SXP: Simplified Extreme Programing Process Model

Article *in* International Journal of Modern Education and Computer Science · June 2017

DOI: 10.5815/ijmecs.2017.06.04

2 authors:

Faiza Anwer
Virtual University of Pakistan
10 PUBLICATIONS **300** CITATIONS
SEE PROFILE

Shabib Aftab
Virtual University of Pakistan
52 PUBLICATIONS **2,197** CITATIONS
SEE PROFILE

I.J. Modern Education and Computer Science, **2017, 6, 25-31**
Published Online June 2017 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2017.06.04

SXP: Simplified Extreme Programing Process Model

Faiza Anwer
Department of Computer Science, Virtual University of Pakistan
Email: faiza.anwer28@gmail.com

Shabib Aftab
Department of Computer Science, Virtual University of Pakistan
Email: shabib.aftab@gmail.com

***Abstract*—**Extreme programming is one of the widely used agile models in the software industry. It can handle unclear and changing requirements with the good level of customer satisfaction. However Lack of documentation, poor architectural structure and less focus on design are its major drawbacks that affects its performance. Due to these problems it cannot be used for all kinds of projects. It is considered suitable for small and low risk projects. It also has some controversial practices that cannot be applied in each and every situation like pair programming and on-site customer. To overcome these limitations a modified version of XP called "Simplified Extreme Programming" is proposed in this paper. This model provides solution of these problems without affecting simplicity and agility of extreme programming.

***Index Terms*—**Agile models, Extreme programming, Drawbacks, Improved Extreme Programming, SXP

I. Introduction

Agile software development models provide an iterative and incremental way of software development that delivers the product with more emphasis on customer satisfaction, team collaboration and managing changing requirements [20]. Agile manifesto contains twelve foundation principles of agile software development. These principles are about frequent team communication, customer satisfaction, managing frequent requirements changing and early delivery of partial working software.

A number of agile software development models exist but extreme programming (XP) is one of the most widely used agile model [1]. It was developed by Kent Beck in 2000 when software industry was seeking for new software development methods to reduce the risk of failure caused by traditional development models. It contains all salient features of agile development. XP is an iterative and incremental model that simply uses small iterations starting from the very basic features of the system to complete software in later releases. XP's Development process consists of six phases called Exploration phase, Planning phase, Iteration to release phase, Productionizing phase, Maintenance phase and Death phase [21].

During exploration phase user requirements are gathered in the form of story cards [35]. Customer writes story cards for each feature. In planning phase, a release plan and iteration plan is prepared [21]. During this phase collected requirements are prioritized using numerical and ranking prioritization technique [26]. Actual development activities take place in Iteration to release phase that incorporate the basic development activities like designing, coding, testing and integration [22]. Programmers (in pair) select tasks to implement, design simply and then write code for these tasks. Unit and integration testing is performed after coding that are good source of instant feedback. In case of any problem code can be refactored to make it according to requirements. These activities are performed iteratively until a workable product is ready to release in productionizing phase [21] [22] [31].

XP practices, values and principles are distinguishing features of XP that provide a guideline throughout the development process. There are twelve XP practices namely planning game, small releases, metaphor, pair programming, refactoring, collective code ownership, on-site customer, continuous testing, simple design, continuous integration, 40 hour work and coding standards [20] [27] [30]. Although XP has a lot of advantages, there are some limitations too. Some of its major drawbacks are weak documentation, absence of proper architectural structure and system design. In XP, focus remains on coding not on software design. Software design activity is performed by developers at the start of iteration for a short time that cannot be considered as proper design phase. No design documents or diagrams are produced during this activity. Absence of good design may distract the development team that result in a lot of refactoring [23]. Lack of documentation in XP is also a big hindrance when the system is maintained over a long time [24].

Along with these problems some of XP practices can cause inconvenience during development like pair programming and on-site customer [23] [25]. Pair programming practices require mutual understanding and common skill set of programmers otherwise it will be difficult for them to work together [25]. On-site customer practice is difficult to implement. Sometimes customer does not understand the importance of their feedback or it

may not be possible for customer to remain at site regularly [23]. It is very rare that professionals have enough time to spend on site regularly.

These problems need a solution. Although a number of customized and extended versions of XP exist that are helpful in using XP for some specific purpose. However these models do not pay attention towards solving above mentioned problems. Furthermore adding more practices to the phases of XP might make it complex and difficult to implement. A modified XP model is needed that can solve these problems without effecting its simplicity and agility.

