optimiser on top of it. Therefore, these previous attempts suggest that some method of improving usability is very important. Moreover, this would allow experts in agent-based modelling in other disciplines unrelated to computer science to immediately harness this technology.

In terms of performance, meta-optimisation on the particle swarm optimiser (PSO) [22] gives clear indications of what to expect when optimising any aspects of an agent-based model. At its heart, the PSO is a modified flocking model [23], [24], as it is inspired from such behaviour as seen in simulations such as Boids [25]. Van Berkel's work was distributed on a set of processors, however, experiments were deemed to still be too expensive to compute (3 or more hours) [18]. Privošnik approached the problem by having a heuristic to reduce fitness evaluations [26]. Coupled with a growing interest in larger-scale agent-based models, and recent breakthroughs in general purpose graphics processing unit usage (GPGPU), parallel ABMs has emerged [27], which is capable of mitigating performance issues.

Here, emphasis is given to performance improvements using Graphical Processing Units (GPUs) for the purpose of parallelisation. In our previous works, we have investigated the use of GPUs in the context of a domain-specific language (DSL) [28], [29], [30]. DSLs are special languages developed for the purpose of servicing a very specific problem domain [31]. The main purpose in doing so was to ensure that optimisation tasks could be specified within the same language that the model is written, and ensuring that it is easy to do so. It has been proposed that DSLs be known by having a well defined domain, a clear notation; the semantics of which is to be formally and informally equivalent [31].

We have previously developed a DSL named SOL, which is primarily intended for lattice-oriented ABM [28], [29], [30]. It includes two optimisation algorithms, one is based on Karva [32], Genetic Programming [33], and Compute Unified Device Architecture (CUDA) [34]. The other is a very simple tree-based evolutionary algorithm.

The language itself is built on Terra [35], which is a very recent multi-stage programming language [36]. It makes use of LLVM [37], which is a very mature compiler architecture. Part of the motivation behind this choice was the release of NVidia's parallel thread execution (PTX) backend compiler for LLVM (NVPTX) [38]. In this article, we aim to introduce the modifications necessary to the SOL language and demonstrate its performance characteristics. We also demonstrate the use of the SOL language to solve a well known problem used in Genetic Programming.

In Section 2 the use of optimisation in the SOL language is described. SOL is extended in Section 3 with GPGPU, and some experiments are then carried out and reported on in Section 4. Finally, we draw some conclusions in Section 5.

2. SOL Model Optimisation

In the optimisation methodology described here, instead of a modeller providing fine-grained and carefully articulated local behaviours in a custom optimiser, the modeller instead provides an objective function and some possible behaviour *within* simulation code. The difficulty in both these tasks are comparable: small variations in local behaviours as well as different choices of objective function can both lead to radically different results. However, the novelty in this approach to the problem lies in the elegance with which a model can be prototyped with all aspects including optimisation encapsulated within it, without suffering performance penalties.

In cases where an objective function is more natural to use in modelling, it can be a great advantage to have the behaviour generated from it, or at least provide a suggestion which can be used as the basis for manual experimentation. By allowing a modeller to express most of a simulation with reasonable certainty, and other parts with annotated uncertainty and clear objectives, an underlying optimiser can usefully generate some behaviours.

One of the factors which sets this new language apart from other model induction attempts is that the syntax used for optimisation is itself a constraint upon the search space. Techniques such as Genetic Programming have been used in the past for evolving entire models [19] which were provided with user-defined actions and perceptions of agents. In the case of SOL, only relevant actions and perceptions are provided, but they are given in terms of a *potential solution* given by the user. This kind of syntax serves to (1) provide a good starting solution and more importantly (2) limit the immense search space. It is reasonable to assume that the dynamics of a system is at least partially known by the expert, and therefore, it is worth optimising only the *uncertain* portions.

Given a method for expressing the objective function and for measuring it, some method of configuring the output desired from the optimiser is required. Certain constraints apart from search space define, in large part, the operation of the optimiser, or even what kind of optimiser is used. Three such configurations are implemented by specifying one of the keywords recombination, single, or permutation. These qualifiers tell the compiler which kind of optimisation is required. What is referred to by these qualifiers is some kind of reorganisation of *statements* marked for optimisation. While single and permutation refer to the selection of a single statement only, or a permutation of statements (with replacement), recombination refers to a *disassembly* of the provided code into terminals and nonterminals for recombination. The recombination qualifier is given the most attention in this article, since the other two qualifiers have been previously described [28], [30].

A complete example of this is given in Listing 1. Lines 34–56 contain what will be referred to as an uncertain construct. This term is purely related to the structure of the model, and not of code semantics. The quantity is count_sheep() and the objective is to minimise this (hence the minimise keyword given on line 34).

In cases where the fitness of a model depends partly on stochastic behaviours, it is important to obtain statistically significant scores. This is typically done by averaging multiple evaluations of the objective function. This has certain implications on performance, which is a major factor in our decision to use parallel computing on GPUs. Though these devices are less expensive than grid computers, effort is required to ensure that a program maximises their computing power.

In this process, a user first provides a program with an uncertain code segment within a model. Upon execution of the host program which is written in C++, the Terra runtime parses the provided code immediately, transforming it into a type-checked abstract syntax tree (AST). This tree is what is modified by the optimiser in a later stage. The host Lua script then duplicates this tree several times until a population of N trees is made. At this point, the optimiser searches for an optimisation statement, of which there can be only one within a program. The optimiser then initialises each program's code segment with randomly chosen statements taken from within the provided construct. Depending on the type of construct, there will be one chosen statement (single), or a combination of statements (permutation) with replacement, or a complete recombination of code (recombination).

In summary, the overall process is detailed in Algorithm 1. This algorithm contains all the components generally expected within an EA, and more specifically one which uses genetic operators. The termination criteria shown is simply a maximum number of generations. Computing a new generation of programs is done by the process shown in Algorithm 2.

3. Parallel SOL

Previously, a lattice along with a temporary write-only lattice was allocated on the host. Should CUDA be enabled in a SOL model, the data is instead allocated on the GPU hardware by using the CUDA API [34] at runtime. These device pointers are provided to the Terra compiled function, which SOL is compiled to in turn, instead of pointers to host memory. The compiled SOL code is therefore able to operate on the lattice, as allocated by the host on the GPU hardware. The code parser and type checker are identical, but a separate CUDA code generator is used in order to accommodate the restrictions imposed by the CUDA GPU architecture. Compiled code is then mostly PTX instructions,

```
 1  sol
 2    defvar count = 1
 3    defvar pred = 1
 4
 5    query neighbours6
 6      if  neighbour == 1 then
 7        count = count + 1
 8      else
 9        if  neighbour == 2 then
10          pred = pred + 1
11        end
12      end
13    done
14
15    defvar predator = 2
16    defvar prey = 1
17
18    if  me == predator then
19      defvar temp = get_closest_prey
20      move towards temp
21      if  pred == 6 then  die   end
22      split
23    end
24
25    defvar eq = count_sheep()
26
27    if  me == prey then
28      defvar closepred = get_closest_predator()
29      if  (distance to  (closepred)) < 2
30          then
31        die
32      else
33        defvar closeprey = get_closest_prey()
34        select  permutation to  minimise(eq)
35          -- flee predator (F)
36          if  (distance  to  (closepred)) < 3
37                    then
38            move awayfrom closepred
39          end
40          -- breed (B)
41          if  (distance  to(closeprey)) < 2
42                    then
43            split
44          end
45          -- overcrowding
46          if  (count > 7)  then
47            die
48          end
49          -- move randomly
50          move random 4
51          -- seek mate (M)
52          if  (distance  to(closeprey))  >= 2
53            then
54            move towards closeprey
55          end
56        end
57      end
58    end
59  end
```

Listing 1: A program written in the SOL DSL for the Predator-Prey model, containing an uncertain construct (lines 42–64).

Algorithm 1 The complete simulation process eliminating uncertain constructs.

```
Terra custom parser reads agent description
Read user-provided parameters
Allocate & initialise space for $n$ candidates
{small variations when no uncertainty is present}
Compute and compile a new generation of candidates
Zero all scores
while Termination criteria not met do
  for $x$ frames do
    Execute model programs
    Collect new scores into running totals by model
    Visualise the result
  end for
  Compute a new generation using collected scores
end while
Output best candidate in final population.
```

Algorithm 2 The process of generating a new generation of candidate models for evaluation.

```
Collect scores
for $x$ candidate models do
  Optimiser performs evolutionary operators
  Pass modified typed tree through code generator
  Wrap generated function code with arguments
  Emit wrapped code
end for
Overall generated code is compiled to machine code
Fn pointer to compiled function passed to C++ via Lua
```

wrapped with the necessary host code to launch CUDA kernels with the correct thread grid and block dimensions. Once a timestep is computed, the data is copied back from the GPU to the host and then passed to the visualiser.

Compiling Terra code for CUDA is straightforward, provided that the boundaries of the device in terms of memory and thread resources are respected. The usual Terra code generated is essentially compiled into a single CUDA kernel, which is launched with a grid and block configuration, and its arguments, by a separate host Terra function. Given that an appropriate grid and block must be provided, this presents an opportunity to discuss different parallelisation techniques.

Three parallelisation strategies are implemented from which the user may freely choose. The first is a simple "one-thread, one-model" (1T1M) strategy, where a single CUDA thread is assigned a candidate model. This CUDA thread is then responsible for executing the entire model simulation once per time step. This is unsuitable most of the time, especially when one candidate model operates on a larger lattice, or the model is demanding of processing time required. The second strategy is named "one-block, one-model" (1B1M), in which an entire CUDA block is dedicated to computing a single model simulation once per timestep. While this may seem the obvious choice in nearly all circumstances, the limitations of block sizes (1024 threads maximum at the time of writing), mean that the lattice sizes have a limit. A great many candidate model simulations can be executed concurrently at reasonable speeds using this, but the limitation in lattice size is a considerable issue. The third strategy is termed "many-blocks, one-model" (*B1M), where multiple blocks are assigned to a single candidate model. This allows much larger model sizes, but race conditions become more difficult to eliminate, which require further strategies.

Another considerable issue is the source of random deviates on the GPU hardware. There are various methods to accomplish this [39]. In this case, this was implemented by maintaining a separate GPU array with three unsigned long integers for every candidate model. These integers represent the u, v and w parameters of the Ran random number generator [40]. They are initialised by the host before being copied to the device. The host computes a new Ran state for each candidate model using a master Ran for providing a seed. This allows the SOL code to use as many random deviates as it needs to operate, since Ran also provides a colossal period, approximately $3.138(10^{57})$ [40]. To generate a random number, code is automatically generated from a macro function to update the Ran state and compute a random deviate of the specific thread.

4. Selected Results

Two experiments are conducted to evaluate convergence and computing performance. The first is The Santa Fe Ant Trail problem which Koza solved using Genetic Programming [41]. This problem was approached using a select recombination structure, in order to evolve the appropriate decision tree (expression tree) for an ant to collect all food placed on an irregular trail. The second experiment pertains to a more classical model: the Predator-prey model [42], [43]. The chosen objective in the Predator-prey model is to evolve a list of ordered rules which are most suitable for the predators to catch the prey.

The parameters used for these experiments are shown in Table 1.

4.1 Santa Fe Ant Trail

The fitness plot for this experiment is shown in Figure 1. There is a gradual (though small) decrease in mean fitness up to generation 120, and followed later by a slight drop around generation 280. The minimum fitness does decrease over time, indicating progress in the search. Though, during this run, the optimum fitness (zero) was not achieved, a number of semi-suitable programs were generated. It is worth noting that no generated program can achieve maximum fitness by brute force iteration of the entire lattice due to the 400 time step maximum [41].

Santa Fe Parameter	Value
Population Size	500
P(mutate)	0.2
P(crossover)	0.8
Program head length	12
Timesteps per Generation	400
Predator-prey Parameter	**Value**
Population Size	500
P(mutate)	0.2
P(crossover)	0.8
Timesteps per Generation	200
Generation Repeats	3

Table 1: Optimisation parameters used for the Santa Fe Ant Trail problem and the Predator-prey model.

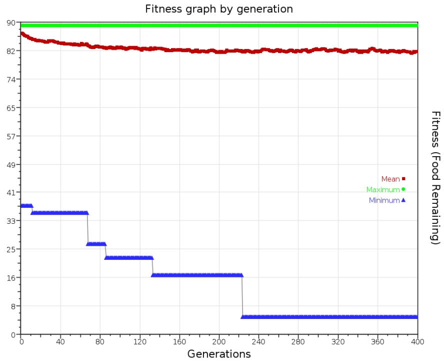
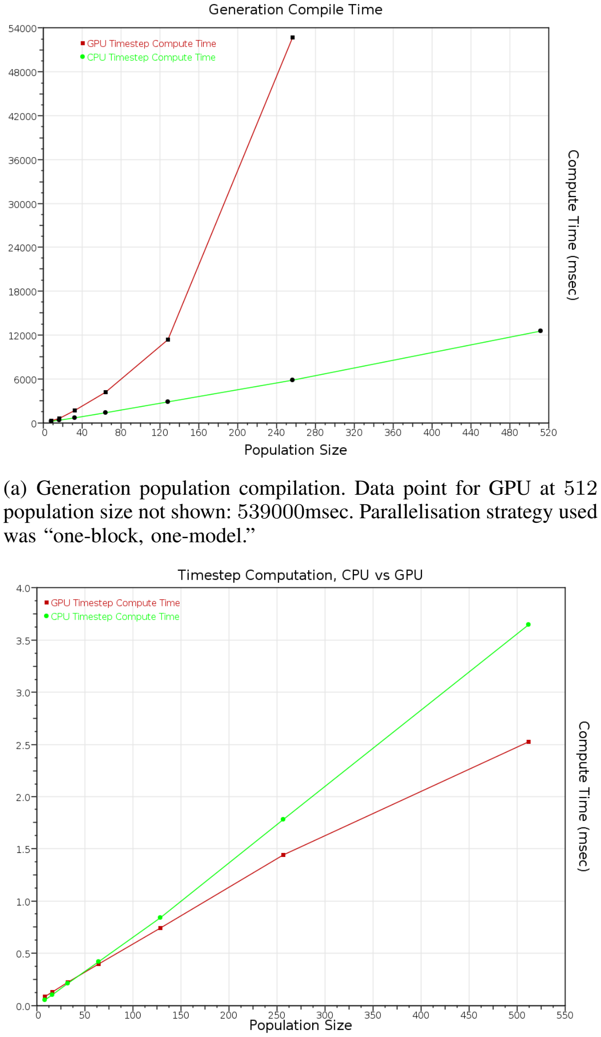


Fig. 1: Fitness plot by generation for the Santa Fe Ant Trail problem computed by the host using the single-threaded version of SOL.

The time taken to compile all 500 typed Terra trees to *host* code (in single-threaded SOL) is approximately 14.8 seconds (averaged five times). Whereas, the time to compile 500 unique typed trees to PTX is approximately 473 seconds. This is clearly an undesirable amount of time, considering that this dwarfs the evaluation time.

The time taken to compute a single timestep of a system with 500 candidates is approximately 3.57msec (averaged five times) for the single-threaded host version, whereas the GPU-parallel version computes a frame in 2.5msec (averaged five times). These are both of the order of a second for evaluating the entire population in the current generation. Additional scaling data for increasing population sizes are shown in Figures 2(a) and 2(b). In these plots it is clear that unless there is excessive computation necessary in the evaluation of a population of candidates, then it is likely that the GPU version of SOL is not necessary. At this point it is not clear why there appears to be an exponential rise in compute time required for compiling larger simulations at runtime for GPU, though the use of CUDA run-time code generation (RTCG) in Terra is experimental at this point. Also interesting in the timestep plot is that the GPU code surpasses the CPU code at relatively small population sizes. Though the CPU code is faster for population sizes of 32 and less, it was expected that the CPU code will surpass the GPU until at least 128 candidate programs. Altogether, the total lattice size operated on for the largest population was sized $512(32) = 16384$ by 32 lattice sites. This is a colossal 524,288 lattice sites, which makes the computing time for one frame considerably more reassuring for both CPU and GPU code.



(a) Generation population compilation. Data point for GPU at 512 population size not shown: 539000msec. Parallelisation strategy used was “one-block, one-model.”

(b) Timestep computation.

Fig. 2: Performance plots of timestep computation and population compiling between CPU and GPU compiled SOL code. Data is averaged over the first 300 time steps of randomly initialised runs.

4.2 Predator-Prey Model

For the Predator-prey model, timestep computation times were measured for the single-threaded and GPU-parallel compilations for different population sizes and lattice sizes. This model is more computationally expensive to compute per timestep than the Santa Fe Ant Trail. The reason for this is that both predators and prey must execute a SOL program, meaning that at times, every lattice site will execute a program, whereas in the Santa Fe Ant Trail, there was only one program being executed, that of the ant.

The optimisation objective is to select a permutation (with replacement) of rules for prey, to maximise the number of prey. Usually, the evasion strategy for prey is to find the closest predator, and move directly away from it. This is a simple but effective strategy. Emphasis here is given on performance results instead of convergence results.

The measured data in log-linear plots are shown in Figure 3. Different configurations of candidate model lattice sizes (8x8,16x16,32x32,64x64 and 128x128) were used, along with the "one-block, one-model" (1B1M) CUDA parallelisation strategy, the "many-blocks, one model" (*B1M) strategy, as well as single-threaded CPU configurations. The largest candidate model lattice (128x128) proved prohibitively expensive to compute by the single-threaded CPU configurations. There exist rapid increases in computing time between candidate lattice sizes, but increases in population size of these are relatively slower. The mere ability to simulate a lattice of size 128x128 with 64 heterogeneous candidates is enormously encouraging, considering that a population of this size is advantageous with regard to population-based optimisers. Unfortunately, while timestep computation scales well, compile time does not, however.

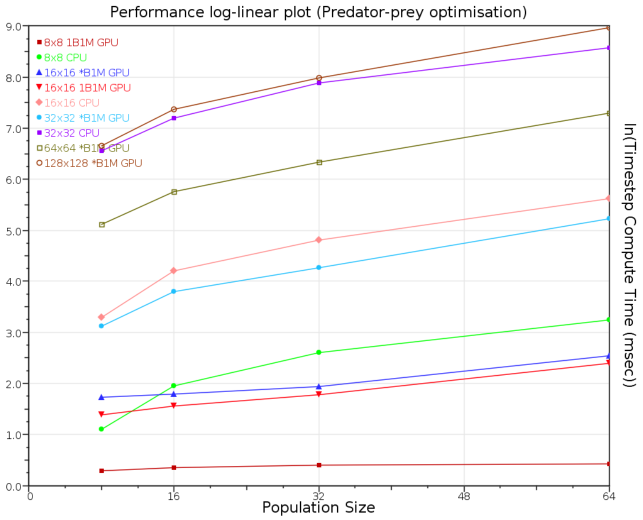


Fig. 3: Performance plot for different system sizes both in candidate model lattice sizes and number of candidates in populations for both the parallelisation strategies "one-block, one-model" (1B1M) and "many-blocks, one model" (*B1M). CUDA block sizes in *B1M were restricted to 16x16.

For the complete program shown in Listing 1, compile times from a SOL typed syntax tree to Terra code takes between 25msec and 30msec, however, some programs take up to 130msec to compile for the predator-prey model with heterogeneous programs (caused by the optimiser exploring the search space). The CUDA kernel compile time for 32 heterogeneous programs is approximately 40 seconds. For a trivially simple program (sol move left end), compile time from SOL typed tree to Terra is approximately 1msec, and kernel compilation is 530msec for 32 identical programs. A slightly more complex trivial program, but one still containing an optimisation construct such as the following takes on average 5msec to compile from a SOL typed tree to Terra, and approximately 1msec for a CUDA kernel to compile:

```
sol select recombination to minimise(1) move left end end
```

5. Conclusion

The use of run-time code generation is a good method for improving performance, provided that the evaluation phase of a population of candidate models is sufficiently complex. For models in which evaluation is less expensive (such as the Santa Fe Ant Trail model), it is more appropriate to use the single-threaded version of SOL. This allows programs to be compiled faster, while suffering a very small drop in timestep computation performance.

Systems larger than 512 candidate models appear to be out of reach of GPU run-time code generation. Such a result was expected, given that Cupertino et al. chose to evolve PTX code itself rather than use the CUDA run-time library to compile C code [44] noting that the latter would be too computationally expensive.

These limitations reaffirm that it is unwise to ignore the power of the newer multi-core processors available [45]. At the same time, the GPU should be applied when most or all of its theoretical computing power can be achieved. This demands proper choice in parallelisation strategy, which is anticipated to be automatically selected using relevant model information in the future. It is certainly possible to generate multi-threaded code on the host processor instead of CUDA code. This is a promising area for future work.

In summary, our SOL system is considerably more powerful and useful when it is supported by underpinning parallel computing capabilities. At the time of writing, SOL is a useful tool for experts, but there is scope for additional debugging and other user-friendly features that would make it useful for application domain users. We believe this approach has great power in providing a basis for developing initial application domain ideas into practical running simulations.

References

[1] Macal, C.M., North, M.J.: Tutorial on agent-based modeling and simulation part 2: How to model with agents. In: Proc. 2006 Winter Simulation Conference, Monterey, CA, USA. (3-6 December 2006) 73–83 ISBN 1-4244-0501-7/06.

[2] Figueredo, G.P., Siebers, P.O., Aickelin, U.: Investigating mathematical models of immuno-interactions with early-stage cancer under an agent-based modelling perspective. BMC Bioinformatics **14**(6) (2013) 1–38

[3] Macy, M.W., Willer, R.: From factors to actors: Computational sociology and agent-based modeling. Annual Review of Sociology **28** (2002)

[4] Gorochowski, T.E., Matyjaszkiewicz, A., Todd, T., Oak, N., Kowalska, K., Reid, S., Tsaneva-Atanasova, K.T., Savery, N.J., Grierson, C.S., di Bernardo, M.: BSim: An agent-based tool for modeling bacterial populations in systems and synthetic biology. PLoS ONE **7**(8) (August 2012)

[5] Manson, S.M.: Agent-based modeling and genetic programming for modeling land change in the Southern Yucatán peninsular region of Mexico. Agriculture Ecosystems & Environment **111** (2005) 47–62

[6] Birks, D., Townsley, M., Stewart, A.: Generative explanations of crime: Using simulation to test criminological theory. Criminology **50** (2012) 221–254

[7] Lotka, A.J.: Elements of Physical Biology. Williams & Williams, Baltimore (1925)

[8] Volterra, V.: Variazioni e fluttuazioni del numero d'individui in specie animali conviventi. Mem. R. Accad. Naz. dei Lincei, Ser VI **2** (1926)

[9] Luke, S., Spector, L.: Evolving teamwork and coordination with genetic programming. In: Proceedings of the First Annual Conference on Genetic Programming, MIT Press (1996) 150–156

[10] Jim, K., Giles, C.: Talking helps: evolving communicating agents for the predator-prey pursuit problem. Artificial Life **6**(3) (2000) 237–254 Summer.

[11] Calvez, B., Hutzler, G., et al.: Adaptative dichotomic optimization: a new method for the calibration of agent-based models. In: A. Tanguy C. Bertelle J. Sklenar et G. Fortino, éditeurs, Proceedings of the 2007 European Simulation and Modelling Conference (ESMâĂŹ07). (2007) 415–419

[12] Calvez, B., Hutzler, G.: Automatic tuning of agent-based models using genetic algorithms. In: Proceedings of the 6th International Workshop on Multi-Agent Based Simulation (MABS 2005). (2005) 41–57

[13] Stonedahl, F., Wilensky, U.: Finding forms of flocking: Evolutionary search in abm parameter-spaces. In: Multi-Agent-Based Simulation XI. Springer (2011)

[14] Said, L.B., Bouron, T., Drogoul, A.: Agent-based interaction analysis of consumer behaviour. In: Proceedings of the First International Joint Conference on Autonomous agents and multiagent systems: part 1, ACM (July 2002) 184–190

[15] Epstein, J.M.: Agent-based computational models and generative social science. Generative Social Science: Studies in Agent-Based Computational Modeling (1999) 4–46

[16] Ryan, C., Collins, J., O'Neill, M.: Grammatical evolution: Evolving programs for an arbitrary language. In: Proceedings of the First European Workshop on Genetic Programming. Volume 1391 of LNCS., Paris, Springer-Verlag (April 1998) 83–95

[17] Tisue, Wilensky: NetLogo: A simple environment for modeling complexity. In: International Conference on Complex Systems. (2004)

[18] van Berkel, S.: Automatic discovery of distributed algorithms for large-scale systems. Master's thesis, Delft University of Technology (2012)

[19] van Berkel, S., Turi, D., Pruteanu, A., Dulman, S.: Automatic discovery of algorithms for multi-agent systems. In: Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference companion. (July 2012) 337–334

[20] Junges, R., Klügl, F.: Evaluation of techniques for a learning-driven modeling methodology in multiagent simulation. In Dix, J., Witteveen, C., eds.: Multiagent System Technologies. Volume 6251 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2010) 185–196

[21] Privošnik, M., Marolt, M., Kavčič, A., Divjak, S.: Construction of cooperative behavior in multi-agent systems. In: Proceedings of the 2nd International Conference on Simulation, Modeling and optimization (ICOSMO 2002), Skiathos, Greece, World Scientific and Engineering Academy and Society (2002) 1451–1453

[22] Husselmann, A.V., Hawick, K.A.: Particle swarm-based meta-optimising on graphical processing units. In: Proc. Int. Conf. on Modelling, Identification and Control (AsiaMIC 2013), Phuket, Thailand, IASTED (10-12 April 2013)

[23] Kennedy, Eberhart: Particle swarm optimization. Proc. IEEE Int. Conf. on Neural Networks **4** (1995) 1942–1948

[24] Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: Evolutionary Computation Proceedings. (1998)

[25] Reynolds, C.: Flocks, herds and schools: A distributed behavioral model. In Maureen C. Stone, ed.: SIGRAPH '87: Proc. 14th Annual Conf. on Computer Graphics and Interactive Techniques, ACM (1987) 25–34 ISBN 0-89791-227-6.

