I.J. Information Engineering and Electronic Business, 2015, 1, 37-42
Published Online January 2015 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijieeb.2015.01.05

Proposal of Enhanced Extreme Programming Model

M. Rizwan Jameel Qureshi, Jacob S. Ikram
Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Kingdom of Saudi Arabia
Email: anriz@hotmail.com, jacob.sayid@hotmail.com

Abstract—Extreme programming is one of the commonly used agile methodologies in software development. It is very responsive to changing requirements even in the late phases of the project. However, quality activities in extreme programming phases are implemented sequentially along with the activities that work on the functional requirements. This reduces the agility to deliver increments continuously and makes an inverse relationship between quality and agility. Due to this relationship, extreme programming does not consume enough time on making extensive documentation and robust design. To overcome these issues, an enhanced extreme programming model is proposed. Enhanced extreme programming introduces parallelism in the activities' execution through putting quality activities into a separate execution line. In this way, the focus on delivering increments quickly is achieved without affecting the quality of the final output. In enhanced extreme programming, the quality concept is extended to include refinement of all phases of classical extreme programming and creating architectural design based on the refined design documents.

Index Terms—Software Engineering, Agile, Extreme Programming, Pair Programming, Parallel.

I. Introduction

Agile software development methods are emerged in 2001 at the signing of Agile Manifesto [1]. There are two models in systems development projects; predictive model and adaptive model [2]. In predictive models, the scope of the project is clearly defined which makes it possible to anticipate the cost and time precisely. On the other hand, the adaptive model has no clear scope and it is mainly a mission driven. Agile methods adapt well with such projects that have ambiguous and changing requirements.

The agile manifesto has defined twelve principles for agile. The principles are highly focused around customer satisfaction and involvement, incremental delivery of software and stakeholders' collaboration and cooperation. These principles obviously help projects with changing requirements to succeed. For this reason, the agile methods are proper for adaptive model.

One of the broadly used agile methods is extreme programming (XP). XP inherits all the previously mentioned features of agile. In XP, programmers develop the system in pairs. Code is extensively tested and reviewed. Only the functional requirements are focused on without any additional features that are not yet needed. XP focuses on collaboration between all team members including managers, customers and developers. XP consists of 4 main phases: plan, design, code and test. There are activities and practices performed in each phase. For example, users' stories are written along with release planning in planning phase. These stories are given size and effort amount by estimation. The estimation process in XP relies on a practice called planning poker technique. This technique asks from each and everyone in the team to attend a meeting where stories are presented briefly and the attenders give estimation for each story. The estimations are discussed and reasoned till all members agree on a single estimation. Other interesting practice is pair programming. Pair programming means that two developers work on one machine for development. They share ideas and work with collaboration to finish their task. The pair programming leads to another practice in XP called collective code ownership. This means that anyone can edit the code anytime since the ownership of the code is shared. In design phase, system metaphor (abstract design) and CRC cards are created. The simplicity of design is the key requirement in XP. As the design is simplified, the time taken to finish it is less compared to a complex design. To decide whether a design is simple or not is a subjective task. One common rule is that to work on something needed currently without trying to work on something may be needed in future but not asked for currently. Another way is that the simplicity of design can be clear when the project is progressed for a while. This makes the process easier and leads to know that design in XP comes after coding for refining purposes. In coding phase, the customer is involved to help the team and be considered one of the team members. The code is written based on previous tests that are developed. In other words, XP follows test driven development manner where the test is written before code. Testing in XP includes unit testing, system testing, integration testing on increments and the acceptance testing which marks an increment to be finished and approved. An increment in XP may take

Copyright © 2015 MECS

single iteration to multiple iterations to finish. This depends on the customer acceptance and the changes he asks for the same increment.

Despite the benefits offered by XP, several drawbacks are noted. For instance, XP suffers from weak documentation and lack of overall design for the system. Moreover, extreme tests and refactoring activities are done sequentially during an iteration which reduces the agility and increases response time. The previous drawbacks make XP not suitable for medium and large size projects. There are other limitations in XP that make it not suitable in some scenarios. For instance, XP is not suitable with outsourced team. This is because the XP needs highly competent members in team. These members need to collaborate, trust, respect and be self-organized. Such skills are hard to find in outsourced team members who work just for the project that is assign to them. Another limitation is that XP doesn't work with distributed development environment. The reason is that the practices and activities of XP require high collaboration, involvement, face-face meeting and customer to be with the team. Due to the high agility of XP and simplicity of its design, the system is developed very quickly without taking reusability of newly added components into account. This means that XP doesn't take the advantage of component based development into account.

