Presenting and Evaluating Scaled Extreme Programming Process Model

Muhammad Ibrahim[1], Shabib Aftab[2], Munir Ahmad[3], Ahmed Iqbal[4], Bilal Shoaib Khan[5]
Muhammad Iqbal[6], Baha Najim Salman Ihnaini[7], Nouh Sabri Elmitwally[8]
Department of Computer Science, Virtual University of Pakistan, Lahore, Pakistan[1, 2, 4]
School of Computer Science, National College of Business Administration & Economics, Lahore, Pakistan[2, 3, 6]
Department of Computer Science, Minhaj University Lahore, Lahore, Pakistan[5]
Department of Computer Science, College of Science and Technology, Wenzhou Kean University, China[7]
Department of Computer Science, College of Computer and Information Sciences, Jouf University, KSA[8]
Department of Computer Science, Faculty of Computers and Artificial Intelligence, Cairo University, Egypt[8]

***Abstract*—Extreme programming (XP) is one of the widely used software process model for the development of small scale projects from agile family. XP is widely accepted by software industry due to various features it provides such as: handling frequent changing requirements, customer satisfaction, rapid feedback, iterative structure, team collaboration, and small releases. On the other hand, XP also holds some drawbacks, including: less documentation, less focus on design, and poor architecture. Due to all of these limitations, XP is only suitable for small scale projects and doesn't work well for medium and large scale projects. To resolve this issue many researchers have proposed its customized versions, particularly for medium and large scale projects. The real issue arises when XP is selected for the development of small scale and low risk project but gradually due to requirement change, the scope of the project changes from small scale to medium or large scale project. At that stage its structure and practices which works well for small project cannot handle the extended scope. To resolve this issue, this paper contributes by proposing a scaled version of XP process model called SXP. The proposed model can effectively handle such situation and can be used for small as well as for medium and large scale project with same efficiency. Furthermore, this paper also evaluates the proposed model empirically in order to reflect its effectiveness and efficiency. A small scale client oriented project is developed by using proposed SXP and empirical results are collected. For an effective evaluation, the collected results are compared with a published case study of XP process model. It is reflected by detailed empirical analysis that the proposed SXP performed well as compared to traditional XP.**

Keywords*—Extreme Programming Process Model; XP; modified XP; scaled XP; customized XP; empirical comparison; empirical analysis

I. Introduction

Agile process models replaced the conventional and traditional software development methodologies due to effective features which were not available in conventional models [34]. These features include: emphasis on customer satisfaction, team collaboration and managing changing requirements [20],[45],[50]. Agile models follow an iterative and incremental way of development which delivers a high quality software [2-3], [32],[46]. Agile process models are backed by Agile Manifesto which is considered a parent document of agile family. This document explains the foundations of agile software development in the form of 12 basic principles and practices. These basic principles are about frequent team communication, customer satisfaction, managing frequent changing requirements even at later stages of development and early delivery of partial working software [1],[28],[31],[33],[35],[47],[48]. Many agile process models are used by the software industry now a days such as: Extreme Programming (XP), Scrum, Feature Driven Development (FDD), and Dynamic System Development Method (DSDM) [3],[5],[11],[37]. Extreme programming (XP) is one of popular agile process models for the development of small scale projects as well as widely used by the software industry [5],[41],[42],[43],[51]. XP is a light weight approach for software development, designed and developed by Kent Beck in 2000 [6]. It develops a qualitative software in limited time and lower cost by using some of the best engineering practices, principles and values in a disciplined way. The XP development life cycle (Fig. 1) has six phases: "Exploration phase, Planning phase, Iteration to release phase, Productionizing phase, Maintenance phase and Death phase" [7],[12],[13],[38]. Exploration phase deals with the requirement gathering and it is also responsible for the selection of particular architecture for development. Project planning phase deals with the overall planning, including: number of iterations, no of requirements to be implemented in each iteration, cost and time etc. Iteration to release phase deals with the development of a workable software, this phase may consists of one or more iterations. Productionizing phase deals with the testing of developed module. Maintenance phase deals with the addition of any new functionally (if required) by keeping the old ones intact and finally death phase deals with the completion of software as per client's requirement and ends with the release of software product. All of these phases are backed by twelve best practices of software engineering, including: "planning game, small releases, metaphor, simple design, continuous testing, refactoring, pair programming, collective ownership, continuous integration, 40-hour work per week, on-site customer and particular coding standards [8],[9],[10]. The structure of XP process model along with the umbrella of these 12 practices is best suited for small scale project and also can handle frequently changing requirements very

effectively [4],[39],[40],[44]. However the structure of XP cannot handle medium, complex and large scale projects [13],[36],[49],[52]. To handle this issue, many researchers have introduced improved versions of XP process models. However real issue arises when XP is selected for the development of small scale project and gradually the scope of the project extends to medium or large scale project due to clients requirements. To resolve this issue, this paper presents Scaled Extreme Process (SXP) Model. The proposed model can be used as an effective alternative of XP which can handle small as well as medium and large scale projects. Moreover, in the situation of sudden change in requirements and extension of the scope of small scale project to medium or large scale project, SXP can be effective as well. This research also evaluates the proposed SXP with an empirical case study in which a real time client oriented project is developed and results are compared with another published case study where XP is used for the development of client oriented small project. Comparative analysis reflects the effectiveness of proposed SXP process model.

This paper is further organized in the following sections. Section II highlights and discusses some of the related studies. Section III elaborates the problem definition. Section IV presents the proposed SXP process model. Section V empirically evaluates the proposed SXP. Section VI presents the critical analysis on results. Section VII finally concludes the paper.