[26] Privošnik, M.: Evolutionary optimization of emergent phenomena in multi-agent systems using heuristic approach for fitness evaluation. In: Evolutionary Computation, 2009. CEC '09. IEEE Congress on. (May 2009) 1829–1834

[27] Perumalla, K.S., Aaby, B.G.: Data parallel execution challenges and runtime performance of agent simulations on GPUs. In: SpringSim '08: Proceedings of the 2008 Spring simulation multiconference, New York, NY, USA, ACM (2008) 116–123

[28] Husselmann, A.V., Hawick, K.A.: Multi-stage high performance, self-optimising domain-specific language for spatial agent-based models. In: The 13th IASTED International Conference on Artificial Intelligence and Applications, Innsbruck, Austria, IASTED (February 2014)

[29] Husselmann, A.V., Hawick, K.A.: Towards high performance multi-stage programming for generative agent-based modelling. In: INMS Postgraduate Conference, Massey University. (October 2013)

[30] Husselmann, A.V., Hawick, K., Scogings, C.: Model structure optimisation in lattice-oriented agent-based models. Technical Report CSTN-222, Computer Science, Massey University (2014) Submitted to the International Journal of Modelling and Simulation (ACTA Press).

[31] Taha, W.: Domain-specific languages. In: Pro. Int. Conf. Computer Engineering and Systems (ICCES). (25-27 November 2008) xxiii – xxviii

[32] Ferreira, C.: Gene expression programming: A new adaptive algorithm for solving problems. Complex Systems **13**(2) (2001) 87–129

[33] Koza, J.R.: Genetic programming as a means for programming computers by natural selection. Statistics and Computing **4**(2) (June 1994) 87–112

[34] NVIDIA: CUDA C Programming Guide. 5.0 edn. (July 2013)

[35] DeVito, Z., Hegarty, J., Aiken, A., Hanrahan, P., Vitek, J.: Terra: a multi-stage language for high-performance computing. In: PLDI. (2013) 105–116

[36] Taha, W.: A gentle introduction to multi-stage programming. In: Domain-Specific Program Generation. Springer (2004) 30–50

[37] Lattner, C., Adve, V.: Llvm: A compilation framework for lifelong program analysis & transformation. In: Code Generation and Optimization, 2004. CGO 2004. International Symposium on, IEEE (2004) 75–86

[38] LLVM: User guide for NVPTX back-end. http://llvm.org/docs/NVPTXUsage.html accessed 12 March, 2014.

[39] Leist, A.: Experiences in Data-Parallel Simulation and Analysis of Complex Systems with Irregular Graph Structures. PhD thesis, Massey University, Auckland, New Zealand (November 2011)

[40] Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes. Cambridge University Press (2007)

[41] Koza, J.R.: Genetic Programming: On the programming of computers by means of natural selection. Massachusetts Institute of Technology (1992)

[42] Hawick, K.A., Scogings, C.J., James, H.A.: Defensive spiral emergence in a predator-prey model. Complexity International **12**(msid37) (October 2008) 1–10 ISSN 1320-0682.

[43] Scogings, C.J., Hawick, K.A.: Altruism amongst spatial predator-prey animats. In Bullock, S., Noble, J., Watson, R., Bedau, M., eds.: Proc. 11th Int. Conf. on the Simulation and Synthesis of Living Systems (ALife XI), Winchester, UK, MIT Press (5-8 August 2008) 537–544

[44] Cupertino, L., Silva, C., Dias, D., Pacheco, M.A., Bentes, C.: Evolving CUDA PTX programs by quantum inspired linear genetic programming. In: Proceedings of GECCO'11. (2011)

[45] Chitty, D.M.: Fast parallel genetic programming: multi-core cpu versus many-core GPU. Soft. Comput. **16** (2012) 1795–1814

Source Code Control Workflows for Open Source Software

Kevin Gary
Department of Engineering
Arizona State University
Mesa, AZ 85212)
kgary@asu.edu

Ziv Yaniv, Ozgur Guler, and Kevin Cleary
The Sheik Zayed Intitute for Pediatric Surgical Innovation
Children's National Medical Center
Washington, D.C. 20001
zyaniv,oguler,kcleary@childrensnational.org

Andinet Enquobahrie
Kitware, Inc.
Carrboro, NC, 27510
andinet.enqu@kitware.com

Abstract—**Many open source projects rely on the dedicated and highly skilled members of distributed development teams. These teams often employ agile methods, as the focus is on concurrent development and fast production over requirements management and quality assurance. The image-guided surgical toolkit is an open source project that relies on the collaboration of a skilled distributed development team, yet addresses a safety-critical domain. Due to this rare intersection of agile and open source development processes and a safety-critical domain, the IGSTK team has had to enhance the process with key elements and a set of best practices to augment commonly applied agile methods. This paper presents our experiences and lays out some research questions for the future.**

Index Terms—**agile, open source, safety-critical software.**

I. Introduction

As agile methods have matured, so has the realization that these methods are not dogmatic in their approach. Agile methods encourage *the right amount* of ceremony; therefore if a safety-critical system requires a greater emphasis on non-coding process activities like documented design and requirements management, then an agile approach will include these as necessary activities and not ceremony. Furthermore, we argue that agile and open source approaches focus more on code-level quality that most classic software engineering process models, which often talk about quality in every phase of the lifecycle *except* implementation.

We present our experiences on the image-guided surgical toolkit (IGSTK) project as a backdrop for this discussion. IGSTK is an open source software project that has employed agile best practices for the past nine years. In that time we started with the assumption that a lighter process is better, focusing on evolving code, and only adding in process elements where the need has arisen. The IGSTK team just released version 5.2 to the community, and in the past year has adopted modifications to its software processes.

II. Related Perspectives

Boehm [2] articulated the widespread belief that agile methods, due in part to their lack of emphasis on documentation, requirements stabilization, planning, and other large-scale synchronization points in the software process. But recently, literature has started to appear suggesting this may not always be the case, particularly in healthcare applications. Dwight and Barnes [5] describe a "lean-to-adaptive (L2APP)" variant on agile methods in a clinical research setting to streamline value delivery by utilizing a parallel flow side-by-side with laboratory validation. The three-phase L2APP model (speculation, collaboration, learning) strikes us as an agile way to manage requirements in an innovation-driven domain, though it does not say much about downstream processes related to design review and in-construction change management. The authors acknowledge the process is leveraged in the clinical lab to develop innovations for later large-scale production by shops prepared to take a technology to market. To us this represents a reasonable tradeoff in upstream efficiency but may punt too much responsibility downstream – how early does evidence of traceability and design rigor have to be accumulated?

Ge, Paige, and McDermid [8] present a detailed discussion of Agile versus plan-driven methods using Boehm's central premise as the framework for discussion. The authors then go on to suggest a semi-agile process that incorporates aspects of traditional plan-driven processes such as up-front design and hazard analysis with iterative development *and* iterative development of a safety argument. The presentation admits there is a lot of devil-in-the-details, with no current general principles for deciding how much agility is too much or too little. The key seems to be in calibrating the process to do just enough at each point in the process, a complex process goal.

Despite the complexity, we philosophically agree with [8] in our own recent paper [7]. Agile methods do embrace activities like planning, design, and validation as long as they are *without ceremony*, meaning they are not performed for performance sake; they are performed at the right times and to the extent needed (and no more) to achieve product requirements (for example, certification). Exploration of general principles, reference process frameworks, or evaluation criteria to guide practitioners in adopting "just enough agile" is a worthy pursuit. Finally we note that none of this discussion contradicts Boehm's original assertions; Boehm noted that knowing the right places to apply the right process is critical, and we view these explorations as an investigation into where agile can fit as a means of harnessing its benefits in the healthcare domain.

III. IGSTK

IGSTK is an open source framework for creating surgical applications. IGSTK is distributed under a BSD-like license that allows for dual-use between academic research labs and commercial entities. Image-guided surgery involves the use of preoperative medical images to provide image overlay and instrument guidance during procedures. The toolkit contains the basic software components to construct an image-guided system, including a tracker and a four-quadrant view, incorporating image overlay. IGSTK also leverages other open source projects, specifically ITK for segmentation and registration, VTK for visualization, and FLTK and Qt for the user interface.

IGSTK has geographically distributed developers, complex application requirements, and framework constraints for extensible and reusable architecture components. This is obviously a tremendous challenge compounded by the safety critical nature of the domain. The IGSTK team has created its software processes to balance an agile development philosophy with an integrated requirements elicitation and management approach, and consequently has arrived at a methodology that is fast and flexible, yet meets the stringent needs of this application domain.

IGSTK development presents interesting challenges from a methodology perspective. These complexities derive from the nature of the requirements, the makeup of the team, the dependence on pre-existing software packages, and the need for high quality standards within this domain.

The first challenge to IGSTK development derives from the nature of the framework-level requirements, which are difficult to completely understand before applications are constructed upon it. Waterfall-style methodologies [11] that attempt to define requirements completely before development begins are not considered suitable. Rational Unified Process (RUP) use-case driven modeling [10] is selectively applied through a customized process C-PLAD [1], as we cannot assume non-functional requirements derived from a set of applications known today represent a complete set of requirements for the future.

The second challenge to IGSTK development is the makeup of the team, comprised of academic and commercial partners collaborating in a distributed setting. Most if not all of the team members have other demands on their time. These factors create challenges for setting project deliverables and expectations over medium- and long-term horizons. Fortunately, most developers are deeply familiar with the domain and with a common set of tools and libraries from which to begin development. The requirements, team composition, and use of pre-existing software suggest that agile methods [4] should be applied to IGSTK. All team members have significant exposure to agile methods; some have even developed agile-ready tools that are employed on IGSTK [12].

Another challenge to IGSTK development – the high quality standards demanded the application domain – suggests that some agile practices need to be reinforced. For example, FDA guidelines for approval of medical devices require traceability of requirements through implementation and testing. To address these complexities, IGSTK adopted an agile approach augmented by a set of best practices we have previously described in detail [6] so we merely list here:

Best Practice #1: Recognize people as the most important mechanism for ensuring high quality software. This agrees with the philosophy espoused by the agile community [3].

Best Practice #2: Promote constant communication.

Best Practice #3: Produce iterative releases.

Best Practice #4: Manage source code carefully. A paradox of open source development in this space is that on the one hand you want to encourage community contributions and innovation, but on the other you need to manage change to software artifacts carefully. We expand on our recent process modifications to address this issue (in part).

Best Practice #5: Validate the architecture. This best practice is a nod to the specialty of the domain, and is discussed in more detail in the next section.

Best Practice #6. Emphasize continuous builds and testing.

Best Practice #7. Support the process with robust tools.

Best Practice #8. Emphasize requirements management in lockstep with code management.

Best Practice #9. Focus on meeting exactly the current set of requirements; it makes traceability easier, not harder.

Best Practice #10. Allow the process to evolve.

IV. Best practices for Community Source Control

These best practices are not foreign to agile practitioners, or even to non-agile practitioners. In the safety critical domain, following only these practices is unusual and not sufficient. Key process elements need augmentation to ensure safety. In a previous paper [7] we explored architecture validation (best practice #5) and requirements management (best practice #8). In this section we describe our past and new activities to support best practice #4, *manage source code carefully*.

Agile methods are certainly highly iterative; the predominant agile process models, Scrum and XP, use short time-boxed iterations as a mechanism for managing change. But beyond short iterations, agile methods have other practices facilitating an almost continuous checkpointing of the process – the daily scrum, pair programming, scrumboards, and continuous integration and testing dashboards. However these practices are typically best implemented when the team is physically co-located and dedicated to the project. Hurdles, ideas, and other communications are addressed in real-time. Even with powerful online tools, geographically distributed teams can only rarely achieve this real-time interaction. Time boundaries, language barriers, network infrastructure issues, and local distractions and responsibilities at multiple sites are common causes for this degradation. It is exacerbated in open source communities, where participants are often dispersed individuals working for different organizations and only part-time in that community. Further, the community has to have either formal or informal rules regarding who can do what with which source code modules. IGSTK operates under such constraints, with global participation from researchers in hospital labs and universities, industry partners, and practitioners who made partial contributions over time. IGSTK has employed some traditional mechanisms for managing this collaboration, including developer meetings, user group meetings, online wikis and support forums, two mailing lists (adopters and core developers), and restrictions on core component development to only the core team.

Source code control is a critical practice in managing change in an agile and open source environment. At any point in time a community developer may be working on a defect fix, a new core feature, a new non-core feature, a refactoring, or an application-specific behavior or integration. That developer may be working in isolation, with little visibility in the rest of the community until the time arrives that s/he desires submission of the changed code. Should the code be accepted? Does it adhere to defined quality policies? Has it been code reviewed? It is an experimental or application-feature or a feature identified as desired by the community? These and more questions arise in this situation, and all pose risks when developing in a safety critical domain.

Over the past decade IGSTK has used a traditional, centralized approach to source code control common in many software projects. This approach supports a "branch-and-merge" centralized workflow. IGSTK further adopted a multiple codeline practice known as *sandboxing* to allow for experimental features to be developed under lower quality conditions. But problems arose over time. The sandbox codeline grew larger, much larger, than the main product codeline, to the extent that less rigorous traceability on the sandbox led to inevitable technical debt. In other words, the sandbox repository became filled with incomplete features whose owners may have gone inactive and whose documentation and issue management in other tools was outdated. Even if a community member identified a desire to complete a sandboxed feature, they were often forced into significant rework or to scrap the sandbox module and start over in order to meet the quality policies on the mainline.

In the past year IGSTK has moved to the popular distributed version control system, Git. Git enjoys significant popularity now, though many projects use it in the same manner as traditional centralized *delta* repositories. Git's distributed repository model encourages many practices that go against low-level practices taught in the traditional model – for example, instead of "check-in early and check-in often" (to minimize merge conflicts in optimistic centralized tools), the distributed model encourages local branches with infrequent merges, preferring to merge only when a feature is complete and up to quality policy. The need for a sandbox is gone. Further, it encourages self-sustaining communities; community practitioners may maintain their own forked versions of repositories without burdening the core team with constant review of their features. Gone are the days of "contrib. modules" one may "use at their own risk" from the centralized repository; now one may publish their own forked repository and leave it to the market of adopters to decide what to use. A concern in this model is with the overall safety properties of the forked repositories – who has the overall ability to validate the safety properties of these forks with the core? A good research question! For the time being, this model saves the IGSTK core team valuable time in reviewing non-critical development.

Because of the peer distributed repository model used by Git and like-minded tools, a large variety of workflows may be employed on a project [3]. The IGSTK team reviewed the Git workflow literature and practices from related communities like ITK to adopt a variation of a *branchy workflow*. In this workflow two branch types are defined, a *topic branch* and an *integration branch*. The topic branch commits represent work on a single new feature or fix. The local developer(s) who work(s) on it name it locally but the branch is not public on the blessed IGSTK repository, so no other developers can branch from it. The integration branch is where merges of two or more topics happen. These branches have quality policy constraints enforced, and are publicly named on the blessed repository (one may pull from it).

Figure 1 depicts the relationship between topic and integration branches.

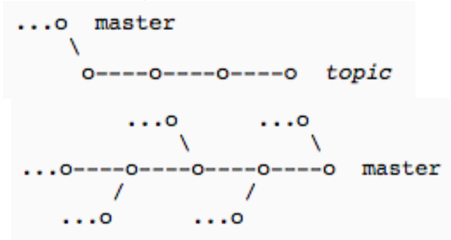


Figure 1. Topic and Integration branch commit patterns

Neither topic nor integration branches represent the main codeline, this is maintained separately in Git as the default branch (*master*). In other words, merges are not done directly into master, but into new integration branches, which are then merged into master after the integration is deemed stable and up to quality policies. Figure 2 shows what a sequence of commits may look like in IGSTK between the master, a topic branch, and an integration branch named next. Further details on the IGSTK Git workflow may be found on the IGSTK Wiki at http://www.igstk.org/Wiki/Git/Workflow/Topic.

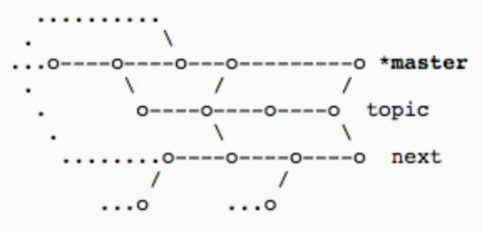


Figure 2. A possible sequence of commits on all branch types

This was a detailed presentation of a particular best practice in the agile IGSTK open source community. The level of rigor in daily collaborative practices of community developers is significant, and suggests if the trail of data of these practices can be harnessed and analyzed, it could provide a basis for safety case evidence for software in healthcare.

V. Discussion

IGSTK's agile approach is neither as rigorous nor as complete as it could be for a safety-critical domain. IGSTK is principally used in academic research centers and some small commercial endeavors outside the USA, which can afford to be more forgiving. Yet, the tale of IGSTK's agile evolution, we think, offers lessons and hope for applying agile methods to safety-critical domains. The work is laborious; to create and faithfully execute agile practices in a distributed open source community, every detail of the daily practices must be examined for the right amount of ceremony. We presented our revised approach to source code control as an example.

As we indicated at the end of section II, we believe there is an opportunity to create guidelines, models, and/or quality process criteria for the introduction of agile methods in the healthcare domain. Foremost, we believe it is a necessity – the explosion of personal medical devices and information management platforms such as the fitbit (fitbit.com) or smartphone-based sensor apps has serious implications for future patterns of individual-to-clinician healthcare. It will eventually become a necessity in systems development in healthcare (for example, tele-robotic surgery). Changes in medical device regulatory evaluation to a more evidentiary case-based approach [9] opens the door for agility. If agile methods can be instrumented to collect and aggregate daily practices into such evidence, then the possibility exists for expanded opportunities in healthcare development. Certainly the economic drivers are there. In our view research is needed on how to instrument agility to collect the evidence required for safety cases. Agile's benefits include the lightweight but constant management of detailed activity, and making this visible and transparent to all stakeholders. The increasing adoption of tools within the agile process focuses on communication between stakeholders (chickens) and developers (pigs). The identification and instrumentation of daily safety-related activities needs to be included in such toolsets to make this evidence collection continuous, feasible, and complete. Safety-based micro-evidence may then be aggregated to uncover macro-trends and introduce process improvements. This is a current focus of our research in this area.

Acknowledgment

This work was funded by NIBIB/NIH grant R01 EB007195. This paper does not necessarily reflect the position or policy of the U.S. Government.

References

[1] Blake, M.B., Cleary, K., Ibanez, L., Ranjan, S.R., and Gary, K., "Use Case-Driven Component Specification: A

Medical Applications Perspective to Product Line Development," ACM Symposium on Applied Computing, Santa Fe, NM (2005).

[2] Boehm, B. Get ready for agile methods, with care. *IEEE Computer* 2002; **35**(1):64–69.

[3] Chacon, S. *Pro Git*. APress, 2009.

[4] Cockburn, A.: "Characterizing People as Non-linear, First-order Components in Software Development." 4th International Multi-Conference on Systems, Cybernetics and Informatics, Orlando, Florida, (2000).

[5] Dwight, Z. and Barnes, A. Laboratory Driven, Lean-to-Adaptive Prototyping in Parallel for Web Software Project Identification and Application Development in Health Science and Research. *Software Engineering and Applications*, 2012; 5:62-68.

[6] Gary, K., Ibanez, L., Aylward, S. Gobbi, D., Blake, M.B., and Cleary, K. IGSTK: An Open Source Software Toolkit for Image-Guided Surgery. *IEEE Computer, vol. 39, no. 4, pp.46-53*, April 2006.

[7] Gary, K., Kokoori, S., Muffih, B., Enquobahrie, A., Cheng, P., Yaniv, Z., & Cleary, K. "Agile Methods for Safety-Critical Open Source Software", *Software: Practice and Experience*, 41:945-962, April 2011.

[8] Ge, X., Paige, F., and McDermid, J.A. An Itaretive Approach for the Development of Safety-Critical Software and Safety Arguments. *AGILE Conference* (2010).

[9] Geisler, J. "Software for Medical Devices", in *Mission-Critical and Safety-Critical Systems Handbook Design and Development for Embedded Applications* (Fowler, K. ed.) 2010 Elsevier Inc.

[10] Kruchten, P. *The Rational Unified Process—An Introduction, 2nd Edition*, Addison-Wesley (2000).

[11] Royce, W.W.: "Managing the development of large software systems: concepts and techniques." IEEE WestCon, Los Angeles, 1970.

[12] Schroeder, W.J., Ibanez, L. Martin, K.M.: "Software Process: The Key to Developing Robust, Reusable and Maintainable Open-Source Software." Proceedings of the IEEE International Symposium on Biomedical Imaging. Arlington, VA 2004.

DIRCE - Design of Interaction and elicitation of Requirements focusing on the Communication and Exploration of ideas - Experiences of use in content creation systems for digital television

Marilia S. Mendes[1]
[1]Estácio University of Ceará (FIC)
Fortaleza, CE – Brazil
mariliamendes@gmail.com

Elizabeth Furtado[2]
[2]University of Fortaleza (Unifor)
Fortaleza, CE - Brazil
elizabet@unifor.br

Abstract—*This paper aims to present an approach to help professionals focus on interaction aspects since the early stages of the process of development of an innovative system. This approach guides the application of techniques addressing the integration between the processes of requirements elicitation and interaction design by considering both the experiences of users and other factors which influence the context of use of a system under development. Such approach was applied to a system which creates content for DTV, resulting as major contributions the description of pre-patterns for the context of content creation for DTV, as well as an analysis of the implications of the use of techniques of user experience for the activities of software engineering and interaction design.*

Keywords:*User experience, Experience Prototyping, Human Computer Interaction, pre-patterns, innovative systems, Digital Television.*

1. Introduction

This paper presents an approach to help Human Computer Interaction (HCI) professionals (such as developers, designers and usability engineers) understand the experiences of target users in a project of innovative computing systems. An innovative system is characterized by the fact that target users — and in many cases developers too — are not yet familiar with the technology being studied. The team that will develop the system must go through a process of understanding the experiences of target users in order to assure the usability of such system. The usability notion of [14] was adopted, which decomposes usability into technical quality (the system should work) and user quality, and then decomposes user quality into functionality (the system should provide users with the right functionality that most users need or strongly want), ease of use (it should be easy to learn and easy to operate) and user experience (most users should have positive experiences when using such system). From this understanding, they could define usability goals and system requirements. Developers (including designers) have difficulties in creating such definitions. Features and system constraints involving the usability as a whole refer to a series of doubts concerning possible combinations among HCI concepts (such as: why will some users probably not accept using a specific modality for a task in a use environment?). To answer this question, a lot of approaches have focused on the interaction study exclusively with users, disregarding the team who will develop the system. As it was previously mentioned, target users are not familiar with an innovative system under development. This is the main reason why users have difficulties when talking about their expectations during experiences with a specific system. Our assumption is that the most relevant professionals developing the system should also have such experiences for better describing the system.

The methodology developed by this work is based on the user experience techniques. When applied to relevant stakeholders of the project (such as target users and HCI professionals), those should be able to:

Improve the understanding of relevant stakeholders about the interactive possibilities of the system under development. Throughout the design process, there are situations in which the designer is forced to be creative: be able to see people, things and situations in a new perspective [7]. This situation is reinforced when the system is innovative. In this case, the designers have to "imagine" both the operation of the system — which does not yet exist— and the use of the system by users. Dow et. al [5] argue that designers do not have enough information on new technologies to understand the limitations of feasibility, physical properties and the vectors of change. The designers find it difficult to design innovative applications due to the level of complication, the lack of technological support and the unstable nature of new technologies [5]. From this came the need to support them to facilitate their understanding of the system under development.

Promote an awareness process among stakeholders of the project regarding the contextual factors that influence the use of a system in an environment since the beginning of the interaction design. In innovative projects, specific techniques have to be used to place the context of use as the principal concept to observe the experiences of use by the target users. Ethical factors (what people like or do not like and how the privacy of information is handled in the context of interaction), social factors (how users are able to overcome the risks of ownership of technology) and environmental factors (such as weight and portability of the equipment as for the weather, amount of people in streets, etc..) influence the use of the system and are difficult to identify without a detailed understanding of the system. The development team of a system should be able to handle the unexpected, knowing the reaction of users and the return of their interaction. Users can report their concerns, problems that have come from similar systems, etc. By that, the team can improve the interaction design by analyzing contextual factors, checking whether the scenarios presented are consistent with the reality of users, and whether the system will be well received by them.