Several studies are proposed to extend the XP so that it adapts to custom requirements in some types of projects. However, most of these studies are not focused to directly solve the drawbacks of XP. Instead, they embed new phases and activities in XP to achieve a specific purpose such as security of application [3]. As a consequence, there is a need to implement XP with a customized model that can overcome its drawbacks. This paper proposes a customized model of XP that increases the agility and improves quality, documentation and overall design.

The rest of the paper is organized as follows: Section II is about related work. Section III describes the research problem. Section IV covers the details of the proposed solution. Last section presents conclusion and future work.

II. Related Work

There are several papers proposed to improve or extend XP model so that it adapts to the needs of various projects. Bala et al. [3] presented improved XP framework that takes into account the security controls and tighten them. It includes both the development team and business representatives at initial stages to identify and deal with all security concerns. The framework introduces security checks in almost all phases of XP. However, the continuous security checks in XP iterations can affect the agility of XP negatively.

Luigi et al. [4] used the analytic hierarchy process (AHP) to determine the prioritization of CRC cards efficiently and effectively. Using of AHP presents the simplicity and agility in the process of prioritizing. Responsibility, collaboration and stability are the criteria used in CRC prioritization. AHP is implemented to structure 3-lv hierarchy where the top level has Prioritization CRC, the second LV includes the criteria of CRC, and the last LV includes alternative CRCs. However, the proposed solution needs to be evaluated in real test cases. Moreover, the shown experiment lacks covering all teams' results. Team members were still learning some skills to be able to evaluate the proposed solution.

Elmuntasir et al. [5] proposed a solution to adapt XP for development of large-scale distributed projects. The suggested practices are daily standup meetings, adaptive planning, code control, contentious Integration, visual Indicators, XP project management and Code Gallery. These practices are implemented in Sudan Automated Traffic Violations Project as a case study. However, the authors didn't address the issues that arise when collaboration is missed in this type of projects.

Feng et al. [6] made comparison of two software development methodologies based on three months project data with 4 developers at most. These methodologies are XP and Waterfall. The same project is developed repeatedly for five years by fifty teams. The result showed that the completed features and lines of code were almost the same before and after transition to extreme programming method. Therefore, the authors concluded that it doesn't matter what method is used when the project is with previously mentioned criteria. However, the collected data needs diversity in terms of source and characteristics.

Stephen et al. [7] addressed post-adoption performance and the role of teams in engineering methods which are neglected in previous studies. The solution through studies found that client and team foci are the critical active ingredient of XP.

Irina et al. [8] conducted observational studies on two different industrial projects in terms of size and time. These studies have shown that there are different types of interactions between team members such as collaborative and cooperative backup behavior. This distinction leads to appropriate use of pair programming and also can explain why there are contradictions in the results of observing pair programming benefits. However, the observational studies conducted by the authors are limited to only two different projects data. The data collected needs to be more than what is available so that the result is going to be more precise.

Nick [9] divided introductory java course students into two 65 teams. One uses solo programming while the other uses VPP in the last four-assignments where metrics like LOC, defects per 1000 LOCs, quality and productivity are recorded for both teams. The comparison results show that VPP team was more productive, has fewer defects in their code (50% less) and their deliverables are of higher quality.

Gert et al. [10] addressed the role of customer by assisting the release plan in XP. The author has developed an optimization model that generates release plan which is based on story size, business value,

Copyright © 2015 MECS

precedence relations and themes. The solution through studies found that client and team foci are the critical active ingredient of XP. However, the optimization tool consumes a lot of time to gather precise data from different dimensions to produce accurate output. This can affect responding speed in XP which is one of the main advantages in agile methodologies.