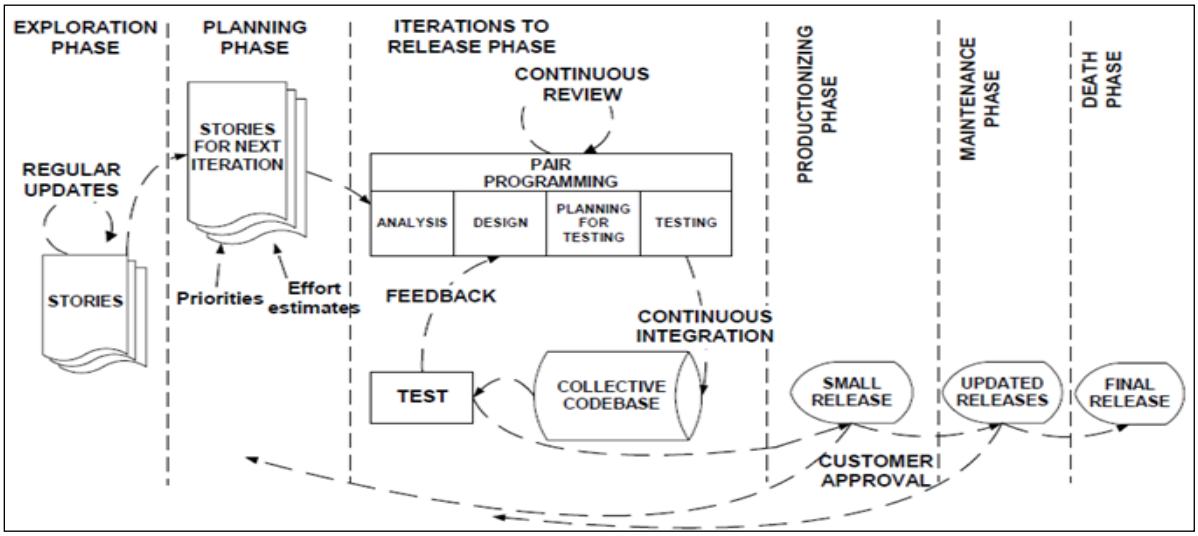

Fig. 1. XP Process Model.

II. Related Work

Many researchers have proposed modified versions of XP process models to reduce its limitations, some of the related studies are discussed here. Researchers in [14], customized the conventional XP process model in order to resolve: design, documentation and quality related issues. The proposed customized model performs the activities regarding non-functional requirements in a separate iteration. The proposed version of XP has some drawbacks including the need of extra staff members to refine each deliverable in each phase. Moreover, this process model is not good for that project which has higher interdependencies among subsystems. This research also lacks the empirical validation of proposed model. In [15], the researchers, proposed an extended iterative maintenance life cycle using XP practices for software maintenance. This approach uses RC (Request for Change) stories and old software as an input and performs all the phases & produce upgraded software. This study is validated through academic projects. However, academic projects are less complex than real-time oriented projects. In [16], the researchers proposed a hybrid model named DXPRUM by combining three agile process models DSDM, XP, and Scrum. The DXPRUM is proposed in order to achieve various features in one framework including: business solution, project management, agile team management, and core engineering practices. The proposed model is validated through the empirical case study in [17]. The DXPRUM process model performed much better as compared to DSDM. Researchers in [18], adopted XP for the development of large scale distributed project and introduced some new practices such as: stand-up meetings, code control, visual indicators, adaptive planning, XP project management, and code gallery. In [19], the authors proposed a new solution for development. It is a combination of seven principles of SOA and XP practices. The proposed solution did not resolve the issue of SOA complexities and did not sustain the agility of XP. In addition, the proposed solution has lacked empirical proof. In [20], the authors used Analytical Hierarchy (AHP) for the CRC cards prioritization process. The APH helps the developer to identify the most significant classes for simple designs. However, the proposed model is not evaluated on real time test cases. In [21], the author studied 40 different teams that use extreme programming for the development of small scale projects. This study provided comprehensive factors and practices which provide positive effects on team performance including: release planning, planning game, on-site customer, small releases, and stand-up meeting. On the other hand the researchers have also highlighted that unit testing, acceptance-testing, test-first design (using TTD), pair programming, and refactoring impacts negatively. In [22], the authors proposed an integrated XP process model. This model has the best engineering practices & management practices of XP, Scrum

and DSDM process model. They suggested a new role named "Technical Writer", who writes effective documentation that enhances understandability and future maintenance. In [23], the authors customized the XP process model for medium and large scale projects. This proposed solution is appropriate for parallel and incremental project development. Extended-extreme programming is Omni-direction in nature and it has five phases including the risk management phase. However, this study did not provide statistical proof regarding large scale project about parallel development. In [24], the author proposed an optimization model that assists in the activity of release planning in XP. The proposed solution essentially supports the development team of XP and the client in the release planning phase. In this model, the release plan is developed based on stories and their relations along with the priorities. However, it consumes a lot of time for data collection which ultimately loses the agility.

In [25], the authors presented a controlled empirical case study of XP and Waterfall methodology. Same project was developed multiple times over five years. The purpose of this research work is to evaluate the efficiency of the XP and Waterfall process models. This research shows unexpected same results of both process models. However, this research has a lack of diversity of data source and data characteristics. In [26], the researchers proposed a hybrid process model named eXRUP for small to medium scale projects by integrating XP and RUP. The proposed solution has been validated through a controlled case study. However, the proposed eXRUP has minimal interaction of programmers with customers and needs higher management. In [27], the authors proposed tailored extreme programming (TXP), a simple version of the XP process model. The author removed unnecessary practices and phases of the XP model to modify it for small teams and small projects with predefined requirements. In [29], the authors identified the need of software process improvement (SPI) in small firms. These small firms face the same SE challenges as large software industries face about SPI. This research develops an SPI structure for small firms by using XP process model. In [30], the researcher introduced a modified XP for medium and large scale projects with large team size. The proposed solution extends the capability of the conventional XP by resolving the design, and documentation related issues. A new phase named "Analysis and risk management" is also introduced to handle failure risks. The new XP model is validated through two case studies on two independent software houses.