Provide better communication between stakeholders enabling a consensus on interactive alternatives of the system. Preece; Rogers and Sharp [10] emphasize the importance of multidisciplinary teams in interaction projects. Gathering a group of people with different backgrounds and training promotes the combination of skills with an understanding of the different areas of application needed as to design the new generation of interactive systems. However, the authors also emphasize the difficulty of communication within the team: the more people with different backgrounds in a design team, the harder it can be to make them communicate and advance the projects developed [10].

In software engineering, there is a globally accepted principle: good communication between members of a development team is needed in order to have a better use of their skills as for the system. The developer needs to have good communication with the designer so that the result of the development will be that which they have designed for it and, finally, also the designers have to communicate with the users and usability engineers so that the system will have a good design guide.

This article is organized into seven sections: the second emphasizes the reasoning of the methodology; the third describes the proposed approach; the fourth section explains the application of the methodology in the case study; the fifth presents the results obtained; and the sixth brings a discussion. As for conclusion, some final considerations and future work are presented.

2. Rationale of the methodology

This work focuses on the requirements elicitation and the design of the interaction of innovative systems considering the experiences of users as well as the experiences of the team developing the system. The activity of requirements elicitation is responsible for the specification of requirements for the operation and development of software systems. The design of interaction goes beyond the interface, and its study involves dealing with issues that often arise during the use in a process of participatory design. The participatory design is characterized by the active participation of end users of the software throughout the design cycle and development. There are several techniques of participatory design. The following ones are cited as important for this work:

- User experience: encompasses all aspects of user interaction during their use of the system [14];
- Experience prototyping: this is a technique that combines user experience with prototyping and proposes to provide developers / designers, users and customers of a system with the possibility of "experiencing it by themselves" before the system is developed [1];
- Space-time representation: allows stakeholders of the project to report their experiences of interaction in terms of space and time, and such representation can be made by use of post-it, notes, drawings, photographs, etc. [5];
- Pre-patterns: are emerging patterns that are not yet in common use by the design community and end users. They are used in the exploratory stages of the project of innovative systems, in which the problem to be solved is not yet known with a certain pattern [3].

These techniques are associated in the approach proposed in this work, which is described in the next item.

3. Approach DIRCE

The approach proposed in this paper is called DIRCE – Design of Interaction and Elicitation of Requirements focusing on the Communication and Exploration of ideas. Such approach is suggested to be applied before or simultaneously with the phases of requirements engineering and design of interaction. As a reference point for integrating the approach, the process of engineering requirements of Sommerville [13] and the activities of interaction design of Preece; Rogers e Sharp [10] were taken. The process engineering requirements of Sommerville is composed of four phases, which are: (1) Feasibility Study, (2) Elicitation and analysis of requirements, (3) Specification of requirements and (4) Validation of requirements [13]. Preece; Rogers and Sharp define four basic activities for the interaction design: (1) Identifying needs and establishing requirements, (2) Developing alternative designs, (3) Building interactive versions of designs and (4) Evaluating designs [10].

The proposed approach considers these activities involving the techniques cited by the following steps (Figure 1): (1) Study of similar applications, (2) User experience, (3) Identification of system requirements, (4) Definition and implementation of pre-patterns; and (5) Definition of Interaction Design. In the first step of the methodology, the applicators of the experience do a study of applications which are similar to the system to be developed. The technique used for this is the analysis of competitors, which focuses on the identification of the strengths and weaknesses of competing products before they start working on the design of the system to be developed. Also in this step, the applications studied with the team and participants are presented so that they could acquire some knowledge about the innovative system.

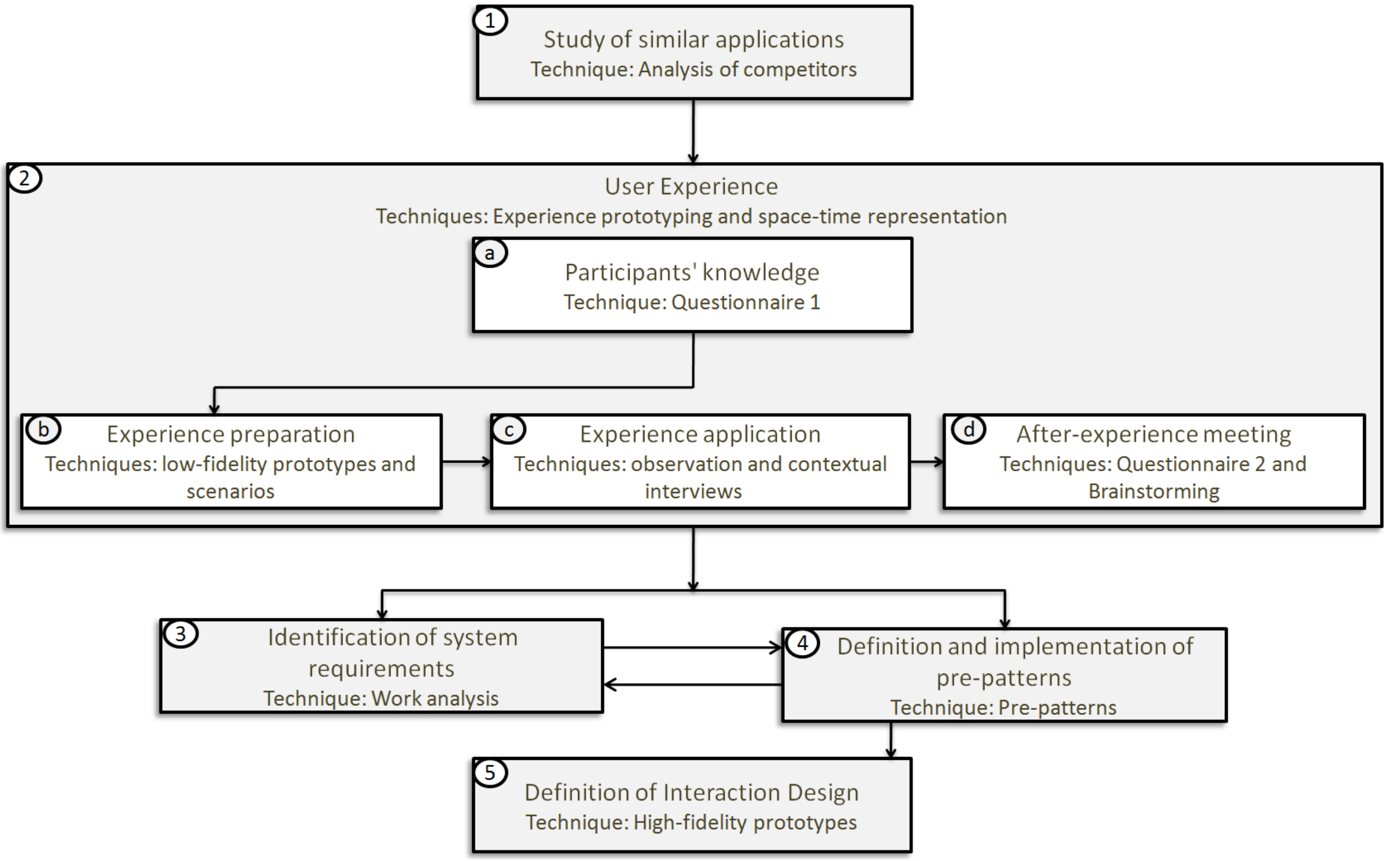

1
Study of similar applications
Technique: Analysis of competitors
2
User Experience
Techniques: Experience prototyping and space-time representation
a
Participants' knowledge
Technique: Questionnaire 1
b
Experience preparation
Techniques: low-fidelity prototypes and scenarios
c
Experience application
Techniques: observation and contextual interviews
d
After-experience meeting
Techniques: Questionnaire 2 and Brainstorming
3
Identification of system requirements
Technique: Work analysis
4
Definition and implementation of pre-patterns
Technique: Pre-patterns
5
Definition of Interaction Design
Technique: High-fidelity prototypes

Fig. 1. Methodology DIRCE

In the next step of user experience, the techniques experience prototyping and space-time representation are applied. This phase is divided into four steps: (a) Participants' knowledge, (b) Experience preparation, (c) Experience application; and (d) After-experience meeting. In order to evaluate the participants' knowledge, a questionnaire is provided, aiming to assess how well those participants understand the system to be developed and their experiences as for similar systems. In (2) the applicators should prepare simple low fidelity prototypes, which will serve as communication tools to demonstrate how participants can use the system. Scenarios of use can also be defined and thus used as a guide for the experience activity.

In (c) the experience should be applied to the participants with the presence of facilitators / observers. In (d) the participants should report their experience and answer a questionnaire in order to provide more details about the user experience.

The last step is the identification of system requirements. In this step, the results of the previous steps are analyzed (interviews, questionnaires, prototypes, sketches, images, etc.) in order to integrate and organize the results.

From the results of this analysis, the pre-patterns for the system are defined. These pre-patterns are then applied, validating scenarios and ethical, social and environmental factors of the system under study and having a relationship with the interaction design for the construction of the final prototypes.

By suggesting techniques to be applied in the phases of Requirements Engineering (RE) and Interaction Design (ID), the approach of this work enabled the definition of a conceptual framework (Figure 2). A conceptual framework consists of a set of concepts used for solving a problem of a specific domain [12].

The framework proposed is a generalization of the methodology, relating the techniques applied in the methodology to the processes of RE and ID. The process of the methodology provides results for all phases of the requirements engineering step, since the feasibility study until the requirements validation. For the interaction design, the needs and requirements are important for the preparation of the user experience and, as a result, alternative designs, interactive versions of the design and evaluation of resulting designs are obtained.

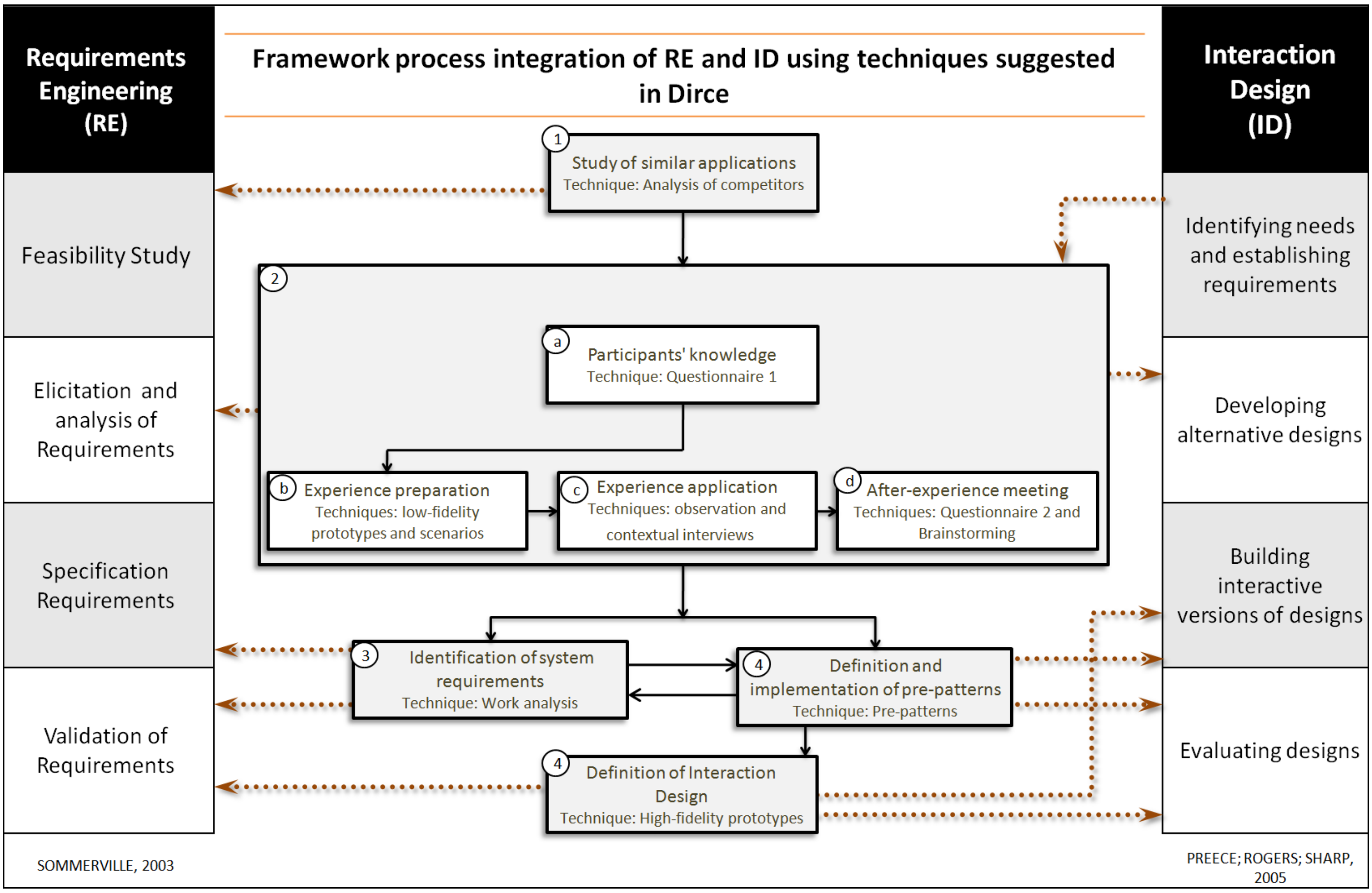

Requirements Engineering (RE)
Framework process integration of RE and ID using techniques suggested in Dirce
Interaction Design (ID)
Feasibility Study
Elicitation and analysis of Requirements
Specification Requirements
Validation of Requirements
Identifying needs and establishing requirements
Developing alternative designs
Building interactive versions of designs
Evaluating designs
1 Study of similar applications
Technique: Analysis of competitors
2
a Participants' knowledge
Technique: Questionnaire 1
b Experience preparation
Techniques: low-fidelity prototypes and scenarios
c Experience application
Techniques: observation and contextual interviews
d After-experience meeting
Techniques: Questionnaire 2 and Brainstorming
3 Identification of system requirements
Technique: Work analysis
4 Definition and implementation of pre-patterns
Technique: Pre-patterns
4 Definition of Interaction Design
Technique: High-fidelity prototypes
SOMMERVILLE, 2003
PREECE; ROGERS; SHARP, 2005

Fig. 2. Framework process integration of RE and ID using techniques suggested in Dirce

4. Case study

DIRCE approach was applied to a project whose objective was developing a system for creating web content for DTV (DTV). It was applied to 12 participants of the project team: 7 researchers teachers (who will be the end users of the system) and 5 professionals from the development team (3 developers, 1 designer and 1 usability expert). The team consisted of 7 women and 5 men, aged between 20 and 35 (8) and above 35 (4). It was noticed that 7 of those have experience in preparing lessons, from whom 6 had experience in distance learning, but none of those had experience in DTV. The participants answered a questionnaire which enabled the identification of the characteristics described above, besides some other important information as for the system, such as: all of them have a television at home, use a computer and the Internet, normally create content such as blogs, news, warnings to people or websites.

1) Study of similar applications. Two applications were presented to stakeholders aiming to improve their understanding of the interactive possibilities of this new system. The applications presented are meant to create content on the web for viewing in the DTV, such as the system being designed. However, neither has educational context to create classes for DTV.

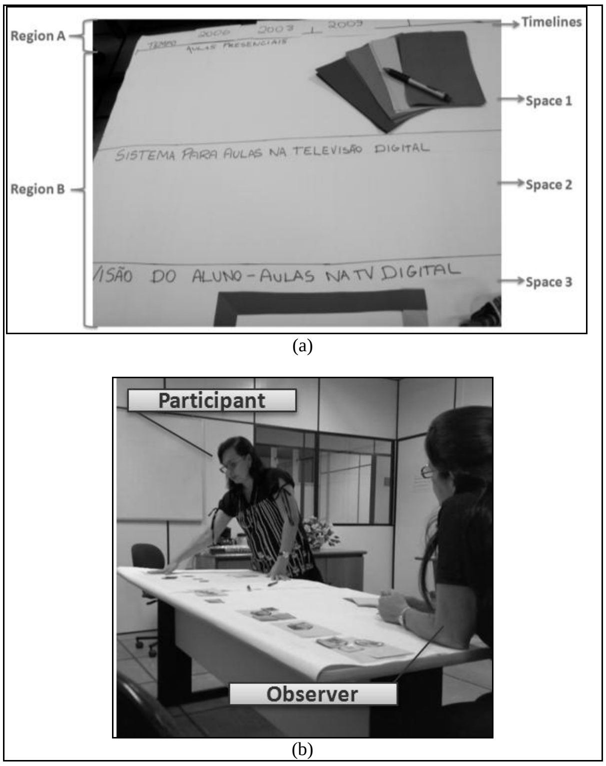
The sample of similar applications facilitated and clarified a lot of questions, and the following points of interest for research arose: how to have teachers (who are accustomed to preparing classroom lessons) create lessons for a web system to be viewed on DTV? How would teachers prepare such lessons? How would they use what they have already done? Which would their difficulties be?

In this sense, this work focused on a way for providing user experience for teachers to verbalize how they would like the system to be. The starting point was the need for supporting the development of courses in an easy way and which does not disrupt their productivity.

2) Preparation of the experience. In order to provide users with closer scenarios, the experience was contextualized in terms of time and space. Time was chosen based on the assumption that teachers would organize their classes based on a time sequence and that they would like to use these classes later. Space was chosen for defining experienced scenarios of use. The prototype was built by using low cost materials such as paperboard, icons, colored cards, pen and paper.

The paperboard was prepared on a table and was divided into 3 parts grouped into 2 regions (Figure 3). The time representation (Region A of Figure 3) was treated by means of a time line, and the space representation (Region B of Figure

3) was treated by defining three scenarios whose results were freely arranged in the spaces of region B in Figure 3: (1) the 1st horizontal line set the space for the classroom scenario; (2) the 2nd horizontal line defined the space for the scenario in which the system would be used for the classes on DTV; and (3) the last line defined the space that would correspond to the student's vision on DTV. The participants were instructed to use the time line for presenting their experiences in teaching classes (years could be freely written on papers) and to develop scenarios in the context of spaces.

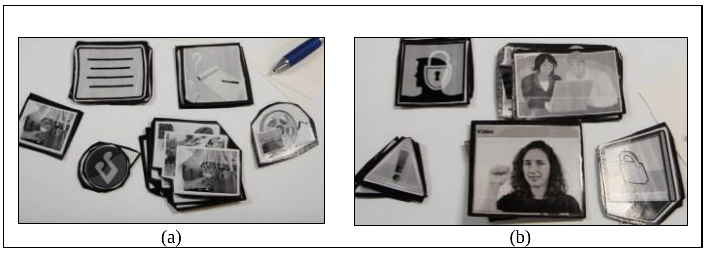


(a)

(b)

Fig. 3. Low-fidelity prototype (a) and Application of the experience (b)

Classes would be prepared by using content cards (Figure 3-a), a pen for the participants to make notes and a set of icons representing the following objects of content creation for a class: text, images, audios, videos, presentations and questionnaires (here called as media, which are represented in Figure 4-a). In order to represent factors that could influence the use of the system, some icons were previously created, with the following concepts: privacy, safety, accessibility, collaboration and copyright (represented in Figure 4-b). In case the participant would not choose any of these media or concepts, they could use white papers for expressing such concepts.



(a)

(b)

Fig. 4. Icons

The scenarios of the experience prototyping which were set to be presented to participants were:

- Scenario 1 – Preparation of classroom lessons. Reflection: Do you currently prepare your classroom lessons?
- Scenario 2 – Preparation of lessons for DTV. Reflections: How could this new system help you prepare your lesson for DTV? What difficulties would you have in using such a system?
- Scenario 3 – Creation of the students' vision on DTV. Reflections: How would students attend such lessons on DTV? What factors would influence their use for content creation or the student's use for the interaction with the content?

3) Application of the experience. The application of the experience in this case study had the following characteristics: individual session; duration: from 30 minutes to 1 hour and; monitoring: 1 or 2 observers. During the application of the technique, the participants were provided with a short explanation, ensuring that they would not be tested, but actually participate in an experiment to help the final product which will be developed. The scenarios are explained to the participant before they start using the system; some contextual interviews are applied while they use it. These contextual interviews involve talking to the user while they perform their tasks, by combining the techniques of interview and observation.

The first scenario, focusing on classroom lessons, aimed to motivate the participant in the context that was previously known. This scenario intended to rescue the participant's teaching practices for creation of classroom lessons. The participants were asked to start this experience by explaining how they prepared the class content by using the resources available for improving communication.

The second scenario aimed to introduce the participant in the application context - a system for creating content for DTV. As they have explained their classroom lessons in the first scenario, in this scenario they are encouraged to think about the following: "*You have just explained how your classroom lessons work. Now imagine that the institution where you work has provided teachers with a system for creating distance classes for DTV. Therefore, you should adapt your previously created class by using the system for DTV*". In this scenario, the participant should imagine the system and explain what the difference would be between the classroom lessons and the DTV lessons.

Still in the second scenario, the next step consisted in describing what features the system would have. The applicator asked the participants what features they believed such a system would have. The participant makes a comparison between the way they usually teach (classroom lessons) and the way they think it would be in DTV, by thinking of which features would be important for the preparation of their classes, for example whether they would use videos or apply group activities, and what they would like to use in the system. They were supposed to imagine how that content would be handled on DTV.

In the third scenario (student's vision) the user is asked: "*How do you think the student will see this class you have created on DTV?*" The participant would think as a student, reflecting about how the student would attend that class on DTV. During the course of user experience, the participant takes the position of both a teacher and a student. Then, there is a reflection on contextual factors which may influence the use of the system. The participant is asked whether they have thought of any factor which could influence the use, such as: accessibility, copyright, security, sharing, collaboration or privacy. The images of Figure 3-b represent the application of the technique. In this case study, participants were supported as for the use of the system. They were also monitored as for their experience with the system, both through observation and through questions.

4) After-experience meeting. In this meeting, the process and the experience description were explained, and some questions about the experience were raised. Each participant detailed how their user experience with the future system had been. The participants were then encouraged to point out the possible features of the system previously imagined throughout the three scenarios of use. They talked about their ideas and expectations regarding the system. Afterwards, they filled out a post-experience questionnaire. A week later, the applicator of this methodology together with the participants explored the data resulted from this phase of the methodology and created a requirements document and a prototype of the interfaces considering the data collected. Such prototype was presented to participants and thus validated.

5. Analysis of results

The analysis of results was based on two moments: (1) during the explanation of the experiment, through observation, contextual interviews and use of prototypes and; (2) after the experience, through brainstorming and questionnaires.

The results obtained from the application of these techniques are described below, which are basically the analysis of the information obtained and of the participants' behavior, noticed through their speech and behavior.

Observation of the experience: This experience brought a lot of ideas and suggestions. By imagining how the system should be, the teachers described important features it should have. They said: "*I would like the system to provide us with media such as pictures, audios and videos*", "*I would like to know the student's level*". They made comparisons between their classroom lessons and how these lessons would be applied in the system: "*When I teach in the classroom, it's possible to know whether the student is or isn't interested in my class, so how will I realize it through the DTV system?*". They also talked about what they did not wish the system to have: "*I don't want to have much work to search for content*". Therefore, they would list what was and what was not necessary for the system, which would then be transformed into requirements and constraints for the system.

Contextual interview: The contextual interview facilitated the conduction of the experiment during its use. The questions were supposed to have the participant reflect about the system. The participants were asked: "*How would this system help you prepare your lesson?*" and some of the answers were: "*The system should be simple and easy to use*"; "*The system should give me feedback so that I know whether my class was good*"; "*The system could send me questions from students*", and then the applicator asked the following question: "*What features do you think this system should have?*", to which the following responses were obtained: "*Teaching tips on how to prepare a lesson and technology tips on how a good class on DTV would be, for instance: not using long texts*"; "*Possibility of the teacher to determine a logical sequence for the student to view the content, for example: the advanced subject would be seen only after the basic subject has been studied*".

Another question proposed to participants was: What are the main difficulties for the teacher whose habit was to prepare classroom lessons, when they face such a digital system? Some of the answers were: "*When I prepare my lesson and go to classroom, the lesson totally depends on the interaction with the class; I can change it and add something more. When using the digital system, how can I do this?*", "*In DTV, more people have access to this content, so that will demand me more care when I prepare my content!*", "*I'll have to quote all the references*", "*I won't have direct interaction with the students and thus I won't be able to see their interest or their possible questions as to the content of the lesson*".

In the third scenario, the following question was asked: "*How do you think the student will see this class on DTV?*", to which some participants answered: "*First of all, they should have access to a menu, from which they would choose what to display.*", but all the participants explained visually by using drawings and screen assembly of the prototype.

The participants were asked to freely think of factors that could influence the use of the system. The following factors were suggested: communication, location, sharing, privacy, trust, collaboration, reuse, and usability requirements. For each of those factors, they would say whether they had thought of it or not. Some users stated to have thought of it, whereas others had not. Sometimes they did not think of it during the process, but at the time they were asked about it, they began to think of it. One participant has confidentially said: "*I did not think of it because the DTV is open*", yet another participant has affirmed not to have thought of copyright for they thought the system would automatically deal with this subject later.