Table 1. Summary of the Related Work

Title	Summary
Service Agile Development Using XP [13]	• Despite the proposed solution gives very detailed guidelines and practices on how to combine both XP and SOA; it lacks the real life evaluation and case studies. • Adapting XP with SOA in the way described can affect the overall agility negatively.
Comparing Extreme Programming and waterfall Project Results [6]	• The proposed comparison is limited to only one project data. • The data collected needs diversity in terms of source and characteristics.
Successful extreme programming: Fidelity to the methodology or good team working? [7]	• The performance measure in the proposed solution is based on assessments that rely on subjective interpretations.
Quantitative release planning in extreme programming [10]	• The optimization model consumes a lot of time to gather precise data from different dimensions to produce accurate output. This can affect responding speed in XP which is one of the main advantages in agile methodologies.
Cooperation, collaboration and pair-programming: Field studies on backup behavior [8]	• The observational studies conducted by the author are limited to only two different projects data. • The proposed solution gives good inspiration on how to make the pair programming more effective in the future work.
Measuring the Effects of Virtual Pair Programming in an Introductory Programming Java Course [9]	• The results are better to stand on bigger sample space. For example, 4-6 classes recordings need to be collected over 3-4 semesters.
Pair Programming and Software Defects A Large, Industrial Case Study [11]	• The number of defects to be reduced by practicing PP should be increased. • The results should be implemented on projects where programmers' are of different levels of experience
The impact of Absorptive Capacity on the Ex-Post Adoption of Agile Methods: The Case of Extreme Programming Model [12]	• Only one site is used to study the collected data which is not enough to generalize the insights and results to other projects
Agile Software Engineering as Creative Work [14]	• Creativity perspective needs to be evaluated through proposing well explained ways to improve XP and then making case studies on proposals
Research on Requirement for High-quality Model of Extreme Programming [15]	• The effectiveness of the proposed solution needs to be evaluated in real projects
Extreme programming applied in a large-scale distributed system [5]	• The proposed solution has been implemented on only one project which is not enough to prove the feasibility of adapting XP as a software development methodology in large-scale projects. • The proposed solution hasn't addressed the issues that arise when collaboration is missed in this type of projects.
Improved Extreme Programming Methodology with Inbuilt Security [3]	• The proposed solution needs to be applied in real business projects that need security to decide its applicability and effectiveness. • The continuous security checks in XP iterations can affect the agility of XP negatively.
Prioritizing CRC Cards as a Simple Design Tool in Extreme Programming [4]	• The proposed solution needs to be evaluated in real test cases. • The shown experiment lacks covering all teams' results. • Team members were still learning some skills to able to evaluate the proposed solution.
Application of Agile Method in the Enterprise Website Backstage Management System [17]	• The author advices and practices were generally said without mentioning examples of any previous experience. Most of the advices and practices were not clear enough.
Agile software development methodology for medium and large projects [16]	• The size of the sample space is not enough to validate the proposal statistically. • The proposed solution lacks for detailed description on how XP model can be adapted for parallel development.

Enrico et al. [11] had done a case study that is based on 14-months dataset collected from a team of professional developers working in an IT department of a large Italian manufacturing company. The results of the study are carefully collected and validated internally/externally. The analysis results show that as PP practiced, new defects are decreased.

Bahli et al. [12] studied two information system development projects in a Canadian organization which is switching from Waterfall to XP model. TAM (technology acceptance model) is extended by the author to include absorptive capacity. The collected results from ex-post adoption to XP switch showed that IS developers asserted that they are capable to apply XP even in future projects which is a clear realization to the high absorptive capacity they have. However, only one site is used to study the collected data which is not enough to generalize the insights and results to other projects.

Felipe et al. [13] proposed guidelines and best practices that employ XP and SOA concepts for service development. Their proposed solution focuses on the seven principles of SOA and how they are supported by XP. The guides show how different types of metaphor should be designed for different principles. It also shows which stakeholders should be aware of which metaphor in each principle of SOA. Despite the proposed solution gives very detailed guidelines and practices on how to combine both XP and SOA; the authors didn't address the negative effect of the proposed solution on the overall agility.

