An Architecture for Dynamic Self-Adaptation in Workflows

Sheila Katherine Venero Ferro¹ and Cecilia Mary Fischer Rubira¹
¹Institute of Computing, University of Campinas (UNICAMP), Campinas, Sao Paulo, Brazil
ra144653@students.ic.unicamp.br, cmrubira@ic.unicamp.br

Abstract - Today several organizations use many kinds of Process-Aware Information Systems to support their processes. A typical example of such systems is Workflow Management Systems. However, due to the complexity of business processes and its continuously changing environment, there is an inevitable need to expand the dynamic behavior of these computational solutions. One of the drawbacks of the workflow platforms is that they usually cannot dynamically adapt their processes. The aim of this research is to develop an architecture for workflow management systems that provides means of flexibility to dynamically adapt the workflows during runtime. In order to validate our solution, a case study was conducted in nursing processes based in a real diagnosis scenario to show a practical applicability of the proposed adaptive architecture. Preliminary results have shown that the architecture successfully supports business logical adaptations.

Keywords: Workflow Management Systems; dynamic process adaptation; Process-Aware information systems; software architecture.

1 Introduction

Due to the continuous dynamism of the business environment, organizations should be prepared to respond to different situations and unpredictable events in order to maintain leadership. To do so, many kinds of Process-Aware Information Systems are used by the organizations to support their processes. Workflow Management Systems (WFMSs) are typical examples of such systems [1] which partially or totally automate business processes. WFMSs define, create, and manage the execution of workflows through software, executing one or more workflow engines that allow process definition, user interaction, and provide means for invoke IT tools and other applications [2].

Traditional Workflows Management Systems usually work with well-structured processes and typically for predictable and repetitive activities [3]. However, modern processes are often required to be flexible in order to reflect foreseeable or even unforeseeable changes in the environment. Thus, WFMSs face some limitations in means of flexibility; they cannot support dynamically changing the business process or just support them in a rigid manner [1]. Dynamic workflow changes can be either in a single instance or an evolutionary change in the process schema, so the adaptations can be at the instance level or at the type level [4].

Workflow technology can deliver the right information to the right person and at the right time reflecting all the changes. And if this technology could adapt the processes dynamically according to the environment or context they also would help in the decision-making process and certainly improve the efficiency of business processes to respond to unexpected changes. As an illustrative example, considering a nursing process, a system can alert the nurse that a patient is allergic to a particular drug, suggest other similar drugs or redefine on-the-fly the care plan of a patient, because when planning (at design time) it is almost impossible to prevent all the situations since the number of different possibilities is too high.

Lately adaptability in workflow technology is one of the hot topics in the academic world [5]. Nevertheless, just a few of approaches treat adaptation at the business logic level. Most of the approaches deal adaptation at the technological or performance level, treating exceptional behavior caused by errors in the application or errors in the infrastructure or middleware components on which the process runs. This paper is focused on dynamic adaptation in workflows at the business logic level, treating exceptional behaviour caused by the result of a breach of a business rule, a constraint violation, a data issue, or an unexpected behavior.

Most approaches support changes in the environment, failures, variations and exception using policy/rule-based frameworks [3, 6, 7, 8, 9, 10] and explicitly represent paths or schemas. Many of them use ECA (Event-Condition-Action) rules to define constraints. These kinds of approaches work well with well-defined business process; however they are not suitable for more complex and dynamic processes or for weakly structured or non-routine process. Other modern approaches uses knowledge-based techniques [11,12, 13, 14, 15, 16, 17], case-based reasoning [18] or ontology-based reasoning [19, 20, 21]. These kinds of approaches help with unpredictability and uncertainty of business processes, and with non-routine processes and provide human thinking and decision-making techniques that provide sufficient flexibility and adaptability for business process [13].

We believe that the usage of autonomic computing principles along with some cognitive capabilities can satisfactory cope with foreseen and unforeseen changes in
business process and permit adaptation at runtime. In this paper, we propose an architecture based on the MAPE-K reference model that provides flexibility for workflow management systems to dynamically adapt business processes to suit new conditions at runtime. This paper is organized as follows: Section 2 describes the background of this work. Section 3 shows the proposed solution. Section 4 presents the case study and finally Section 5 presents the conclusions.

2 Background

This section describes the theoretical foundation used in the proposed solution.

2.1 Workflow Management System (WFMS)

A workflow Management system can be defined as a software system that defines, creates, and manages the execution of workflows. It possesses one or more workflow engines capable to interpret the process definition, enable user interactions, and invoke different IT Tools and applications [2]. The workflow execution has a specific order of execution according to business logic [22].