Prototypes: Participants began the experiment by planning their classroom lessons as a function of time. They would put the years when they began to teach and would explain the content and its evolution. The main phrases that show how the technique was used are: "*In that year I would use only texts, but in the following year I began to use videos...*", "*In that year I would prepare the lectures by using slides and questionnaires, and in the following year I took advantage of such content but added group activities*".

More than one participant would divide their lessons into periods. They explained their classes as follows: "*The class* *usually starts as a lecture, then a debate is raised, and at the end I suggest some more literature on the subject*"; "*Firstly I expose the contents, then I assign an exercise for evaluating student's understanding...*"

Some participants (2 out of 7 teachers) focused on teaching methods of planning the educational concepts in the application. One participant used free cards to describe aspects of Didactic Engineering (preliminary analysis, a priori, posteriori experimentation and analysis [4]). The system would support the a priori analysis when the teacher would use existing material to generate a new one. The experimentation refers to the use of content by the student, which is followed by a posteriori analysis of learning by the teacher.

In the second scenario, system for classes in DTV, users continued to explain step by step how their classes would be in this context, considering issues such as the reuse of content and media for the preparation of the digital lessons. They said: "*Firstly, I would provide the students with a text on the subject, then I would have them answer a questionnaire, then I would show a video...*". And this way they were actually making the lesson for DTV. The same also happened when the participants were asked about how they thought the students would see the class on DTV. Also for this, the participants numbered a sequence of steps: "*The student would first see a video, and then they would see a text on the subject...*".Some participants also represented the vision of the student with representative television frames.

Brainstorming: Throughout this step, each participant has listened to the others' thoughts regarding the system and have argued about the ideas that emerged. Due to the diversified profile of those involved, different expectations as for the system were detected. Whenever a participant would tell their experience, and their expectation about the system, it was common to hear from other participants: "*I have already thought otherwise*" or "*I think it could be that way*", encouraging them to have a healthy discussion in order to reach a consensus. The applicators of the experience were also mediators during the brainstorming.

It was also noticed that the user experience provided them with more security so that they could discuss reporting their opinions about the system, making their contribution more active compared to meetings happening before the application of the methodology.

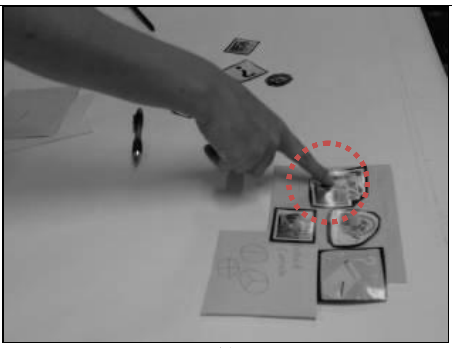
In the next paragraphs, the results were grouped into 4 items: (1) Results for the requirements elicitation, (2) Alternative design of interaction, (3) Pre-patterns for the system and, (4) Definition of project interaction.

(1) Results for the requirements elicitation. At the end of the experiment, ideas and issues were raised by participants. The technique of space-time representation associated to the experience prototyping have encouraged the participants to think of factors such as reuse and logical organization of content versus time (see Table 1). An amount of 32 valid ideas for the scenario 2 and 12 valid ideas for scenario 3 were raised. These ideas were transformed into requirements, and factors into pre-patterns explained in the following item. The full analysis is described in [8].

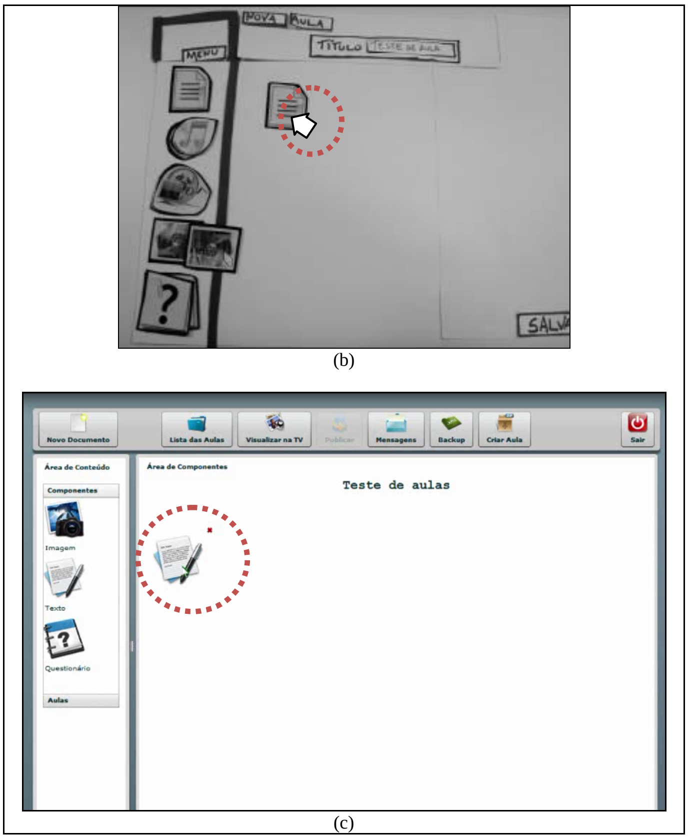
TABLE I. IDEAS AND FACTORS ARISING FROM THE PROTOTYPING EXPERIENCE

Results	**Experience prototyping (associated with the space-time representation, contextual interview and prototypes)**		
Title (centered)	***Scenario 1 (Classroom lessons)***	***Scenario 2 (Teacher vision: web application for the construction of content)***	***Scenario 3 (Student vision: for application in DTV)***
valid ideas	-	32	12
Factors	Reuse	Reuse, Collaboration, Sharing, Copyrights, Usability, Pedagogical	Collaboration, Accessibility, Usability

(2) Alternative design of interaction. An important contribution of this technique was the emergence of alternatives for interaction design suggested by the participants. For instance, the alternative considering a table for the organization of contents. The main screen of the web application would be like a table, and the builder objects of the content would be freely moved, enabling the content to be created on a virtual table, in terms of space and time, like the experience application. In Figure 5-a, the table where the experience was applied is shown, and Figure 5-b shows the alternative proposed (in a paper prototype) by the development team who participated in the experiment. The alternative suggests the User to be able to drag the media in order to build content on the main screen. The Figure 5-c shows the final version of the system. The creativity of the participants involved is originated mainly by the list of alternative solutions to a specific problem. Presenting and exploring multiple prototypes helps to better understand the capabilities of each alternative to meet the needs of users and the requirements of the system under development [2].



(a)



(b)

(c)

Fig. 5. Alternative of table

(3) Pre-patterns for the system. As a result of the experience and the requirements generated by it, new pre-patterns for the system were found out. The categories in which the pre-patterns were identified are:

A. Accessibility. Refers to the inclusion and extent of use by people with disabilities or limited mobility so that they can participate in activities that include the use of products, services and information;

B. Collaboration. Refers to the action developed jointly by two or more people who understand each other, by sharing values, services or products;

C. Copyright. Refers to names used in reference to the list of rights for authors of their intellectual works that can be literary, artistic or scientific;

D. Security. Refers to protecting the system from possible contingencies;

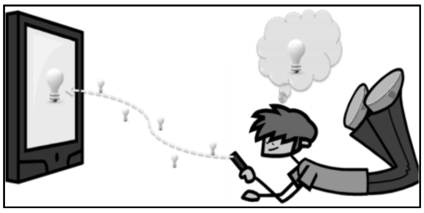
E. Usability. Refers to how easily the system or component can be operated by the User;

F. Reuse. Refers to using a product or file more than once;

G. Pedagogical. Refers to the educational context addressed in the system.

An example of a collaboration pattern created for the viewing on DTV is shown below:

B2: Collaborative participation of users of DTV content



Synopsis: Students can collaborate on the content by sending questions.
Context: Students may have questions or interesting ideas to contribute to the enrichment of the content. Therefore, when accessing the content on DTV, they can send their questions or suggestions for the system.
Problem: The student (User of DTV) wishes to send their questions and suggestions on the content published.
Solution: Enable the students to participate by sending questions and suggestions.
References: The system will provide users with an option for sending the questions about the content. These questions will be sent to the teacher responsible for the publication of that content.

Some pre-patterns have come from the first study [9] and others have arisen throughout the experiment, as discussed after the experience through brainstorming and raised through the questionnaire. Some were considered in the design of the interface and as non-functional requirements of the system.

After the definition of pre-patterns, it was made an association from which it can be contextualized in the web interface and in the DTV interface of the system.

(4) Definition of Interaction Design. The pre-patterns for DTV have helped to define interface patterns for the system. The accessibility pre-pattern enabled the legend design and the cooperation pre-pattern made it possible to send questions and suggestions from student to teacher.

6. Discussion

The purpose of the application of this methodology was to apply a more effective requirements elicitation and develop the interaction design in innovative systems. In this study, effectiveness refers to how useful the application of associated techniques was while applying the methodology in order to achieve the following objectives which will be discussed below.

Supporting the project stakeholders as for improving their understanding about the interactive possibilities of the system under development. Such objective was developed in the methodology by means of the following decisions: (a) the participants were both users and the system development team. The HCI area has directed its studies in order to increasingly include the user in the software development process. The importance of including also the team who will develop the system is highlighted. The involvement of both components is important for the project, once it not only provides a mutual learning, but also promotes interaction and understanding of both on the use of the system; (b) there was a competitor analysis and a presentation of results to stakeholders so that they could have an initial knowledge of the future system. This step was important for it provided the participants with an idea about the purpose of the future system and; (c) when moments of user experience were provided for those involved. At the time in which the participants reflect on the system within a context of an user experience, they understand the system better and facilitate their exploration of ideas.

An important point proving that this goal was achieved is the collection of alternatives for the interaction design generated by participants. When users or even the development team do not have enough understanding of the system, they do not even formulate alternatives.

The rich data collection from this experiment would have been insufficient if only techniques such as questionnaire, interview and brainstorming had been applied. Therefore, the application of the experience techniques provided all stakeholders with the opportunity to disclose their ideas and possible alternatives to the system.

Promoting an awareness process among stakeholders of the project regarding the contextual factors that influence the use of the system since the beginning of the interaction design. This objective was addressed in the methodology in the following moments: (a) During the application of the experience prototyping, when those involved were led to think of possible factors that would influence its use; (b) at the moment when these factors are considered for the development of requirements and pre-patterns for the system and; (c) at the time when the pre-patterns are applied in the interaction design of the system. When the participants of the experience were asked whether they had thought of any factor which could influence its use, only 3 of them answered yes and described factors such as reuse, collaboration and copyright. However, when some factors were suggested, the participants who had not thought of those so far would start to think of such factors. All of them (12) gave explanations about the factors, by analyzing them according to the system.

Still during the experience to prepare a lesson, the participants would use the media icons, asking whenever they did not understand any icon. When they had no representation of the media they wanted, they would write it on paper. They would look at the icons for the context of use, but only one participant used them in their experience. After a background alignment and sample of contextual factors that could influence the use of the system, other participants would begin to use them.

The contextual factors that influence the use were considered during the elaboration of requirements and pre-patterns for the system. From the 14 pre-patterns designed, 11 were applied in the interaction design of the system.

The pre-patterns were used for educating users as for the importance of contextual factors for the interaction of the system. The contextual factors have been raised and considered in the definition of requirements and pre-patterns and also considered in the system interface.

Providing better communication between stakeholders enabling a consensus as for the interactive alternatives of the system. At the beginning of the project, there were issues as for the communication and interaction between stakeholders, mainly referring to the vocabulary used. Education professionals would often express themselves by using unfamiliar terms for technology professionals, such as: "*preliminary analysis*", "*a priori analysis*", "*posteriori analysis*", "*Fedathi sequence*", "*didactic engineering*", "*epistemology*", among others. The development team would also use technical terms, often not understood by the educational team, such as: "*storyboard*", "*prototyping*", "*personas*" and other technical terms of the domain. The difficulty with communication would generate other interaction difficulties. Professionals from different backgrounds would feel intimidated when trying to consider topics unknown to them during the project meetings. Usually, the meetings did not allow a concrete result due to the number of factors to be considered and the difficulty in expressing them.

In this study, it was observed that the application of the methodology enabled a consensus of ideas between those involved and a clear definition of the system scope. The clearest example of improvement in the communication referred to the moment when participants explained their user experience, ideas and suggestions, which led to the perception that people would work in distinct ways. For instance, in the courses structure, teachers from the educational domain had concepts such as lessons- units-content, whereas teachers from the technological domain had the structure of courses as lessons-files. From these suggestions, alternatives were designed, and with improved communication it was possible to reach a consensus for the final choice. For this factor, a flexible structure for courses was desired, in a more free setting called label. The user would create a label and name it the way they want.

In order to sum up and give a final figure on how effective the result of applying a set of techniques for requirements elicitation and for interaction design, it is worth saying that these three techniques have helped on requirements elicitation and design interaction. The application of the brainstorming technique by itself would not have brought the intended results if an user experience had not been applied afterwards, for instance. The proposed methodology had the objective of organizing these techniques so that the benefit of each technique could be extracted from another one.

As a result of the application of the methodology, documents of requirements were obtained for both modules, definition of 13 pre-patterns and sketches of alternatives of design interaction

7. Conclusion and future works

In this paper, was presented the DIRCE methodology and its application in a case study of development of a system for creating educational content for digital TV from the web, as well as the results of applying the methodology and analysis of results.

The DIRCE approach focuses on the user experience, with a combination of the experience prototyping and space-time representation techniques, which together could provide an experience whose focus was on communication and exploration of ideas from those involved. The user experience, applied together with the techniques experience prototyping and space-time representation, was effective in providing experience of use for those involved. The use of a set of techniques for collecting requirements was important for capturing data from the user experience.

The focus of this work was not to research the users' needs, but the requirements elicitation from experiences of use. When preparing a design project, most designers in our study are based on qualitative methods such as ethnography, focus groups and informal interviews, as well as observations of everyday life. This research mainly focuses on the design and prototyping for interaction design and not on the depth investigation of user which often occurs long before design ideas emerge.

The contributions of this research are divided into methodological and products generated:

(1) methodological: (i) Exploration of the prototyping experience for identifying alternatives of interaction design; (ii) Demonstration of how contextual factors that influence the quality of use of systems can be used for the interaction design and for requirements elicitation; (iii) Step-by-step description of application of techniques of experience through a real example and with significant results; (iv) Presentation of various relations of the results with the techniques applied. These can be useful for providing an analysis of return on investment, for instance: pre-patterns vs. functional and non-functional requirements; pre-patterns vs. interface; factors vs. requirements; factors vs. pre-patterns; ideas considering different points of view vs. system functions; ideas vs. alternatives for interaction design; and

(2) Products generated: (i) Pre-patterns; (ii) Methodology; (iii) Generic Framework.

This work is concluded with the following question: How is it possible to achieve more efficient requirements? A possible response resulting from the application of this methodology would be: all stakeholders should participate, be reflective, creative and involved with experiences of use, through good communication.

As future work, more contextual factors that influence the use shall be analyzed, by continuing working with all potential stakeholders of the project (users and development team) and apply again the pre-patterns defined in other projects related to the creation of content for DTV.

In this work, the application of the technique experience prototyping was carried out individually, and data collection (brainstorming) occurred in a collective way, so that one participant would not influence the other participants throughout the process of experience of use, but still allowing them to interact and discuss during the process of experience of use, in which each participant would tell their experience to others. However, [6], in his work Prototyping Social Action, argues that the prototype is composed of human action rather than the technology that supports it. He believes it is important to understand how people interact with others while using a prototype, and how these interactions affect the manner in which individuals use the prototype.

Based on this context, it is intended to apply the DIRCE approach once more for studying factors of social interaction and on how they influence the quality of use. Indeed, television is a social technology and, therefore, it could be interesting to analyze the interaction of people together.

8. References

[1] Buchenau, M., Suri, J. F., “Experience Prototyping”, Proceedings of DIS00: Designing Interactive Systems: Processes, Practices, Methods, & Techniques, Brooklyn, New York, 2000. pp. 424-433.

[2] Blind Review.

[3] Chung, E. S., Hong, J. I., Lin, J., Prabaker, M. K., Landay, J. A., Liu, A. L.. “Development and Evaluation of Emerging Design Patterns for Ubiquitous Computing”, Proceedings of Designing Interactive Systems, Cambridge, Massachusetts, USA, 2004, pp. 233-242.

[4] Damore, B. Epistemologia, Didática da Matemática e Práticas de Ensino. In: Bolema, v. 20, n. 28, 2013. Available <www.dm.unibo.it/rsddm/it/articoli/damore>.

[5] Dow, S., Saponas, T. S., Landay, Li, Y., Landay, J. A., “External representations in ubiquitous computing design and the implications for design tools”, Proceedings of DIS06, 2006. pp. 241-250.

[6] Kurvinen, E. Prototyping Social Action. Gummerus Printing. Printed in Vaajakoski, Finland 2007

[7] Löwgren, J., Stolterman, E., “Thoughtful interaction design: A design perspective on information technology”, MIT Press, Cambridge, Massachusetts, USA, 2004.

[8] Blind Review.

[9] Blind Review.

[10] Preece, J., Rogers, Y., Sharp, H. “Interaction design: Beyond human-computer interaction”, John Wiley & Sons, New York, NY, 2002.

[11] Saponas, T. S., Prabaker, M. K., Abowd, G. D., Landay, J. A., “The impact of pre-patterns on the design of digital home applications”, Proceedings of the 6th ACM conference on Designing Interactive systems, University Park, Pennsylvania, USA, 2006, pp. 189- 198.

[12] Blind Review.

[13] Sommerville, I., “Software engineering”, Addison Wesley Longman Publishing Co., Redwood City, CA, USA, 2003.

[14] Ole N., Dybkjaer. Multimodal Usability. Springer.2009.

Probabilistic Alias Analysis for Parallel Programming in SSA Forms

Mohamed A. El-Zawawy[1] **and Mohammad N. Alanazi**[2]
[1,2]College of Computer and Information Sciences,
Al Imam Mohammad Ibn Saud Islamic University (IMSIU)
Riyadh, Kingdom of Saudi Arabia
[1]Department of Mathematics, Faculty of Science
Cairo University
Giza 12613, Egypt
Email[1]: maelzawawy@cu.edu.eg
Email[2]: alanazi@ccis.imamu.edu.sa

Abstract—*Static alias analysis of different type of programming languages has been drawing researcher attention. However most of the results of existing techniques for alias analysis are not precise enough compared to needs of modern compilers. Probabilistic versions of these results, in which result elements are associated with occurrence probabilities, are required in optimizations techniques of modern compilers.*

This paper presents a new probabilistic approach for alias analysis of parallel programs. The treated parallelism model is that of SPMD where in SPMD, a program is executed using a fixed number of program threads running on distributed machines on different data. The analyzed programs are assumed to be in the static single assignment (SSA) form which is a program representation form facilitating program analysis. The proposed technique has the form of simply-strictured system of inference rules. This enables using the system in applications like Proof-Carrying Code (PPC) which is a general technique for proving the safety characteristics of modern programs.

Keywords: Probabilistic Analysis, Alias Analysis, Parallel Programming, SSA Forms.

1. Introduction

Considerable efforts of research have been devoted to achieve the static alias analysis of different type of programming languages. Algorithms for calculating alias relationships for all program points exist for basic programming techniques. Classically, alias relationships fall in two groups: definitely-alias relationships and possibly-alias relationships. The former is typically true for all possible execution paths and the later might typically be true for some of the possible execution paths. However the information calculated by most existing algorithms for alias analysis is not precise enough compared to the needs of modern compilers. This is so as modern compilers need finer alias-information to be able to achieve tasks like code specialization and data speculation. In other words information calculated by most alias analysis techniques do not help compilers to do aggressive optimizations. More specifically, possibly-alias relationships is not rich enough to inform the comfier about the possibility that constraints for the executions. Hence compilers are somehow forced to follow a conservative way and assume the conditions validity for all execution paths [1], [2], [3].

A dominant programming technique of parallelism for large-scale machines equipped with distributed-memories is the single program, multiple data (SPMD) model. In SPMD, a program is executed using a fixed number of program threads running on distributed machines on different data [4]. SPMD can be executed on low-overhead and simple dynamic systems and is convenient for expressing parallelism concepts. This parallelism model is used by message-sending architectures such as MPI. SPMD is also adapted by languages whose address spaces are globally partitioned (PGAS) such as UPC, Co-Array Fortran, and Titanium. Specific deadlocks can be prevented using the SPDM model which can also be used to achieve probabilistic data races and specific program optimizations [5], [6].

Static single assignment (SSA) [7], [8] is a program representation form facilitating program analysis. SSA forms are important for software re-engineering and compiler construction. Program analysis needs data-flow information about points of the program being analyzed. Such information is necessary for program compilation and re-engineering and is conveniently collected by SSA. For program variables, some analyses need to know assignment statements that could have assigned the used variable content. In Static single assignment (SSA) form exactly one variable definition corresponds to a variable use. This is only possible if the algorithm building the SSA form is allowed to insert auxiliary definitions if it is possible for different definitions to get into a specific program point.

A general technique for proving the safety characteristics of modern programs is Proof-Carrying Code (PCC) [9], [10]. PCC proofs are needed and typically constructed using logics annotated with inference rules that are language-specific. The proofs ensures safety in case there are no bugs in the inference rules. One type of Proof-Carrying Code is Foundational Proof-Carrying Code (FPCC) which uses theories of mathematical logic. The small trusted base of FPCCs and the fact that they are not tied to any specific systems make them more secure and robust.

This paper presents a new technique for probabilistic alias analysis of parallel programs. The technique has the form of simply-strictured system of inference rules. The information calculated by the proposed technique are precise enough compared to information needed by modern compilers for compilations, re-engineering, aggressive-optimization processes. The proposed technique is designed to work on the common and robust data-flow representation; SSA forms of parallel programs. The use of inference systems in the proposed technique makes it straightforward for our technique to produce justifications needed by Foundational Proof-Carrying Code (FPCCs). The proofs have the form of inference rules derivations that are efficiently transferable. The parallelism model treated in this paper is that of single program, multiple data (SPMD) in which the same program is executed on different machines on different sets of data.

Motivation

The paper is motivated by need for a precise probabilistic alias analysis for SPMD programs running on a hierarchy of distributed machines. The required technique is supposed to associate each analysis result with a correctness proof (in the form of type derivations) to be used in proof-carrying code applications.

Contributions

The contribution of the paper is a new approach for probabilistic alias analysis of SPMD programs running on SSA forms of programs and producing justifications with analysis results.

Paper Outline

The outline of this paper is as follows. Section 2 presents the langauge model, *SSA-DisLang*, of the paper. This section also presents an informal semantics to the langauge constructs. The main content of Section 2 is the new technique of the probabilistic alias analysis of SPMD programs. Section 4 concludes the paper and suggests directions for future work.

2. Probabilistic Alias analysis for SPMD

This section presents a new technique for probabilistic alias analysis of parallel programs. The parallelism model used here is that of SPMD where the same program is executed on distrusted machines havering different data. However communications between the distributed machines is allowed in a predefined contexts. For example a command running on machine 1 may request machine 3 to evaluate a specific expression using data of machine 3 and to return the result to machine 1.

The syntax of the langauge used to present the new probabilistic alias analysis technique is shown in Figure 1. We call the langauge mode *SSA-DisLang* for ease of reference. A program in *SSA-DisLang* consists of a sequence of statements where statements are of wide diversity. Statements use (distributed) expressions, *DExpr*. The machines to run *SSA-DisLang* programs are typically organized in a hierarchy. The distributed expressions include the following:

- malloc() : allocates a dynamic array in memory and return its base address.
- run (e, m) : evaluates the distributed expression e on machine m and return the result.
- reform(alis m, int m) e : casts the location denoted by the distributed expression e as an integer rather than a pointer to a memory location on machine m.
- reform (int m_j, int m_i)) e : casts the location denoted by the distributed expression e as a pointer a memory location on machine m_i rather than as a pointer to a memory location on machine m_j.