Copyright © 2015 MECS

Broderick et al. [14] fixes the relevance between creativity and knowledge management which are both essential for software engineering. To address this important relationship, they compare phases and roles of XP with phases and roles in creativity process. Based on the comparison, creativity of XP can be improved. Despite the authors have provided good basis and realization for future researches on improving XP based on creativity perspective, this basis needs to be evaluated through proposing well explained ways to improve XP and then making case studies on proposals.

Zhai et al. [15] established XP high quality analysis model getting the benefit of the quality feature in Kano model. The established model has improved customer awareness and reduced misunderstanding of requirements.

Rizwan [16] has proposed an extended XP model to fit medium and large projects with better documentation, stronger architectural design. Extension is done through modifying the phases of XP process model so that it includes the following phases: Project Planning, Analysis & Risk Management, Design & Development and testing. The first phase defines the project scope and focuses on major milestones. The second phase provides proper documentation and risk management. The third phase is combination of design and code phases of previous model to increase speed. It relies on demos to verify requirements and develop incrementally. The last phase is all about testing as in the previous model. Despite the proposed solution proved its success in the case studies mentioned, it lacks for detailed description on how XP model can be adapted for parallel development. The summary of the literature review regarding this paper is shown in Table 1.

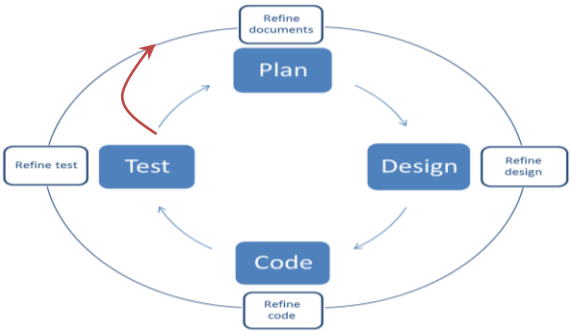

Fig. 1. The proposed Enhanced XP Model

III. Problem Definition

Extreme Programming is an agile development methodology that is intended for adaptive projects where requirements are vague and not clear. One of the key features of XP is the fast responsiveness to changing customer requirements. Several papers are proposed to adapt XP to different project sizes [16], with additional features [3]. Other papers are focused to improve the activities of XP [4] or to offer implementation guidelines [5] and case studies for different types of projects [17].

However, these proposed solutions lack to offer an implementation model that can, at the same time, increase the agility and solve common problems in XP such as bad documentation and architectural design. The question arises is:

How to design a customized model for XP that improves documentation, architectural design and the agility at the same time?

IV. The Proposed Enhanced Xp Model

The enhanced XP methodology (EXP) is expected to deliver a high quality software system in terms of documents, code and design. Therefore, it can be implemented on projects of different sizes unlike the classical XP that is focused to be highly responsive to customer requirements without having robust documentation and architectural design. Because the classical XP activities of quality and functionality are implemented in a sequential manner, there is going to be an inverse relationship between agility and quality. On the other hand, EXP takes parallelism into account to avoid the inverse relationship between quality and agility. Rizwan described that his extended model is suitable to be implemented incrementally or in parallel [16]. However, detailed description of parallel development is not presented.

EXP mostly keeps the main phases of classical XP intact without radical changes. It only takes out activities that serve non-functional requirements from these phases into a parallel refinement phase. For this reason, EXP offers high level of backward compatibility with other extended models. For instance, Bala et al. [3] extended XP by introducing inbuilt security as an activity that needs to be embedded in all XP phases. Securing the system can be achieved in the same way described in EXP with taking all additional steps embedded in classical XP phases and including them in refine iteration phases.

In EXP, Iteration through main phases is classified into four categories: initial, incremental, final and quality iteration. To achieve a milestone, we need an initial iteration, n-incremental iterations and a final iteration. The fourth iteration starts in parallel with the first incremental iteration and keeps iterating through its refining phases till everything is checked and refined. The proposed EXP model is shown in fig.1. A detailed explanation of how to implement EXP on these iterations is as follows:

A. *Initial Iteration*

In this iteration, the four main phases of XP are implemented normally as in classical XP. The only difference is in coding phase. During coding, a member of code refactoring team which will work in parallel starting from the next iteration will join the pair programmers. The task of this member is to understand and monitor what is being coded by the pair programmers. He may discuss with them code issues and help them if needed. Later in the next iteration, this member will give assistant for code refactoring team when they face any difficulties to understand the written code. Moreover, the

Copyright © 2015 MECS

member has to join all pair programming sessions throughout the system or subsystem development. The best role that can be assigned for this task is a technical team leader who will be able to assist both teams: the pair programming team to code quickly overcoming all obstacles, and the refactoring team to find the issues and to check quality.

