UTM Computing Proceedings
Innovation in Computing Technology and Applications
Volume: 2| Year: 2017 |ISBN: 978-967-0194-95-0

Comparative study on differences of various agile methodologies and supporting tools in software maintenance

Nur Atiqah bt Mohamed Adanan [1] and Dayang Norhayati bt Abang Jawawi [2]

[1] Faculty of Computing, Universiti Teknologi Malaysia (UTM), Malaysia
[2] Faculty of Computing, Universiti Teknologi Malaysia (UTM), Malaysia
ikadanan@gmail.com, dayang@utm.my

Abstract. Software maintenance has always been a critical issue in every software project. Maintenance of a software product has been estimated to take 40 percent up to 70 percent of the life-cycle cost. Maintenance of aging software tends to become more difficult year by year since updates gradually destroy the original structure of the application thus increase its entropy. Aging software may also contain troublesome regions with very high error densities called as error-prone modules. Due to the undiscovered flaws, it is common that numbers of defects have been passed from the development process to the maintenance process. Therefore, software maintainer teams always found out it hard to face tremendous amount of changes throughout a rigid timeline. A rigid timeline demands an exhaustive planning, in a way of keeping a balance between amount of responding to users' change requests yet meet time and cost goal. Every change has the massive power of altering the original structure of the exisiting operational software system, thus governing the success of a software maintenance project. Time is a main driver for determining number of changes that can be resolved, whereas cost is a main consideration in other cases. Ergo, exerting traditional methodologies on maintaining a plan-driven software system in this modern unpredictable IT world is the worst mistake. A little over three decades ago, software engineering field experienced a lot of problems and obstacles by implementing the traditional methodologies on maintaining process of a software product. Nonetheless, emergence of agile manifesto in 2001 has brought a huge upshot to the modern unpredictable IT world. Ever since, numbers of software projects have changed their existing methodology to a more flexible method which is Agile methodology.

Keywords: Software maintenance, Agile methodologies, DSDM, Scrum, XP, FDD

1 Introduction

Maintenance has been defined as **"**the work of keeping something in a proper order.**"** In the context of Software Engineering field, it is referred as the maintenance of a software product. Based on former research studies, software maintenance can be enunciated as a process of modifying existing operational software system while keeping its primary functions intact [1]. Software maintenance can be said as the continuity of software development process. Software maintenance generally includes sustaining engineering and developing new functions, corrective changes, adapting to new requirements as well as perfecting or improving the existing functions [1]. Capers Jones said series of studies estimated that within thousand lines of software codes, there are ranges from 49.5 to 94.5 percent of defect density occurred during the development process [2]. Thus, in addition to the undiscovered flaws, it is common that numbers of defects have been passed from the development process to the maintenance process [1].

Software development methodology is known as a formalized approach that used to plan and manage the process of developing a software system [3]. As maintenance process is the continuity of the development process, thus same type of software development methodology may be used in maintaining a software product. Since there are many software development methodologies, it has been a challenge faced by software developer teams to decide the suitable software development methodology to be applied in a software development project [3]. In fact, it is possible that the maintenance of a software product uses the same methodology as the previous development process, but changing to a new methodology takes a whole new challenge perspective. Based on series of former studies, there are numbers of

software project that changes their existing methodology to new ones during the software maintenance process as they aimed to improve the flexibility and agility on maintaining their software product [4], [5].

Waterfall is one of the software development methodologies that is generally used. It was formally introduced as an idea by Winston W.Royce [6] whom critized sequential development in 1970. He asserted that software should not be developed as an automobile on assembly line, in which piece is added and completed before the next phase can begin. Waterfall methodology has been known for years among the software developer as well as maintainer teams that emphasized meticulous planning and output comprehensive documentations. Besides, Waterfall methodology is predictable due to its sequential software development process and rigid timelines [3]. However, fault initial requirements collected at the beginning of the projects can affect the quality of the final software products. Plus, customer will not have the abilities neither to change scope nor add requirements once the project has begun. Hence, Waterfall methodology is recommended to be used if there is a clear picture of what the final products should be.

