I.J. Modern Education and Computer Science, **2017, 12, 26-37**
Published Online December 2017 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2017.12.04

Latest Customizations of XP: A Systematic Literature Review

Faiza Anwer, Shabib Aftab
Department of Computer Science, Virtual University of Pakistan
Email: {faiza.anwer28, shabib.aftab}@gmail.com

Received: 10 August 2017; Accepted: 16 October 2017; Published: 08 December 2017

***Abstract*—**Software development process model plays a key role in developing high quality software. However there is no fit-for-all type of process model exist in software industry. To accommodate some specific project's needs, process models have to be tailored. Extreme Programming (XP) is a well-known agile model. Due to its simplicity, best practices and disciplined approach researchers tried to mold it for various types of projects and situations. As a result a large number of customized versions of XP are available now days. The aim of this paper is to analyze the latest customizations of XP. For this purpose a systematic literature review is conducted on studies published during 2013 to 2017. This detailed review identifies the objectives of customizations, specific areas in which customizations are done and practices & phases which are being targeted for customizations. This work will not only serve the best for scholars to find the current XP states but will also help researchers to predict the future directions of software development with XP.

***Index Terms*—**Extreme Programming, XP, Agile, Customized XP, Modified XP, Tailored XP, Systematic Literature Review.

I. Introduction

Agile software development models provide a good, light weight and cost-effective option for quality software development. Agile manifesto defines the values and principles which are applied in iterative fashion to obtain quality software in limited time [25] [26] [36] [37] [38]. A number of agile software development models emerged with the time having potential to handle various project types with agility. Most commonly used agile models incudes Extreme programming (XP), Scrum, Feature Driven Development (FDD), Dynamic System Development Method (DSDM), Kanban, Lean Software Development (LSD), and Adaptive Software Development (ASD) [27] [39].

Extreme Programming (XP) is one of the oldest known agile models that gave new directions to software development. Kent Beck presented XP model in 1999 but still today it is one of the most debating topics in software industry. XP revealed a new perspective of software development that gives much importance to customer satisfaction, changing requirements and team collaboration than plan driven software development models [30] [31] [36] [37]. XP works well for small scale, low risk projects [28] [36]. It uses best software practices in the disciplined way to develop high quality software [27]. It can easily accommodate changing requirements with good level of customer satisfaction and can deliver qualitative software within limited time [32] [35]. XP practices like pair programming, on-site customer, collective code ownership, continuous integration and continuous testing were new for software industry but their satisfactory results enforced developers to adopt them even in diversified projects. Due to XP's flexibility and simplicity, researchers showed great interest in customizing XP. They tried to make it suitable for various scenarios by tailoring its phases or by adding more practices for some specific needs. As a result there are a number of modified versions of XP available now days. In this paper a systematic literature review is conducted to explore the latest transformation of XP. This SLR considered related literature published during 2013 to 2017 by applying inclusion and exclusion criteria.

Further organization of this paper is as follows. Section II describes related work, section III defines research methodology used for this SLR. Section IV presents critical review of selected papers. Section V enlists and discusses the finding of this detailed review. Section VI finally concludes the paper.

II. Related Work

Wide acceptance of agile methods fascinated many researchers to explore different aspects of agile software development. There are a number of SLR's mentioned in this section which provide valuable information about different aspects of agile software development. However it is observed that very little contribution is made for extreme programming. Few literature reviews are available which does not provide clear picture of XP's current state. Agile methods are famous due to user involvement during software development phases especially in requirement engineering process. In [10] authors conducted a systematic literature review to explore the art of requirements engineering in agile methods. This study mainly focused on various methodologies used to invite stakeholders by presenting

their perspective during requirements engineering and management process. Another study in this regard is conducted in [11], for this review authors selected literature from 2002 to 2013 to extract data about requirements engineering practices and challenges in agile methods. A SLR was conducted in [12], to find the direct and indirect effect of agile release practices on software projects. Agile release practices are used for fast and low cost software development. In [13], SLR was conducted to find out the challenges, and success factors identified during agile method's transformation for large scale industrial projects. Authors listed 35 challenges and 29 success factors for which selected papers are analyzed. In [14], authors grabbed an emerging trend of combining agile practices with outsourced software development. This type of development faced the coordination and communication problems. This study focused on finding useful communication practices and then it differentiated these practices from classical practices used in non-distributed environment. In [15], authors presented different agile methods tailoring aspects and criteria of practice's selection used for tailoring. This SLR considered the literature published during 2002 to 2014 to find the common trends and criteria used for process tailoring. Another area of software development is covered in [16]. Authors conducted SLR to present the effort estimation methods used in agile software development. They concluded that expert judgement based technique, planning poker and use case points technique are the most commonly used estimation methods in agile development. In [17], authors used SLR to find the security related issues in software development using extreme programming (XP). Authors used the literature published during 2002 to 2012. They concluded that XP practices can be successfully combined with security based practices for effective results. In [18], authors studied the literature regarding integration of user centered design with agile software development models. They concluded that this integration is mostly used for design and usability evaluation however there is a lack of studies that provide its empirical proof. Pair programming is one of the distinguishing practices of XP, to find its effect on quality and effort, a meta-analysis is conducted in [19]. Results of this analysis showed that pair programming has a little better performance in term of quality but also has negative effect on effort used for software development.

Another systematic review to study the effect of pair programming is conducted in [20]. The authors identified the factors which can affect the usefulness of pair programming. In [34], authors studied impact of user involvement in the project success by performing a systematic literature review. They analyzed 87 empirical studies and found that user involvement plays a positive role in project success.

III. Research Methodology

To conduct a successful SLR, we need a proper research methodology that can help in achieving the complete research objectives. Different studies are available that provide the guidelines for systematic literature review [22] [23] [24] [33]. By consulting these studies we formulated a systematic research strategy to follow.