B. Incremental Iteration

The incremental iteration follows the first iteration and keeps iterating n-times until it reaches a point where the subsystem or system is iteration away from a milestone. In incremental iteration, all deliverables of previous iteration are given to the team of quality iteration that works in parallel. The main team members who work in incremental iteration don't bother themselves in refining previous design, code and tests. Instead, they only take the new requirement and implement it. While in classical XP, refining and refactoring activities are included in the four phases. It is necessary to denote the quality iteration in this section as it starts in parallel with the first incremental iteration. In quality iteration, the previous design, code, tests and documents are iteratively refined and double checked via a separate team. This team works in parallel. Firstly, the design is refined which gives coders hints on how to refactor the code and make it structured, understandable and easy to be modified. Moreover, refining design includes working on the architectural design incrementally through studying all previous design documents. Secondly, code refactoring is done according to the refined design and by the help of the technical team leader who was monitoring the previously written code in the previous iteration. The code refactoring team will work in a different fork of the system/subsystem that is to be implemented to avoid any conflicts. Additional functionality will be grabbed to the forked system for refactoring, whereas changes in the previous code will be implemented again in the refactored code without grabbing them as is. Thirdly and after code refinement, tests are refined by applying more comprehensive tests on the code. Found bugs will be solved directly without waiting for the next incremental iteration. Finally, Documents refining team starts refining all poorly written documents as well as they document extensively anything that is missing.

C. Final iteration

The final iteration directly precedes a milestone in system development. A milestone is met by the end of finishing requirements of this iteration. The phases in this iteration are dealt with in the same way as in incremental iterations. Once the final iteration is done, it means that a milestone in project is met. Because meeting milestones make the vague project requirements clear enough for designers, it is possible to deliver a first release of the architectural design for the system so that the subsequent requirements and big view of the system will be expected and clear for all stakeholders. As a result, progressing in the project becomes faster and in confident steps.

If the project is very large and there is no high dependency level between subsystems of the whole project, it is possible to start separate iterations with each of these subsystems so that parallelism is presented again in a different level. However, integrating all subsystems is needed in the final phase of the project as a new requirement. The final requirement is dealt with in the same way using EXP.

V. Conclusion And Future Work

Extreme programming is an agile methodology for software development that performs very well with changing requirements. XP is one of the most commonly used methods among other agile methods. However, it is implemented sequentially on all activities including those which are not functional. Therefore, the agility is reduced. Moreover, classical XP suffers from weak documentation and architectural design. Therefore, there is a need for extended model that can overcome these problems. Several papers are proposed to adapt XP to different project sizes with additional features. Other papers are focused to improve the activities of XP or to offer implementation guidelines and case studies for different types of projects. However, these proposed solutions lack to offer an extended model that can solve the common problems in XP without affecting the agility negatively. In this paper, an extended model of XP is proposed which is called Enhanced Extreme Programming (EXP) to address the problem mentioned. Higher agility can be achieved when non-functional activities are done in parallel in a separate iteration. To have better documentation in XP along with good architectural design, the concept of monitoring and refinement needs to be applied on design, documentation and testing. This paper gives clear steps on how to implement EXP model. Although EXP offers all these features, it is considered very costly as it requires more staff to work, monitor and refine each deliverable of XP phases.

References

[1] K. Roebuck, Agile Software Development: High-impact Strategies - What You Need to Know, *Emereo Pty Limited*, 2011.

[2] K. Schwalbe, Information Technology Project Management, *Cengage Learning*, 2004.

[3] S. Musa, N. Norwawi, M. Selamat and K. Sharif, "Improved Extreme Programming Methodology with Inbuilt Security," *in Computers & Informatics (ISCI)*, Kuala Lumpur , 2011.