To defeat rapid change in the organization and business need in traditional methods, [7]–[9] there have been several studies and ideas on improving and enhancing the existing software development process thus a new methodology emerges in February 2001 [10]. This methodology is called as Agile methodology. It is an alternative to the traditional software development methodologies and is mainly focusing on people [9], [11]. Customer and each team member in Agile team can either be the key success or failure factor in an Agile process. Agile methodologies are a set of software development rules that are based on iterative development. Each phase in the software development process is added incrementally, and iteratively been evaluated and executed throughout the software development process [11]. Thus, Agile methodologies are known in promoting rapid and flexible response to changes, simply known as agility [11].

Taking in consideration of unprecedented requirement changes rate in the software development process, consequently leads to the search of methodologies that can deal with such circumstances [8], [9], [11]. No doubt that Agile methodologies have the capability on dealing to rapid changes. This research focuses on the comparative study on differences of each Agile methodologies and tools in software maintenance. This include identifying and analyzing two most suitable Agile methodologies to be implemented in the maintenance process of MySEFAM plan-driven software system. Hence, a comparative study on Dynamic Systems Development Method (DSDM), Scrum methodology, Extreme Programming (XP) methodology, Lean Software Development methodology, Feature Driven Development (FDD) methodology and Crystal methodology for maintaining a plan-driven software system using Agile methodologies has been carried out.

2 Background and Motivation

Software development methodologies encompass programming, documenting, testing, and bug fixing. While this remains valid for software development process, choosing new suitable Agile methodologies for maintaining a plan-driven software system opens a new perspective.

[9] had carried out a study on examining current Agile methodologies and practices, understanding strengths and weaknesses of Agile methodologies and various issues of their applicability. They had used integrated research methodologies, both qualitative and quantitative to meet their research objectives. However, the research did not clearly underline the strengths and weaknesses of Agile methodology as well as highlight the issues on applicability. In an advance study, [11] had referred to [9] and studied more on the strengths and weaknesses of Agile methodology. Nonetheless, the findings were too general, rather than emphasizing and focusing on strengths and weaknesses of each Agile methodology. As most of choosing a methodology process is based on its existing strengths and weaknesses, thus researches are supposed to clearly defined and described them on their studies. Therefore, a research on describing the differences of strengths and weaknesses of each Agile methodology still needs to be done.

[3] had proposed selection criteria on selecting the appropriate software development methodology which comprise of project time, clarity of user requirements, familiarity with technology, system complexity, system reliability and schedule visibility in context of Expert system (ES). The findings resulted Prototyping, Iterative development and XP, which turn out to be the suitable methodologies when flexibility of project is concerned. Furthermore, researchers in [12] studied on the comparison between variations of traditional development and variations of Agile development. The research stated that Agile methodologies can incorporate changes more easily

and demonstrate business value more efficiently than traditional projects especially XP. Thereby, both researches had demonstrated that Agile methodologies do support flexibility issues compared to traditional methodologies. Nevertheless, this research is still in need of improvements and future works to cover up more Agile methodologies.

[13] explored a detailed comparative analysis of two well-known Agile methodologies which are XP and Scrum using previously published 4-Dimensional Analytical Tool (4-DAT). 4-DAT analysis is based on four characterization perspectives: those of scope, agility, Agile values and software process. Its analyzation used both qualitative and quantitative approach to evaluate Agile both at the phase level as well as at the practice level. A report generated from 4-DAT assists organizations in making decisions about the selection or adoption of Agile methodologies. However, a study by [14] found out a useful tool for making out decisions which is SuperDecisions software. SuperDecisions implements Analytic Network Process (ANP) in its implementation of making decisions. ANP is a generalization of the Analytic Hierarchy Process (AHP) which a theory of relative measurement with absolute scales of both tangible and intangible criteria based on the judgement of knowledgeable and expert people. Thus, there is a need to study in deeper the effectiveness and efficiency of both decision methods regarding on the decision-making.