In a broader view, SLR has three basic steps namely plan review, conduct review and document review, however further detail can be added to make it more elaborative. Detailed steps are extracted from the guidelines of [22] [23] [24]. The research methodology which is followed includes following steps: 1) Define research questions, 2) Find keywords to form query string, 3) Define research space to get data, 4) Set criteria to include or exclude papers, 5) extract literature using criteria, 6) Assess study quality, 7) Synthesize required data, and finally 8) document results and outcomes Fig. 1.

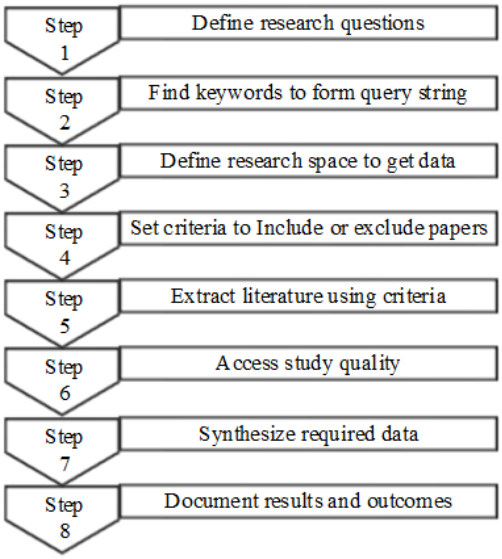

Fig.1. Steps of Systematic Literature Review

A. Research Questions

Research questions represent the research objectives. Answers to these research questions help in concluding SLR. According to step 1 of our research strategy, here are the research questions which will cover our research objectives.

- **RQ1:** Which are the customized versions of XP process model?
- **RQ2:** Do the customized versions tailor XP phases?
- **RQ3:** Which practices, roles or events are included in XP to make it more effective and efficient?
- **RQ4:** What objectives are achieved through XP customization?
- **RQ5:** Do these modified or improved models are validated through empirical proof?

B. Search Space and Query String

Keywords extracted from the research questions are "Agile", "Extreme Programming", "XP", "process",

"model", "method", "modified", "improved", "tailored", "changed", "enhanced" and "customized".

These keywords are arranged in following query.

(Modified OR improved OR customized OR tailored OR changed OR enhanced AND (Agile AND (Extreme Programming OR XP AND (Process OR Model OR Method)))).

Search space represents libraries and repositories from where data can be collected. In our case, we selected Google Scholar to find the papers from 2013 to 2017. Above mentioned query string is used to extract required literature.

C. Selection Criteria

Selection of related material is done based on inclusion and exclusion criteria defined in this section. IC represents inclusion criteria whereas EC represents exclusion criteria.

1) Inclusion Criteria

Inclusion criteria consist of following rules to extract related material for this SLR.

- **IC1:** Papers which are published during 2013 to 2017.
- **IC2:** Papers which are available in journals, conferences, proceedings of conferences or workshops.
- **IC3:** papers which have presented modified form of XP with the help of figure.
- **IC4:** papers which have tried to enhance XP practices.
- **IC5:** papers which have provided practical proof of modified XP model.

2) Exclusion Criteria

Exclusion criteria consist of following rules to exclude un-related material for this SLR.

- **EC1:** Papers which are not published in the duration of 2013 to 2017.
- **EC2:** Papers which are not written in English language.
- **EC3:** papers whose full text is not available.
- **EC4:** Literature which is under review or part of thesis report.
- **EC5:** Literature which is a part of any book.
- **EC6:** Papers that contain survey or review about previous work.
- **EC7:** papers which provided only XP related material other than modified model.
- **EC8:** papers that merged XP with any other process model to form a hybrid model.
- **EC9:** papers which did not provide any pictorial representation of proposed model.
- **EC10:** papers that proposed XP application in some field other than software development.
- **EC11:** papers that introduced new tools to use XP process models.

Using search strings based on identified keywords, initially we found 9795 results. Step by step filter is applied to select the most relevant papers based on our selection criteria as shown in Fig. 2.

D. Quality Assessment

For a successful and useful review, the focus on quality is very necessary. To make our SLR more useful, every step is completed under the quality umbrella. To ensure the higher quality, following measures are taken.

- All the literature is selected from authentic and renowned libraries and databases.
- Only reputed journals publications are included in review.
- All the literature is collected without any biasedness and discriminations.

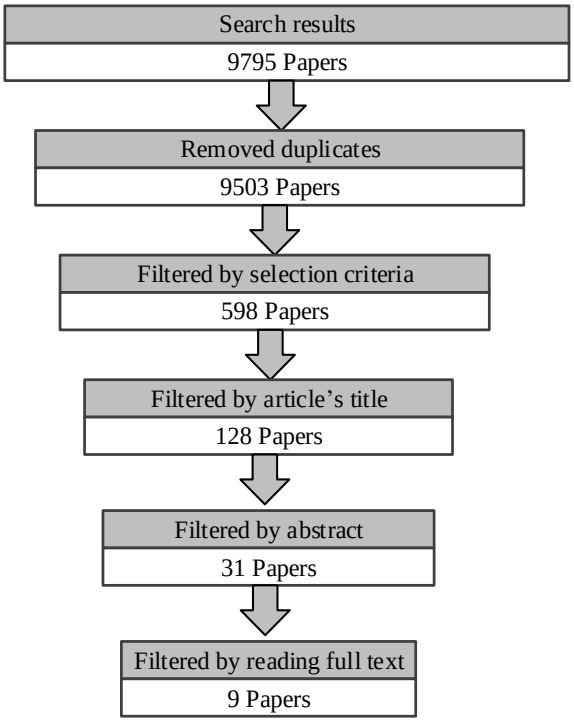

Fig.2. Search Process for Final Selection of Papers

E. Data Extraction and Classification

Data is extracted by considering the guidelines given in [29]. By screening the papers finally nine most relevant papers are selected for review. Data extraction and classification is done according to format given in table 1.

