Empirical evaluation of software development methodology selection consistency: A case study using Analytical Hierarchy Process

Benson Moyo¹, Gonde, Peeps², Ndabezinhle Soganile¹, Gilbert Dzawo¹, Kudakwashe Madzima¹

¹Computer Science and Information Systems Department, University of Venda, Thohoyandou, Limpopo, South Africa; {benson.moyo, ndabezinhle.soganile, gilbert.dzawo, kudakwashe.madzima} @univen.ac.za
²Computer Science Department, National University of Science and Technology, Bulawayo, Zimbabwe peeps.gonde@nust.ac.zw

Abstract - When developing software, the selection of an appropriate software development methodology is an essential decision. The experience, knowledge, expertise, of the software developer and organizational development context are assumed to have a great influence in selecting a methodology. In this research, we examine factors affecting the selection of software development methodologies and the consistency in which the methodology selection process is carried out. Based on Analytical Hierarchy Process (AHP), we evaluate the consistency in software development methodology selection in a particular software development company. We investigate the importance of a number of factors by first soliciting the criteria from practitioners before methodology selection and then later observing the actual implementation of a software development methodology. The paper identifies the predictor variables for development methodology selection and the dynamics triggered by situational variables. The results of our findings as well as recommendations for further work are presented in this paper.

Keywords: Software development methodology, methodology selection consistency, AHP

1 Introduction

One of the most critical decisions when developing software is the selection of an appropriate software systems development methodology. It is believed that a rich repository of systems development methodology exists [1]. Conversely, there is no universally accepted documented guide on how to select software systems development methodologies from a myriad of systems development methodologies in existence. Some empirical studies indicate that systems development methodologies are selected and used in practice [2]. It is possible to select, tailor and adapt systems development methodologies and/or methodology segments to specific systems development context [3],[4],[5]. The amount of expertise and time needed to select, tailor and match methodologies may present stumbling blocks. The experience, knowledge and expertise of the systems developer is assumed to have a great influence in selecting a methodology [6]. Naumann and Palvia[7] posit that selection is biased towards experience and familiarity with the methodology.

Selecting systems development methodologies or a system development methodology from many available options is not only demanding, but also confusing as often selection criteria or guidelines might neither be clearly stated nor justified [3],[6]. Naumann and Palvia[7] note that selecting a systems development methodology from the numerous existing methodology classes is a challenge with technical, social and financial consequences. Not only is the difficulty presented on the selection among methodology classes, but also on the instances of these methodology classes. Iivari et al.[9] presents a classification of methodologies in an effort to demystify the tenet of “methodology jungle” identified by Avison and Fitzgerald[10], however, the suitable methodology search space is a nondeterministic polynomial hard problem. Despite the complexity of selecting a suitable systems development methodology, it is prospected that an appropriate methodology should standardise the development process, organise work and resources and direct appropriately the perception of each member of the development team [11],[3].

2 Related work

This section overviewes methodology models and frameworks proposed in literature.

2.1 Software development methodology selection

There is a growing literature on development of methodology selection theory, frameworks and models. It has been shown that a single systems development methodology is not sufficient to address the requirements and demands of all existing scenarios of systems development [11],[12],[13]. Avison and Fitzgerald [14] express the basis for methodology selection as the target problem domain. Yaghini et al. [15] propose a methodology selection framework based on a multi-faceted approach. Methodologies are first classified as hard or soft and then compared according to six basic features; the philosophy, systems development model, systems
development scope, systems development tools, systems development background and participants. This model has limitations for example, it compares Soft Systems Methodology [16], and Structured Systems Analysis and Design Methodology (SSADM) which are methodology instances grounded on different paradigms [9], therefore they may not be viewed as competing as they have different philosophical assumptions. The criteria for determining the scope of each systems development phase is not precisely and explicitly stated in this model. Scope problems are inherent in a methodology as one of the dimensions of inconsistency [17]; therefore the selection framework might suffer from objectively scoping the phases. This model introduces only a set of six methodologies and it would be challenging to include any methodology not included in the list provided.