![Figure 1. Workflow Reference Model - Components & Interfaces][2]

Figure 1 shows The major components and interfaces within the architecture according to [22], which are described below:

- **Process Definition Tools**: These tools define and model business process and its activities, translating them from the real world to a formal computerized representation. These tools may be supplied as part of a workflow product or as a separate. The interface between those tools and the runtime workflow enactment service is called the process definition import/export interface [22].

- **Workflow Enactment Service**: Provides a runtime environment in which one or more workflow management engines create, manage, and execute workflow instances and interact with external resources by means of workflow application programming interface. Several functions can be handled by the workflow engine, including interpretation of the process definition, control of process instances, navigation between process activities, sign-on and sign-off of specific users, identification of work items for user attention and an interface to support user interactions, maintenance of workflow control data and workflow relevant data, passing workflow relevant data to/from applications or users, an interface to invoke external applications and link any workflow relevant data, supervisory actions for control, administration, and audit purposes. It interacts with the external resources through the interfaces [22].

- **Process Definition Tools** are different tools that define and model business process and its activities, translating them from the real world to a formal computerized representation. These tools may be supplied as part of a workflow product or as a separate. The interface between those tools and the runtime workflow enactment service is called the process definition import/export interface [22].

- **Workflow Client Applications**: Activities that require involving human resources. The interface between these client applications and workflow engine interacts with the Worklist handler, responsible for organizing the user interaction with the process instance. It is the responsibility of the Worklist handler to choose and advance each element of the Worklist [22].

- **Invoked Applications**: Other potential applications without user interaction in a heterogeneous product environment. The interface allows the workflow engine to activate a tool to perform a particular activity, for this reason, there must exist a common format to transfer data among them [22].

- **Administration and monitoring tools**: Provide operations such as user management, role management, audit management, resource control, process supervisory functions, etc. [22].

There should be interoperability functions that provide communication between heterogeneous workflow systems [22].

2.2 Self-Adaptive Systems

A self-adaptive system adjusts its artifacts or attributes in response to changes. To accomplish its goal, it should monitor the software system (self) and its environment (context) to detect changes, make decisions, and act appropriately. The basis of self-adaptive software is the adaptation of dynamic/runtime changes [23]. This kind of software tries to fulfill its requirements at runtime in response to changes [24]. These systems make decisions on their own, using high-level rules and policies. They constantly check and optimize their status and automatically adapt themselves to changing conditions, keeping the system’s complexity invisible to the user and operators.

According to [24], to contemplate their goals they should have some features known as self-* properties: **Self-configuration**, it is the ability of automatic configuration of components according to high-level goals. **Self-optimization**, it is the ability of automatic monitoring and control of resources to ensure the optimal functioning. The system may decide to initiate a change to the system proactively in an attempt to improve performance or quality of service. **Self-healing**, it is the ability of automatic detection, diagnosis, correction, and recovery of faults. **Self-protection**, it is the
ability of identification and protection of malicious attacks but also from end-users who inadvertently make software changes.

[23] discusses these properties along with some others and provides a unified hierarchical set, Figure 2 describes a hierarchy of self-* properties in three levels. In this hierarchy, self-adaptiveness is in the general level which is decomposed into major and primitive properties at two different levels. The primitive properties are Self-awareness and Context-Awareness. Self-awareness means aware of its self-states and behaviors according to [25] and context-awareness means aware of its context its operational environment according to [26].

2.3 MAPE-K

MAPE-K is the reference model for autonomic control loops suggested by IBM [27]. It is a logical architecture that defines the main architectural blocks for building an autonomic manager either in a monolithic or distributed approach.

The MAPE-K derives its name from the main tasks in the feedback control loop of self-adaptive systems [27]:

- **Monitor**, it collects information from the managed resources. The monitor function aggregates, correlates, and filters the information until it determines a symptom that needs to be analyzed.
- **Analyze**, it performs complex data analysis and reasoning on the symptoms provided by the monitor function. This analysis is made with stored knowledge data. If there is a need of changes, the request is logically passed to the plan function.
- **Plan**, it structures the actions needed to achieve goals and objectives. The plan function creates or selects a procedure to enact a desired alteration in the managed resource to cope with the new situation.
- **Execute**, it carries out the adaptation, changes the behavior of the managed resource using effectors based on the actions recommended by the plan function.
- **Knowledge**, standard data shared among the monitor, analyze, plan, and execute functions. The shared knowledge includes data such as topology information, historical logs, metrics, symptoms, and policies. Knowledge is created by the monitor part while execute part might update the knowledge.