Table 1. Data Extraction and Classification Format

Description	Detail
Bibliographic Detail	Paper's title, author's name, publication year, type (journal, conference, workshop)
Data Extraction	
Model	Model name/ Model title
Methodology used	survey, empirical proof.
Project suitability	Small scale projects, medium scale projects, large scale projects.
Data Synthesis/ Classification	
Objectives of Customization	Enhancement of model, elimination of drawbacks, addition of new functionality.
Strategy used for customization	Alteration of phases/practices, addition or removal of practices, addition of new roles
Targeted XP practices	Which practices are used for customization?

IV. Customized Forms of Extreme Programming

This section aims to explore the 9 selected papers to identify the key problems and their proposed solution. Moreover this section also highlights the modules, practices, roles, phases and principles which are being added, customized or removed by the authors in their proposed solutions.

A. SXP: Simplified Extreme Programing Process Model.

SXP (Simplified Extreme Programming) is a modified version of XP process model that provides a simpler way of software development for small and medium scale projects [1]. In this model, authors tried to cover the limitations of classical XP related to design, documentation and customer's involvement. Authors discussed that absence of proper architectural structure and design can affect agility of development process and software quality. Moreover authors highlighted some of the XP practices like pair programming and on-site customer, which cannot be used in all type of projects. It is already discussed in previous researches that pair programming and on-site customer practices are not useful for each type of projects. Authors mentioned a number of studies which tried to solve these problems but a common lapse in these solutions was the loss of process simplicity and agility. To overcome the limitations of classical XP without effecting its simplicity and agility, authors proposed SXP that introduced design and documentation activities in the model. Unnecessary rituals are eliminated from development process to maintain the simplicity and agility. The proposed SXP model consists of five development phases: 1) Initialization, 2) Analysis, 3) Design, 4) Development & Testing and 5) Release. SXP restricted the customer's involvement in first and last phase only. Main activities performed during initialization phase are requirement gathering and project planning. This phase involved personnel's from both customer and developer's side to decide the project scope, cost and technology used for development. Requirements are collected through story cards that have a small description of required functionality along with priority assigned. A project plan is created to document necessary details about project. Next is the analysis phase, in which activities like architectural structure, iteration plan and effort & cost estimation is performed and documented. SXP introduced an explicit design phase to overcome design related issues. In this phase software is designed using use-case and sequence diagrams which provide better development guidance to the developers. Test planning activity performed in this phase relates to the test first strategy of classical XP. Development and testing phase is an iterative phase in which coding and functional testing is performed repeatedly until successful code is integrated. After integration testing final acceptance testing is performed by customer in release phase. In case of successful testing final product is released to customer with user manuals.

Shortcomings:

SXP model incorporated analysis and design phase and produced documentation during each phase of the model. Although it is a better approach for medium scale projects having continuously changing requirements but for small and simple projects, this can create extra overhead for the development team. Furthermore there is an obvious need of empirical proof to check the effectiveness and efficiency of proposed model.

B. Component Based Software Architecture Refinement and Refactoring Method into Extreme Programming.

In [2], classical XP is modified by incorporating architectural and design related framework. This framework adds ability of component based architecture reusability in XP. Authors found that short term development time in XP is a hazard in developing reusable components. That's why newly required functionality can only be developed from scratch. To overcome this problem authors suggested component based architecture refinement framework for XP. They claimed that incorporating this framework in XP can reduce development time, effort and the cost of development. The steps in proposed reuse process model includes; i) Component search and retrieval ii) Identify component to extend and refine iii) Generate target component and iv) Repository management.

Proposed model suggested that instead of developing a new component for required functionality, reusability of already existing components should be checked. For this purpose reusable repository is maintained and searched before development of new functionality. In this step, components that can be reused in particular requirements

are selected. In next step selected components are being examined keenly to check which part is according to requirements and which should be eliminated. Based on their suitability these are identified as candidate for customization, refactoring, or extension. If there are some components that partially implement the functionality, then these components can be customized or extended to adopt required functionality. Refactoring can also be applied to the selected components. Refactoring is a stepwise activity in which a component is improved without changing its external behavior. These extended, customized or refactored components are refined further for the sake of efficiency. In last step of this framework repository is updated with newly created components for later use.

Shortcomings:

Although proposed framework covers a very important limitation of XP however reusability of existing components requires some extra effort in managing components repository. Moreover, in the absence of empirical proof or case study it is difficult to understand the working of framework in XP.

C. Proposal of Enhanced Extreme Programming Model.

In [3], authors proposed "Enhanced Extreme Programming model that claimed to improve agility and quality at same time. XP lacks documentation and design activities and uses extreme testing and refactoring during development iteration for quality enhancement. Due to sequential execution of testing and refactoring in classical XP, agility is affected badly. Authors found an inverse relationship between quality and agility. Lack of documentation and upfront design make it suitable for small projects only. To overcome quality, agility, documentation, design and response time related issues, authors proposed an enhanced XP model. Proposed EXP model introduced parallel refinement iteration along with actual development iteration. This helped in handling inverse relationship between agility and quality.

Proposed model has four phases; Initial iteration, Incremental iteration, Final iteration and Quality iteration. EXP keeps the classical XP development phases intact and adds a parallel refinement phase for non-functional requirements.

EXP development process starts with initial phase. In this phase development starts with basic development activities of XP like plan, design, code and test. However during coding, a refactoring team member works with programmers to understand and monitor coding activity. Later during refactoring of code, this team member helps other team members to resolve confusion and problems about written code. Authors suggested that team leader can be the best candidate for this role as he has to control both coding and refactoring teams. He should have good technical knowledge that helps to manage project easily.