III. Problem Definition

XP process model was designed for small teams to develop small scale projects having limited scope. In XP, the collection of good engineering practices and simple SDLC steps help to produce high quality software product within scheduled time but with limited scope (small project). Many researchers have explored the capabilities of XP and customized its practices for various projects types (such as medium and large) and nature (simple and complex) [30]. XP is ideal for small scale project however issue arises when the requirements of client constantly changes with the gradual passage of time which increases the scope of project from small to medium and large scale projects. In such cases, the features of XP like simple design, less documentation and limited testing and absence of proper change management activities can create hurdles to manage the quality as well as delivery of the product within specified time. To handle such issue, this research presents Scaled XP process model which works well for small scale to medium and large scale projects. Proposed SXP can tackle the issue of change management in such cases where project starts with limited scope but gradually extends to medium and large scale projects due to client's frequently changing and increasing requirements.

IV. Proposed Solution

This research proposes a customized XP process model called Scaled Extreme programming (SXP) for small to medium and large scale projects. The proposed solution is equally suitable for small, medium and large scale software projects, unlike the conventional XP process model. The SXP personalized the current practices of XP model to eliminate its limitations without effecting its agility. These limitations are eliminated by adding new phases and practices in SXP. Some effective activities which are included in the proposed model, include: managing the Risk Register, addition of UML artifacts, Effective Testing Mechanism, Formal Refinement Techniques and a formal procedure of Requirement Change Management (RCM). The RCM provides management support to the development team and customers to produce software products in a controlled & monitored environment. The workflow of SXP consists of seven phases as shown in Fig. 2. These phases are named as: Start Phase, Planning & Analysis Phase, Design Phase, Development phase, Acceptance Phase, Refinement Phase, and Release Phase.

A. Start Phase

The first phase of the SXP model is similar to the first phase of XP Model. In this phase, requirements are gathered from the clients by writing user stories. Writing story card is a very effective XP practice. User stories provide a high-level summary of the requirements for the desired system, and these are used as a primary input into estimating and scheduling. However, these user stories do not contain any technical detail of the desired software. In addition, Non Functional Requirements (NFR) are also explored with customer by keeping in view the Functional Requirements.

The extraction of NFR is also vital to the success of the project as these are extracted in order to get rid of undesirable results like unsatisfied client, as well as schedule & budget overruns, etc.

B. Planning and Analysis Phase

This phase consists of very important activities for the initiation of project and initiates with the input of detailed requirements which are further explored to estimate the risk, time, cost, budget and effort. Essential decisions regarding planning are made & documented including: Iteration plan, team size, estimation of cost, budget & effort. Activities of RCM are assigned to a team. Identification of the potential risk, Monitoring the risk and perform any actions required to mitigate the risk are included in the activities of RCM. Risk registers are used to document the complete actions during the process of risk management.

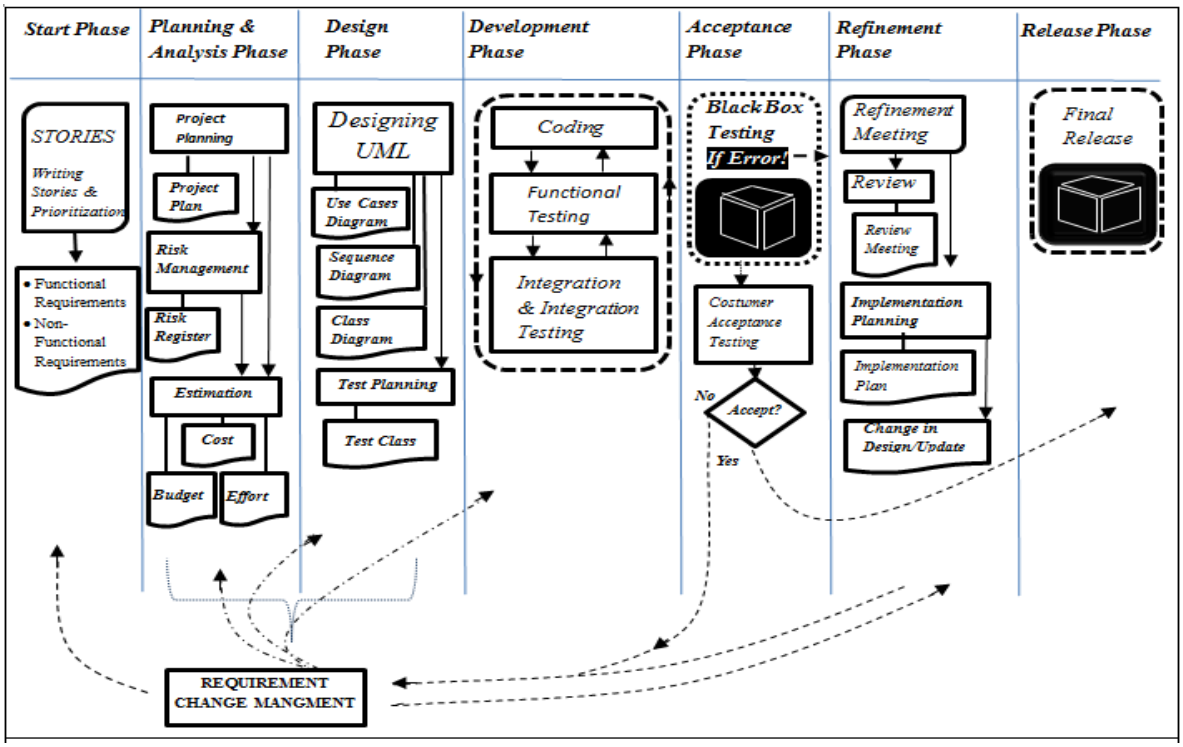

Fig. 2. Scaled Extreme Programming.

C. Design Phase

The design phase of SXP initiates with the development of UML diagrams including: Use Case, Sequence, and Class. These diagrams aim to help the developers to understand the functionalities of system in an easy way. These diagrams also help in change management activity. Moreover, by keeping in view the developed UML diagrams, test planning is completed. During test planning, test classes are written that verifies whether the certain pieces of code & classes are properly working or not.