Remaining part of this paper contains related work in section II. Section III defines problem statement. Section IV provides detailed description of proposed SXP model. Finally section V concludes this paper.

II. Related Work

In [1] an enhanced extreme programming model is proposed that tries to cover the problems of documentation, design and quality without effecting agility. This is done by executing parallel quality iteration to basic XP iteration. However proposed model does not support development of software with higher interdependencies among subsystems. This paper lacks empirical proof of the model.

A process model based on XP is proposed for software maintenance in [2]. This model uses XP practices in software maintenance process to improve the productivity. Proposed model is evaluated using academic projects only, whereas real business projects are far more complex than academic projects. The proposed model should be evaluated using real projects.

In [3] authors conducted field studies on backup behavior of developer's interaction in different environments. The results showed that there are different kinds of interaction among team members which require different level of formality of pair programming. This study also explains why there are so many conflicts in perceiving the benefits of pair programming. However the observational results presented in this paper are limited to only two project's data.

Authors proposed a model based on XP for large scale distributed projects in [4]. This model was applied on Sudan Automated Traffic Violations project. Proposed model introduced some new XP practices like code control, adaptive planning, visual indicators, XP project management and code gallery. This model is validated for only one project that cannot guarantee the suitability of XP for all large scale projects. Interaction and collaboration among team members is difficult and sometimes not possible in large scale projects as needed in XP. This model does not guide how to deal with such team collaboration problems.

In [5] authors proposed a new solution for service development by combining XP and SOA (Service Oriented Architecture) best practices. In proposed solution seven SOA principles are combined with supported XP practices for this purpose. This paper lacks empirical validation to prove the effectiveness of proposed solution. Furthermore the proposed solution remains silent on the issue of SOA complexities that can reduce the agility of XP.

In [6] CRC cards prioritization process is done using AHP (Analytical Hierarchy Process). AHP is a well-structured decision making tool. Use of AHP in cards prioritizing process helped the designers to identify most influential classes for simple and appropriate design. This paper lacks real test cases evaluation of proposed model. Furthermore proposed model is tested by graduate students, which might have not the enough skills to evaluate the solution.

In [7] a study was conducted to evaluate the role of agile techniques and factors associated with the performance of team using XP. Results showed that customer and development team are both very important for XP process however measure of performance was greatly dependent upon subjective interpretation in this study.

Authors conducted a study in [8] to prove the effectiveness of pair programming in defects reduction. For this purpose data is collected from professionals working in a large Italian manufacturing company. Statistical results showed that new defects tend to reduce with pair programming. But this case study did not consider the factors like task complexity, skill level and experience of developers.

In [9] an extended XP model is presented that can be applied to medium and large scale projects. Proposed model introduced project planning, analysis & risk management and design & development phase to handle medium and large scale projects. Extended XP model does not provide any detail about parallel development in large projects. Statistical validation is also needed with more accurate sample size.

In [10] authors proposed a model that helped development team as well as the customer in the release planning activity. This model helped in developing a release plan by keeping in view the size of stories, priorities and precedence relations. However tool used in this model requires a lot of time in data gathering that can affect responsiveness of XP.

A comparative study was conducted in [11] to compare the outcomes of XP and Waterfall methodologies on same project. In this study same project was developed by fifty different teams over the period of five years. Results were unexpectedly same for both methodologies. However this study lacks diversity of data characteristics and data source as comparison is conducted for same project repeatedly.

An improved XP framework is proposed in [12] that fulfills security requirements in e-commerce projects. This framework incorporate security checks in all phases of XP but this can affect the agility of the model. It is also needed to prove the effectiveness of model by using real life projects.

In [13] a study was conducted by author to find the effectiveness of virtual pair programming as a replacement of classical pair programming.