Incremental iteration is an iterative phase that keeps iterating until the set goal is reached. It is not main development team's responsibility to refine design, code or test. Development team takes new requirements for each iteration and repeat plan, design, code and test activities with these requirements. On the other side, refactoring team starts quality iteration in parallel to incremental iteration to save time after completing first development iteration. For that purpose all artifacts of previous iteration are handed over to refactoring team. This refactoring is completed in supervision of team member that has previously monitored the designing and coding phase and has good understanding about developed code and design. First of all design and architectural documents are refined that give sufficient support for code refactoring. During code refactoring technical team leader works with refactoring team. Additional functionality can be added or previous code can be changed during refactoring. Then tests are refined according to added or changed functionality. In case of bugs, new changes can be made accordingly. Finally document refining team refines poorly written documents. Final iteration is completed by successfully implementing selected requirements of the iteration. Vague requirements now more clear to programmers that give better understanding about system.

Shortcomings of model:

This model used a parallel refinement cycle to ensure quality however the software projects having higher interdependencies among modules are difficult to build using this model. Another problem with this model is the need of more resources than usual. Parallel execution of development and refinement cycle demands more team members and other resources that increase development cost.

D. Mapping Formal Methods to Extreme Programming (XP) –A Futuristic Approach.

FXP is a modified version of XP that specially designed for life critical projects. Classical XP is usually used for small projects with no security and safety issues involved. In [4] authors proposed a new model that map formal methods on XP to incorporate agility in formal methods. Formal methods are used for life and safety critical projects. These methods are more precise and mathematical in nature to handle critical projects. However these models are laborious and costly to use both in term of effort and resources. Whereas XP model has limitations related to software design and architecture. To overcome these drawbacks, authors combined the strengths of both models in [4]. FXP used formal methods like Software Cost Reduction (SCR), Algebraic Specification and Design by Contract (DbC) in different phases to make it suitable for safety critical projects.

First step of FXP is about requirement gathering. FXP used story cards to collect requirement. This is same as used in classical XP where customer writes story cards to describe the required functionality in the system. The only difference is that a formal method Software Cost Reduction (SCR) is used in this process to formally represent the requirements. SCR used four tables (Condition Table, Event Transition Table, Linkage Table, and Directory) to represent requirement's conditions and

their association [4]. In condition table all the possible conditions used for a mode are listed. This is done for each mode to represent complete functionality of the system. Event transition table showed output regarding each mode, event and value. Whereas linkage table represents the association among different modes. Directory is used to keep the record of each datatype. Second phase of FXP is release plan. As the name suggest, a complete plan is developed for the project keeping in view the requirements collected in previous phase. Priority table is built to define order of development using condition table. In next step a Gantt chart is drawn to show the development time for each task. Concurrent task are also identified here to speed up development. Important decision about team size, code ownership, working hours, sitting arrangement and pair programming are taken in this phase. The authors have adopted most of the things from XP perspective however in pair programming it is considered that one of the programmer must be expert in formal methods. Next phase of FXP is iteration to release phase in which tasks are grouped in iterations. Each iteration can include new task to implement or task that failed to pass acceptance test in previous iteration. All possible test cases are also written in this phase. For this purpose a formal method called algebraic expression is used. Algebraic expression is a mathematical way of representing functions with their signature, return type and axioms. After completing this, programmers convert this specification in code. In next phase, implemented code is tested for final acceptance. FXP used Design by Contract (DbC) formal method. In DbC a contract is written using languages like java, JML or Eiffiel. This contract includes preconditions, invariants and post conditions. All the implementation is checked against this contract. In case of any nonconformity, code is fixed later by developers. Finally after successful acceptance testing recently implemented built is released to customer.

Shortcomings of model:

Incorporation of formal method in XP demands some extra training and expertise of development team. To prove the validity of FXP model, authors only relied on expert's opinion. A case study or empirical proof is strongly needed to prove the efficiency and authenticity of model.

E. Extended Iterative Maintenance Life Cycle Using eXtreme Programming.

Software maintenance is a continuous and unavoidable process for a software. Good maintenance can increase the quality and operational life of a software. in [5] authors found that most of the existing models of software maintenance are derived from traditional water fall model and hence not suitable for handling problems related to unstructured code, team morale, poor project visibility, communication and test suits. Authors suggested that these problems can be solved by using agile methodologies. Their iterative and incremental nature and emphasis on team collaboration, customer interaction makes them suitable for software maintenance [5]. Authors chose XP for this purpose due to its best practices and proposed an extended model for software maintenance that used IEEE 1219 standard from XP perspective. This extended model consists of seven phases that includes identification and categorization, planning, analysis, design revision, change implementation, acceptance testing and release. Request of change stories (RC stories and old software are input for this extended model. Authors have explained all the steps of this extended maintenance model using a case study however its general description is discussed here.

In first phase of identification and categorization end users submit change request in the form of RC stories. These stories are then analyzed by system analyst or service engineers to categorize in corrective, adaptive or perfective maintenance. This phase is conducted using planning game practice of XP moreover on-site customer practice is used to get customer opinion about change requests. In planning phase, release plan and iteration plan is developed. These plans are created by considering priority of RC stories however urgent RC can bypass planning and analysis phase. In this phase, estimation error can be occurred due to non familiarity of existing code. To handle this problem planning poker technique is used. Effort and cost estimation is completed along with decision about final release date. In analysis phase feasibility of requested maintenance is checked to create detailed analysis and feasibility report. Metaphor and on-site customer practices are used during this phase. Next phase is design revision phase that used feasibility and analysis report as input. Using these documents design of desired artifacts is changed without effecting overall integrity of the system. An updated design baseline is generated as output. Customer collaboration and prototypes are used to aid this process. Change implementation phase is the phase where change is actually made. Code is written which is then go through unit and integration testing. This phase used pair programming, test driven development, collective code ownership, continuous integration, standup meeting and refactoring practices of XP. Acceptance testing is conducted to check whether implemented change can give desired results. In case of successful testing user manuals, installation guide and training material is reviewed. Finally, release phase consists of activities like installation of software, final testing, user notification, deployment and user training. Authors presented the results of case studies which showed that using XP practices during software maintenance give much better results including improved team productivity and confidence.