[4] has carried out a research to address the gap of a large-scale implementations of Agile sets and incremental practices through an industrial case study at Ericsson AB. Ericsson AB is a telecommunication company that has experienced the effects of changing from plan-driven method to a more flexible method which is Agile. This study focuses on comparing the issues and advantages of using Agile and incremental methods extracted from the existing studies and the case study as well as describing the changing effects to Agile methodology indirectly. Besides, [5] outlined an Agile evolution and maintenance process models as well as evaluated it within two Canadian software organizations. Both companies have implemented two most widely accepted models which are Scrum and XP during their maintenance process, then the effects of the implementation have been recorded. However, there are still a need to study more regarding the maintenance issues of changing from plan-driven methodology to Agile methodology.

Unfortunately, differences of strengths and weaknesses of each Agile methodology, methodologies that promote Agile Manifesto other than XP, the effectiveness and efficiency of 4-DAT analysis compared to SuperDecisions analysis, the effects of maintaining a plan-driven software system in Agile environment, and either different Agile methodologies can work together or does not remains largely a black box. As a conclusion, this research aims to fill this literature gap by addressing the comparative study on differences of various Agile methodologies in software maintenance.

3 Research Methodology

This research has been carried out within four phases which are the first is, studying and identifying phase. At this phase, several chosen Agile methodologies and supporting tools have been studied. The findings will be a guidance to to identify and analyze two most suitable Agile methodologies and supporting tools to be used in the maintenance phase of MySEFAM plan-driven software system. The next phase, the two Agile chosen will be implemented to MySEFAM plan-driven software system by following their respective key practices and principles separately then, the limitations of each of them will be studied and analyzed. Next, the third phase, an integration model of these two chosen Agile software development methodologies will be proposed and implemented in the next version of MySEFAM software system. Final phase, the evaluation of the suitability of the integration and effectiveness of the integrated process modelling is studied.

3.3 Phase 1

As an initial work, a careful and thorough study on the differences various of Agile methodologies name specifically DSDM, Scrum methodology, XP methodology and FDD methodology in general context besides the supporting tools. A comparison study has been carried out to prove a distinctive deviation of each Agile methodologies based on their strengths and weaknesses in general context. Identification to select two most suitable Agile methodologies and supporting tools to be used in the maintenance of MySEFAM plan-driven software system has been conducted. Previously published decision-making frameworks which are 4-DAT and SuperDecisions software have been used on assisting the selection of the Agile based on the identified suitable terms in Phase 1. The results of each framework are then being compared to gain relevant choices.

3.4 Phase 2

The chosen Agile methodologies and supporting tools have been implemented separately in the maintenance process of MySEFAM plan-driven software system. The first stage, first chosen Agile key practices and principles as well as supporting. tool have been implemented to MySEFAM plan-driven software system. After the completion of the first stage, the second implementation has been carried following the second chosen Agile key practices and principles as well as supporting tool to MySEFAM plan-driven software system. The implementation results of both implementation have been studied and evaluated. A conclusion has been made for proposing an implementation of integration of both Agile later.

3.5 Phase 3

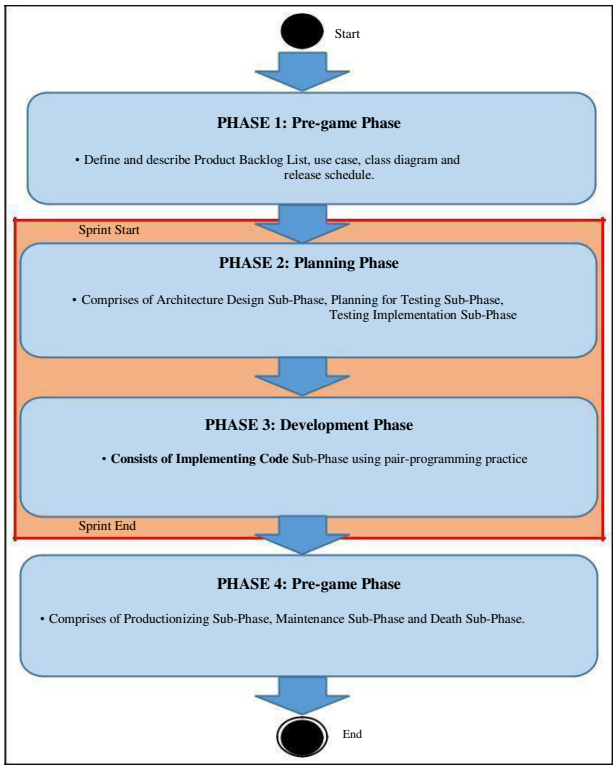
Subsequently, based on the findings studied in Phase 3 both software process models have been integrated. The integration software model has been implemented to the next version of MySEFAM system maintenance. The integrated software process model has been analyzed throughout the maintenance process of MySEFAM system. The limitations found from the implementation of the integrated software process model have been evaluated. A conclusion has been made on the suitability of the integration of both chosen Agile and the effectiveness as well the efficiency of the new integration of software process model has been made.