Our proposed technique assumes that the given program, that is to be be analyzed for its probabilistic alias competent, has the static single assignments form. Therefore the input program would contain annotations (added by any efficient SSA such as algorithm [11]). The program annotations will have the form of new statements added to the original program. Therefore statements of *SSA-DisLang* include the following:

- $l := e$: this is a classical assignment command. However the design of the langauge allows using this command to assign a value evaluated at a machine to a location on a different machine of the machines hierarchy.
- run (S, m) : allows evaluates a specific command S on a specific machine m regardless of the executing machine. This command is necessary when some commands are convenient only to run on data of certain machine of the hierarchy. The command is also used when security is a concern as S would not have access to all machines.
- $x_i := f(x_j, x_k)$: this command is to be added by the supposed SSA algorithm and it semantics is that variable x_i were created specifically for avoiding multiple assignments to variable x. The range of this definition if form definition of variable x_j to that of variable x_k.
- $x_i := md(x_j)$: this is the second sort of annotations *SSA-DisLang* programs. The semantics of this statement is that it is highly likely that variable x_i is used to define variable x_j. Recall that our main technique of the paper cares about possibility of assignments to occur in

$$\begin{aligned}
& && x \in \text{lVar},\ i_{op} \in I_{op},\ b_{op} \in B_{op},\ \text{and } m \in M \subseteq \mathcal{M} \\
l \in \text{Loc} &\ ::= && x \mid l \to y \mid [l]. \\
e \in \text{DExpr} &\ ::= && l \mid e_1\ i_{op}\ e_2 \mid \&l \mid \text{malloc()} \mid \text{run}\ (e, m) \mid \\
& && \text{reform(alis } m, \text{int } m)\ e \mid \text{reform (int } m_j, \text{int } m_i))\ e. \\
S \in \text{Stmts} &\ ::= && l := e \mid \text{run}\ (S, m) \mid S_1; S_2 \mid x_i := f(x_j, x_k) \mid x_i := md(x_j) \mid \\
& && mu(x_j) \mid \text{if } e \text{ then } S_t \text{ else } S_f \mid \text{while } e \text{ do } S_t.
\end{aligned}$$

Fig. 1: Programming Language Model; *SSA-DisLang*

$$\frac{P(x) = \{(a_1, p_1), \ldots, (a_n, p_n)\} \qquad i = \max(p_1, \ldots, p_n)}{x : P \to_l a_i}\ (\text{x}^p)$$

$$\frac{\begin{array}{l} P(l) = \{(a_1, p_1), \ldots, (a_n, p_n)\} \qquad P(y) = \{(b_1, q_1), \ldots, (b_m, q_m)\} \\ i = \max\{p_j \times q_j \mid a_j \in P(y)\rceil_1\} \end{array}}{(l \to y) : P \to_l b_i}\ (\to^p)$$

$$\frac{\begin{array}{l} P(l) = \{(a_1, p_1), \ldots, (a_n, p_n)\} \qquad foralli.P(a_i) = \{(b^i_1, q^i_1), \ldots, (b^i_m, q^i_m)\} \\ i = \max\{p_i \times q^i_j \mid 1 \leq i \leq n\&1 \leq j \leq m\} \end{array}}{[l] : P \to_l b_i}\ ([l]^p)$$

Fig. 2: Probabilistic Alias Analysis (PAA): Locations.

$$\frac{e : P \to a_e \qquad \text{probability of arriving at this memory point} \geq p_t}{\text{reform(alis } m \to \text{int } m)\ e : P \to_l a_e}\ (\text{reform}^p_1)$$

$$\frac{\text{probability of arriving at this memory point} < p_t}{\text{reform(alis } m \to \text{int } m)\ e : P \to_l \bot}\ (\text{reform}^p_2)$$

$$\frac{e : P \to a_e \qquad \text{probability of arriving at this memory point} \geq p_t}{\text{reform (int } m_j \to \text{int } m_i))\ e : P \to_l a_e}\ (\text{reform}^p_3)$$

$$\frac{\text{probability of arriving at this memory point} < p_t}{\text{reform (int } m_j \to \text{int } m_i))\ e : P \to_l \bot}\ (\text{reform}^p_4)$$

$$\frac{e : P \to a_e \qquad b = \text{reform(\_, int } m)a_e}{\text{run}\ (e, m) : P \to_l b}\ (\text{run}^p_e)$$

$$\frac{a_i \text{ is a fresh memory location on machine } m_i}{\text{malloc()} : P \to_l a_i}\ (\text{malloc})^p$$

$$\frac{e_1 : P \to a_{e_1} \qquad e_2 : P \to a_{e_2}}{e_1\ i_{op}\ e_2 : P \to_l a_{e_1} + a_{e_2}}\ (+^p)$$

Fig. 3: Probabilistic Alias Analysis (PAA): Distributed Expressions.

percentages; it is not a $0/1$ technique.

- $mu(x_j)$: this is the third and last sort of annotations *SSA-DisLang* programs. The semantics of this command is that variable x_j is highly likely to be used in the following de-reference command of the program.

Figures 2, 3, and 4 present elements of our proposed technique for probabilistic alias analysis of *SSA-DisLang* programs. The proposed technique has the form of a type system which consists of set of alias types denoted by P and set of inference rules presented in the technique figures. An alias type is a *partial* map. The domain of this partial map is a subset of the set of all variables (denoting registers) allowed to be used on different machines of the distributed hierarchy plus the set of all addresses of memories of machines on hierarchy. The codomain of the alias type is the power set of the set of all *probabilistic pairs*. A probabilistic pair is a pair of variable (register) or a memory location and a number p such hat $0 \leq p \leq 1$.

Judgment produced by the system have the forms $e : P \rightarrow a$ and $S : P \rightarrow P'$. The judgement $e : P \rightarrow a$ means that evaluating the expression e in a memory state of the type P results in the memory address a. The semantics of the judgement $e : P \rightarrow a$ is that running S in a memory state of the type P results (if ends) in a memory state of the type P'. The proposed technique is meant to be used as follows. given a distributed program S, one constructs (using inference rules of the system) an alias type P' such that $S : \perp \rightarrow P'$. The base type is the partial map with an empty domain is denoted by $\perp$. The construction of P' is a type derivation process and results in annotating program with the required probabilistic alias information.

Inference rules for distributed expressions are shown in figure 3. Some comments on the rules are in order. The rules for *reform* expressions only considers the address evaluated from e if there is a considerable probability (probability threshold $> p_{th}$) of arriving at the concerned program point.

Inference rules for statements are shown in figure 4. Some comments on the rules are in order. The rule $([]_3^p)$ uses the base address of the array denoted by l and the address returned for e by the inference rules of expressions. The image of these addresses under the pre-type also contribute to calculating the post type of the de-reference statement.

The soundness of our proposed technique is guarantied by the following theorem. The theorem requests the existence of robust operational semantics for the langauge *SSA-DisLang*. Many semantics candidates exist. Due to lack of space we only reference to the semantics in this paper. From the authors's experience and based on some experiments, the simplicity of the theorem proof deeply relies on the choice of the langauge semantics.

Theorem 1: Suppose that S is a *SSA-DisLang* program and $S : \perp \rightarrow P'$. Suppose also that using a convenient operational semantics for *SSA-DisLang*, the execution of S is captured as $S : M \rightarrow M'$. Then the final memory state M' is of the the probabilistic alias type P'.

3. Related Work

The changing associations characteristics property of pointers makes the points-to analysis a complicated problem [3]. Much research [12], [13], [14], [15] have been developed to solve the pointer analysis problem. Each of these techniques evaluates either points-to or aliases relationships at program points. Points-to and aliases relationships are classified into two classes: definitely-aliases (or must-points-to) relationships and may-points-to (or possibly-aliases relationships). While the later relationships are true on some executions, the former relationships are true on all executions. Wether possibly-aliases or may-points-to relationships are true on most executions or on few executions is not measurable by most of these techniques. For specific transformations and optimizations these missed information are beneficial. Few attempts were made to fill this gap.

Using traditional data-flow analysis, in [16], [17] a theoretical formulation is presented to compute measurable information. More specifically, this work evaluates, for each program point, the predicted count that specific conditions may hold. Aiming at evaluating, among array references, the probabilities of aliases, [18], [19], [20] presents a probabilistic technique for memory disambiguation. A probabilistic, interprocedural, contextsensitive, and flow-sensitive techniques for alias analysis were proposed in [3], [11], [21]. On alias relationships, these technique evaluate measurable information. MachSUIF and SUIF compiler infrastructures provided the bases for the implementation of these techniques. The probabilities of pointer induced, loop carried, and data dependence relationships were evaluated in [22], [23]. Using sparse matrices, as efficient linear transfer functions, [24], [25] modeled probabilistic alias analysis. The results of this research were proved accurate. [26] presents an algorithm to evaluate measurable alias information. A technique for memory disambiguation, evolution of probabilities that pairs of memory pointers point at the same memory location, is presented in [26].

For array optimizations and analysis, probabilistic techniques for memory disambiguation were proposed [18]. These techniques typically present data speculations [27] necessary for modern architectures of computers.

For distributed parallel machines with shared-memory, an important problem is that of compiler optimizations for programs that are pointer-based. This is so as the host processor of an object can be determined using data distribution analysis [28] and affinity analysis [29].

In, pointer-based programs, a reference is referencing a group of objects with may-points-to. For such cases, traditional affinity analysis [30] can be integrated with traditional pointer analysis. The result of this integration is a technique that evaluates the parts of objects on a processor's list

$$\frac{P(x_j) = (x_i, a) \qquad P(x_i) = (x_k, \_)}{x_i := md(x_j) : P \rightarrow_s P[x_i \mapsto \{(x_k, a), (x_j, 1-a)\}]}\ (md^p)$$

$$\frac{p_j(p_k) \text{ is the probability of executing the path from definition of } x_j(x_k) \text{ to that of } x_i}{x_i := fi(x_j, x_k) : P \rightarrow_s P[x_i \mapsto \{(x_j, p_j), (x_k, p_k)\}]}\ (fi^p)$$

$$\frac{a \text{ is the base address of array } l}{x := \&l : P \rightarrow_s P[x \mapsto \{(a, 1)\}]}\ (\&_1) \qquad \frac{}{mu(x_j) : P \rightarrow_s P}\ (mu^p)$$

$$\frac{\begin{array}{l} a_1, a_2 \text{ are the base addresses of arrays } l_1 \text{ and } l_2 \\ i \text{ is the index of } y \text{ in } l_2 \end{array}}{l_1 \rightarrow y := \&l_2 : P \rightarrow_s P[a_1 \mapsto \{(a_2 + i, 1)\}]}\ (\&_2^p)$$

$$\frac{\begin{array}{l} a_1, a_2 \text{ are the base addresses of arrays } l_1 \text{ and } l_2 \\ P(a_1) = \{(b_1, p_1), \ldots, (b_n, p_n)\} \end{array}}{[l_1] := \&l_2 : P \rightarrow_s P[b_1 \mapsto \{(a_2, p_1)\}, \ldots, b_n \mapsto \{(a_2, p_n)\}]}\ (\&_3^p)$$

$$\frac{\begin{array}{ll} l \neq [\ldots] & e : P \rightarrow a_e \\ a_l \text{ is the base addresses of array } l & P(a_e) = \{(b_1, p_1), \ldots, (b_n, p_n)\} \end{array}}{l := [e] : P \rightarrow_s P[a_l \mapsto \{(b_1, p_1), \ldots, (b_n, p_n)\}]}\ ([]_1^p)$$

$$\frac{\begin{array}{ll} e \neq [\ldots] & \\ a_l \text{ is the base addresses of array } l & P(a_e) = \{(b_1, p_1), \ldots, (b_n, p_n)\} \\ e : P \rightarrow a_e & P(a_l) = \{(c_1, q_1), \ldots, (c_m, q_m)\} \end{array}}{[l] := e : P \rightarrow_s P[c_i \mapsto \{(b_1, \min(p_1, q_1)), \ldots, (b_n, \min(p_n, q_n))\} \mid 1 \leq i \leq m]}\ ([]_2^p)$$

$$\frac{\begin{array}{ll} & P(a_e) = \{(b_1, p_1), \ldots, (b_n, p_n)\} \\ a_l \text{ is the base addresses of array } l & \forall i.p(b_i) = \{(d_1^i, t_1^i), \ldots, (q_k^i, t_k^i)\} \\ e : P \rightarrow a_e & P(a_l) = \{(c_1, q_1), \ldots, (c_m, q_m)\} \end{array}}{[l] := [e] : P \rightarrow_s P[c_i \mapsto \{(d_1^i, \min(p_1, q_1 \times t_1^i)), \ldots, (q_k^i, \min(p_n, q_n \times t_k^i))\} \mid 1 \leq i \leq m]}\ ([]_3^p)$$

$$\frac{\begin{array}{ll} e \neq [\ldots] \quad l \neq [\ldots] \quad e \neq \&\ldots & e : P \rightarrow a_e \\ a_l \text{ is the base addresses of array } l & P(a_e) = \{(b_1, p_1), \ldots, (b_n, p_n)\} \end{array}}{l := e : P \rightarrow_s P[a_l \mapsto \{(b_1, p_1), \ldots, (b_n, p_n)\}]}\ (:=^p)$$

$$\frac{\begin{array}{c} S_1 : P \rightarrow_s P'' \\ S_2 : P'' \rightarrow_s P' \end{array}}{S_1; S_2 : P \rightarrow_s P'}\ (;^p) \qquad \frac{S : P \rightarrow_s P'}{\text{run } (S, m) : P \rightarrow_s P'}\ (\text{run}_s^p)$$

$$\frac{S_t : P \rightarrow_s P_t \qquad S_f : P_f \rightarrow_s P'}{\text{if } e \text{ then } S_t \text{ else } S_f : P \rightarrow_s P_t \biguplus P_f}\ (\text{if}^p)$$

$$\frac{n \text{ is the exepcted execution time of } S_t \qquad S_t : P \rightarrow_s P_t}{\text{while } e \text{ do } S_t : P \rightarrow_s \biguplus_n P_t}\ (\text{whl}^p)$$

Fig. 4: Probabilistic Alias Analysis (PAA): Statements.

of task executions. This is necessary for many program optimizations.

There are many examples of aggressive optimizations such as data speculations, speculative multithreading (thread partitioning), and code specialization [31], [32]. To boost performance of modern architectures, these optimizations are typically achieved by compilers. Compilers can only do such tasks if they are able to measure the possibility of dynamic pointer associations. Using interval analysis, irreducible flow graphs, and the elimination technique, intraprocedural analysis can be used to handle pointer analysis of programs [33]. Extensions to such techniques to cover context-sensitive analysis that is interprocedural is achievable as well.

Examples of analysis for speculative multithreading model include thread partitioning [34], [35], [36]. Such analysis boosts compilers performance via running speculative threads in case of low possibilities for conflicts. In this scenario for threads with high possibilities are turned off [22].

4. Conclusion and Future Work

This paper presented a new technique for probabilistic alias analysis of SPMD programs. The new approach has the form of system of inference rules. This has direct applications in proof-carrying code area of research. The proposed technique also has the advantage of assuming SSA forms of analyzed programs.

Directions for future work include the following. Producing probabilistic techniques for important analyses (such as dead-code elimination) for SPMD programs that uses the results of the analysis proposed in this paper would be an important contribution. Producing other analyses for the langauge model of this paper in the spirit of [37], [38], [39] is another direction for future work. There is also a need for precise probabilistic operational semantics for SPMD programs. This semantics would be important to accurately measure probabilities of statements executions and probabilities of executions order.

5. Acknowledgment

The authors acknowledge the support (grants numbers 340918 & 330911) of the deanship of scientific research of Al Imam Mohammad Ibn Saud Islamic University (IMSIU).

References

[1] U. P. Khedker, A. Mycroft, and P. S. Rawat, "Liveness-based pointer analysis," in *SAS*, ser. Lecture Notes in Computer Science, A. Miné and D. Schmidt, Eds., vol. 7460. Springer, 2012, pp. 265–282.

[2] M. A. El-Zawawy, "Flow sensitive-insensitive pointer analysis based memory safety for multithreaded programs," in *ICCSA (5)*, ser. Lecture Notes in Computer Science, B. Murgante, O. Gervasi, A. Iglesias, D. Taniar, and B. O. Apduhan, Eds., vol. 6786. Springer, 2011, pp. 355–369.

[3] P.-S. Chen, Y.-S. Hwang, R. D.-C. Ju, and J. K. Lee, "Interprocedural probabilistic pointer analysis," *IEEE Trans. Parallel Distrib. Syst.*, vol. 15, no. 10, pp. 893–907, 2004.

[4] P. Pacheco, *An Introduction to Parallel Programming*. Elsevier, 2011, 1 edition (2011).

[5] H. Li, G. Fox, G. von Laszewski, and A. Chauhan, "Co-processing spmd computation on cpus and gpus cluster," in *CLUSTER*. IEEE, 2013, pp. 1–10.

[6] M. Tsuji, M. Sato, M. R. Hugues, and S. G. Petiton, "Multiple-spmd programming environment based on pgas and workflow toward post-petascale computing," in *ICPP*. IEEE, 2013, pp. 480–485.

[7] W. Amme, T. S. Heinze, and J. von Ronne, "Intermediate representations of mobile code," *Informatica (Slovenia)*, vol. 32, no. 1, pp. 1–25, 2008.

[8] W. Amme, N. Dalton, M. Franz, and J. von Ronne, "Safetsa: A type safe and referentially secure mobile-code representation based on static single assignment form," in *PLDI*, M. Burke and M. L. Soffa, Eds. ACM, 2001, pp. 137–147.

[9] F. Pfenning, L. Caires, and B. Toninho, "Proof-carrying code in a session-typed process calculus," in *CPP*, ser. Lecture Notes in Computer Science, J.-P. Jouannaud and Z. Shao, Eds., vol. 7086. Springer, 2011, pp. 21–36.

[10] R. Jobredeaux, H. Herencia-Zapana, N. A. Neogi, and E. Feron, "Developing proof carrying code to formally assure termination in fault tolerant distributed controls systems," in *CDC*. IEEE, 2012, pp. 1816–1821.

[11] M.-Y. Hung, P.-S. Chen, Y.-S. Hwang, R. D.-C. Ju, and J. K. Lee, "Support of probabilistic pointer analysis in the ssa form," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 12, pp. 2366–2379, 2012.

[12] Y. Ben-Asher and N. Rotem, "Using memory profile analysis for automatic synthesis of pointers code," *ACM Trans. Embedded Comput. Syst.*, vol. 12, no. 3, p. 68, 2013.

[13] S. Staiger-Stöhr, "Practical integrated analysis of pointers, dataflow and control flow," *ACM Trans. Program. Lang. Syst.*, vol. 35, no. 1, p. 5, 2013.

[14] L. Li, C. Cifuentes, and N. Keynes, "Precise and scalable context-sensitive pointer analysis via value flow graph," in *ISMM*, P. Cheng and E. Petrank, Eds. ACM, 2013, pp. 85–96.

[15] B. Huang, X. Ling, and G. Wu, "Field-sensitive function pointer analysis using field propagation for state graph extraction," *JSW*, vol. 8, no. 7, pp. 1592–1603, 2013.

[16] Q. Shao, Y. D. Chen, and L. Zhang, "An extension of three-parameter burr iii distribution for low-flow frequency analysis," *Computational Statistics & Data Analysis*, vol. 52, no. 3, pp. 1304–1314, 2008.

[17] F. Miwakeichi, P. A. Valdes-Sosa, E. Aubert-Vazquez, J. B. Bayard, J. Watanabe, H. Mizuhara, and Y. Yamaguchi, "Decomposing eeg data into space-time-frequency components using parallel factor analysis and its relation with cerebral blood flow," in *ICONIP (1)*, ser. Lecture Notes in Computer Science, M. Ishikawa, K. Doya, H. Miyamoto, and T. Yamakawa, Eds., vol. 4984. Springer, 2007, pp. 802–810.

[18] R. D.-C. Ju, J.-F. Collard, and K. Oukbir, "Probabilistic memory disambiguation and its application to data speculation," *SIGARCH Computer Architecture News*, vol. 27, no. 1, pp. 27–30, 1999.

[19] B. Guo, Y. Wu, C. Wang, M. J. Bridges, G. Ottoni, N. Vachharajani, J. Chang, and D. I. August, "Selective runtime memory disambiguation in a dynamic binary translator," in *CC*, ser. Lecture Notes in Computer Science, A. Mycroft and A. Zeller, Eds., vol. 3923. Springer, 2006, pp. 65–79.

[20] C. Fang, S. Carr, S. Önder, and Z. Wang, "Feedback-directed memory disambiguation through store distance analysis," in *ICS*, G. K. Egan and Y. Muraoka, Eds. ACM, 2006, pp. 278–287.

[21] A. D. Pierro, C. Hankin, and H. Wiklicky, "A systematic approach to probabilistic pointer analysis," in *APLAS*, ser. Lecture Notes in Computer Science, Z. Shao, Ed., vol. 4807. Springer, 2007, pp. 335–350.

[22] P.-S. Chen, M.-Y. Hung, Y.-S. Hwang, R. D.-C. Ju, and J. K. Lee, "Compiler support for speculative multithreading architecture with probabilistic points-to analysis," in *PPOPP*, R. Eigenmann and M. C. Rinard, Eds. ACM, 2003, pp. 25–36.

[23] A. Zhai, J. G. Steffan, C. B. Colohan, and T. C. Mowry, "Compiler and hardware support for reducing the synchronization of speculative threads," *TACO*, vol. 5, no. 1, 2008.

[24] J. D. Silva and J. G. Steffan, "A probabilistic pointer analysis for

speculative optimizations," in *ASPLOS*, J. P. Shen and M. Martonosi, Eds. ACM, 2006, pp. 416–425.

[25] S. Roy and Y. N. Srikant, "The hot path ssa form: Extending the static single assignment form for speculative optimizations," in *CC*, ser. Lecture Notes in Computer Science, R. Gupta, Ed., vol. 6011. Springer, 2010, pp. 304–323.

[26] Y.-M. Lu and P.-S. Chen, "Probabilistic alias analysis of executable code," *International Journal of Parallel Programming*, vol. 39, no. 6, pp. 663–693, 2011.

[27] L. Xiang and M. L. Scott, "Compiler aided manual speculation for high performance concurrent data structures," in *PPOPP*, A. Nicolau, X. Shen, S. P. Amarasinghe, and R. Vuduc, Eds. ACM, 2013, pp. 47–56.

[28] J. K. Lee, D. Ho, and Y. C. Chuang, "Data distribution analysis and optimization for pointer-based distributed programs," in *ICPP*. IEEE Computer Society, 1997, pp. 56–63.

[29] M. C. Carlisle and A. Rogers, "Software caching and computation migration in olden," *J. Parallel Distrib. Comput.*, vol. 38, no. 2, pp. 248–255, 1996.

[30] T. Pitkäranta, "Affinity analysis of coded data sets," in *EDBT/ICDT Workshops*, ser. ACM International Conference Proceeding Series, M. Mesiti, T. M. Truta, L. Xiong, S. Müller, H. Naacke, B. Novikov, G. Raschia, I. Sanz, P. Sens, D. Shaporenkov, and N. Travers, Eds., vol. 360. ACM, 2009, pp. 177–184.

[31] R. Zhang, S. Debray, and R. T. Snodgrass, "Micro-specialization: dynamic code specialization of database management systems," in *CGO*, C. Eidt, A. M. Holler, U. Srinivasan, and S. P. Amarasinghe, Eds. ACM, 2012, pp. 63–73.

[32] M. A. Khan, "Feedback-directed specialization of code," *Computer Languages, Systems & Structures*, vol. 36, no. 1, pp. 2–15, 2010.

[33] Q. Sun, J. Zhao, and Y. Chen, "Probabilistic points-to analysis for java," in *CC*, ser. Lecture Notes in Computer Science, J. Knoop, Ed., vol. 6601. Springer, 2011, pp. 62–81.

[34] B. Liu, Y. Zhao, Y. Li, Y. Sun, and B. Feng, "A thread partitioning approach for speculative multithreading," *The Journal of Supercomputing*, vol. 67, no. 3, pp. 778–805, 2014.

[35] Y. Li, Y. Zhao, P. Yin, and Y. Du, "Speculative thread partitioning using fuzzy c-means clustering," in *CSE*, W. Qu, K. Lin, Y. Shen, W. Shi, D. F. Hsu, X. Jin, F. C. M. Lau, and J. Xu, Eds. IEEE, 2011, pp. 199–206.

[36] W.-J. Kim, K. Cho, and K.-S. Chung, "Multi-threaded syntax element partitioning for parallel entropy decoding," *IEEE Trans. Consumer Electronics*, vol. 57, no. 2, pp. 897–905, 2011.

[37] M. A. El-Zawawy and H. A. Nayel, "Type systems based data race detector," *IJCSNS International Journal of Computer Science and Network Security*, vol. 5, no. 4, pp. 53–60, July 2012.

[38] M. A. El-Zawawy and N. M. Daoud, "New error-recovery techniques for faulty-calls of functions," *Computer and Information Science*, vol. 5, no. 3, pp. 67–75, May 2012.

[39] M. A. El-Zawawy and H. A. Nayel, "Partial redundancy elimination for multi-threaded programs," *IJCSNS International Journal of Computer Science and Network Security*, vol. 11, no. 10, pp. 127–133, October 2011.

ExpTool: a Tool to Conduct, Package and Replicate Controlled Experiments in Software Engineering

Joao Pucci Neto, Lilian Passos Scatalon,
Rogerio Eduardo Garcia, Ronaldo Celso Messias Correia, Celso Olivete Junior
Department of Mathematics and Computer Science
Faculty of Science and Technology
University State Paulista “Julio de Mesquita Filho”
Street Roberto Simonsen, 305 - CEP 19060-900 Presidente Prudente - SP, Brazil
E-mail: puccineto,lilian.scatalon@gmail.com, rogerio,ronaldo,olivete@fct.unesp.br

***Abstract*—Running multiples experiments in Software Engineering introduces the need of recording data as well as transferring knowledge across them, especially considering that several researchers are involved on replicating experiments. For that, experimental evaluations generate knowledge that must be registered into a so-called lab package. Researches have reported difficulties on sharing lab packages due to lack of standardization. In this paper we present a tool to support experimenters to conduct controlled experiments, packaging experimental data. In this first version, experimental data are kept into XML file organized based on an ontology proposed to controlled experiment. The tool support aims at organizing lab packages focusing on facilitating creating and sharing lab packages.**

Keywords: Controlled Experiment, Lab Package, Experimental Software Engineering.