Table 1. Summary of Related Work

Title	Limitations
Proposal of Enhanced Extreme Programming Model [1]	• Proposed model does not support development of software with higher interdependencies. • Adding parallel refinement cycle along with development cycle demands more resources and team members.
Extended Iterative Maintenance Life Cycle Using eXtreme Programming [2]	• Proposed solution is validated through academic projects only, it should be evaluated by real business projects that are more complex than academic projects.
Cooperation, collaboration and pair-programming: Field study on backup behavior [3]	• Observational results presented in paper are based on the data of only two projects.
Extreme Programming Applied in Large Scale Distributed System [4]	• Proposed solution is validated for only one project that cannot guarantee the suitability of XP for all large scale projects. • Team interaction and collaboration is difficult and sometimes not possible in large scale projects as needed in XP. This model does not guide how to deal with such team collaboration problems.
Service Agile Development Using XP [5]	• This model integrates XP with Service Oriented Architecture (SOA) practices that effect simplicity of XP. • SOA complexities can degrade agility of XP.
Prioritizing CRC as a Simple Design Tool in Extreme Programming [6]	• Proposed model is tested by graduate students, who might not have enough skills to evaluate a solution. • This paper lacks real test case evaluation of proposed model.
Successful extreme programming: Fidelity to the methodology or good team working? [7]	• In proposed solution, performance evaluation depends upon subjective interpretation.
Pair Programming and Software Defects- a large, industrial case study [8]	• Case study presented in this paper does not consider the factors like skill level, experience of programmers and task difficulty etc.
Agile software development methodology for medium and large projects [9]	• Proposed XP model does not provide any detail about parallel development in large projects. • Statistical validation is needed with more accurate sized sample space.
Quantitative release planning in extreme programming [10]	• The optimization model presented is of exponential complexity. • Precise data gathering require a lot of time that can affect responsiveness of XP.
Comparing Extreme Programming and Waterfall Projects Results [11]	• Comparison is conducted for one project's data only. • Lacks diversity of data characteristics and data source.
Improved Extreme Programming Methodology with inbuilt security [12]	• It is needed to prove the effectiveness of proposed model using real life projects with security requirements. • Security checks implementation in each XP iteration can affect agility of whole process.
Measuring the effect of Virtual Pair Programming in an Introductory Programming Java Course [13]	• A large sample space can better support the results. • Data should be collected over more semesters.
Research on Requirement for High Quality Model of Extreme Programming [14]	• Despite of improvements in communication model it also makes it lengthy and complicated. • This paper lacks empirical evaluation of proposed method.
The impact of absorptive capacity on the ex-post adoption of agile methods: The case of extreme programming model [15]	• Data should be collected from various sites to generalize the results of study.
Agile Software Engineering as Creative Work [16]	• Evaluation is needed by proposing detailed method to improve XP using case studies.
An adoptive Software Development Model [17]	• Adoptive model has no guidance about project and team management issues in large projects.
An Improved XP Software Development Model [18]	• Proposed model lacks implementation detail about different analysis and risk management activities.

Results from two teams were compared where one team was using solo programming. Data is compared using metrics like Lines of Code (LOC), defects per 1000 LOC, code quality and productivity for both teams. Results showed that virtual pair programming is better than solo programming. But to validate the results of this study and prove its authenticity data should be collected over more semesters.

In [14] authors tried to solve the problems regarding customer such as bidirectional communication, information barriers and misconception about development process in XP by presenting an analysis model. This model improves XP demand module by using Kano model's quality features. This paper lacks real life project evaluation of proposed method.

Authors studied two projects in Canadian organization in [15]. These projects were shifted from waterfall to XP process model by using absorptive capacity in Technology Acceptance Model (TAM). Results showed the feasibility of XP for future projects but data collected from single site limits the accuracy of results. Data should be collected from various sites to generalize the results of study.

In [16] authors conducted a comparative study among phases and roles of XP and creativity process. However this study needs evaluation by proposing detailed method to improve XP using case studies.

In [17] authors proposed an adoptive XP model. Proposed model provided better adoptability to different software projects by including analysis, design and deployment phases. However there was no empirical proof in the paper to support the claim. This adoptive model also remained silent about project and team management aspects of large projects.

An improved model of XP is proposed for medium and large scale projects in [18]. Improved version of XP support component based development with risk management in distributed environment with large team. However this paper lacks implementation detail of analysis and risk management activities. Furthermore there was no empirical proof given to support the claim.