D. Development Phase

In this phase, designed artifacts are transformed into working software or modules. The UML artifacts and test classes are the input of this phase. The design artifacts help developers to code in object oriented languages. The basic activities performed in this phase include: Coding, Functional Testing, and Integration & Integration Testing. After completion of each component, functional testing is performed to check and analyze the working of that developed component. If the developed component is working fine then it is integrated with previously developed component and then integration testing is performed in order to check whether the integrated components are working fine or not.

E. Acceptance Phase

It is a short phase in which testing is done by the tester in the presence of customer. The tester is a member of the development team who is assigned the task of testing the product externally. Black-Box testing is performed to examine the functionality and features of the system to meet client requirements. If the tester found any error during the test then this phase is aborted and refinement phase is initiated. However, if the tester passes the software then, the product is ready for the acceptance testing which is performed by the customer. This activity is essential and crucial in order to satisfy the customer. Moreover in this phase, the feedback is collected on the software and if the new request or dissatisfaction is reflected by the customer then it will be further handled and catered through RCM.

F. Refinement Phase

The phase initiates if the issues are found in Black-Box testing. The refinement starts with a formal meeting in which a detail review is performed to check the stories, developed artifacts, test classes, and codes in order to identify the issues. At the end of review meeting, identified issues are documented and resolved through an implementation plan. RCM takes necessary action against the refinement decision. In addition, some important documents are also updated by RCM like, Risk register, Change request register and design.

G. Release Phase

This is the last phase, in this stage software is ready to release or deliver to the client. The team moves to this stage when all user stories are implemented, and the customer is satisfied with the software. In addition, training, and documentation are provided to the client after deployment.

H. Role and Responsibilities of RCM

Requirement Change Management (Fig. 3) is a supporting framework of SXP. It is introduced to cater the change requests properly without dropping the team productivity. The RCM can consist of one or more team members.

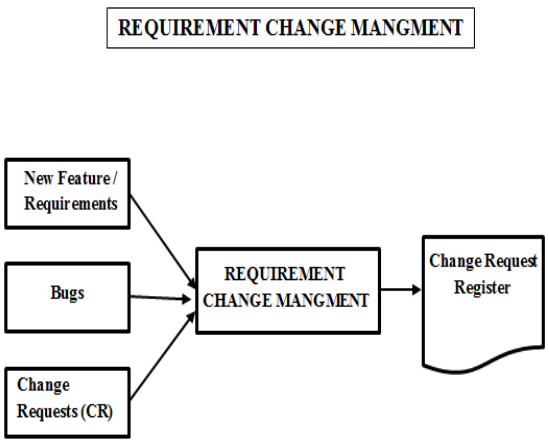

Fig. 3. Requirement Change Management.

Some key duties of RCM are as follows.

- It directly interacts with the customer and the team.
- It collects requirements from the customers in case of change request or requirement for new feature.
- It provides support to the customer and the development team in the entire project cycle.
- It records in change request register if 1) client wants new features 2) clients want to change existed features 2) client is not satisfied by acceptance testing.
- If errors are found in the acceptance phase by tester then it records the changes in change request register.

V. Empirical Evaluation of Proposed SXP

This research proposed a modified version of XP process model named SXP in order to resolve the issues of conventional XP. To analyze the performance of proposed model, an empirical case study is conducted in which a real time client oriented project is developed by using the proposed SXP. The case study was conducted in a software house situated in Karachi, capital of Sindh Province, Pakistan. All of the team members, who participated in the case study have higher degrees in computer science discipline (BS or MS) along with software development experience of at least one year. During the development, data of various software metrics are collected for empirical analysis. For effective evaluation, the results of SXP case study are compared with another case study presented in [26], where conventional XP process model was used to develop a small scale project. Characteristics of both the case studies are reflected in Table I.

XP case study was conducted by the students of computer science programs of BS and MS level with no or little knowledge of agile development. However a training session of 10 days was organized before the development. The reason of choosing the XP case study for comparison with the case study of proposed SXP is to critically analyze the performance of SXP with empirical data, so that any gap or deficiency can further be identified for further improvement. This study compares the proposed SXP with XP process by providing detailed empirical results of all of the iterations as per the guidelines provided by [10]. Both of the case studies (SXP and XP) are empirically compared in Table II. First column of the table shows the serial no whereas second column reflects the particular metrics for which the data was collected during the development for comparison. These metrics include: development time, cost, productivity etc. All of these measures are considered as an effective way to analyze the quality of any process model. The columns (Release 1 to Release 4) shows release wise measures/values of the attributes of column 2 and finally the last columns reflect the aggregated/average values of all the releases.

TABLE I. Selected Case Studies

Characteristics	**SXP**	**XP**
Product Type	Social Media platform	Real Estate Management
Size	Small	Small
Iterations	4	3
Programming Approach	Object Oriented	-
Language	JavaScript	PHP
Documentation	Ms Office & JS Doc	MS Office
Testing	Desktop & Mobile browser testing	-
Project Complexity Type	Average	Average
Team Size	4 Members	3 Members
Development Environment	Visual Studio, Ionic SDK & SQL	Macromedia Dream Viewer and Net Beans
Other Tools	MS Visio	MS Visio

TABLE II. EVALUATION OF XP AND SXP PROCESS MODEL

No	Parameters	Release 1		Release 2		Release 3		Release 4	Total	
		XP	SXP	XP	SXP	XP	SXP	SXP	XP	SXP
1	Completion time (Week)	2	1	1	1	1	1	1	4	4
2	Number of Modules	2	8	1	5	1	7	3	4	23
3	No of User Stories	17	1	13	2	11	2	2	41	7
4	Budget Effort in (h)	240	128	120	128	120	128	128	480	512
5	Actual Effort in (h)	210	120	90	120	90	96	120	390	456
6	No. of user Interface	2	2	1	3	1	2	2	4	9
7	No of class designed	46	10	34	8	30	4	4	110	26
8	Total Line of Code	4500	5300	3200	3900	3300	2900	3200	11000	14700
9	KLOC	4.5	5.3	3.2	3.9	3.3	2.9	3.2	11	14.7
10	No of integration	20	14	12	12	12	18	6	44	50
11	Post Release Defects	2	3	2	1	4	2	1	8	7
12	Post Release Defects per KLOC	0.44	0.56	0.62	0.25	1.212	0.68	0.31	0.727	0.47
13	Productivity= LOC / Actual effort	21.4	44.1	35.6	32.5	36.7	30.2	26.6	28.2	32.2
14	No of prerelease change requests	3	2	2	3	2	1	3	7	9
15	Total change requests per KLOC	0.66	0.37	0.62	0.76	0.60	0.34	0.93	0.636	0.61
16	Time to implement changes in hour	4	2	3	2	1	1	5	8	10