I. Introduction

Controlled experiments can be used to test and validate new methods, techniques, languages and tools. Only the results of isolated experiments are not sufficient - they are no enough to be considered trustworthy. The results of experimental studies can only be considered if they are consolidated into a significant body of knowledge for the community that operates in the area. Therefore, it is needed to replicate experimental studies to generate data and compose the body of knowledge [1]. According to [2], the execution of experiment in different contexts allows verifying if conclusions are valid for a wider population. And if a replication present different result, it is possible to analyze the reasons why such difference was obtained [3].

A replication by other experimenters requires reviewing the information from the original experiment in order to understand how it was designed, conducted and analyzed [4]. So, the information about original study must be available for that. All information (process, artifacts, procedures, results and conclusions) are stored into a so-called lab package [5]. Shull cites that researchers have faced some problems on interpreting and understanding lab packages, and consequently, transferring of knowledge between research groups. In addition, Mendonca [6] proposed a framework (FIRE - Framework for Improving the Replication of Experiments) that suggests the sharing of knowledge generated intra and inter groups, but they do not suggest how to organize lab packages. In a previous paper, Garcia [7] proposed an ontology to organized data from controlled experiments. In this context, the aim of this paper is to present a computational tool called ExpTool to assist the researcher in conducting a controlled experiment based on experimental process proposed by Wohlin [2] as well as the creation of a lab package using the ${}_{Exper}Ontology$. Using the ExpTool the lab package is created: a XML file containing all data from the study is generated according to ${}_{Exper}Ontology$. The tool maps a sequence of activities in a workflow that allows the experimenter to define their experiment and artifacts used.

In order to present the proposed tool, the paper is organized in the following: in Section 2 is discussed the importance of using controlled experiments, the packaging of data and the importance of knowledge transfer through replication of experiments; in Section 3 is presented the use of ExpTool on creating a lab package (for that, we use a case study for conducting the experiment showing the creation of lab package); in Section 4 we presented related works selected from literature; in Section 5 we presented our final remarks and future works.

II. Controlled experiments

To study a technique (method, language or tool) in a controlled experiment, participants are selected from a population, which apply such technique under controlled conditions, according to what was defined by the responsible researchers in the experimental design. From the analysis, conclusions are drawn about the population from which the participants are considered representative.

There are sources of variation in several parts of this process that influence in the results [5], [8]. The study participants can be from diverse cultural environments or can be under a different set of conditions during execution [9], [6]. Thereby, conclusions keep limited by these factors.

In order to generalize conclusions in a more comprehensive way, these variations should be explored and dealt in replications. So the results can be confirmed or the influence of these variations can be identified and produce more relevant conclusions on the topic. Through replications based in a experiment that investigates some technique, the knowledge about it is consolidated. A body of knowledge in Software Engineering with the goal of support decision making related to software development require the accomplishment of families of experiments in different environments [10].

Undertaking a replication requires the access to information about all the study procedures. Such information are stored into the lab package, which registers the experiment documentation. However, Shull et al. [5] pointed out that the lab packages review present difficulties, since a static lab package can not accommodate relevant aspects of the experiment to the researcher that intends to replicate it.

Noticing knowledge transfer problems as barriers to the realization of replications, Mendonça et al. [6] proposed the **FIRE** (*Framework for Improving the Replication of Experiments*), which is composed by the two activities cycles illustrated in Figure 1. The internal cycle represents the execution of an isolated study. In the external cycle are the activities responsible for integrate the knowledge generated by the study in a common body, in order that its lab package can be effectively reviewed by eventual replicators.

Create/evolve package affect the external activity *share knowledge*. As in the other point that the circles intercept themselves, *understand lab packages* is crucial to *set experiment goals* for the replication.

Thus, the lab package represents the output of the internal cycle carrying the information about a study and also the input of the internal cycle of a possible replication, since researchers must review and understand the lab package of the original experiment. So, to share knowledge and aid a better understanding of lab packages (in the external cycle), the way that its information must be represented is very important, what is confirmed also by the activity *standardize packages*.

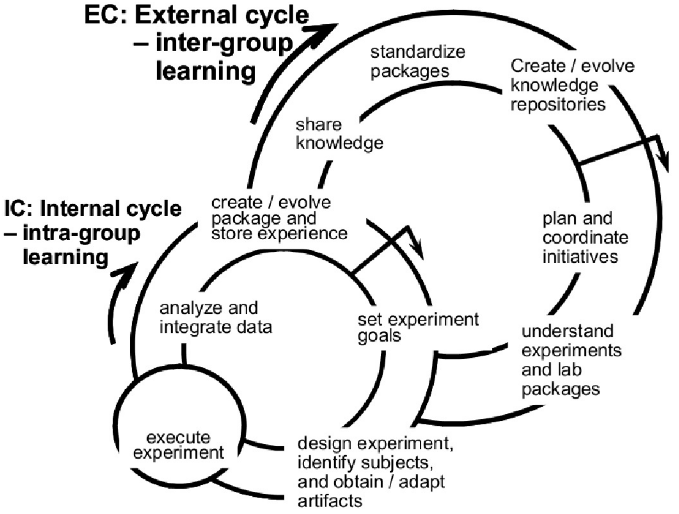


Fig. 1: Cycles of FIRE [6]

A. Experiments in Software Engineering: Main Concepts and ExperOntology

Any experimentation field comprises two types of investigation: Primary and Secondary Studies [11]. Primary studies use specific designs addressed to evaluate the hypothesis formulated by the researcher, to be tested under well-established conditions. Secondary studies intend to produce comparisons between individual investigations selected from a set of primary studies in order to allow generalizing of results. Wohlin et al. [2] have pointed out different sorts of primary studies: Survey, Case Study and Controlled Experiment. The the *ExperOntology* [7] focuses on Controlled Experiment. Its main concepts are highlighted throughout Experimentation Process, described next.

The Experimentation Process follows a sequence of phases [2]: *Definition*, *Planning*, *Operation*, *Analysis* and *Packaging*. In the *Definition* phase, **hypotheses** are clearly stated and the **experiment goals** are established. Based on the definition, in the *Planning* phase, an **execution plan** must be detailed, defining the **execution environment**, the **subjects** involved and their **profile**, the **dependent** and **independent variables** and their scales. At this stage it is important to discuss the **validity** of the expected results. These two initial phases are iterative, since it is possible to return to a previous phase or redo the current one.

The *Operation* phase is divided into three steps: *Preparation*, *Execution* and *Data Validation*. *Preparation* concerns to preparing the required **material** to run the experiment, such as **data collection forms** and **training materials**. The *Execution* must ensure that the experiment is conducted as planned. Finally, during *Data Validation*, replicators try to check the **data collected** for correctness. These three steps are also iterative. After *Operation*, the data collected is **analyzed** (*Analysis*). The *Packaging* phase is concerned to documentation, including experimental **artifacts**, **procedures** and **results** into a so-called **lab package** for future replications. Amaral et al. [12] suggest that such phase should be conducted in parallel throughout the experimentation process. These concepts highlighted in this section are mapped by Garcia et al. [7] into the conceptualization of the ontology in two refinement levels and axioms to formalize it. The first level comprises concepts like controlled experiment, replication, experiment validity and lab package. In the second level, the main aspect is the lab package ontology, which models the concepts that should be considered in the experiment packaging. Scatalon, Garcia and Correia [13] pointed out an approach to package controlled experiments using an evolutionary approach based on ontology based on Garcia et al. [7], but focusing only on creating lab packages – the proposed tool do not support conducting an experiment.

B. The Lab Package

In this section, we present the lab package elements in terms of concepts. The concepts that compose a lab package are presented throughout the experimentation process, highlighted in the following. At first, the **initial hypothesis** of a controlled experiment is established. It is composed by the **object of study**, in agreement with a **purpose**, under a **quality focus**, and in a specific **context**.

The *Definition* phase is the basis for the *Planning* phase and the **initial hypothesis** generates the **hypotheses formalized**. These hypotheses have **null hypothesis** and the **alternative hypothesis**, as attributes. From the **hypothesis formalized**, the experimenter defines the experiment variables – **dependent** and **independent variables**. During the planning phase s/he also defines the **experiment objects**: **technologies** to be studied (**techniques**, **methods** or **tools**) and **artifacts** (**documents**, **tools** or **forms**) to be used.

Copyright © 2014 CSREA Press, ISBN: 1-60132-286-0; Printed in the United States of America

Each **subject** has his/her **profile** recorded to characterize his/her background. Capturing the subject background aims at identifying possible influence on results. For instance, previous knowledge about experiment objects or domain application might influence the results obtained. The subjects' profile must be considered to create the **experimental design**, which is built combining experiment objects, independent variables and subjects, in agreement with the hypothesis under investigation. In addition, the subjects' profile must be considered in analysis.

Based on the experimental design, an **execution plan** must be elaborated in order to describe the entire controlled environment to conduct the experiment. Such plan must consider the training activity, which comprises both theoretical and practical approaches for teaching the involved technology. The plan is obtained by establishing the **tasks** to be executed, their sequence and their period. During the execution, each task must have its initial and final time recorded, and differences between **task planned** and **task performed** must be considered as a threat to validity.

The main objective of *Definition* and *Planning* phases is to establish the experimental design, which must satisfy the requirements to the *Analysis* phase. Such phases culminate in the experimental design and in the execution plan, defining an environment as controlled as possible to test the hypothesis and minimize the threats to validity. Both of them are the core to guide the *operation* phase.

The data set gathered during the execution represents the concept **results**. The **analysis** of these results is based on hypotheses formalized and on experimental design focusing on dependent variables. **Confirmatory analysis** aims to test the hypotheses formalized.

From conducting an *original experiment* a *lab package* is generated. A replication uses a lab package from previous experiments as the basis for its motivation as well as for generating a new lab package. Both the original experiment and the replication have to be evaluated regarding to their *validity*. An original experiment is created by a designer, who has his/her profile related to the experiment as a parameter to define a possible threat to validity. In the same sense, a replicator has also his/her profile associated with the replication. It is important to highlight that both designer and replicator profiles might influence, negatively or positively, the conduction of the experiment/replication. The lack of experience, for instance, is a negative influence since it can be difficult to isolate the factors of risk when defining an experiment. Regarding the replication, the lack of experience can also influence the execution fidelity of the original experiment. On the other hand, the high experience is a positive influence since it minimizes the effort for defining the experiment and helps to analyze the lab package both to identify opportunities and combine results of different experimental treatments. So, the designer and replicator profiles must be taken into account during the analysis of the results as an influence on the experiment validity.

The validity evaluation is an issue to be addressed through all phases of the experimentation process. Wohlin et al. [2] pointed out that there are four types of threat to validity: (1) conclusion validity – refers to the relationship between the treatment and outcome; (2) internal validity – refers to the points that assure there is a causal relationship between the factors and the outcome; (3) construct validity – concerns with the relation between theory and observation; and (4) external validity – concerns with generalization. Each type of validity is constrained by threats. A threat to validity constrains the validity of an original experiment or a replication. However, when there are threats, they are identified in the lab package. The influences to any element that integrate a lab package (or the combination of them) cause a threat to validity.

The concepts presented must be organized into the lab package. One might find several papers suggesting guidelines on items to be [14], [15], [2], [16], [17], [18]. Table I summarizes guidelines for packaging, showing at high level what should be packaged – all highlighted items previously presented must be put into the suggested organization of lab package.

There is a large amount of information handled during the experimentation process. The lab package must keep data and decisions during the conception and execution of controlled studies. The packaging phase of a controlled study might be facilitated using a tool. Besides, the tool must be able to support not only the record of data, but also the workflow of controlled experiments. By integrating data collecting with workflow make possible to register data long the process. In the next section we present the proposed tool to support both conducting an experiment and collecting data from its execution.

III. ExpTool: a tool to support controlled experiments

ExpTool was developed using Java programming language as a web application. Servlets and JSP pages (JavaServer Page) were used for treatment and data presentation, as well as traditional Java classes as basic application. Assistive technologies such as jQuery , Ajax and some auxiliary library packages were also essential for the implementation. Its architecture is organized in four layers that exchange data among themselves. They were separated according to their level and function. As a web application, it was necessary to introduce features about security. For that, *Sessions* between the client (browser) and server are kept while a user is logged.

A. Funcionalities to Experimenters

We present the features following the workflow. First, it is necessary to register some experimenter data (name, email and password), what is made by an administrator. The experimenter registered will receive an email to confirm his/her login and password. There is a specific interface to experimenter access their area in ExpTool. The experimenter migh register an organization to be the experiment context by registering name, acronym and country. Also, each subject in the experiment must be registered (name, email, phone and link to a previously registered institution). Each subject receives an email (login and password) to access his/her area in ExpTool.

The experimenter can create new experiments and record data for the initial definition (Definition Phase): name, description, theme, technical area, type, domain and language. Also, according the GQM (Goal, Question and Metric) [2], it

TABLE I: Overview of proposed guidelines for packaging, adapted from Jedlitschka [19]

	Author					
	Singer [14]	Wohlin et al. [2]	Kitchenham et al. [15]	Juristo e Moreno [16]	Kitchenham [17]	Jedlitschka et al. [18]
type of study	experimental research	experimental research	experimental research	controlled experiment	systematic review	controlled experiment
phases of the study	packaging	all	all	all	all	packaging
Structure proposed	*	*	*	*	title	title
	*	*	*	*	authors	authors
	abstract	*	*	*	structured abstract	structured abstract
	*	*	*	*	keywords	keywords
	introduction	introduction	*	setting goals	background	introduction
		problem statement				
		experiment planning	experimental context			
	introduction	problem statement	experimental context	setting goals	background	background
	method	experiment planning	experimental design	project	review questions	experiment planning
	procedure	operation of the experiment	conduct of the experiment and data collection	experiment execution	review of methods included and excluded studies	deviations from planned
	results	Data Analysis	analysis	experimental analysis	results	analysis
	discussion	discussion and conclusion	*	experimental analysis	conclusion	conclusions and future work
	-	-	-	-	acknowledgments	acknowledgments
					conflict of interest	
	references	references	*	*	references	references
	appendices	appendix	*	*	appendices	appendices

(*) indicates that the elements are implicitly required, although not explicitly mentioned.

is necessary to set goals, registering: object of study; purpose; focus; perspective; context factors.

The experimenter can create new experiments and record data for the initial definition (Definition Phase): name, description, theme, technical area, type, domain and language. Also, according the GQM (Goal, Question and Metric) [2], it is necessary to set goals, registering: object of study; purpose; focus; perspective; context factors. The Figure 2 shows a screenshot of ExpTool interface during the GQM items definition.

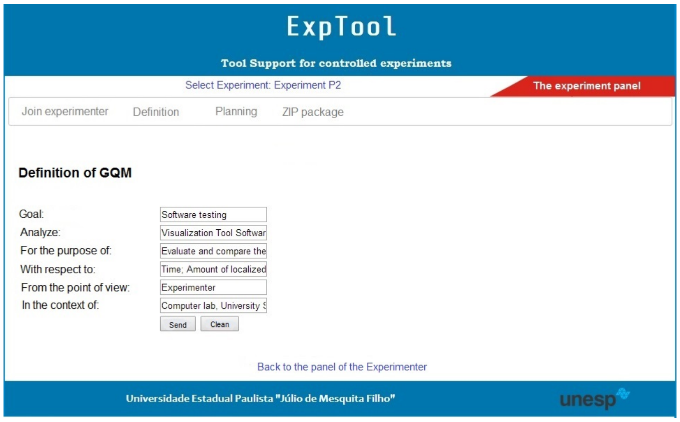
To create the plan, the experimenter has five steps, organized in ExpTool as follow:

1) Planning: consists in five items
   - context: it is defined whether type is in-vitro or in-vivo; whether team is made by students or professionals; whether the problem is real (from industry) or fictitious (classroom); whether domain is specific or general.
   - Hypotheses : the formulation of at least one null hypothesis with a question and the hypothesis itself and the formulation of one or more alternative hypotheses is necessary.
   - Variables: dependent and independent variables are defined according to the hypotheses previously registered.
   - Threats to Validity: set of threats to validity (internal, external and construction threats) are registered.
   - Metric: it is defined metrics and their description shown how data must be collected, registered and analyzed.
2) Instrumentation: consists in three items:
   - Technique: name, description and whether is a method, a technique or a tool under evaluation. Also, it includes description and files of the external artifacts (for example, if a tool is under study, version, files and data about how to install must be kept).
   - Artifact: name, description (questionnaire, document requirement, source code), as well as any included files.
   - Technique versus Artifact: to establish the application of techniques with the artifacts included in the experiment.
3) Participants: there are two items:
   - Selection: select subjects to participate of the study.
   - Characterization: a questionnaire about participants is created to define their profile.
4) Experimental Design: it is necessary to associate each participant to a specific task (apply technique, method or tool) and artifact. For that, it is possible to create groups of participants, and assign or remove participants. So, treatments are associated to groups and the respective artifacts. A schedule about application of treatments by participants must be defined (the experimenter must select the group that will perform a task using the artifact in order to perform the treatment).
5) Planning Conclusion: this step consists in close the planning. After that, it will not be possible to modify the planning.

During the operation phase is not possible to modify any parameter defined during previous phases. At this point, each participant can access his/her area and execute the task designated and record the data obtained during the execution. The features for participants are presented in next section.

During Analysis and Interpretation phase the experimenter evaluates the data obtained from the previous phase, verifying its validity, and makes the relationship between the results and the assumptions, as well as registering the results obtained, with artifacts and conclusions.

Finally, the packaging is done as discussed. The Figure 3 shows the options to generate the lab package: into a XML or a ZIP file. The Figure 4 shows the lab package created (XML


ExpTool
Tool Support for controlled experiments
Select Experiment: Experiment P2
The experiment panel
Join experimenter
Definition
Planning
ZIP package
Send
Clean
Back to the panel of the Experimenter
Universidade Estadual Paulista "Júlio de Mesquita Filho"
unesp

Definition of GQM

Goal: Software testing
Analyze: Visualization Tool Softwar
For the purpose of: Evaluate and compare the
With respect to: Time; Amount of localized
From the point of view: Experimenter
In the context of: Computer lab, University S

Fig. 2: Screenshot of GQM interface – Definition Phase

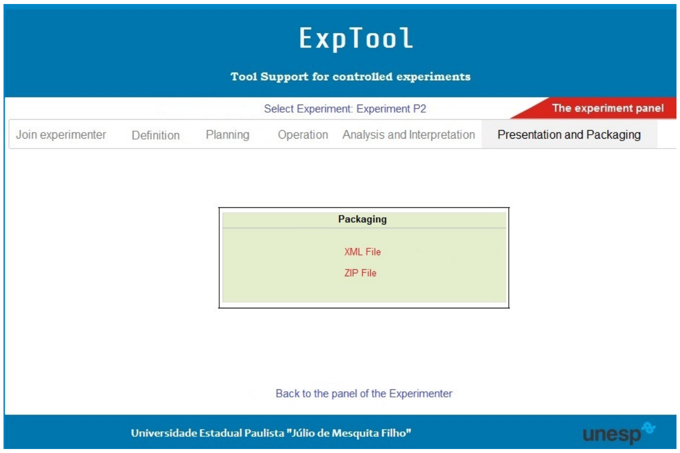
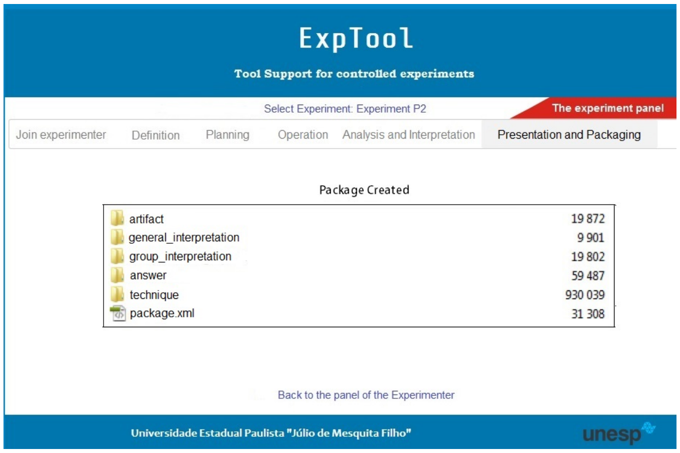

ExpTool
Tool Support for controlled experiments
Select Experiment: Experiment P2
The experiment panel
Join experimenter
Definition
Planning
Operation
Analysis and Interpretation
Presentation and Packaging
Packaging
XML File
ZIP File
Back to the panel of the Experimenter
Universidade Estadual Paulista "Júlio de Mesquita Filho"
unesp

Fig. 3: Screenshot of Creation of lab package

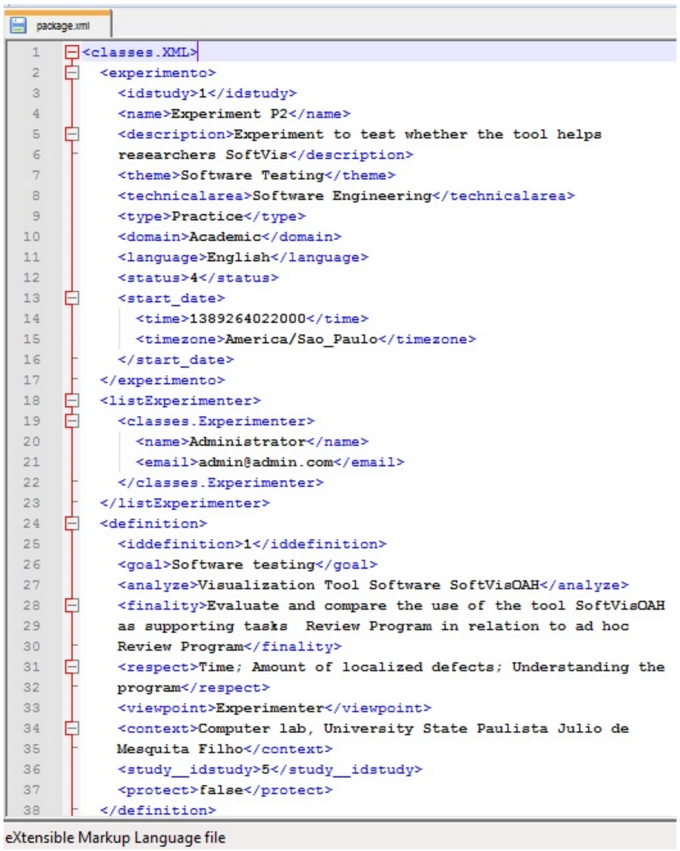

ExpTool
Tool Support for controlled experiments
Select Experiment: Experiment P2
The experiment panel
Join experimenter
Definition
Planning
Operation
Analysis and Interpretation
Presentation and Packaging
Back to the panel of the Experimenter
Universidade Estadual Paulista "Júlio de Mesquita Filho"
unesp

Package Created

artifact	19 872
general_interpretation	9 901
group_interpretation	19 802
answer	59 487
technique	930 039
package.xml	31 308

Fig. 4: Screenshot with Lab package generated

and related artifacts). The Figure 5 shows partially the content of XML file – it is possible to observe the GQM elements instantiated.



```
<classes.XML>
  <experimento>
    <idstudy>1</idstudy>
    <name>Experiment P2</name>
    <description>Experiment to test whether the tool helps
    researchers SoftVis</description>
    <theme>Software Testing</theme>
    <technicalarea>Software Engineering</technicalarea>
    <type>Practice</type>
    <domain>Academic</domain>
    <language>English</language>
    <status>4</status>
    <start_date>
      <time>1389264022000</time>
      <timezone>America/Sao_Paulo</timezone>
    </start_date>
  </experimento>
  <listExperimenter>
    <classes.Experimenter>
      <name>Administrator</name>
      <email>admin@admin.com</email>
    </classes.Experimenter>
  </listExperimenter>
  <definition>
    <iddefinition>1</iddefinition>
    <goal>Software testing</goal>
    <analyze>Visualization Tool Software SoftVisOAH</analyze>
    <finality>Evaluate and compare the use of the tool SoftVisOAH
    as supporting tasks  Review Program in relation to ad hoc
    Review Program</finality>
    <respect>Time; Amount of localized defects; Understanding the
    program</respect>
    <viewpoint>Experimenter</viewpoint>
    <context>Computer lab, University State Paulista Julio de
    Mesquita Filho</context>
    <study__idstudy>5</study__idstudy>
    <protect>false</protect>
  </definition>
```

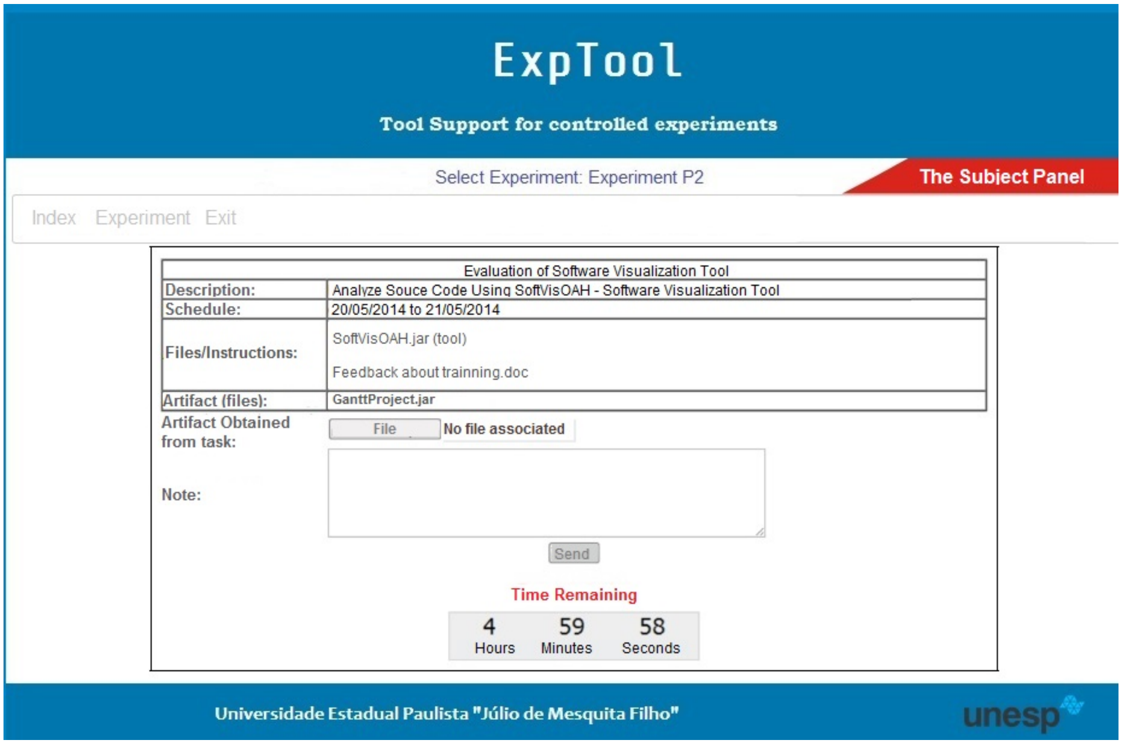
Fig. 5: Screenshot with partial view showing the lab package generated by the ExpTool into XML file

B. Funcionalities to Subjects

Each subject receive by e-mail a confirmation of registration and a login and password. S/He can access the system on access to your login and password by selecting the option to access as a participant interface. When the subject open an experiment associated, should answer the questionnaire to characterize the participant before having access to the activities of the study. The questions are presented and, when answered, the access to the list of activities (tasks) is presented.