3.6 Phase 4

In the final phase, the evaluation has been performed. The integrated process model has been evaluated on its suitability of the integration of both methodologies. The effectiveness of the integrated process modelling is also studied and evaluated through its limitations during the implementation in MySEFAM software system.

4 Results

An integration software process modelling has been constructed based on Scrum and XP process modelling, flow as well as their key practices. The integrated process modelling comprises of four main phases which are P1: Pre-game Phase, P2: Planning Phase, P3: Development Phase and P4: Post-game Phase.



Phase 1: Pre-game Phase

At the beginning of a software project, an exploration of the requested requirements has been studied and analyze. In this phase, Product Backlog List (PBL) which is contains all the currently known requirements are performed. These requirements are prioritized and the effort needed is estimated. For better understanding of the requirements, a use case as well as class diagram of a software system are also constructed. To cope with the project, a release schedule is constructed as it provides a clear guideline of the project flow.

Phase 2: Planning Phase

After the requirements have been fully explored, the Sprint is started. In the Planning Phase, there are three other sub-phases which are Architecture Design Sub-Phase, Planning for Testing Sub-Phase and Testing Implementation Sub-Phase. In the Architecture Design Sub-Phase, Sprint Backlog List, use case description, sequence diagram, activity diagram as well as user interface design have been performed. Under Planning for Testing Sub-Phase, test-driven development technique has been used to assist on developing a tested code developed and the implementation of the tested code is in Testing Implementation Sub-Phase.

Phase 3: Development Phase

After the testing has been implemented in previous phase, in this phase codes that followed the test cases constructed. The code is developed following the test cases, thus it will pass all the testing process in future. The last iteration marks the that the software product is ready to be release.

Phase 4: Post-game Phase

Under Pre-game Phase there are three other sub-phases which are Productionizing Sub-Phase, Maintenance Phase and Death Sub-Phase. After the software product meets all the requested requirements, then it will be passed to Productionizing Sub-Phase for future release. Meanwhile, if the software product does not meet the requirements then it will be discussed either to improve in the current release or the next release. Death Phase is when the requirements are no longer valid or the maintenance is impossible then, the requirements have been disposed.

4 Discussion

Based on the implementation of the integrated process modelling on MySEFAM software system version 3, the comparison of the integrated process modelling, Scrum and XP phases as well as practices have been performed.

	Integrated process modelling	Scrum	XP
Phases	4 phases Phase 1: Pre-game Phase 2: Planning ▪Architectural design ▪Planning for Testing ▪Testing Implementation Phase 3: Development ▪Functional Testing Phase 4: Post-game ▪Productionizing ▪Maintenance ▪Death	3 phases Phase 1: Pre-game ▪Planning ▪Architectural design Phase 2: Development Phase 3: Post-game	6 phases Phase 1: Exploration Phase 2: Planning Phase 3: Iterations to Release ▪Analysis & Design ▪Planning for Iterative Testing ▪Iterative Testing ▪Functional Testing Phase 4: Productionizing Phase 5: Maintenance Phase 6: Death
Practice	▪Iterations ▪Time-boxing ▪On-site customer ▪Self-organizing teams ▪Empirical process ▪Adaptive planning ▪Requirements prioritization ▪Frequent integration ▪Simplicity of design ▪Refactoring ▪Pair-programming	▪Iterations ▪Time-boxing ▪No change of started project ▪On-site customer ▪Self-organizing teams ▪Empirical process ▪Adaptive planning ▪Requirements prioritization ▪Fast decision making ▪Frequent integration	▪Iterations ▪Time-boxing ▪On-site customer ▪Self-organizing teams ▪Empirical process ▪Sustainable discipline ▪Adaptive planning ▪Requirements prioritization ▪Frequent integration ▪Simplicity of design ▪Refactoring ▪Team code ownership ▪Pair-programming