VI. CRITICAL ANALYSIS

Some significant differences are reflected in the performance of both models (XP and SXP) in Table II. Complexity level of both the projects is different as reflected by: KLOC, No of code integration, No of modules, and No of interfaces. “KLOC” and “Actual Effort” both are considered as important software metrics to analyze the performance of software process models. KLOC developed during SXP case study and XP case study is reflected in Fig. 4. Actual Effort (h) in both the case studies is reflect in Fig. 5. Release wise Actual Effort in both the case studies is also shown in Fig. 6. In XP project, 11 KLOC were produced with 390 hours of actual effort. However during the development of SXP project, 14.7 KLOC were produced with 456 hours of actual effort. It can be seen that the proposed SXP model slightly performed better in these metrics as 3.7 more KLOC were developed with 66 more hours of actual effort as compared to XP. However, it should also be noted that in XP project, there were 3 team members whereas in SXP, there are 4 team members.

During XP case study, 41 requirements were implemented however in SXP case only 7 requirements are implemented (Fig. 7). It should also be noted here that no of modules designed and developed during the implementation of 41 requirements of XP case study were only 4 as compared to 23 in SXP case study. Moreover, 4 interfaces were developed in XP case study as compared to 9 in SXP. Another important metric which should be discussed along with “No of implemented requirements” is the “No of code integrations”. There were 44 integrations in XP as compared to 50 in SXP. So it can be analyzed that only the no of requirements implemented in a project cannot reflect the performance of a model as client can write only one requirement which might have many modules, interfaces and backend integrations. The performance of SXP is better in all of these metrics as the team of SXP done more work as compared to the team of XP.

The “No of designed classes” is also an important software metric to analyze the performance of teams especially when this metric is analyzed along with developed KLOC. In XP case study, 110 classes were designed as compared to 26 in SXP (Fig. 8). As the development approach in SXP was object oriented so the very less no of classes with higher no of KLOC is justified. Object oriented principles used in SXP case study is also one of the reasons of good performance as it increases the re usability of code with an effective and efficient way.

The defects which are discovered by the client after the release is an important quality parameter which also reflects the customer satisfaction. The software application developed with XP reflected 8 defects as compared to 7 in SXP case study (Fig. 9). This metric raises the questions on the quality assurance activity and particularly the testing strategy of software process model. In SXP, efforts are made to produce the quality software even it performed better than XP (reflected from the implemented case study) but 7 defects after the release are not acceptable. However, this issue can be raised if the testing mechanism of the model is not implemented properly by the team.

Software productivity is a crucial metric to analyze the performance of any software process model. It reflects the effort of whole development team during the project. It shows the amount of effort, the team has put to complete the project within defined time. However in order to analyze the effectiveness and efficiency of the model, this single parameter is not enough, instead all of the software metrics

shown in Table II collectively reflect the performance of model. The team of XP reflected the productivity 28.2 as compared to 32.2 in SXP (Fig. 10, Fig. 11). The SXP model produced more lines of code in less time. If the Productivity is analyzed by keeping in view the complete list of parameters in each of the given iterations then it can be said that the proposed SXP performed well.

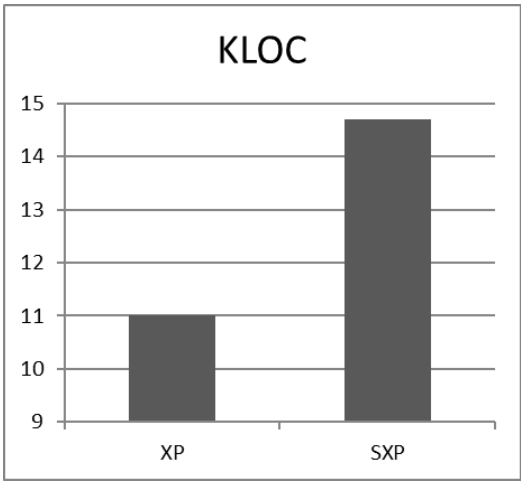

Fig. 4. KLOC.

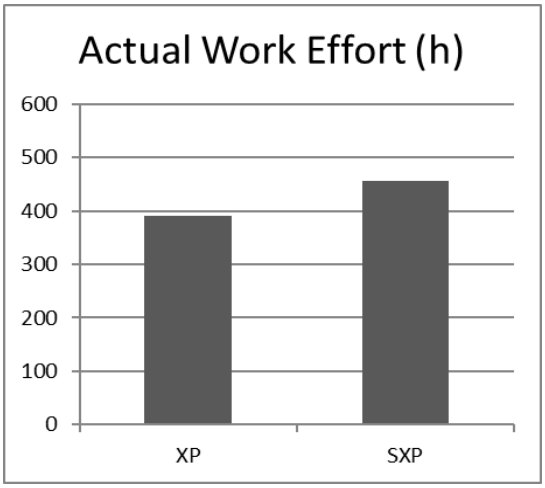

Fig. 5. Actual Work Effort.

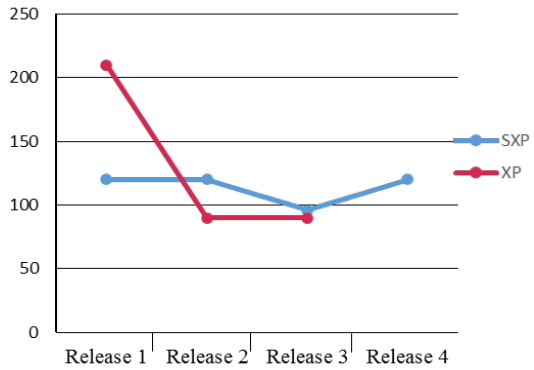

Fig. 6. Releasee Wise Actual Work Effort.