Naumann and Palvia[7] present a selection model centred on quantitative scoring method called Delphi to collaboratively evaluate and recommend essential methodology functions. The candidate methodology is selected based on the scores awarded to it. The drawback of this model is the subjectivity of the methodology function definition and the concentration on the system development techniques and neglecting the other methodology components. Cockburn [18] put forward a decision model based on evaluating appropriateness of each member of the Crystal methodology family instances to a target systems development problem domain. The stumbling block of this model is its being restricted to a limited methodology instances. Rashmi and Anithashree [13] recommend a selection framework for Rapid System Development (RSD) Methodologies built on a comparative analysis of a set of essential aspects of rapid development methodology instances under consideration. However, this selection is limited to Rapid System Development (RSD) Methodology family. Burns and Dennis [19] advocate for a two-dimensional framework for selecting the most suitable systems development methodology. This contingency strategy classifies projects in terms of project complexity and uncertainty factors [19]. The project complexity is determined by four aspects: the project size, the number of system users, the quantity of new generated information, and the complexity of generating new information [19]. On the other hand project uncertainty consist of three characteristics; the level of structure, the extent of users’ knowledge on their duties and system developer’s experience and expertise. The methodology selection process in this strategy involves a straightforward reading of the two dimensional array contents based on the level of complexity and uncertainty of the project. However, the drawback of this selection strategy is that it considers only two methodology instances. Yusof et al. [6] present yet another variation of selection criteria based on complexity and uncertainty, quality criteria and scope of methodology phases as key factors. The researchers select eight methodologies and state that they are the most common ones and in addition they give a formula for calculating the score for each methodology. Perhaps the drawback of this model is the determination of methodology scores. The approaches mentioned so far have a large likelihood of subjectivity when selecting a software development methodology.

Zhu [20] suggests three contingency approaches to software systems development methodology selection grounded on the dynamics of situational variables. The first strategy is “contingency at the outset” [20] and assumes contextual variables as static and thereby allows the selection of a methodology or methodologies prior to the development process and when chosen, a methodology or methodologies remain(s) invariant up to systems development project completion. The methodology and the contingency variables achieve a state of equilibrium throughout the development process.

The second strategy is “contingency with a fixed pattern” [20] which permits deterministic selection of systems development methodology or methodologies as the development process progresses. The possible future adjustments, variations and reconfiguration of the situational variables are considered predictable and follow some known archetype. The strategy assumes specific predictable expectations in different phases of a systems development life cycle. Systems development phases form decision points and allow the systems development methodology to be changed at each stage of systems development [21].

“Contingency along development dynamics” [20] is the third strategy which relies on selecting methodologies, or/and methodology fragments, tools and techniques into the development process as dictated by the dynamics of the evolving software systems development context. The strategy suggests a high level of uncertainty in the development process and therefore does not prescribe any set of systems development methodologies prior to any confrontation with particular contingency variable configurations at any point in time. This strategy allows multiple-decision points throughout the systems development process. In each stage, therefore it is assumed that new contextual features emerge that demand appropriate methodologies, or/and methodology fragments, tools and techniques to be employed. The suitability of systems development methodology is viewed as the achievement of equilibrium between the methodology and the situational variables. Therefore systems development methodology has to be adjusted from time to time in order to maintain development variables equilibrium.

2.2 Software systems development contingency variables

Systems development problem situations are different, some development environment are well-understood while some are ill-understood. These different systems development circumstances demand different methodologies to handle them if predictable results are expected. Even in single organisational settings the contingency variables configuration may differ on a software project to project basis. Carroll [3] found in a particular case study, that contingency factors affected the selection of methodologies throughout the development process. The contingent factors strategy suggests
that each development situation demands an appropriately selected methodology from a portfolio of methodologies. However, the challenge is that there is no single repository with all the methodologies compared and contrasted, classified and analysed on their normative principles, strengths, weaknesses, and contextual appropriateness. Methodology engineering goes a step further to suggest selection of methodology fragments from a repository and construct an appropriate framework or adapt, configure or tailor methodologies to fit the specific systems development projects. However, experience and a high degree of expertise may be needed to apply this strategy. The derivations are more biased on theoretical deductions than empirical evidence which make them more of pieces of advice on what should be done.