Shortcomings of model:

The proposed model is applied in academic environment that is far more different from real scenarios. Proposed model is needed to be checked for real life problems.

F. A Framework for Partial Implementation of PSP in Extreme Programming.

In [6] authors proposed a modified version of XP by combining it with Personal Software Process (PSP). The aim behind this study was to combine the strengths of XP and PSP to develop high quality software. Authors suggested that using PSP, developers can improve their planning and estimation capabilities and can strive for quality by lowering the defect rate. Modified XP model presented in this paper showed that how all these personal qualities can be used with XP phases. This model includes six practices form XP and six practices from PSP. Proposed model in [6] consists of five phases named as Exploration phase, Planning phase, Personal Planning Phase, Iteration to release phase and Productionizing phase.

Exploration phase is same as in classical XP where requirements are collected through story cards. Development team also considers different possible architectural structures, tools and techniques for the developed system. In planning phase, stories are prioritized and selected for the current release. Development schedule and cost is estimated on the basis of selected stories. In this phase developers are free to take their own decisions. While in personal planning phase individual developer plan their activities and estimate time required to fulfill assign tasks. Use of coding standards, time and defect recording logs helped developers to evaluate their daily performance. Iteration to release phase can consists of many iterations to complete a release. After successful code completion, system moved towards productionizing phase. Some more testing is performed to check the validity of the developed system.

Shortcomings of model:

The proposed model can only be used for small scale projects where teams are small and familiar with PSP. Due to absence of practical application of this model, it is difficult to decide about its usability.

G. Estimation of the New Agile XP Process Model for Medium-Scale Projects Using Industrial Case Studies.

In [7], author tried to extend classical XP for medium scale projects with large development teams. This model tried to break the concept that XP can only be used for small scale projects with small teams. Author introduced analysis and risk management activity to overcome the project failure risk. Proposed model also tried to cover classical XP drawbacks like lack of up front design and no documentation. Proposed model is validated through two industrial case studies one for small scale projects and other for medium scale projects.

Proposed model consists of four phases that are project planning phase, analysis and risk management phase, design and development phase and testing phase.

In project planning phase, proposal document is prepared by using Cost-Benefit Analysis (CBA) technique. This document contains economic, technical and operational feasibility reports that help to check the overall feasibility of the project. Furthermore development team members are also selected in this phase. Team size can be vary by considering project size, and schedule. Analysis and risk management phase starts after checking the feasibility of the project. Detailed requirement are collected in this phase which are documented properly. This model used story cards for requirement elicitation like classical XP. Customer prioritizes these story cards on the basis of his needs. These story cards are then selected for current release using planning poker technique. In design and development phase, author combined design and development activities for the sake of agility and efficiency. Author used prototyping technique for design and requirement verification whereas refactoring is used during design and coding activities. This model used pair programming for coding. Programmers write code for current release's stories and in the meanwhile Interface Specification document for next release is also prepared. Design and coding activities are repeated until whole stories are implemented for the project. In the meanwhile implemented code is integrated continuously. In next phase implemented code is tested using unit testing, integration testing and system testing. Finally acceptance testing is conducted for customer verification. In case of successful acceptance testing, system is deployed at customer's site. Deployment further consists of installation, training and security activities.

Shortcomings of model:

Adding analysis and risk management phases in classical XP can affect agility of development process.

H. Role –Based Extreme Programming (XP) for Secure Software Development.

Although XP gives good performance in developing software within limited time and cost however it is also a fact that there is no emphasis on developing secure software in this model. To overcome this limitation authors proposed an additional role in [8]. Authors explored that there is no practices or role that help to develop secure software. In such situation developed software is open to security threats. Although using XP, a product can be delivered quickly but later it may require a lot of repair due to security risks. Authors proposed that by introducing a new role called "Security Master", we can lower down the effort and cost of later fixes which is required to make the software secure. Authors actually extend the model presented in [21] to develop secure software using XP. Authors introduced a new role and some additional security elements in XP practices to make it suitable for secure software development. They mentioned different practices for five major roles to implement security measures in XP. Here is the detail of XP practices for each role.

Customer: planning game, small releases, metaphor and on-site customer.

Coach: small releases, coding standards.

Manager: planning game, small releases, metaphor, small design, continuous integration and coding standards.

Programmer: planning game, small releases, metaphor, small design, refactoring, pair programming, collective code ownership, continuous integration, 40-hours week, coding standards.

Tester: testing, 40-hours week

However authors suggested that these practices cannot be implemented correctly until a professional person provide the guidance. Security master will be responsible for providing proper training to other team members regarding security, types of security attacks and their effects on software quality. XP practices related to security master includes planning game, small releases, metaphor, small design, refactoring, pair programming, collective code ownership, continuous integration, 40-hours week and coding standards.

Shortcomings of model:

Paper does not provide the implementation detail about role of security master. There is no guidance what type of rights he has and what type of rules he has to follow while interacting with other team members. There is no practical application of this model to get any idea about model authenticity.

I. Prioritizing CRC Cards as a Simple Design Tool in Extreme Programming.

Some researchers tried to enhance XP practices for better results. In [9], authors used Analytical Hierarchy Process (AHP) for prioritizing Class Responsibility Collaboration (CRC) cards. Authors found that among different XP design tools CRC cards are more effective tool for developing simple object oriented design. These cards provide flexible and quick help in finding object class, members and their relationship. However, prioritizing these cards has great effects on decisions regarding design. For example classes that are linked with many responsibilities have great effect on system coherence and need a lot of refactoring during implementation whereas collaboration among classes indicates the system coupling. To make this process simple and systematic, authors recommended the use of AHP in CRC cards prioritizing process.