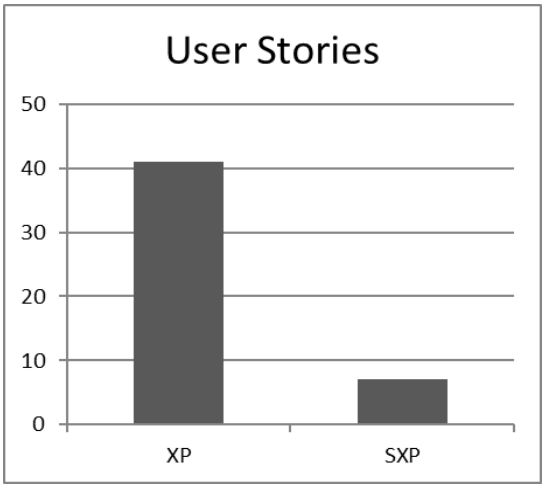

Fig. 7. User Stories.

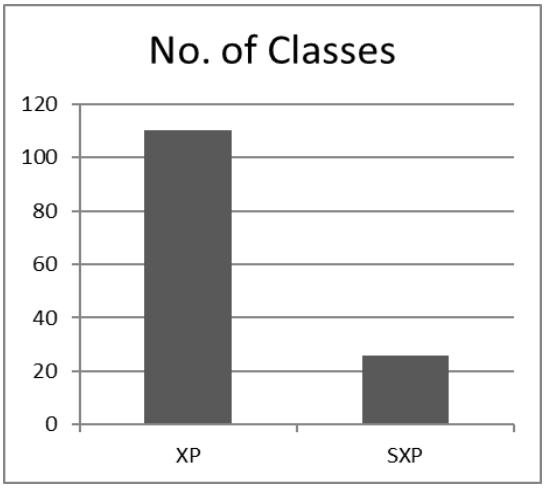

Fig. 8. No of Classes.

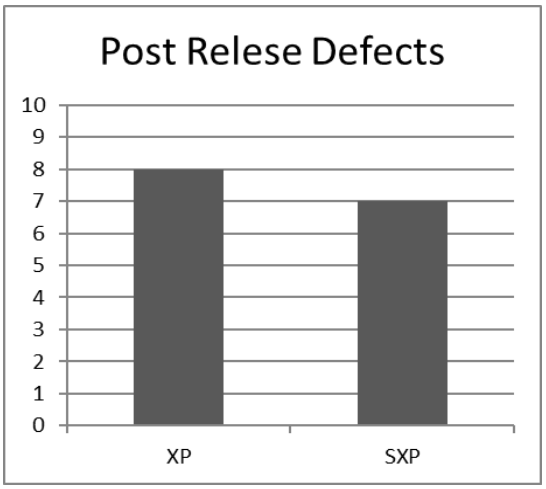

Fig. 9. Post Release Defects.

The projects implemented in both the case studies were same in nature (web based) but different in complexity level. The project implemented with SXP was complex as compared to XP project. Moreover development language, project size and no of team members were also different. Results of all the

quality parameters reflect the effectiveness of proposed SXP model however 7 defects after the release of complete software product raises the question on quality assurance activities. There might be other reasons of these defects including the human error of testing personnel or lack of quality test cases etc.

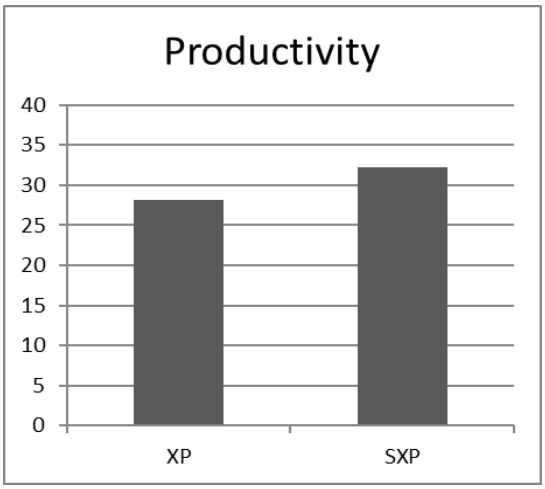

Fig. 10. Productivity.

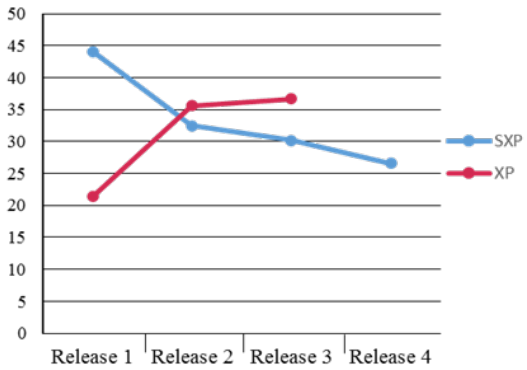

Fig. 11. Release wise Productivity.

VII. Conclusion

XP is considered as one of the widely used agile process model by the software industry for the development of small scale projects. Its practices include: quick response to changing requirements, customer satisfaction, rapid feedback, team collaboration, and small releases. However, besides the featured practices, XP has a major drawback as well and that is: its ability to handle only small projects. To resolve this issue many researchers have proposed its customized versions specifically to handle medium and large scale projects. However real problem arises when the conventional XP process is selected for a small scale and low risk project but with the gradual passage of time, the frequently changing requirements due to modern business change drags the scope of project from small scale to medium or even large scale. At this stage, some characteristics of conventional XP don't let its life cycle to handle medium or large projects. The characteristics include: poor architectural structure, lack of documentation, less focus on design and absence of proper change management procedure. This research has proposed a scaled version of XP process model which can handle such situations very effectively. Moreover, the proposed model can be equally effective for small, medium and large scale projects. In the proposed model, more focus is given on designing, testing and particularly on change management procedure. Due to these features, SXP can handle any extension in the scope of the project. An empirical evaluation is also performed in order to analyze the effectiveness of proposed SXP. For this purpose, a case study is conducted in which a real time client oriented project is developed. Empirical results of software quality metrics are collected during the development and then compared with another published case study in which XP was used for the development of a client oriented project. A detailed empirical analysis is performed and it is observed that the proposed SXP performed well almost in every important quality parameter. However to further evaluate the proposed model, medium or large complex project should be chosen for development.