III. Problem Definition

Extreme programming is one of the most commonly used agile models. It's flexible, lightweight and iterative nature can easily handle changing requirements even in late phases of software development [1] [19] [29] [30]. XP's actual strength lies in its principles, values and practices that provide actual guidance for the software development process [32]. It works well for small projects however lack of documentation, complicated structure and poor system design make it inappropriate for medium and large scale projects [23] [24]. There is no upfront architectural structure available in XP as well as no explicit design activity is performed. Without architecture and design activities programmers do not get better understanding of the task. In such situation they have to rely on code refactoring that increase time and effort. Furthermore distribution of tasks among team members is problematic due to absence of system design [25] [34]. XP lacks the documentation and only the oral communication among stakeholders cannot be as effective as documenting the different tasks. For effective software development and maintenance proper documentation in different phases of the software development model is required. [24].

Along with these deficiencies, XP has some unnecessary norms like pair programming and on-site customer. Pair programming requires high level of coordination between two programmers. Difference in skill level, experience and personalities can degrade its effectiveness [23] [28] [34]. Sometime programmers feel easy to work alone without other person's interruption. Similarly on-site customer practice is difficult to implement in its true sense if customer does not understand importance of feedback [23]. Mostly professionals from customer's side have not enough time to remain present all the time and if this task is assigned to some inexperienced then it may lead to chaos [33]. Wrong information provided by that person can mislead the development team.

In the quest of solution to these problems, researchers tried to extend or modify XP process model for varying size and type of projects [1]. Although these models tried to cover limitations but also added complexity by adding more practices and modifications which brought negative effects on simplicity and agility. There is a need of a model that solves these problems and cut unnecessary norms from the practices of XP such as use of pair programming and on-site customer. By keeping in view these problems, we tried to find answer of the following question in this paper.

How to eliminate XP's limitations without affecting its true sense of simplicity and agility for small and medium scale projects?

IV. Proposed SXP

Simplified Extreme Programming (SXP) is proposed to overcome the limitations of classical XP. It provides more flexible and simple approach for small to medium scale projects due to explicit focus towards architecture, design and documentation issues. It also removes constraints of pair programming and on-site customer to avoid unnecessary conflicts and interference. There are five major phases of SXP; initialization phase, analysis phase, design phase, development & testing phase and release phase as shown in fig.1. Customer involves in initialization and release phase only. All other phases are executed by development team with the complete coordination. Necessary documentation is produced during each phase that helps to resolve documentation and maintenance related issues. Proper analysis and design phase provide opportunity to explicitly focus on design after analysis.

A. Initialization Phase

This is the first phase of SXP that focus on requirement gathering and preparing overall project plan for the system to be developed. Initialization phase has two major activities called "Story Writing & Prioritization" and "Project Planning". In this phase personnel's from both customer and developer's side sit together and complete the following tasks.

Story Writing & Prioritization: During this activity all requirements of the desired system are gathered and

arranged in accordance to their importance. Requirements are collected by writing story cards. Customer writes story cards for each feature/functionality that should be added in system. A story card consists of name of feature, type, priority and short description of required functionality. Customer has to describe feature in a small paragraph without any technical detail. Customer assigns priorities to these features that help in defining the order of their implementation. . For this purpose numerical Priorities are assigned to these requirements. High priority requirements are implemented prior to low priority requirements. Collected requirements are further categorized in functional and nonfunctional requirements.
Project Planning: During this activity important decision are taken regarding project scope, cost and tools/ technology to be used for the development. Development team and customer finalize the project scope and cost. For the selection of suitable tools and technology, different available options are considered. After agreement of both parties, detail about scope, cost and tools to be used is documented in project plan document.

B. Analysis Phase

In this phase budget and schedule related issues are addressed. Activities of this phase are conducted by development team only. Estimation is made about budget required for the successful completion of the project. An iteration plan is documented having detail about number of iterations needed for project completion, number of stories implemented in each iteration and iteration time. This iteration plan helps to keep project on track.