Systems development contingency factors may be considered at both micro and macro levels. Micro-context level deals with specific localised and bounded systems development problem situation. The micro-context level dynamics may include how the methodology deals with the social, technical, management, and economic factors confined to a same development environment or a similar development environment. Organisational structure and culture, each systems development team member’s previous experiences, existing knowledge, tacit knowledge, skills, culture, roles, rights and level of expertise constitute part of both social and technical contextual factors.

The macro-context level tends to be universal and may impact on micro-context level dynamics [22]. Ghaffarian [22] explains one of the reasons for the failure of the Effective Technical & Human Implementation of Computer-based Systems (ETHICS) methodology to propagate as probably the development context level dynamics. It is our contention that the development context is imperative when selecting a systems development methodology. Each methodology selection should be based on project to project specifics and the choice should be consistent and rigorous.

It is assumed that organisations should be able to select a systems development methodology that is best suited for a specific systems development project. Unfortunately, not much research has been performed to guide organisations in this regard. Research into contingent use of systems development methodologies is relevant to organizations aiming at selecting suitable methodologies for specific projects, in specific organisations, with specific organisational cultures and political structures. The appropriate selection of a methodology is purported to reduce failure probability. It is also expected to increase systems development process efficiency, improve quality of developed systems, and deliver systems on schedule and within budgetary constraints. Organisations are aware of the software crisis and the implications of project failure on reputation, employee morale, costs and business continuity. This is probably one of the reasons of sticking to one proven and tested methodology to avoid uncertainty associated with a new methodology.

All illustrations, drawings, and photographic images will be printed in black and white. We recommend that you examine a printed copy of your paper (in black and white) and make the final adjustments before submission. All illustrations must be numbered consecutively.

3 Methodology

In this work we endeavour to investigate consistency between what the practitioners say they do and what they really do. We aim to explore software systems development methodology selection consistency. Therefore we use the Analytic Hierarchy Process (AHP) as a rigorous subjective multi-criteria decision evaluation tool.

Given a set of software systems development methodologies, preference of one from the other can be established through knowledge solicitation techniques like observations, questionnaires or interviews and subjecting the data collected to a thorough statistical analysis. Selecting a software development methodology is a multi-criteria decision making process. AHP converts a multi-criteria decision making process into the solution of an Eigen value problem. Eigen values have their greatest significance in that dynamic problems tend towards a steady state under some mathematical operations. The appeal of AHP in the selection process is on its ability to verify consistency of subjective measures. Ratio scales are derived from paired comparisons and both quantitative and qualitative measures can be scientifically verified and validated. The ability to detect inconsistent judgements makes it a good candidate for selecting a software systems development methodology. New ideas and methodologies are viewed as prone to failure and risky. Learning from failures is not acceptable to organisations as failure may have a serious negative impact on reputation, employee morale, and continuity of business. The essence of AHP involves the construction of a square matrix expressing the relative values of a set of attributes. For example: What is the relative importance to the developers the market window of a software system as opposed to quality of the software system? What is more important responding to change over sticking to a plan? The fact that the human mind is capable of making a single pairwise comparison at any given time is taken advantage of in AHP. Each selection made is mapped onto a numerical value.

The structure of a problem is comprised of a hierarchy of components in terms of a goal, criteria, and alternatives. The priority setting of the criteria based on pairwise comparison allows the determination of the relative importance of the criteria within each level. The typical question is “How important is social issues interaction relative to technical issues?” The respondent selects from descriptive comparisons. The selection is then mapped into a numerical scale that expresses the intensity of importance. The values range from 1(Equal importance-when two activities contribute equally to the objective) to 9(Extreme importance-when evidence favouring one activity over another is of the highest possible order of affirmation). The numbers 2,4,6,8 represent
intermediate values where a compromise has to be met. The reciprocal of each of these values is assigned to the other criterion in the pair. The weightings are then normalized and averaged in order to obtain an average weight for each criterion.