References

[1] S. Ashraf and S. Aftab, "Latest Transformations in Scrum: A State of the Art Review," Int. J. Mod. Educ. Comput. Sci., vol. 9, no. 7, 2017.

[2] L. Williams, "Agile software development methodologies and practices," in Advances in computers, vol. 80, pp. 1–44, 2010.

[3] C. R. Kavitha and S. M. Thomas, "Requirement gathering for small projects using agile methods," IJCA Spec. Issue Comput. Sci. Dimens. Perspect. NCCSE, 2011.

[4] J. Newkirk, "Introduction to agile processes and extreme programming," in Proceedings of the 24th International Conference on Software Engineering. ICSE 2002, 2002, pp. 695–696.

[5] A. Begel and N. Nagappan, "Usage and perceptions of agile software development in an industrial context: An exploratory study," in First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007), pp. 255–264, 2007.

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

[7] F. Anwer, S. Aftab, S. S. M. Shah, and U. Waheed, "Comparative Analysis of Two Popular Agile Process Models: Extreme Programming and Scrum," Int. J. Comput. Sci. Telecommun., vol. 8, no. 2, pp. 1–7, 2017.

[8] M. N. Swamy, L. M. Rao, and M. P. KS, "Component Based Software Architecture Refinement and Refactoring Method into Extreme Programming," architecture, vol. 5, no. 12, 2016.

[9] S. Ashraf and S. Aftab, "Scrum with the Spices of Agile Family: A Systematic Mapping," Mod. Educ. Comput. Sci., vol. 9, no. 11, 2017.

[10] S. Aftab, Z. Nawaz, F. Anwer, M. Ahmad, A. Iqbal, A. A. Jan, and M. S. Bashir, "Using FDD for small project: An empirical case study," Int. J. Adv. Comput. Sci. Appl., vol. 10, no. 3, pp. 151–158, 2019.

[11] J. A. J. Builes, D. L. R. Bedoya, and J. W. B. Bedoya, "Metodología de desarrollo de software para plataformas educativas robóticas usando ROS-XP," Rev. Politécnica, vol. 15, no. 30, pp. 55–69, 2019.

[12] 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," Int. J. Nat. Eng. Sci., vol. 8, no. 3, pp. 35–42, 2014.

[13] F. Anwer and S. Aftab, "SXP: Simplified Extreme Programing Process Model," Int. J. Mod. Educ. Comput. Sci., vol. 9, no. 6, p. 25, 2017.

[14] M. R. J. Qureshi and J. S. Ikram, "Proposal of Enhanced Extreme Programming Model," Int. J. Inf. Eng. Electron. Bus., vol. 7, no. 1, p. 37, 2015.

[15] J. Choudhari and U. Suman, "Extended iterative maintenance life cycle using eXtreme programming," ACM SIGSOFT Softw. Eng. Notes, vol. 39, no. 1, pp. 1–12, 2014.

[16] M. Fahad, S. Qadri, S. S. Muhammad, and M. Husnain, "Software Quality Assurance of Medium Scale Projects by using DXPRUM Methodology," Int. J. Nat. Eng. Sci., vol. 8, no. 1, pp. 42–48, 2014.

[17] M. Fahad, S. Qadri, S. Ullah, M. Husnain, R. Qaiser, S. Ahmed, W. A. Qureshi, and S. S. Muhammad, "A Comparative Analysis of DXPRUM and DSDM," IJCSNS, vol. 17, no. 5, p. 259, 2017.

[18] E. Abdullah and E.-T. B. Abdelsatir, "Extreme programming applied in a large-scale distributed system," in 2013 International Conference On Computing, Electrical And Electronic Engineering (Icceee), 2013, pp. 442–446.

[19] F. Carvalho and L. G. Azevedo, "Service agile development using XP," in 2013 IEEE Seventh International Symposium on Service-Oriented System Engineering, pp. 254–259, 2013.

[20] S. Alshehri and L. Benedicenti, "Prioritizing CRC cards as a simple design tool in extreme programming," in 2013 26th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), 2013, pp. 1–4.

[21] S. Wood, G. Michaelides, and C. Thomson, "Successful extreme programming: Fidelity to the methodology or good teamworking?," Inf. Softw. Technol., vol. 55, no. 4, pp. 660–672, 2013.

[22] S. Sultana, Y. H. Motla, S. Asghar, M. Jamal, and R. Azad, "A hybrid model by integrating agile practices for pakistani software industry," in 2014 International Conference on Electronics, Communications and Computers (CONIELECOMP), 2014, pp. 256–262.

[23] M. R. J. Qureshi, "Agile software development methodology for medium and large projects," IET Softw., vol. 6, no. 4, pp. 358–363, 2012.

[24] G. van Valkenhoef, T. Tervonen, B. de Brock, and D. Postmus, "Quantitative release planning in extreme programming," Inf. Softw. Technol., vol. 53, no. 11, pp. 1227–1235, 2011.

[25] F. Ji and T. Sedano, "Comparing extreme programming and Waterfall project results," in 2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T), 2011, pp. 482–486.

[26] G. Rasool, A.Shabib, H.Shafiq, and S.Detlef, "eXRUP: A Hybrid Software Development Model for Small to Medium Scale Projects.", Journal of Software Engineering and Applications, vol. 6, pp. 446-457, 2013.