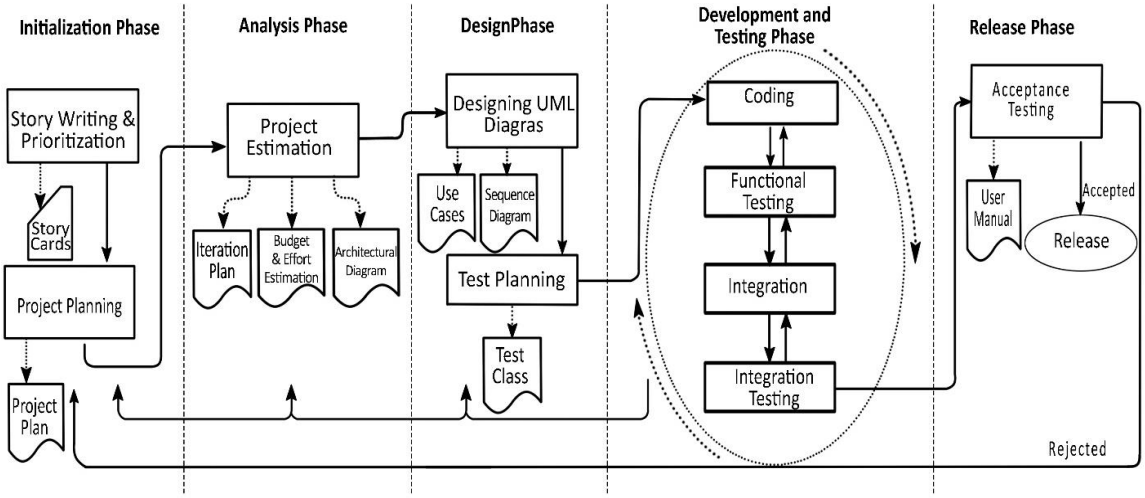

Fig.1. Phases of Proposed SXP

For budget and effort estimation, resources such as hardware and softwares are identified with their availability. To understand overall project structure, an architectural diagram is designed by the programmers. Also the training can be conducted in this phase to make the development team familiar with tools and technology.

C. Design Phase

This phase consists of two activities namely "Designing UML Diagrams" and "Test Planning".

Designing UML Diagrams: System design is very important for successful software development. To simply design the system, this model uses only use case diagrams and sequence diagrams.

Test Planning: In this activity developer writes test cases for the features to be included in iteration. Writing tests prior to code help the development team to understand different design opportunities. Successful completion of this phase gives a good start of development phase.

D. Development and Testing Phase

This is an iterative phase in which actual development activity take place. This phase further consists of activities namely coding, functional testing, integration and integration testing.

Coding: In coding activity, a programmer writes code for selected stories by keeping in view the design document which was developed during design phase.

Functional Testing: Implemented modules are tested using test cases written during test planning activity. In case of any problem during functional testing, coding activity can be repeated. These tests are performed by programmers and results are also noted to keep track of defects.

Integration: After successful completion of functional testing code is integrated with previously implemented code.

Integration Testing: After integrating the code, integration testing is performed to check whether all the implemented modules are working correctly as a whole or not. In case of any incompatibility, previous activities can be repeated. Feedback arrows from development and

testing phase indicates that any identified problem during this phase may require the revisit design, analysis or initialization phase.

E. Release Phase

This is final phase of SXP in which customer performed acceptance testing. After the customer approval a workable product which is developed during current iteration is released. User manual is also completed in this phase before handing over the workable product.

If developed product does not satisfy customer then whole development process is repeated again with changed or modified set of requirements.

V. Conclusion

Extreme programming is a well-known, most widely used agile model in software industry. It has more focus towards customer satisfaction, quick response to changing requirements, team collaboration, rapid feedback and small releases. Despite of these advantages there are some limitations also. Lack of documentation, poor architectural structure, and less focus on design are big problems of XP. Some of XP practices like Pair programming and on-site customer are a bit difficult and controversial from implementation point of view. Many factors involved in the implementation of these practices make their effectiveness questionable. A number of studies were conducted in which researchers extended or customized XP process model. These models were proposed to handle different projects varying in size, type or nature. Some of the proposed models pay more attention towards customizing its phases and some added new practices and provided implementation detail. But do not provide proper guidance about handling XP's drawbacks and limitations. Furthermore adding more practices or modification in phases affects its simplicity and agility that make it difficult to implement. In this paper a modified version of XP called Simplified Extreme Programming (SXP) is proposed that can help to cover all these limitations without effecting simplicity and agility. Detail description of each phase gives stepwise solution of XP's problems with keeping it simple to implement.

References

[1] 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.

[2] 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.

[3] 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.

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

[5] F. Carvalho and L. Azevedo, "Service Agile Development Using XP," in *Service Oriented System Engineering* (SOSE), Redwood City, 2013.

[6] 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.

[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] 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.

[9] M. Rizwan Jameel Qureshi, "Agile software development methodology for medium and large projects," *IET Software*, vol. 6, no. 4, pp. 358 - 363, 2012.