2.4 Intelligent Agents

An agent is an autonomous software entity situated in some environment where it takes autonomous actions to achieve their goals. They are capable of making decisions to proactively or reactively respond changes in its environment in real-time [28].

According to [29], agents are autonomous (operates without direct human intervention and control their internal states), social (interact with human and other agents), reactive (perceive changes in the environment and responds to it in a timely fashion), proactive (takes the initiative to satisfy its goals, goal-directed behavior).

2.5 Flexibility in Process

[30] defines flexibility as the ability to yield to change without disappearing, without losing identity. In processes, flexibility is defined as the ability to deal with both foreseen and unforeseen changes, adapting or varying the affected parts of the business process and maintaining the essential form of the parts not impacted [31].

According to [31] the flexibility types can be classified as: Flexibility by Design, design alternative execution paths within a process model at design time. The selection of the most appropriate path is made at runtime for each process instance. Flexibility by Deviation, a process instance might temporarily deviate at runtime from the execution path prescribed in order to deal changes in their environment without altering the process model. Flexibility by Underspecification is the ability to execute an incomplete process model at runtime. Placeholders are variable points marked as underspecified, where is not possible design the activities because the lack of information. E. g. Late binding, late modeling, etc. Flexibility by Change is the ability to modify a process model at runtime. This means to migrate all current process instances to the new process model.

3 The Proposed Solution

To enable dynamic adaptation in workflows, the proposed architecture (Figure 3) uses the reference model for autonomic control loops MAPE-K which provides to the workflow autonomic properties such as self-configuration, self-healing, self-optimizing, and self-protecting. With these self-* properties along with self-awareness and context-awareness, the workflow fulfills the requirements to be self-adaptive and dynamically respond to changes at runtime.

An important quality attribute of the architecture is the separation of concerns between the business logic and the application logic. The adaptation layer is separated from the workflow engine, therefore the workflow adaptation is performed in a meta-level, and consequently it is transparent.
to the workflow engine, so we do not modify the structure of any workflow engine. In addition, it gives us an easy and independent way to manage changes in the business process. This paper is focused in the adaptation layer.

The adaptation layer uses intelligent agents which make the workflow autonomous, proactive, reactive, and goal oriented. Since these operate without direct human intervention, can communicate to each other, perceive the environment, respond to changes, and take initiatives to achieve the goals. To achieve their goals, agents need to use knowledge to reason about its current state and the environment, to generate plans and execute them, for this reason, a shared centralized knowledge repository is provided. The knowledge repository is the core element of our architecture, which centralizes all the information related to the business goals, business rules or policies, case data, a process repository (set of activities that can be chosen at runtime), and other external knowledge repositories related to the business domain. According to this information, the agents monitor, analyze, make decisions, and plan new activities. The information should be represented in the form of declarative knowledge to do logical inferences and solve problems.

The adapter agent comprises the analyzer, the planner, and the simulator separated according to their capabilities or functions. All the agents share the knowledge repository.

The adaptation layer has a monitor agent, adapter agent, and executor agent, consecutively the adapter agent comprises the analyzer, the planner, and the simulator separated according to their capabilities or functions. All the agents share the knowledge repository.

The monitor agent continuously evaluates the current state of the process instance and its context provided by the context manager during all the process execution until it determines a symptom that need to be analyzed, that could be caused by the result of a breach of a business rule, a constraint violation, an unexpected data value or output or any other unexpected business behavior. This symptom is delivered to the analyzer.

The analyzer observes, identifies, and reasons over the situation according to the business rules and goals and diagnoses the situation. The analyzer counts with artificial intelligence techniques to make the data analysis, reason and make real-time decisions, the analysis is influenced by all the knowledge base provided. It is responsible for identify potential adaptations during the process execution. If adaptations are required, the planner is activated.

The planner creates or selects activities from the process repository, it checks if the detected situation has a solution, if not it makes some inferences of a possible solution also it counts with artificial intelligence planning techniques to reconfigure the activities, according with preconditions and postconditions, interdependencies between activities, business rules, etc. It reformulates and reconfigures the process at a high level of abstraction. The solution can be a single activity or a complex process. It also employs predictions techniques to make predictions about the planned activities, to watch their impact.