AHP provides a systematic way of analyzing a problem that have multiple criteria. AHP is a hierarchical model that reflects human thinking process. As discussed in [9] AHP consists of five steps. In first step a hierarchy model is developed by breaking down the problem into interrelated decision elements. In next step a criteria is being defined to construct a pair wise comparison matrix. Each criteria on the same level is compared with other criteria with respect to their importance to the main goal. In next step a pairwise matrix for alternatives is constructed. Whereas consistency of judgment errors is calculated using consistency ratio. Finally to choose the highest score weighted average rating is calculated. Using AHP authors defined following criteria to prioritize CRC cards [9].

1: Which class responsibility has more effect on system?

2: Which classes have strongest relationship with other classes?

3: Which classes are stable and have fewer tendencies towards change?

Authors conducted an experiment consisting of 12 Master's students divided in two teams. These teams built same projects with different design tools. Authors found that team using AHP for CRC cards prioritization gave better results than other team.

Shortcomings of model:

Proposed solution requires handful expertise in applying AHP technique for CRC cards prioritization. That can be achieved by giving training or by hiring some expert in this field. However in both cases it will overburden development process and may affect agility and cost of development.

V. Findings and Discussion

This section provides the descriptive results of this mapping and gives answers to the research questions, whereas in table 2 these results are summarized for the quick view.

This mapping reveals that after 16 year of agile method's advent, it is still a hot area of research. Researchers are working on the improvement of agile process models, especially on customization and integration of these models. XP is one of the oldest and famous agile methods used in software industry. A large number of customized forms of XP are proposed by different researchers in past twenty years however this research is focusing to find the recent customized versions. The published literature which is considered for this research is from 2013 to 2017. After applying the selection criteria, we finally selected nine best appropriate papers. After critical analysis of these papers, following answers are found to the research questions.

RQ1: Which are the customized versions of XP process model?

All the papers discussed from [1] to [9] are the customized versions of XP. All these studies have done customization of XP model by either integrating or removing any particular practice, phase or role in the process lifecycle. The summary of these tailored models is presented in section IV of this research and also is available for quick view in table 2 and 3.

RQ2: Do the customized versions tailor XP phases?

All papers except [8], have changed the XP phases to make them according to their needs by introducing or removing different practices, activities and roles as shown in Table 2. In [8], a new role of security master is added.

Table 2. Summary of Analyzed papers in SLR

Model	SXP	Component Based Architecture refinement in XP	Enhanced Extreme Programming	FXP	Extended Maintenance Cycle Using XP	PSP in Extreme Programming	New XP Process model For medium scale projects	Role Based Extreme Programming	Prioritizing CRC cards in Extreme programming
Objective of customization	Resolving issues related to design and documentation for medium scale projects	Adding strength of component based architectural refinement and reusability in XP	Improving software quality without effecting agility of XP	Tailoring XP process for safety critical systems	Using XP for improvement of maintenance process by writing maintainable code	Increasing developer's productivity by integrating PSP practices in XP	Customizing XP for large and medium scale projects having large teams.	Using XP for developing secure software	Enhancing and simplifying design process
Suitable Project	Medium and Small Scale	Small scale	Small scale	Safety critical	N.A	Small Scale	Medium scale	Security critical software	N.A
Targeted practice(s)/ role(s)	Pair programming, On-site customer	Simple design, Refactoring	All XP practices	Pair programming, Collective Code ownership, 40 hours per week	All XP practices	XP and PSP practices	All XP practices	All XP practices	Not mentioned
Strategy used for customization	Added analysis phase and design phase in XP	Added component based reusability refinement in XP	Added a parallel refinement cycle to development cycle of XP for the sake of quality	Integrated formal methods in XP	Software maintenance activities are introduced using XP	Added PSP practices to improve developer's productivity in XP	Changed XP phases	Added a new role in XP	Used AHP technique for better design decisions
Changed XP phases	Yes	Yes	Yes	Yes	Yes	Yes	Yes	N.A	No
Validation	No	No	No	Yes	Yes	Yes	Yes	No	Yes
Methodology used for Validation	N.A	N.A	N.A	Survey	Empirical Proof	Survey	Empirical Proof	N.A	Empirical Proof

RQ3: Which practices, roles or events are included in XP to make it more effective and efficient?

The selected papers shows that the different practices, values, phases and roles are included by researchers during the customization. For improvement, they have targeted almost all activities of XP life cycle for software process improvement such as requirements gathering, designing, testing, maintenance, security, quality control. The detail regarding the nature of customization along with the practices, roles, events, activities and phases is provided in section IV of this research and also is available for quick view in Table 2.

RQ4: What objectives are achieved through XP customization?

It has been found that the primary objective of the customization of each selected paper is to achieve the good quality end product by improving software process life cycle in an effective and efficient way. Researchers have tailored the classical XP model by integrating or removing particular phases, practices, events and roles to make it fit for certain projects in certain circumstances. The proposed model in [1] helped to develop medium scale projects with better design and documentation opportunities. Model proposed in [2] introduced concept of reusability to overcome architectural issues. Model proposed in [3] tried to improve quality without effecting agility of the process. In [4], formal methods are used in XP phases to make it suitable for safety critical projects. In [5], an XP based model is proposed for software maintenance. Paper [6] used Personal Software Process (PSP) to improve the quality and project planning abilities of developers. In [7], XP is customized to handle large scale projects with big teams. In [8], a new role is introduced to develop secure software. Paper [9] introduced a new prioritizing technique for better design decisions.

RQ5: Do these modified or improved models are validated through empirical proof?

Papers [1] [2] [3] and [8] gives no empirical validation of proposed model whereas in paper [4] a survey is conducted from software professionals to validate proposed model. Papers [5] [6] [7] and [9] used empirical proof to validate their studies.