[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] F. Ji and T. Sedano, "Comparing extreme programming and Waterfall project results," in *Software Engineering Education and Training* (CSEE&T), Honolulu, HI, 2011.

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] M. R. J. Qureshi and S. A. Hussain, "An adaptive software development process model," *Advances in Engineering Software*, vol. 39, no. 8, pp.654-658, 2008.

[18] M. R. J. Qureshi and S. A. Hussain, "An Improved XP Software Development Model," arXiv preprint arXiv:1202.2501, 2008.

[19] G. S. Matharu, A. Mishra, H. Singh, and P. Upadhyay, "Empirical study of agile software development methodologies: A comparative analysis," *ACM SIGSOFT Software Engineering Notes*, vol. 40, no. 1, pp. 1-6, 2015.

[20] P. Abrahamsson, O. Salo, J. Ronkainen and J. Warsta, "Agile software development methods: Review and analysis," 2002.

[21] K. Beck, "Extreme programming explained: embrace change," addison-wesley professional, 2000.
[22] M. C. Paulk, "Extreme programming from a CMM perspective," *IEEE software*, vol. 18, no. 6, pp. 19-26, 2001.
[23] A. Dalalah, "Extreme Programming: Strengths and Weaknesses," *Computer Technology and Application*, vol. 5, no. 1, 2014.
[24] R. Fojtik, "Extreme Programming in development of specific software," *Procedia Computer Science*, vol. 3, pp.1464-1468, 2011.
[25] R. Crocker, "The 5 reasons XP can't scale and what to do about them," *Proceedings of XP*, 2001.
[26] J. A. Khan, I. U. Rehman, Y. H. Khan, I. J. Khan and S. Rashid, "Comparison of requirement prioritization techniques to find best prioritization technique," *International Journal of Modern Education and Computer Science*, vol. 7, no. 11, p.53-59, 2015.
[27] M. Khalid, S. ul Haq and M. N. A. Khan, "An assessment of extreme programming based requirement engineering process," *International Journal of Modern Education and Computer Science*, vol. 5, no. 2, p.41-47, 2013.
[28] S. Beecham, H. Sharp, N. Baddoo, T. Hall and H. Robinson, "Does the XP environment meet the motivational needs of the software developer? An empirical study," in *Agile Conference (AGILE), 2007* pp. 37-49, IEEE.
[29] J. Newkirk, "Introduction to agile processes and extreme programming," in *Proc. 24th Int. conf. Software engineering*, pp. 695-696, 2002.
[30] E. R. Mahajan and E. P. Kaur, "Extreme Programming: Newly Acclaimed Agile System Development Process," *International Journal of Information Technology*, vol. 3, no. 2, pp.699-705, 2010.
[31] R. Juric, "Extreme programming and its development practices," in. *Proc. 22nd Int. Conf. Information Technology Interfaces*, IEEE, Jun. 2000, pp. 97-104
[32] 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*, May 2006, pp. 544-550.
[33] S. A. J. Khalaf, and K. A. Maria, "An Empirical study of XP: the case of Jordan," in *Information and Multimedia Technology, 2009. ICIMT'09. International Conference* pp. 380-383, IEEE.
[34] K. S. Choi and F. P. Deek, "Extreme Programming Too Extreme," *New Jersey Institute of Technology*, 2002.
[35] S. Shahzad, "Learning from experience: The analysis of an extreme programming process.," in *Information Technology: New Generations, 2009. ITNG'09. Sixth International* Conference, pp. 1405-1410, IEEE.

Authors' Profiles

Faiza Anwer is a MS scholar in Computer Science Department of Virtual University of Pakistan. She completed her undergraduate computer science degree in 2010 and currently working as lecturer in Computer Science Department of Govt. College for Women Samanabad, Faisalabad. Her area of research is Software Process Improvement.

Shabib Aftab is currently working as a lecturer in Department of Computer Science of Virtual University of Pakistan. He obtained his MS degree in Computer Science from 'COMSATS Institute of Information Technology, Lahore. Previously he has done M.Sc 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,"SXP: Simplified Extreme Programing Process Model", International Journal of Modern Education and Computer Science(IJMECS), Vol.9, No.6, pp.25-31, 2017.DOI: 10.5815/ijmecs.2017.06.04