The simulator reproduces the activities and verifies if it has to do some extra changes in the process or if there are some inconsistencies if so, this new situation is passed to the analyzer. Otherwise, the solution is suggested to a domain expert, who will approve the proposal or further adapt it, after the solution will be learned. Depending on the particular application domain, processes can be only altered by the supervision of an expert.

The executor agent makes the process definition: creates a computerized representation of the process that will be interpreted by the workflow engine.

In order to validate our proposal, the next section presents a case study based in a real nursing case.

4 Case study: Nursing Process

This section presents an evaluation of the proposed architecture to prove its effectiveness and feasibility adapting workflows at runtime. Thus, the main goal of this case study is to analyze if the proposed architecture contemplates dynamic adaptation in workflows at runtime in the context of a nursing domain. We chose the nursing domain because the nursing process is typical example of a flexible process. In the nursing practice, nurses monitor patients during their treatments, execute different tasks according to each patient situation and react to unexpected situations. For these reasons explained above, we believe that a nursing process is a very
good example of a workflow with unexpected situations and changes, consequently is excellent to evaluate the proposed architecture.

4.1 Case Study Context

Nursing is a service provided to humans focused on assistance and patient care in different situations related to his health [32]. Nurses adopt practices based on scientific knowledge and develop a systematization of their processes. Thanks to this systematization, the nursing profession achieved its professional autonomy. With this autonomy, nurses can give a diagnosis, different from a medical diagnosis. Thus, nurses should have necessary knowledge and experience in order to take care a patient and make decisions about what treatment or procedures they may do.

A medical diagnosis deals with disease or medical condition. A nursing diagnosis focuses on the person and their physiological and/or psychological response to actual or potential health problems and life processes [33].

According to the Federal Nursing Council Resolution No. 358/2009, the nursing process is a methodological tool that guides professional care and nursing documentation of professional practice. The steps for realization of the nursing process (Figure 4) are: collect nursing data, nursing diagnosis, nursing planning, implementation, and evaluation of nursing. These five steps are interrelated, interdependent, and recurring [34].

![Figure 4. Nursing Process](image)

The first step, collect nursing data, is collect relevant data about the patient, family, and community, to identify needs, problems, worries, and patient’s human reactions [35]. This is basically through anamnesis and physical examination. The nursing diagnosis is the process of interpretation of the data collected, the decision making process based on the concepts of nursing diagnoses which form the basis for selecting interventions and expected results [35]. The nursing planning consists in determining priorities between diagnosed problems, setting the expected results for each problem and their respective prescriptions in an organized way. According of the expected results, a set of nursing interventions is planned. The implementation of the nursing plan attends the whole process to minimize risks, solve or control a problem (nursing diagnosis), assist in daily activities and promote health. Nurses should be constantly aware of both patient responses as well as their own performance because the human being is unpredictable and requires constant monitoring [36]. The evaluation is a deliberate, systematic, and continuous process of verification. It consists in follow the patient responses to the prescribed care through observation notes in the respective medical record. The nurse evaluates the progress of the patient, establishes corrective measures, and if necessary, revises the care plan.

The nursing process helps nurses to make decisions, predict, and assess consequences. It improves the nursing capacity to solve problems, make decisions, maximize opportunities and resources to form habits, and increase their expertise.

4.2 Research Questions

The study answers the following questions.

RQ1: Does the architecture provide means to adapt the workflow under new circumstances or unexpected behaviors during the workflow execution?

RQ2: What kinds of changes does the architecture support?

4.3 Case selection and units of analysis

In case studies, the case and the units of analysis are selected intentionally [37]. So, the main selection criterion was that the nurse case scenario should present unexpected situations or events and provide us a good description of the events in the nursing process. But also the selected unit was limited to its availability. The unit for analysis was selected because its extreme behavior, the real-life medical scenario analyzed is described in [38].

4.4 Data Analysis

The data was collected through a documentary analysis. The medical scenario was reconstructed in order to get the sequence of events to be mapped into the nursing process. We made a time-series analysis in order to denote the set of events that happen over the time and executed a simulation of a process instance (a nursing scenario) in order to verify if the architecture supports all unexpected situations that happen in this nursing scenario. The domain was modeled as follows:

**Knowledge Repository**

It was populated with relevant information of the nursing domain:

- Normal rates for vital signs → Business Rules
- NANDA (North American Nursing Diagnosis Association) → Business Rules
NOC (Nursing Outcomes Classification) → Business Goals
NIC (Nursing Intervention Classification) → Process Repository
Medical Record → Case Data
Drugs information’s, evidence-based medicine → External Knowledge Repositories

**Context information**

- New symptoms (Adverse reactions, allergies, etc.)
- Abnormal vital signs
- Contraindications
- Active health problems
- Active medications

The procedures of each part of the adaptation layer were:

**Monitor agent:** Continuously analyzes the case data until some unexpected events happen; new context elements appear or are identified by the context manager.