Regarding the control flow of activities, there are some important points to consider. According to the schedule defined and the subject action, a status is associated to activities. The activity states are:

- Not available: the activity is not approved for implementation, since its start time (defined by experimenter) has not been reached;
- Available and not started: it is possible to start the activity, but the subject did not do it yet;
- Available and Initiated: it was viewed by the subject, but s/he has not sent the artifact resulting from treatment yet (task not completed);
- Started and not completed: the subject initiated the activity and the deadline for its completion was reached; Not initiated and out of time: even the activity was


ExpTool
Tool Support for controlled experiments
The Subject Panel
Index Experiment Exit
File
Send
Universidade Estadual Paulista "Júlio de Mesquita Filho"
unesp

Select Experiment: Experiment P2

Evaluation of Software Visualization Tool	
Description:	Analyze Souce Code Using SoftVisOAH - Software Visualization Tool
Schedule:	20/05/2014 to 21/05/2014
Files/Instructions:	SoftVisOAH.jar (tool) Feedback about trainning.doc
Artifact (files):	GanttProject.jar
Artifact Obtained from task:	No file associated
Note:	

Time Remaining

4 Hours 59 Minutes 58 Seconds

Fig. 6: Screenshot showing the subject interface during a task

visualized by the subject, and the time limit has expired.

In order to establish and compute the timing, the clock of the server where the application is hosted is used. The activity can only be opened when it is available or has already started. In Figure 6 is presented a screenshot of the interface presented to a subject during a task associated – it is possible to observe the time still available to the task at botton region.

The experimenter set a deadline to each task and whether the treatment (in hours). The time to begins at the point when participant view the activity for the first time. A regressive counter to the participant execute a task is presented with the remaining time. When deadline is reached, the tool prevents the transmission of results to the server. If for some reason the subject performed the activity, but was unable to load the files on time, s/he should contact the experimenter responsible in charge and the experimenter must assess the situation and decide whether use or not the data generated by subject.

IV. Related Works

Some applications have been developed in order to support the controlled experiments. We presented two similar works, pointing out their features and contrainst that motivated our proposed tool. The first one is the eSEE (Experimental Software Engineering Enviroment): a computational infrastructure based on web services that aims to facilitate the instantiation of environments to support some activities of experimental studies. According to Santos and Travassos [20], the insfraestructure supports both primary and secondary studies in Software Engineering. Also, they present a propotype tool named eSPA (experimental Studies Planning Assistant) that amins to analyze data repository about experimental studies and recover decisions about their design. Lopes and Travassos poited out that eSEE have been used to evolve a glossary of terms concerned Experimental Software Engineering [21] .

Another tool to supoprt experimental studies is PontoLab [13]. This tools focuses on instantiate concepts of *ExperOntology* [7]. The interface provides tabs to edit the information related to the experimental process [2], according to the ontology concepts. In the tab corresponding to the definition phase, are inserted the general directions of the experiment, represented by the concepts *object of study*, *quality focus*, *purpose* and *context*. Next, this directions are formalized in hypotheses, that express the cause-effect relationship to be analyzed through the experiment execution. From the hypotheses formulation, in the tab of planning, are defined the *dependent* and *independent variables*, with the respective treatment that can be assumed by each. Also, are selected the *subjects* of the experiment. Once established the independent variables and the participants, it is possible to define the study schedule, that is, the *experimental project*: assign to each subject a set of treatments, each of a independent variable. These assignments, represented in the ontology by the predicate Design, generate *tasks*, inserted in the *execution plan* in the next tab, of the operation phase, in which is also possible to register about the *results* obtained with the execution. Finally, in the last tab, of analysis phase, it is registered about statistical tests of *confirmatory analysis* of the hypotheses and observations that emerged from a *exploratory analysis*, leading to unforeseen relations in the experimental project. It is important to note that the PontoLab tool focuses on packaging an experiment using the ontology, but do not support the execution of controlled experiments.

V. Final Remarks

Experimental Software Engineering attempts to evaluate and measure the performance of models and techniques in practical contexts, in order to establish a body of knowledge base to support decision-making. Also, results from multiples experiments can be used in order to support new ideas and theories. To build a body of knowledge, it is necessary to conduct controlled experiments and their replications. Also, it helps to generalize the results packaged and stored into lab packages. To replicate studies. It is necessary review the lab package from original study. Difficulties rise from the lack of standard to organize data stored into lab packages and, consequently, make difficult to understand its contents.

In this context, we proposed a tool to support the conduction of controlled experiments, presented in this paper. ExpTool provides support to the experimenter: each task defined in workflow to execute an experiment is supported, including the packaging. During each task, data collected are registered. It is possible to keep record (and links) to external files used during the experiment, such as tool and artifacts. After the execution, data analysis and results are registered using external files, if necessary – it is allowed to keep spreadsheet and graphics in separate files. Threats to validity, conclusions and further works can be registered in text format. As a last step, the lab package is instantiated into a XML file, organized as suggested by Jedlitschka [19].

In addition, the ExpTool allows experimenter to use an XML file previously created as input to another experiment. In this case, the XML file represents a lab package and the experimenter would use it to replicate the experiment. The ExpTool allows both replication (repetition of original experiment) and replication with variation – the experimenter is allowed to modify the original plan (different parameters or artifacts, for example) to execute the experiment in order to create a family of studies. As main contribution, we point out: 1) the organization of lab package using the workflow steps and related concepts; 2) facilitate exchanging the lab package among experimenters (since adopting ExpTool, experimenters are allowed to exchange their lab package); 3) an lab package can be used as input to an experiment, facilitating the integration among experiments (and experimenters).

As further work, we are developing a new feature: to instantiate lab packages using an ontology. For that, the concepts will be kept according to Exper$_{Ontology}$ [7]. Also, we are aware that exchanging lab package is difficult to validate. So, we intend to create a repository of controlled experiments and to make them available to other experimenters, as well as the ExpTool.

References

[1] J. C. Carver, “Towards reporting guidelines for experimental replications: A proposal,” in *International Workshop on Replication in Empirical Software Engineering Research (RESER)*, 2010.

[2] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering: An introduction*. Boston, USA: Kluwer Academic Publishers, 2012.

[3] F. J. Shull, J. C. Carver, S. Vegas, and N. Juristo, “The role of replications in empirical software engineering,” *Empirical Software Engineering*, vol. 13, no. 2, pp. 211–218, 2008.

[4] B. Kitchenham, “The role of replications in empirical software engineering–a word of warning,” *Empirical Software Engineering*, vol. 13, no. 2, pp. 219–221, 2008.

[5] F. Shull, V. R. Basili, J. Carver, J. C. Maldonado, G. H. Travassos, M. G. Mendonça, and S. C. P. F. Fabbri, “Replicating software engineering experiments: Addressing the tacit knowledge problem,” pp. 7–16, 2002.

[6] M. G. Mendonça, J. C. Maldonado, M. C. F. de Oliveira, J. Carver, S. C. P. F. Fabbri, F. Shull, G. H. Travassos, E. N. Hohn, and V. R. Basili, “A framework for software engineering experimental replications,” in *ICECCS*, 2008, pp. 203–212.

[7] R. E. Garcia, E. N. Höhn, E. F. Barbosa, and J. C. Maldonado, “An ontology for controlled experiments on software engineering.” in *Proc. (Software Engineering & Knowledge Engineering)*. Knowledge Systems Institute Graduate School, 2008, pp. 685–690.

[8] F. Shull, M. G. Mendonça, V. R. Basili, J. Carver, J. C. Maldonado, S. C. P. F. Fabbri, G. H. Travassos, and M. C. Ferreira, “Knowledge-sharing issues in experimental software engineering,” *Empirical Software Engineering: An International Journal*, vol. 9, pp. 111–137, March 2004. [Online]. Available: http://portal.acm.org/citation.cfm?id=966771.966783

[9] J. Miller, “Replicating software engineering experiments: a poisoned chalice or the holy grail,” *Information & Software Technology*, vol. 47, pp. 233–244, March 2005. [Online]. Available: http://dx.doi.org/10.1016/j.infsof.2004.08.005

[10] V. R. Basili, F. Shull, and F. Lanubile, “Building knowledge through families of experiments,” *IEEE Transactions on Software Engineering*, vol. 25, no. 4, pp. 456–473, 1999.

[11] J. C. de Almeida Biolchini, P. G. Mian, A. C. C. Natali, T. U. Conte, and G. H. Travassos, “Scientific research ontology to support systematic review in software engineering,” *Adv. Eng. Inform.*, vol. 21, no. 2, pp. 133–151, 2007.

[12] E. A. G. G. Amaral and G. H. Travassos, “A package model for software engineering experiments,” in *Proceedings of ISESE 2003 - International Symposium on Empirical Software Engineering*, 2003, pp. 21–22.

[13] L. P. Scatalon, R. E. Garcia, and R. C. M. Correia, “Packaging controlled experiments using an evolutionary approach based on ontology,” in *International Conference on Software Engineering and Knowledge Engineering (SEKE 2011)*, 2011, pp. 408–413.

[14] J. Singer, “Using the APA style guidelines to report experimental results,” in *Proceedings of Workshop on Empirical Studies in Software Maintenance*, 1999, pp. 71–75.

[15] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, P. W. Jones, D. C. Hoaglin, K. E. Emam, and J. Rosenberg, “Preliminary guidelines for empirical research in software engineering,” *IEEE Transactions on Software Engineering*, vol. 28, no. 8, pp. 721–734, 2002.

[16] N. Juristo and A. M. Moreno, *Basics of software engineering experimentation*. Kluwer, 2001.

[17] B. Kitchenham, “Procedures for performing systematic reviews,” Departament of Computer Science, Keele University, Tech. Rep., 2004.

[18] A. Jedlitschka, M. Ciolkowski, and D. Pfahl, “Reporting guidelines for controlled experiments in software engineering,” Fraunhofer Institute for Experimental Software Engineering, Germany, Tech. Rep., 2007.

[19] ——, *Guide to Advanced Empirical Software Engineering*. London: Springer-Verlag, 2008, ch. Reporting Experiments in Software Engineering, pp. 201–228.

[20] P. S. M. dos Santos and G. H. Travassos, “esee - ambiente de apoio a experimentação em larga escala em engenharia de software,” in *1st Brazilian e-Science Workshop*, 2007.

[21] V. P. Lopes and G. H. Travassos, “Experimentação em engenharia de software: Glossário de termos,” in *ESELAW´09 - VI Experimental Software Engineering Latin American Workshop*, 2009.

Minimizing the negative impact of emergent properties in component based complex systems

Ndabezinhle Soganile [1], **Benson Moyo**[2]

[1]Department of Computer Science and Information Systems, University of Venda, Limpopo, South Africa

[2]Department Department of Computer Science and Information Systems, University of Venda, Limpopo, South Africa

Abstract *The Software Engineering field is faced with many engineering challenges. Of particular note is the emergent property phenomenon. The concept of emergent property in software engineering originates from component based software development, where in a bid to reduce development time and costs, developers build their systems from readymade components. The emergent property is thought of as an unanticipated system behaviour that is exhibited as a result of interaction between system components. These behaviours can either be positive, meaning that they may add value to the systems original intended functionality or they can be negative, meaning that they may cause harm and at times compromise the entire security of the system. To contain the negative impact and to better understand emergent properties researchers have and are coming up with ways to predict these properties afore hand. In this paper we discuss what emergent properties are, their impact on complex systems, the different types of emergent properties and we also look at the existing methods of predicting emergent properties. Finally we offer suggestions to minimize the impact of these emergent properties.*

Keywords : *Emergent property, EEEP, LEEP, Complex Systems*

3 Introduction

In Software Engineering emergence describes the new behaviours and patterns of complex systems, arising at the systems integrative level due to a multiplicity of fairly simpler components interacting. This interaction is not only component to component but also it is the interaction of these components or the whole system with the environment.[1] points out that it is hard to design and understand complex systems because of the interaction between different components. Different authors have diverse interpretations of the concept of emergent property "There is, however, considerable disagreement about the nature of 'emergent properties'" [1], CW. Johnson[1] attempts to give several alternate views of 'emergence' so as to reduce confusion associated with the use of the word. In this paper we will concentrate on the impact of the emergent property from a software engineering perspective and then try to suggest ways of reducing negative impacts associated with these emergent properties. Emergent properties arise more often when the system is in its deployment phase, though they can be observed in a phase during development. The diagram below shows the emergent property assessment cycle

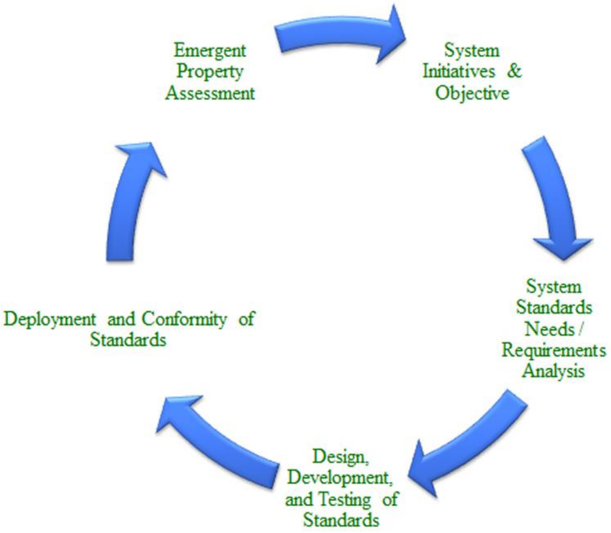

Emergent Property Assessment
System Initiatives & Objective
System Standards Needs / Requirements Analysis
Design, Development, and Testing of Standards
Deployment and Conformity of Standards

Figure 1 : *Emergent property assessment cycle*
Adopted from *Manyphay Viengkham, IEC 2012 Young Professional Leader*

2 Emergent Properties

Unforeseen behaviours and patterns of a system that are observed as a result of an application component interacting with the environment define what is referred to as 'emergent properties'.Johnson[1] highlights that such emergent properties are used to differentiate complex systems from complicated applications. Emergent properties are not always negative, at times users benefit from them. Robust applications are delivered to users; they end up adopting products with more functionality than originally planned by designers.The biggest challenges come when these emergent properties exhibit negative behaviours in the system. The more complex the systems become the higher the potential for unpredictable emergent properties[2].

Emergent properties may arise as a good feature of a system hence adding a functionality that was not originally intended for by the developer, but at a larger scale they are usually problematic since they undermine important safety requirements.

Some engineers try to take advantage of emergent features by trying to model the emergent feature of the real world hoping that they might achieve new functionality. As an example Steve Forest said “interesting systems can be developed by taking advantage of interaction among components.

Generally emergence is a higher level property which cannot be deduced from or explained by properties of low level entities.

Economist Jeffrey Goldstein identifies the following characteristics of emergent properties:

- Features not previously observed in systems
- Integrated wholes that maintain themselves over some period of time.
- A global or macro level (wholeness)
- Emergence evolves
- It can be perceived.
- He defines emergence as “the rising of novel and coherent structures, patterns and properties during the process of self-organization” in complex systems.

Complex systems are built from completely autonomous components adhering to a particular standard. John Synas [3] points out that some systems exhibits behaviours that are not embedded on the source code but rather emergent out of the interaction of the system component with the environment. This phenomenon is referred to as computational emergence.

3 Computational Emergence

Computational emergence is a feature in which software behaves or exhibits behavior that the developer did not instruct it to do. Thus they may be described as features that were not hard coded into the source code of an executable object [3].

Emergent properties are properties of a system as a whole e.g. a car, it is made of separate parts from different manufacturers such as the body, engine, wheels, steering, axel etc. These individual components have their specific functionality that do not emerge until they are all combined into a whole, i.e. these components are useless on their own; their purposed only becomes apparent if they have been combined to form a vehicle. Thus we deduce yet again that they are properties that only emerge when the system is combined

Emergent properties may be a good feature of a system that is if users discover a new advantageous functionality that the developer dint intend to do , but to a larger extent they are usually problematic for example if they undermine important safety requirements.

Some engineers try to take advantage of emergent features by trying to model the emergent feature of the real world hoping that they might achieve new functionality. As an example Steve Forest said “interesting systems can be developed by taking advantage of interaction among components.

Thus in a nut shell I can say Emergence is a higher level property which cannot be deduced from or explained by properties of low level entities.

4 Types of Emergent Properties

4.1 Functional Emergent Properties

This is when parts of a system interact to achieve a particular task [4]. Components are combined to give a particular functionality of the system. An example of a car can give us a clear picture of what functional emergent properties are. A car can only be considered as a transformational vehicle only if it’s a whole. Car parts are useless as individual components but can only function when combined with other parts to form a whole which is a vehicle. The main aim of systems development is to create a system with the desired functional emergent properties.

4.2 Non Functional Emergent properties.

These are properties that represent behaviours of a system in its operational environment. They are critical because failure to meet their minimal requirements might render the system unusable [4]. Non-functional emergent properties are defined by its reliability, its security strength as well as its performance. Reliability, security and performance are dependent on the operational environment. A system can only be declared reliable when all its components are reliable that is the hardware, software and operators. Ian Somerville in his book Software Engineering explains how an error can propagate from one component to another until it brings the whole system down. Therefore to ensure reliability of a system all the components need to be tested as a whole.

The performance of a system cannot be predicted based on individual components but rather it can only be measured after all components have been integrated.

Emergent properties types can further be divided into *weak properties* and *strong properties*. *Weak properties* are those at a low level and can be predicted using observation. These do not need priori analysis. On the other hand *Strong emergent* properties are at a high level, making them difficult to predict. These are the properties that emerge to the system as a whole.

In this paper we identify and categorise these emergent properties into two, the Early Exposed Emergent Properties (EEEP) and the Late Exposed Emergent Properties (LEEP). We define EEEP as those emergent properties that are detected during the component integration stage and are mostly code related. On the other hand the LEEP are those that are detected during system rollout, when the system components begin to interact with the environment. It is relatively easy to deal with the EEEP than the Handle LEEP, the challenge for many engineers is on how to predict and handle the LEEP.

5 The impact of Emergent Properties on Complex Systems

An object like a car is made of separate parts from different manufacturers, suchparts as the body, engine, wheels, steering, axel etc.These individual components have their specific functionality which doesn't emerge up until they are all combined into a whole, i.e. these components are useless on their own, and their purpose only becomes apparent if they have been combined to form a vehicle. Thus we deduce yet again that they are properties that only emerge when the system is combined. In an equal analogy, a system or application is the sum of its simpler standalone components. Ian Somerville [4] defines a component as an independent executable entity that can be made up of one or more executable objects. These components exhibit lesser functionality as individuals than when they are combined into a system. Once built into a more complex system, new behaviours begin to surface as a result of component interaction with the environment. New advantageous functionalities begin to be seen, adding robustness to the whole system. John C. Hsu[5] highlights that these emergent properties also provide abundant potential for applications not only to overcomethe problems of interoperability but also to achieve high levels of adaptability, scalability, and cost effectiveness not possible in traditional systems. Systems exhibit new behaviours as a way of adapting to the environment. Thanks to the emergent properties systems continue to operate in the ever changing environments and that factor increases cost effectiveness since these systems are self-organising, there is no need for costly upgrades that are motivated by environmental changes. Taking advantage of these new behaviours, engineers are able to trace, test and validate the requirements to the good of the development process.

On the contrary, emergent properties can be harmful or undesirable.Their occurrence might have a serious negative impact to the entire system. They may compromise the important security features of the system, thereby escalating the total costs of developing the system. Additional budgets focussed on fixing the problem must then be put in place. The problem is compounded by the fact that in complex systems, it is difficult to identify the origins of the error. More time therefore is wasted on fixing errors. The challenge facing the software engineering world is how to minimize these negative behaviours

5 Existing Strategies

Proper prediction and handling of the emergent properties helps to reduce the risk of negative impact brought by these emergent properties. Engineers and researchers are coming up with useful tools and strategies for dealing with negative emergent properties. The common methods used in predicting the emergent properties are discussed below.

5.1 Simulation models

Before the system is rolled out in the real world simulation models that mimic the actual world are developed and the system is tested on that virtual environment. It is a method implemented at lower level with weak emergence. It involves creating a model that simulates the actual system. The model is studied to derive the behavior of the combined parts. The engineers' first work on a model to see if what they want to achieve is feasible.

5.2 Observation

This prediction mechanism is also applicable at a lower level. This involves closely watching how the individual components work then try and figure out how they would perform when combined.

5.3 Priori Analysis

This method simply means deriving some conclusion without any physical experiment. It involves trying to figure out how the whole system will perform without actually putting it into use. The only major weakness of this strategy is that crucial emergent properties are always left out since the analysis is dependent on the expertise of the engineer doing the analysis

6 Suggested Solution

Considering the gravity on the system and the risk on business should the negative emergent properties be exhibited during the run of business, it is therefore important to find ways of eliminating these negative properties as early as possible before the system is deployed. In the following section we give our suggestion on how best the negative properties can be reduced

6.1 Piloting

Piloting is a strategy that categorises emergent properties into two views, the Early Exposed Emergent Properties (EEEP) and the Late Exposed Emergent Properties (LEEP). The first category, the EEEP is mostly associated with the code and can be detected and dealt with during the component integration phase. We recommend the use of any available strategy as define in section 5 above. The second category, the LEEP are those behaviours that the system exhibits during the deployment phase, they come as consequence of the system interaction with the real environment. They are often risky and harmful if they happen to be negative. We suggest dealing with the LEEP category in two ways, Firstly, the system must be tested on a simulation platform, this will provide an opportunity to identify and eliminate a number of these mal emergent properties. Secondly, the system must be tested on a pilot real world platform before deploying or delivering to the client. This will give engineers an opportunity to further identify and deal with these properties before the final deployment of the system.

7 Conclusion

It is practically impossible to fully predict all the behaviours of a complex system since the environment is not a constant. It is ever changing, therefore complex systems are bound to exhibit some strange behaviours that are dependent on the environmental changes. However the impact can be minimized by adequately testing the systems before putting them to work. In this paper we recommended *piloting as* one strategy that engineers can use to minimize the impact of the emergent properties. Predicting and eliminating the emergent properties before the deployment is a good way of ensuring that their impact is not felt when the system is put to work.