Table 3. List of Selected Papers

Year of Publication	Paper's Title	Paper Type	Journal/ Conference Name	Objective of Research
2017	SXP: Simplified Extreme Programing Process Model [1]	Journal	International Journal of Modern Education and Computer Science	• Using XP for medium scale projects. • Handling design and documentation related limitations
2016	Component Based Software Architecture Refinement and Refactoring Method into Extreme Programming [2]	Journal	International Journal of Advanced Research in Computer and Communication Engineering	• Introduction of reusability in XP • Adding strength of component based architectural refinement in XP
2015	Proposal of Enhanced Extreme Programming Model [3]	Journal	International Journal of Information Engineering and Electronic Business	• Improving software quality without effecting agility • Improving architectural design and documentation
2014	Mapping Formal Methods to Extreme Programming (XP) –A Futuristic Approach [4]	Journal	International Journal of Natural and Engineering Sciences	• Making XP suitable for safety critical system
2014	Extended Iterative Maintenance Life Cycle Using eXtreme Programming [5]	Journal	ACM SIGSOFT Software Engineering Notes	• Writing maintainable code that require less maintenance effort later • Speeding up maintenance process using XP
2013	A framework for partial implementation of PSP in Extreme programming [6]	Journal	International Journal of Engineering Research and Applications	• Improving developer's productivity • Improving software quality and project planning
2013	Estimation of the New Agile XP Process Model for Medium-Scale Projects Using Industrial Case Studies [7]	Journal	International Journal of Machine Learning and Computing	• Making XP suitable for medium/ large scale projects having large team
2013	Role Based Extreme Programming (XP) for Secure Software Development [8]	Journal	Science International	• Using XP for developing secure software
2013	Prioritizing CRC Cards as a Simple Design Tool in Extreme Programming [9]	Conference	IEEE Canadian Conference Of Electrical And Computer Engineering	• Enhancing and simplifying design process

VI. Conclusion

In this paper a systematic literature review is conducted to explore the current state of XP customizations. A research methodology is defined with inclusion and exclusion criteria to extract relevant data. According to defined criteria, nine most appropriate papers are selected for review. Critical analysis of these papers revealed that after almost twenty years of XP's invention, researchers are still working on its transformation to make it more adoptable and effective.

Results showed that researchers customized XP phases and practices to achieve best results in different projects. Some researchers merged more practices and elements to make it applicable for specific needs. XP limitations which make it difficult to use in large and critical natured projects are eliminated by customization. However main objective of these customizations is to make XP suitable for different type and size of projects along with maintaining the agility. High quality software development with reduction in cost, time and effort is a big achievement but lack of empirical proof make it difficult to access the applicability of proposed models. To prove the innovation and effectiveness of proposed model, empirical validation is strongly recommended.

References

[1] F. Anwer and S. Aftab, "SXP: Simplified Extreme Programing Process Model," *International Journal of Modern Education and Computer Science (IJMECS)*, vol. 9, no. 6, pp. 25-31, 2017.

[2] S. Nagalambika, R. Majunath and K.S. Praveen, "Component Based Software Architecture Refinement and Refactoring Method in Extreme Programming," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 5, no. 12, 2016.

[3] M. R. J. Qureshi and J. S. Ikram, "Proposal of Enhanced Extreme Programming Model," *International Journal of Information Engineering and Electronic Business*, vol. 7, no. 1, p.37- 42, 2015.

[4] T. Saeed, S.S. Muhammad, M.A. Fahiem, S. Ahamd, M.T. Pervez and A.B. Dogar, "Mapping Formal Methods to Extreme Programming (XP)–A Futuristic Approach," *International Journal of Natural and Engineering Sciences*, vol. 8, no. 3, pp.35-42, 2014.

[5] J. Choudhari and U. Suman, “Extended iterative maintenance life cycle using eXtreme programming,” *ACM SIGSOFT Software Engineering Notes*, vol. 39, no. 1, pp.1-12, 2014.

[6] N. Iqbal, M.U. Hassan, A.R. Osman and M. Ahmad, “A framework for partial implementation of PSP in Extreme programming,” *International Journal of Engineering Research and Applications*, vol. 3, no. 2, pp.604-60, 2013.

[7] M. Qureshi, “Estimation of the new agile XP process model for medium-scale projects using industrial case studies,” *arXiv preprint arXiv:1408.6228*, 2014.

[8] I. Ghani, N. Izzaty and A. Firdaus, “Role-based Extreme Programming (XP) For Secure Software Development,” in *Special Issue-Agile Symposium*, pp. 1071-1074, 2013.

[9] 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.

[10] E.M. Schön., J. Thomaschewski and M.J. Escalona, “Agile requirements engineering: a systematic literature review,” *Computer Standards & Interfaces*, vol. 49, pp.79-91, 2017.

[11] I. Inayat, S.S. Salim, S. Marczak, M. Daneva and S. Shamshirband, “A systematic literature review on agile requirements engineering practices and challenges,” *Computers in human behavior*, vol. 51, pp.915-929, 2015.

[12] T. Karvonen, W. Behutiye, M. Oivo and P. Kuvaja, “Systematic literature review on the impacts of agile release engineering practices,” *Information and Software Technology*, 2017.

[13] K. Dikert, M. Paasivaara and C. Lassenius, “Challenges and success factors for large-scale agile transformations: A systematic literature review,” *Journal of Systems and Software*, vol. 119, pp.87-108, 2016.

[14] T. Dreesen, R. Linden, C. Meures, N. Schmidt and C. Rosenkranz, “Beyond the Border: A comparative literature review on communication practices for agile global outsourced software development projects,” *System Sciences (HICSS), 49th Hawaii International Conference* pp. 4932-4941, IEEE, 2016.