**Adapter Agent:** Divides tasks between the analyzer, planner, and simulator.

- **Analyzer:** The analyzer observes, reasons about the events, characterizes the situation according to the NANDA taxonomy, search for the defining characteristics, analyze the medical record, and finally give the nursing diagnoses for this situation.
- **Planner:** With this diagnostic, the planner searches interventions (NIC) for the diagnoses in order to create a care plan and structures the activities needed to achieve the goals and objectives (NOC).
- **Simulator:** Finally, the simulator simulates all the process and search for potential inconsistencies or dangerous situations. If some problem is detected, the process is replanned. Finally it will be suggested to the nurse. After her approval, the process is stored in the process repository.

**Executor Agent:** After that process, the executor agent will make the process definition to be delivered to the workflow engine.

This cycle is repeated several times in the execution of the nursing scenario and this information is all the time stored in the case data.

### 4.5 Threats of validity

- **Construct Validity,** The threat of validity is that the chosen nursing scenario is a real life case documented in [38], so it counts with expertise of nurses.
- **External Validity,** it is not possible to say that the study case is exhaustive. Other studies for different processes and domains have to be performed.
- **Internal Validity,** the simulated scenario is a real-life nursing case chosen because it shows an extreme nursing case with many unexpected circumstances. The proposed workflows were validated by a nurse.
- **Reliability,** this study presents a limitation related to its results, which will be considered only as evidence.

### 4.6 Results

**RQ1: Does the architecture provide means to adapt the workflow under new circumstances or unexpected behaviors during the workflow execution?**

Based on the simulated situations during the execution of the nursing scenario, it was shown that the architecture has means to adapt the processes at runtime: the monitor agent, the adapter agent make possible the process adaptation using the knowledge repository and compose the process in high level of abstraction to finally the executor agent make the process definition at runtime in order to be interpretable for the workflow enactment service. The workflow adaptation is only possible with a vast knowledge repository. The dynamic adaptation is imperceptible to the workflow enactment service because it is being made in a higher level of abstraction.

**RQ2: What kinds of changes does the architecture support?**

The architecture supports both foreseen and unforeseen changes in process instances. The foreseen changes are previously modeled in the process repository at design time, so the architecture provides flexibility by design. The unforeseen changes are supported by the adaptation layer at runtime. The nursing planning is a step in which there is insufficient information at design time, so as the architecture provide means to execute an incomplete process model, the architecture provides flexibility by underspecification. Flexibility by deviation is also supported by the architecture because during the process execution we deviate several times from the initial plan to cope changes in the context over the time.

Thus, the proposed architecture contemplates dynamic adaptation in workflows at runtime in the context of a nursing domain. We must mention that for this domain, it is very important to know that every taken decision (diagnosis, interventions, and activities) by the agents, it is first suggested to the nurse (domain expert) who approves it or change it. After its acceptance, the suggested workflow is saved as an experience then the workflow is created to be interpretable for the workflow engine. The suggested information helps medical professionals to identify unexpected situations and make better clinical decisions, nursing diagnosis, and nursing interventions.
5 Conclusions

In this paper was proposed an architecture for workflow systems, we focused in the adaptation layer that permits dynamic adaptation in workflows. The adaptation layer is based in the autonomic control loop MAPE-K that provides the means for adaptation, each module of the MAPE-K is represented by an intelligent agent that follows business rules in order achieve the business goals. The agents are capable to make decisions in order to deal with unexpected changes in both “the self” and “its context”, proactively work together to achieve the business goals and learn from its experience. We believe that our approach is simple but potent.

In order to validate the solution, a case study was made in the nursing domain. An extreme nursing case scenario was simulated in the architecture in order to illustrate the use of our approach. The proposed architecture has showed its capacity to support different situations and dynamically adapt the process according to unexpected circumstances. Therefore, the architecture supports dynamic adaptation at runtime.

Our approach is part of an ongoing work. As such, much remains to be done. As future work, we plan to develop a system that performs dynamic adaptation in workflows in order to empirically evaluate the proposal.

Acknowledgment

Thanks to the National Council for Scientific and Technological Development – CNPq – for financial support.

References