[4] S. Alshehri and L. Benedicenti, "Prioritizing CRC cards as a simple design tool in extreme programming," *in Electrical and Computer Engineering (CCECE)*, Regina, SK, 2013.

[5] E. Abdullah and E.-T. Abdelsatir, "Extreme programming applied in a large-scale distributed system," *in Computing, Electrical and Electronics Engineering (ICCEEE)*, Khartoum, 2013.

[6] F. Ji and T. Sedano, "Comparing extreme programming and Waterfall project results," *in Software Engineering*

Education and Training (CSEE&T), Honolulu, HI, 2011.
[7] S. Wood, G. Michaelides and C. Thomson, "Successful extreme programming: Fidelity to the methodology or good teamworking?" *Information and Software Technology*, vol. 55, no. 4, p. 660–672, 2013.
[8] I. D. Coman, P. N. Robillard, A. Sillitti and G. Succi, "Cooperation, collaboration and pair-programming: Field studies on backup behavior," *Journal of Systems and Software*, vol. 91, p. 124–134, 2014.
[9] N. Zacharis, "Measuring the Effects of Virtual Pair Programming in an Introductory Programming Java Course," *IEEE Transactions on Education*, vol. 54, no. 1, pp. 168 - 170, 2011.
[10] G. v. Valkenhoef, T. Tervonen, B. d. Brock and D. Postmus, "Quantitative release planning in extreme programming," *Information and Software Technology*, vol. 53, no. 11, p. 1227–1235, 2011.
[11] E. di Bella, I. Fronza, N. Phaphoom, A. Sillitti, G. Succi and J. Vlasenko, "Pair Programming and Software Defects--A Large, Industrial Case Study," *IEEE Transactions on Software Engineering*, vol. 39, no. 7, pp. 930 - 953 , 2013.
[12] B. Bahli, Y. Benslimanne and Z. Yang, "The impact of absorptive capacity on the ex-post adoption of agile methods: The case of Extreme Programming model," in *Industrial Engineering and Engineering Management (IEEM)*, Singapore, 2011.
[13] F. Carvalho and L. Azevedo, "Service Agile Development Using XP," in Service Oriented System Engineering (SOSE), Redwood City, 2013.
[14] B. Crawford, C. de la Barra, R. Soto and E. Monfroy, "Agile software engineering as creative work," *in Cooperative and Human Aspects of Software Engineering* (CHASE), Zurich, 2012.
[15] Z. Li-li, H. Lian-feng and S. Qin-ying, "Research on Requirement for High-quality Model of Extreme Programming," *in Information Management, Innovation Management and Industrial Engineering (ICIII)*, Shenzhen, 2011.
[16] M. Rizwan Jameel Qureshi, "Agile software development methodology for medium and large projects," *IET Software*, vol. 6, no. 4, pp. 358 - 363, 2012.
[17] L. Liu and Y. Lu, "Application of agile method in the enterprise website backstage management system: Practices for extreme programming," *in Consumer Electronics, Communications and Networks (CECNet)*, Yichang, 2012.

Authors' Profiles

Dr. M. Rizwan Jameel Qureshi received his Ph.D. Computer Sciences degree from National College of Business Administration & Economics, Pakistan 2009. He is currently working as an associate professor in the Department of Information Technology, Faculty of Computing & Information Technology, King Abdulaziz University. He is the best researcher awardees of King Abdul-Aziz University Saudi Arabia in 2013 and Department of Computer Science, COMSATS Institute of Information Technology Pakistan in 2008.

Yaqoob S. Ikram was born in 1989 and received the bachelor's degree in information technology from the King Abdul Aziz University, Jeddah, Kingdom of Saudi Arabia, in 2013. He is currently working toward the master's degree at the King Abdul Aziz University. His current research interests include agile methodologies, improved agile models and risk management. Yaqoob worked as a developer in different governmental projects of medium-large size.

How to cite this paper: M. Rizwan Jameel Qureshi, Jacob S. Ikram,"Proposal of Enhanced Extreme Programming Model", IJIEEB, vol.7, no.1, pp.37-42, 2015. DOI: 10.5815/ijieeb.2015.01.05

Copyright © 2015 MECS