[27] F. Anwer, S. Aftab, and I. Ali, "Proposal of Tailored Extreme Programming Model for Small Projects," Int. J. Comput. Appl., vol. 171, no. 7, pp. 23–27, 2017.

[28] S. Alam, S. Nazir, S. Asim, and D. Amr, "Impact and Challenges of Requirement Engineering in Agile Methodologies: A Systematic Review," Int. J. Adv. Comput. Sci. Appl., vol. 8, no. 4, pp. 411–420, 2017.

[29] M. Y. Al-Tarawneh, M. S. Abdullah, and A. B. M. Ali, "A proposed methodology for establishing software process development improvement for small software development firms," Procedia Comput. Sci., vol. 3, pp. 893–897, 2011.

[30] M. R. J. Qureshi, "Estimation of the New Agile XP Process Model for Medium-Scale Projects Using Industrial Case Studies," Int. J. Mach. Learn. Comput., vol. 3, no. 5, pp. 393–395, 2013.

[31] S. Ashraf and S. Aftab, "Pragmatic Evaluation of IScrum & Scrum," Int. J. Mod. Educ. Comput. Sci., vol. 10, no. 1, pp. 24–35, 2018.

[32] F. Anwer, S. Aftab, U. Waheed, and S. S. Muhammad, "Agile software development models tdd, fdd, dsdm, and crystal methods: A survey." Int. J. Multidiscip. Sci. Eng., vol. 8, no. 2, pp. 1–10, 2017.

[33] S. Aftab, Z. Nawaz, M. Anwar, F. Anwer, M. S. Bashir, and M. Ahmad, "Comparative Analysis of FDD and SFDD," Int. J. Comput. Sci. Netw. Secur., vol. 18, no. 1, pp. 63–70, 2018.

[34] S. Ashraf, "IScrum: An Improved Scrum Process Model," Int. J. Mod. Educ. Comput. Sci., vol. 9, no. 8, pp. 16–24, 2017.

[35] Z. Nawaz, S. Aftab, and F. Anwer, "Simplified FDD Process Model," Int. J. Mod. Educ. Comput. Sci., vol. 9, no. 9, pp. 53–59, 2017.

[36] S. Aftab, Z. Nawaz, F. Anwer, M. S. Bashir, M. Ahmad, and M. Anwar, "Empirical evaluation of modified agile models," Int. J. Adv. Comput. Sci. Appl., vol. 9, no. 6, pp. 284–290, 2018.

[37] F. Anwer and S. Aftab, "Latest Customizations of XP: A Systematic Literature Review," Int. J. Mod. Educ. Comput. Sci., vol. 9, no. 12, pp. 26–37, 2017.

[38] F. Anwer, S. Aftab, M. S. Bashir, Z. Nawaz, M. Anwar, and M. Ahmad, "Empirical Comparison of XP & SXP," Int. J. Comput. Sci. Netw. Secur., vol. 18, no. 3, pp. 161–167, 2018.

[39] M. R. J. Qureshi, "Empirical Evaluation of the Proposed eXSCRUM Model-Results of a Case Study," Int. J. Comput. Sci. Issues, vol. 8, no. 3, pp. 150–157, 2011.

[40] Z. Mushtaq and M. R. J. Qureshi, "Novel Hybrid Model: Integrating Scrum and XP," Int. J. Inf. Technol. Comput. Sci., vol. 4, no. 6, pp. 39–44, 2012.

[41] M. R. J. Qureshi, "An Evaluation of the Improved XP Software Development Process Model," Strategy, vol. 20, no. 2, pp. 79–82, 2008.

[42] S. Kazi, M. S. Bashir, M. M. Iqbal, Y. Saleem, M. R. J. Qureshi, and S. R. Bashir, "Requirement change management in agile offshore development (RCMAOD)," Sci. Int., vol. 26, no. 1, pp. 131–138, 2014.

[43] M. R. Jameel Qureshi, "Evaluating the Quality of Proposed Agile XScrum Model," Int. J. Mod. Educ. Comput. Sci., vol. 9, no. 11, pp. 41–48, 2017.

[44] A. I. Khan, M. R. J. Qureshi, and U. A. Khan, "A Comprehensive Study of Commonly Practiced Heavy & Light Weight Software Methodologies," no. February, 2012.

[45] M. R. J. Qureshi and S. A. Hussain, "An Improved XP Software Development Process Model," SCIENCE INTERNATIONAL-LAHORE, vol. 20, no. 1, 2012.

[46] M. R. Jameel Qureshi and M. Kashif, "Adaptive Framework to Manage Multiple Teams Using Agile Methodologies," Int. J. Mod. Educ. Comput. Sci., vol. 9, no. 1, pp. 52–59, 2017.

[47] R. J. Qureshi, M. O. Alassafi, and H. M. Shahzad, "Lean Agile Integration for the Development of Large Size Projects," Int. J. Mod. Educ. Comput. Sci., vol. 11, no. 5, pp. 24–33, 2019.

[48] R. J. Q. M and Z. Abass, "Long Term Learning of Agile Teams," Int. J. Softw. Eng. Appl., vol. 8, no. 6, pp. 01–18, 2017.

[49] M. R. J. Qureshi and A. Barnawi, "Kinect Based Electronic Assisting System to Facilitate People with Disabilities Using KXPRUM Agile Model," Life Sci. J., vol. 11, no. 10, pp. 56–62, 2014.

[50] M. Rizwan Jameel Qureshi, "Comparison of Agile Process Models to Conclude The Effectiveness for Industrial Software projects," Sci. Int., vol. 28(5), no. November-December, pp. 5119–5123, 2016.

[51] M. R. J. Qureshi and S. A. Hussain, "A reusable software component-based development process model," Adv. Eng. Softw., vol. 39, no. 2, pp. 88–94, 2008.

[52] S. U. Nisa and M. R. J. Qureshi, "Empirical Estimation of Hybrid Model: A Controlled Case Study," Int. J. Inf. Technol. Comput. Sci., vol. 4, no. 8, pp. 43–50, 2012.