[15] A.S. Campanelli and F.S. Parreiras, “Agile methods tailoring–A systematic literature review,” *Journal of Systems and Software*, vol. 110, pp.85-100, 2015.

[16] M. Usman, E. Mendes, F. Weidt, and R. Britto, “Effort estimation in agile software development: a systematic literature review,” in *Proceedings of the 10th International Conference on Predictive Models in Software Engineering*, pp. 82-91, ACM, 2014.

[17] I. Ghani and I. Yasin, “Software Security Engineering in Extreme Programming Methodology: A Systematic Literature Review,” *Science International*, vol. 25, no. 2, 2013.

[18] T.S. Da Silva, A. Martin, F. Maurer and M. Silveira, “User-centered design and agile methods: a systematic review,” in *Agile Conference (AGILE)*, pp. 77-86, IEEE, 2011.

[19] J.E. Hannay, T. Dybå, E. Arisholm and D.I. Sjøberg, “The effectiveness of pair programming: A meta-analysis,” *Information and Software Technology*, vol. 51, no. 7, pp.1110-1122, 2009.

[20] N. Salleh, “A systematic review of pair programming research-initial results,” in *Proc. New Zealand Computer Science Research Student Conference (NZCSRSC08), Christchurch*, 2008.

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

[22] P. Brereton, B.A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, “Lessons from applying the systematic literature review process within the software engineering domain,” *Journal of systems and software*, vol. 80 no. 4, pp. 571-583, 2007.

[23] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, P. W. Jones, D. C. Hoaglin, K. El Emam, and J. Rosenberg, “Preliminary guidelines for empirical research in software engineering,” *IEEE Transactions on software engineering*, vol. 28, no. 8, pp.721-734, 2002.

[24] B. A. Kitchenham and S. Charters, “Procedures for Performing Systematic Literature Review in *Software Engineering,” EBSE Technical Report* version 2.3, EBSE-2007-01, Software Eng. Group.

[25] A. Alliance. 2001 "Agile manifesto," [Online]. Available: http://agilemanifesto.org/

[26] F. Anwer, S. Aftab, U. Waheed, and S. S. Muhammad, “Agile Software Development Models TDD, FDD, DSDM, and Crystal Methods: A Survey,” *International Journal of Multidisciplinary Sciences and Engineering*, vol. 8, no. 2, 2017.

[27] F. Anwer, S. Aftab, U. Waheed, and S. S. Muhammad, “Comparative Analysis of Two Popular Agile Process Models: Extreme Programming and Scrum*,” International Journal of Computer Science and Telecommunications* vol. 8, no. 2, March 2017.

[28] J. Newkirk, “Introduction to agile processes and extreme programming,” in *Proc. 24th International Conference of Software engineering*, pp. 695-696, May 2002.

[29] K. Petersen, R. Feldt, S. Mujtaba and M. Mattsson, “Systematic Mapping Studies in Software Engineering,” in *EASE*, vol. 8, pp. 68-77, 2008.

[30] K. Beck, “Extreme programming explained: embrace change,” addison-wesley professional, 2000.

[31] P. Abrahamsson, O. Salo, J. Ronkainen and J. Warsta, “Agile software development methods: Review and analysis,” *VTT publ.*, pp. 3-107 2002.

[32] R. Juric, “Extreme programming and its development practices,” in. *Proc. 22nd Int. Conf. Information Technology Interfaces*, IEEE, pp. 97-104, Jun. 2000.

[33] B. Kitchenham, O.P. Brereton, D. Budgen, M. Turner, J. Bailey and S. Linkman, “Systematic literature reviews in software engineering–a systematic literature review,” *Information and software technology*, vol. 51 no. 1, pp.7-15, 2009.

[34] M Bano. and D. Zowghi, “User involvement in software development and system success: a systematic literature review,” in *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering*, pp. 125-130, ACM, 2013.

[35] O. Kobayashi, M. Kawabata, M. Sakai and E. Parkinson, “Analysis of the interaction between practices for introducing XP effectively,” in *Proc. 28th International conference of Software Engineering*, pp. 544-550, May 2006.

[36] F. Anwer, S. Aftab and I. Ali, “Proposal of Tailored Extreme Programming Model for Small Projects,” *International Journal of Computer Applications (IJCA)*, vol. 171, no. 7, pp. 23-27, 2017.

[37] G. Rasool, S. Aftab, S. Hussain and D. Streitferdt, “eXRUP: A Hybrid Software Development Model for Small to Medium Scale Projects,” *Journal of Software Engineering and Applications*, vol. 6, no. 09, p.446, 2013.

[38] S. Ashraf and S. Aftab, “Latest Transformations in Scrum: A State of the Art Review,” *International Journal of*

Modern Education and Computer Science (IJMECS), vol. 9, no.7, pp.12-22, 2017.

[39] Z. Nawaz, S. Aftab and F. Anwer, " Simplified FDD Process Model," *International Journal of Modern Education and Computer Science (IJMECS)*, vol. 9, no.9, pp. 53-59, 2017.

Authors' Profiles

Faiza Anwer is student of MS Computer Science with the specialization of Software Engineering in Virtual University of Pakistan. Her areas of interest are Software Process Improvement and Agile Development Models.

Shabib Aftab is working as Lecturer in Computer Science Department at Virtual University of Pakistan. He completed MS degree in Computer Science from COMSATS Institute of Information Technology, Lahore and previously he got M.Sc degree in Information Technology from Punjab University College of Information Technology (PUCIT), Lahore. His areas of research are Data Mining and Software Process Improvement.

How to cite this paper: Faiza Anwer, Shabib Aftab, "Latest Customizations of XP: A Systematic Literature Review", International Journal of Modern Education and Computer Science(IJMECS), Vol.9, No.12, pp. 26-37, 2017.DOI: 10.5815/ijmecs.2017.12.04