The phases of the integrated process modelling are the combination of both Scrum and XP respectively. The management activities have been extracted from the Scrum practices such as the self-organizing teams and the architectural design sub-phase practice by Scrum. However, to cope with the software engineering practices, the integrated process modelling has adopted some practices of XP which are pair-programming and refactoring. Thus, as the integrated process modelling is the integration of the management of Scrum and the engineering of XP, it covers both the good practices of both model. Therefore, it can be concluded that Scrum and XP is suitable to be integrated as the practices of both complement each other limitations.

Other than that, as the integrated process modelling provides the construction of updated use case as well as updated class diagram of the system thus, it covers the learning process of the system being maintained. As it also practices the testing planning before implementation, thus it provides an activity to reduce the risks of implementing the fault features thus lead to extension of effort on maintaining a software system. Other than that, despite of the excellence learning curve supported in the pair programming, however as the level of knowledge between developer and tester are different thus, limit the efficiency of pair-programming practice. In addition, the planning for testing do provide a good guideline to develop codes, nevertheless as the testing has been done by the tester only thus, the knowledge-bound occurred. As the tester does not have the same level of knowledge of the programming language used, thus it limits the contribution of the testing process. Last but not least, as the stakeholder is located in Kuala Lumpur, thus face-to-face interaction could not be performed. This has been an issue for every Agile methodology as most of them highlighted the used of the practice.

References

1. B. Hunt, B. Turner, and K. Mcritchie, **"**Software Maintenance Implications on Cost and Schedule,**"** vol. 20, no. 12, pp. 1–6, 2008.
2. C. Jones, **"**Geriatric Issues of Aging Software,**"** no. December, pp. 4–8, 2007.
3. M. A. A. L. Ahmar, **"**Rule Based Expert System For Selecting Software Development Methodology,**"** pp. 143– 148, 2010.
4. K. Petersen and C. Wohlin, **"**A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case,**"** *J. Syst. Softw.*, vol. 82, no. 9, pp. 1479–1490, 2009.
5. M. Kajko-mattsson and J. Nyfjord, **"**A Model of Agile Evolution and Maintenance Process,**"** pp. 1–10, 2009.
6. I. Wescon, **"**Managing The Development Of Large Software Systems Dr. Winston W. Rovce Introduction,**"** no. August, pp. 1–9, 1970.
7. A. Eberlein, **"**An Analysis of the History of Classical Software Development and Agile Development,**"** no. October, pp. 3733–3738, 2009.
8. P. Kettunen, **"**Adopting key lessons from agile manufacturing to agile software product development-A comparative study,**"** *Technovation*, vol. 29, no. 6–7, pp. 408–422, 2009.
9. K. N. Rao, G. K. Naidu, and P. Chakka, **"**A Study of the Agile Software Development Methods , Applicability and Implications in Industry,**"** vol. 5, no. 2, pp. 35–46, 2011.
10. A. Cockburn, R. Jeffries, and R. C. Martin, **"**Manifesto for Agile Software Development.**"**
11. A. H. Mohammad, T. Alwada**'**n, and J. "M. A. Ababneh, **"**Agile Software Methodologies : Strength and Weakness,**"** *Int. J. Eng. Sci. Technol.*, vol. 5, no. 03, pp. 455–459, 2013.
12. T. Dybå and T. Dingsøyr, **"**What Do We Know about Agile Software Development?,**"** pp. 0–3, 2009.
13. B. Henderson-sellers, **"**Comparative evaluation of XP and Scrum using the 4D Analytical Tool (4-DAT) Comparative Evaluation Of Xp And ...,**"** no. August 2014.
14. L. Vargas, **"**The Analytic Network Process,**"** no. August, 2014.