Reference List

1. C.W. Johnson, 2003 A Handbook of Accident and Incident Reporting, Glasgow University Press, Glasgow.
2. W.A. Wulf, 2000 Great Achievements and Grand Challenges, *The Bridge*, US National Academy of Engineering
3. David H. Bailey A, 2012, New Kind of Science: Ten Years Later
4. Ian Somerville, 2011, Software engineering, 9th edition, ISBN-13: 978-0-13-703515-1
5. John C. Hsu, 2009, Emergent Behaviour of Systems-of-Systems, Marion Butterfield. The Boeing Company
6. Ferreira S. Tejen, J. (2011) ‘ An evolving understanding for predicting emergent properties’, IEEE International System Conference (SysCon), Texas, pp 479 -483.
7. Forrest, S. (1990). Emergent computation: self-organizing, collective, and cooperative phenomena in natural and artificial computing networks. Physical D, 42, 1–11.

Establishing Testbed for Functional Testing of Embedded System

Jang Yeol Kim[1], Kwang Seop Son[1] , Young Jun Lee[1] , Se Woo Cheon[1], Kyoung Ho Cha[1], Jang Soo Lee[1], Kee Choon Kwon[1]

[1]Instrumentation and Control / Human Factors Division, Korea Atomic Energy Research Institute,
989-111 Daedeok-daero, Yuseong-gu, Daejeon, Republic of Korea 305-353

Abstract – *Software lifecycle consists of requirement phase, design phase, implementation phase, testing phase, integration phase, installation phase, operation and maintenance phase. In particular, testing phase involves component test, software integration test, and software-hardware integration, system test. To qualify as the safety-critical software, exhaustive testing is necessary. However, it still remains as a challenge in embedded system verification and validation due to the time and efforts required for having to test software manually. In this study, we established automatic testbed for small digital devices. The testbed can perform not only on manual mode, but also on automatic mode. It has extended communication ports such as USB, Ethernet, and RS232C. Using these ports, input data and test result values can be easily transferred between testbed and target device. We performed the functional test on a small digital device using our testbed. The result from manually testing the device was same as the result from the automated testbed we developed.*

Keywords: Testbed, Auto mode, Manual Mode, Functional Testing, Performance Testing, System Test

1 Introduction

The purpose of developing testbed for system test is to save time and efforts from functionality and performance tests. We developed the testbed for burning test based on operational scenario. Hardware, software, and man-machine-interface (MMI) are fundamental parts to the testbed. Signals generated from the testbed contain analog input, analog output, digital input and digital output. Using LabView programming, we established the testbed that can generate signal sources consisting of triangle wave, sign wave, and step-wise signal. The testbed can control not only the real-time condition parameters, but also weighted signal test. The testbed can provide a solution for the time discrepancy between input signal and result value during synchronization process.

The sequence of automatic system testing is in the order as shown below.

o Target for small digital device
o Test approach
  - The functional and performance based testing
  - Identify the test items, selection of test cases and test execution
o System configuration and test environment
o Test scenarios, pass/fail criteria
o Test procedures/Test execution and test results

Input/output specification of the test-bed in Automatic Test-bed Establishing System for Small Digital Devices is as follows.

o Analog I/O
  - Voltage Source(-10V~10V) : 64 Channels
  - Voltage Source(-0V~10V) : 48 Channels
  - Current Source (-20mA~20mA) : 8 Channels
o Digital I/O
  - Digital Input (30V max) : 64 Channel
o 60V Programmable Power Supply
o RS232C Serial interface
o Dual port Ethernet interface

2 Establishment of Testbed

The general methods for testing are as shown in Figure 1. After making a test plan, test suite/test case were generated. All the results from the testing were documented and reported as the test summary report.

In general, test process is as follows.[1]

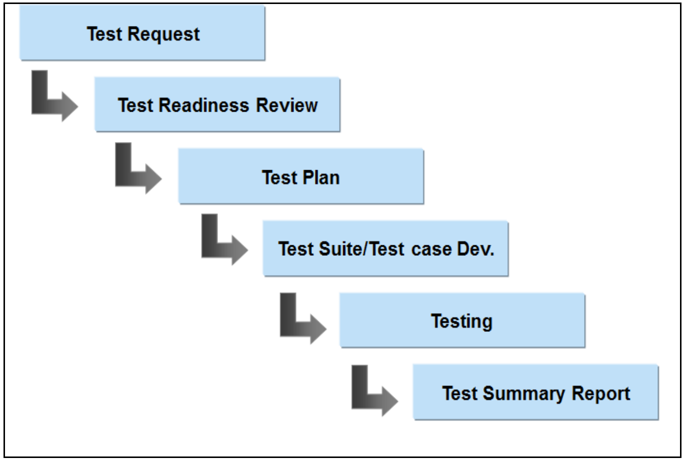


Figure 1. General test process

This study describes the system test environment and system test configuration.

But, in order to establish of automatic testbed system, five items are to be considered.

a) System configuration
b) Environment variable setup
c) Labview Programming
d) Making an automatic testing scenario
e) Time synchronization between input value and result value (i.e. returning output value)

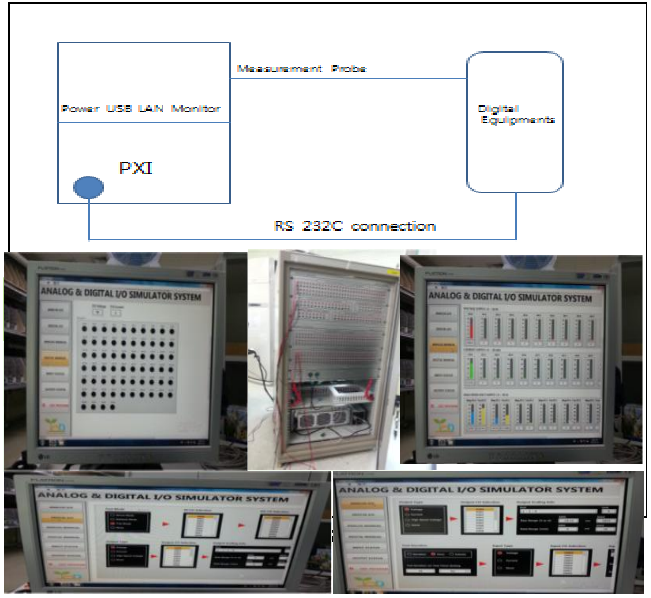

Measurement Probe
Power USB LAN Monitor
PXI
Digital Equipments
RS 232C connection
ANALOG & DIGITAL I/O SIMULATOR SYSTEM
ANALOG & DIGITAL I/O SIMULATOR SYSTEM
ANALOG & DIGITAL I/O SIMULATOR SYSTEM
ANALOG & DIGITAL I/O SIMULATOR SYSTEM

Figure 2. System Configuration of Automatic Test-bed

Mainly, functional test and performance test were conducted. A test case data generation and test execution were done by Automatic Test-bed Establishing System.

NI hardware components in Figure 2 are as followings.

o NI PxI-6511(DIO)

o NI PxI-6512 (DIO)

o NI PxIe-6363 (AIO, DIO)

o NI PxIe-6363 (AIO, DIO)

o NI PxIe-6239 (AIO, DIO)

o NI PxIe-6733 (AO, DIO)

o NI PxIe-6704 (Voltage Out, Current Out)

o NI PXI-8430/4 (4 Port)

o NI PXI-8234

o GPR-6030D

Module-specific software configuration is shown in Table 1.

Table 1 Test Mode on Analog and Digital

Items	Mode	Procedure
Analog Value	Auto	(1) HW setting for test condition (I/O channel, measurement range, measurement unit, Error allowed range)
		(2) START
		(3)Target(small digital device) measurement according to volt or current increment
		(4) Save testing result through RS232C communication
	Manual	(1)Power supply control by communication
		(2)Power supply output control if output value changes
		(3)Output by I/O card
Digital Value	Auto	(1) Test Mode setup (Direct, Delay, Trip)
		(2) Setup for Input / Output channel
		(3) Setting for voltage and current
		(4) START
		(5) Output with setup mode
	Manual	(1) Setting for I/O channel
		(2)Setting for output, voltage, and current
		(3) Output of voltage and current for assigning I/O channel

3 Test Case Design

The following test cases can be written before actual tests. Test case IDs are unique numbers for test items. Input range is either 0~10Volt or 4~20mA. For each input data, several sequence numbers can be assigned. Current value is mapping to volt value using Ohm formal equation. Input value is for testing input data. Expected value is calculated by Engineering Unit Conversion (EUC) Formula in advance. Result value is an actual output value. Tolerance range is acceptable values between actual value and expected value assuming ±0.1%. TRUE or FALSE indicates PASS or FAIL, respectively. More detailed information is shown in Table 2.

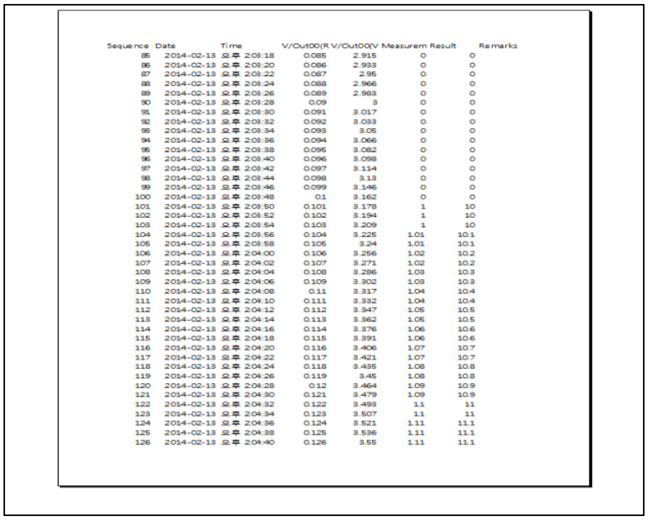
Table 2 Test case generation form for test result measurements

Test case ID	Input Range	Seq. NO	Volt	Current	Input Value	Expected Value (MATH Formula) (%)

Result Value (%)	Tolerance Range		TRUE/ FALSE	Comments
	(Low)	(High)		

4 Automatic test Results

The test case of Table 2 executes under the testbed system in Figure 2. Previous system test had to rely on labor-intensive work for testing small digital devices. The testbed we developed is fully automated and covers virtually all the range of system testing. It was time saving and cost efficient in comparison to the manual testing. Based on our testbed, we have a plan to expand the testbed for Field Programming Gate Array (FPGA)-based controller.



Sequence	Date	Time	V/Out00(R	V/Out00(V	Measurem	Result	Remarks
85	2014-02-13	오후 2:03:18	0.085	2.915	0	0	
86	2014-02-13	오후 2:03:20	0.086	2.933	0	0	
87	2014-02-13	오후 2:03:22	0.087	2.95	0	0	
88	2014-02-13	오후 2:03:24	0.088	2.966	0	0	
89	2014-02-13	오후 2:03:26	0.089	2.983	0	0	
90	2014-02-13	오후 2:03:28	0.09	3	0	0	
91	2014-02-13	오후 2:03:30	0.091	3.017	0	0	
92	2014-02-13	오후 2:03:32	0.092	3.033	0	0	
93	2014-02-13	오후 2:03:34	0.093	3.05	0	0	
94	2014-02-13	오후 2:03:36	0.094	3.066	0	0	
95	2014-02-13	오후 2:03:38	0.095	3.082	0	0	
96	2014-02-13	오후 2:03:40	0.096	3.098	0	0	
97	2014-02-13	오후 2:03:42	0.097	3.114	0	0	
98	2014-02-13	오후 2:03:44	0.098	3.13	0	0	
99	2014-02-13	오후 2:03:46	0.099	3.146	0	0	
100	2014-02-13	오후 2:03:48	0.1	3.162	0	0	
101	2014-02-13	오후 2:03:50	0.101	3.178	1	10	
102	2014-02-13	오후 2:03:52	0.102	3.194	1	10	
103	2014-02-13	오후 2:03:54	0.103	3.209	1	10	
104	2014-02-13	오후 2:03:56	0.104	3.225	1.01	10.1	
105	2014-02-13	오후 2:03:58	0.105	3.24	1.01	10.1	
106	2014-02-13	오후 2:04:00	0.106	3.256	1.02	10.2	
107	2014-02-13	오후 2:04:02	0.107	3.271	1.02	10.2	
108	2014-02-13	오후 2:04:04	0.108	3.286	1.03	10.3	
109	2014-02-13	오후 2:04:06	0.109	3.302	1.03	10.3	
110	2014-02-13	오후 2:04:08	0.11	3.317	1.04	10.4	
111	2014-02-13	오후 2:04:10	0.111	3.332	1.04	10.4	
112	2014-02-13	오후 2:04:12	0.112	3.347	1.05	10.5	
113	2014-02-13	오후 2:04:14	0.113	3.362	1.05	10.5	
114	2014-02-13	오후 2:04:16	0.114	3.376	1.06	10.6	
115	2014-02-13	오후 2:04:18	0.115	3.391	1.06	10.6	
116	2014-02-13	오후 2:04:20	0.116	3.406	1.07	10.7	
117	2014-02-13	오후 2:04:22	0.117	3.421	1.07	10.7	
118	2014-02-13	오후 2:04:24	0.118	3.435	1.08	10.8	
119	2014-02-13	오후 2:04:26	0.119	3.45	1.08	10.8	
120	2014-02-13	오후 2:04:28	0.12	3.464	1.09	10.9	
121	2014-02-13	오후 2:04:30	0.121	3.479	1.09	10.9	
122	2014-02-13	오후 2:04:32	0.122	3.493	1.1	11	
123	2014-02-13	오후 2:04:34	0.123	3.507	1.1	11	
124	2014-02-13	오후 2:04:36	0.124	3.521	1.11	11.1	
125	2014-02-13	오후 2:04:38	0.125	3.536	1.11	11.1	
126	2014-02-13	오후 2:04:40	0.126	3.55	1.11	11.1	

Figure3. Automatic test results (sample)

Calibration factor for testbed is necessary due to the unmodifiable parameters involving displayed values and output data for target digital device.

The testbed was designed for AUTO mode in the generation of voltage and current according to test mode setup, test case, parameter name, engineering unit conversion (EUC) formula, range, unit, expected Pass/Fail criteria, condition value and timing synchronization method. Also, it was developed for MANUAL mode for the generation of voltage and current based on test mode setup and setting the input and output channel numbers. Test results were analyzed by comparing the expected values to the actual values.

The integrity of the automatic test-bed was validated by comparing the Fluke-755 manual test measurements with AUTO mode test results.

All the results from the testing were documented and reported.

5 Conclusion

Following the test procedures, we performed the test scenario-based automatic functional testing for safety-grade digital device as a thirty-party verifier. It was successfully completed. While automation cannot reproduce everything that a software engineer can do, it can be extremely useful for the system test. However, it does require a well-developed test suite of testing scripts in order to be useful. In this study, we developed the multipurpose and cost-efficient testbed using LabView program instead as a testing script.

Future research on safety-critical software engineering will include the following items.

o Design qualification for Safety features

- Explicitly Initialization

- Memory Check during Initial Booting

- Self-Supervision

- Self-Diagnostic

- Mirroring

o Lesson Learned from a wide range of Experience

o How to setup the Acceptable Verification and Validation (V&V) Framework (Safety CASE)

6 References

[1] Jang-Yeol Kim, Soon-Gohn Kim, “Software Qualification Approach for Safety-critical Software of the Embedded System”, The 2012 International Conference on Future Generation Communication and Networking (FGCN), Kangwondo Korea, December 16-19, 2012

[2] IEEE Std-829, “IEEE Standard for Software Test Documentation,” 1998.

[3] IEEE Std-1016, “IEEE Recommended Practice for Software Design Descriptions,” 1998.

[4] J. Y. Kim, Kee-Choon Kwon, “The Commercial Off The Shelf(COTS) Dedication of QNX Real Time Operating

System(RTOS)," International Conference on Reliability, Safety and Hazard-2010, Mumbai India, December 14-16, 2010.

[5] J.Y. Kim, S.W. Cheon, J.S. Lee, Y.J. Lee, K.H. Cha, and Kee-Choon Kwon, "Software V&V Methods for a Safety Grade Programmable Logic Controller," International Conference on Reliability, Safety and Hazard-2005, Mumbai India, December. 1-3, 2005.

[6] 10CFR 50 Appendix A,4/94, "General Design Criteria"

[7] ASME NQA-1-1997 "Quality Assurance Requirements for Nuclear Facility Applications"

[8] USNRC Reg. Guide 1.152, Rev. 02, 2006, "Criteria for Programmable Digital Computers System Software in Safety Related Systems of Nuclear Power Plants"

[9] USNRC Reg. Guide 1.172, Rev. 00, Jul. 1997, "Software Requirements Specifications for Digital Computer Software Used in Systems of Nuclear Power Plants"

[10] IEEE Std. 7-4.3.2-2003, "Standard Criteria for Digital Computers in Safety System of Nuclear Power Generating Stations"

[11] IEEE Std. 829-1998, "IEEE Standard for Software Test Documentation"

[12] IEEE Std. 1008-1987, "IEEE Standard for Software Unit Testing"

[13] IEEE Std. 1012-1998, "IEEE Standard for Software verification and validation"

ACKNOWLEDGEMENT

This work, described herein, is being performed for "Development of licensing and Validation Technology" as a part of the Korea Atomic Energy Research Institute (KAERI) projects and funded by Ministry of Trade, Industry and Energy since on November the 1st, 2013.

The Case for an Open and Evolving Software Assurance Framework

Miron Livny[1], **Barton P. Miller**[2], **and Von Welch**[3]
[1]Morgridge Institute for Research, Madison, WI, U.S.A
[2]Department of Computer Science, University of Wisconsin-Madison, Madison, WI, U.S.A.
[3]Center for Applied Cybersecurity Research, Indiana University, Bloomington, IN, U.S.A.

Abstract – *While software is becoming an increasingly ubiquitous part of our lives, our ability to determine assurances about the resilience of that software with regards to malicious actors lags. The DHS-funded Software Assurance Marketplace (SWAMP) is an instantiation of a framework for software assurance that seeks to address this gap. The vision for the framework is not a software assurance tool in itself, but rather a mechanism to allow for flexible application of different software assurance tools and technologies into automated workflows. This paper describes the desired attributes of such a framework: tools and software packages can be easily added; analysis results, at all levels, can interpreted by different tools; and the entire software assessment process can be readily studied. A key goal of the framework is to provide the foundation for an increasingly large and productive community working on software assurance.*

Keywords: Software assurance, software development, quality assurance.

1 Introduction

Software has become an essential component of every element of our life - from the pacemaker to the national power grid. It has been growing in complexity and size at a rate that exceeds our ability to keep pace with assuring its quality. Recent events, such as Heartbleed, have exposed vulnerabilities in critical and widely used software components, software assurance (SwA) methodologies and technologies failed to prevent or detect these critical weaknesses [1]. The authors are leading a project, the Software Assurance Marketplace (SWAMP) [2], funded by DHS, to tackle the growing gap between the role of software and our ability to provide assurance about software. The SWAMP is not a SwA technology in itself, but a rather a materialization of a framework for SwA, bringing together software packages and SwA tools with the principle of continuous assurance, and forming the foundation for SwA research, development and application. In this paper we describe the framework that governs the design and implementation of the SWAMP facility, and the rationale behind it.

2 The Open and Evolving Framework

Evaluating the assurance of software involves solving a broad spectrum of problems. Each of these problems constitutes a stage in an end-to-end workflow, where each stage is realized by a collection of tools that addresses the specific tasks of that stage. Extending the impact and expanding the reach of software assurance technologies requires a model that captures the different stages in these workflows and a framework that embodies it. Such a framework will cover the stages of code development, analysis, result normalization and labeling, result merging and integration, visualization, result evaluation and annotation, and risk assessment. The model and framework cannot be static, as experience and innovation will evolve them well past the limits of today's technologies. Our understanding of software engineering and assurance challenges as well as novel methodologies will continue to grow through research and experience.

One of the key benefits of a flexible and adaptive framework is the ability to compose a variety of tools to produce a workflow that is tailored to the specific needs of a software assurance task. Operating on a chain of well-defined intermediate results, these "best of breed" tools will join forces to deliver effective assurance capabilities to the end user ranging from a student in a class to a software supply chain specialist. The value and power of such frameworks have been effectively demonstrated in a variety of research and engineering areas as they encourage and facilitate sharing within and across organizations. Easy authoring, exchange and adaptation of workflows facilitate the development and adoption of best practices throughout the community.

Another key impact of such a framework is the ease in which new technologies are adopted. By offering the "glue" needed to incorporate a new technology in a SwA workflow, the framework expedites the "the time to impact" of novel technologies. It minimizes the burden placed on the technology developers who, in most cases, do not have the means to develop the required utilities needed to make their technologies accessible to end users.

An open SwA framework will allow a developer to choose the tools and technologies that best fit their needs, and compose them into complete automated workflows. Developers benefit from the variety of tools available at each stage and the experiences of other developers, as embodied in the stored workflows, who have used the framework to solve similar problems. Developers can also bring their own tools to the framework to experiment with new technologies and methodologies. As developers gain more experience with this global view of the SwA process, they can add more tools to their workflow, and expand their coverage of the problem space. SwA Researchers and tool developers can evaluate capabilities of different methodologies and technologies.

In recent years we have seen several organizations take steps to support a more flexible and composable approach to the software assurance process. These groups include Secure Decisions's CodeDX and Denim Group's ThreadFix tools, which merge and visualize results from multiple code analysis tools, and KDM Analytic's TOIF, which serves to merge analysis tool results and represent it in a common format. Rather than presenting the end user with one monolithic software system that covers multiple stages of the SwA process, the capabilities offered by these groups allow the integration and manipulation of results from different analyzers.

3 Framework Attributes

To meet the diverse and ever changing needs and expectations of the different groups that compose the SwA community, the framework will have to offer the following key elements:

- *An environment where new tools can be added easily and efficiently:* Tool developers and researchers should be able to bring their tools into the framework with no more difficulty than bringing the tool up on their own desktop. This means not only having simply uploading procedures, but also being able to work interactively to address porting issues. Ease of bringing tools into the framework also means that once a tool is available, the operation of running it against a software package should be fully automated. Such automation requires that the framework provide the glue to run the tool against software packages with complex and even non-standard build procedures.
- *An environment where new software packages can be added easily:* As with the case for tools, software package developers should be able to bring their software into the framework with no more difficulty than bringing the package up on their own desktop. Again, interactive access is critical to keep this process simple and familiar. Once a package is successfully built, the effort of the package developer should be done. The framework must provide the glue that automates running selected tools against the package, assessing the package exactly as it would be built, and handling complex directory structures, separate compilation, whole program analysis, and builds that produce multiple executables.
- *Support for tools that integrate and interpret the output of SwA assurance tools:* The step of automating importing tools and software packages, and running the tools against the packages, are only the first steps in the workflow. While running against multiple SwA tools provides a rich source of assessment information, this information must be unified, labeled and presented to the user in a way that allows them to understand it. The SwA framework must provide open access for tools that fill all or parts of this space.
- *Access to software products and results at all levels:* An SwA framework will include analysis products from each step of the workflow. These products include the raw results from SwA tools, normalized raw results in a uniform format, merged and interpreted and labeled results, annotated results that include feedback from the programmer or higher level tools. Tool developers and programmers must have the opportunity to access any of these results and share these results, while not being forced to depend on any of them. Common data representations are key to allow the choice of multiple tools at any stage of the SwA process, and to allow independently developed tools to interact with each other.

A foundation for understanding the process of software assessment: The body of data that will be created by an active and productive SwA framework provides a life history of the software development and assurance process. As such, this data offers raw materials for study to the researcher in software engineering, software assurance, risk management, and software business processes. For example, a researcher might be studying the productivity of a software assessment method or the effectiveness of various tools and techniques. Data can be provided to researchers in both raw and anonymized forms.

4 Foundation for an SwA Community

Despite the power a framework may eventually bring, it is not obvious that any technology in itself will bridge the growing gap between software and SwA. Hence it must at least contribute to a growing community of software developers and SwA researchers working to enable SwA through education and better software development practices. Two thrusts underway are the use of the SWAMP facility in education to teach SwA earlier, alongside software development, and the development of a body of software development practices, such as structuring software in such a way to be assurable. One lesson with the recent experience with Heartbleed was that software of sufficient complexity cannot be successfully analyzed to the point it can be assured. Rather than expecting the SwA tools to bring that gap, we may need the software to meet assurance half way.

5 Current Status

The need for an open and flexible SwA framework has guided the design and development of the Department of Homeland Security Science and Technology Directorate's recent initiative, the Software Assurance Marketplace facility. As a technology-neutral entity, the SWAMP is uniquely positioned to define, implement and evolve such a framework and to make it available to the SwA community. The SWAMP is currently operational with fundamental portions of the framework, primarily for software developers operational. We welcome comments and recommendations on the framework that will help us reach and extend its impact.

6 Acknowledgments

We acknowledge the funding of the Department of Homeland Security, Science and Technology Directorate in funding the SWAMP.

7 References

[1] Kupsch, James A., and Miller, Barton P. "Why Do Software Assurance Tools Have Problems Finding Bugs Like Heartbleed?" Continuous Software Assurance Marketplace, 22 Apr. 2014. https://continuousassurance.org/swamp/SWAMP-WP003-Heartbleed.pdf

[2] "SWAMP Capabilities." Continuous Software Assurance Marketplace, 12 Dec. 2013. https://continuousassurance.org/swamp/SWAMP-WP001-Capabilities.pdf