The case study was conducted in one registered Software Development Company in Zimbabwe from 2010 to 2012. The Company’s core competency is in application software development and it permitted one of the research collaborators to participate. Within this period a total of four new software systems development projects are carried out and other activities involved client support services to already deployed systems. In the start of each project the researcher presents a questionnaire on the selection of development methodology. It solicits a simple pairwise comparison of organisational, project and systems development methodology characteristics. The interpretation of the responses is done using the AHP Fundamental Scale [24]. In case of group selection, a geometric mean is used to aggregate the individual choices into a single representative judgment [24].

The software methodology complex decision problem is structured as a hierarchy as shown in Figure 1. We assume three-tier architecture in decision making. The first layer entails the main goal (select the most appropriate software development methodology); second level has criteria and sub-criteria, and alternatives at the bottom layer. The stakeholders at criteria layer include the analysts (who are leading the proposals for development), management (who have to buy into the project for support), the programmers, and users (the clients of the system).

In Figure 2 we indicate the priority vector to guide the decision. At second layer the project team is the most significant factor in the selection of a software development methodology. It comes slightly ahead of stakeholders. During the study, resignations of a member of the team lead to change of software development methodology. Again expertise is relatively valued highest among experience, team size, and team distribution by location. The values may vary but fundamentally the importance of project team will show some strong intensity. Project complexity is relatively more significant than project uncertainty, however at one decimal place precision these have equal importance.
Alternatives are evaluated against each criterion. It is observed that structured methodologies may deal well with maintainability, organizational politics, and expertise; however, the stumbling block would be on handling requirements volatility. Object oriented handles well organizational politics, maintainability, expertise, moderate on requirements volatility. Agile methodologies deal well with organisational politics, team size, moderate on maintainability, and excellent on requirements dynamics. Lastly agile methodology is slightly more preferred than neither structured nor object oriented methodologies. However, expressing the decision values correct to one decimal place there is no difference in the appropriateness of these methodologies.

4 Discussion

The methodology selection is a fundamental exercise in and of itself. The use of AHP may expose consistency or inconsistency in the selection process of a software development methodology. The selection originates from the goal, trickles down to criteria, sub-criteria, and finally to the alternatives. There seem to be a praxis gap in the selection of a software development methodology due to the fact that a practitioner may strongly evaluate one alternative theoretically superior over another, however, practically recommends another.

5 Results and conclusions

An interesting observation is made during the implementation of a selected systems development methodology. An expert who is one of the well experienced members of the project team resigns from the company. The methodology initially selected is re-evaluated and dropped as part of response to contextual dynamics in systems development. Change in project characteristics may result in change or modification of a systems development methodology.

The AHP was applied to systematically and consistently evaluate the selection consistency of systems analysts who are basically tasked to select the most appropriate systems development methodologies amongst a myriad of existing alternative classes and instances of methodologies.

The study shows that there is need to investigate existence, adoption and use of systems development methodology selection frameworks. Our study contributes to the pool of knowledge in systems development in the following ways. Firstly, the conducted critical analysis of the prior literature on systems development methodology selection helps confirm the knowledge gap in this area. The
important features for selecting a methodology are identified and the dynamics of the development process observed. Second, we investigated consistency in selecting methodology by the relevant actors. The suggested attributes can be used to understand the impact of systems development contextual variables on methodology selection.

One of the possible limitations in the study is that a single organization was considered. Generalization of the results of the study is limited as the study is based on a case study. Case studies are powerful to get the deeper understanding of a particular phenomenon in its actual settings but not for providing general predictor variables for the phenomenon. The findings are considered as a trigger to explore more in the area of systems development methodologies selection.

Further research work can be done on how the selection criteria are used. What are the criteria to select a software development methodology and change it or modify it during the development process? In the first place what is the threshold to change, tailor a selected software development methodology?

6 Acknowledgements

Although it is impossible to give credit individually to all those who participated and supported the project on software methodology selection consistency, the authors would like to express their gratitude and appreciation to all. This research is supported by the University of Venda.

7 References