Empirical Evaluation of Modified Agile Models

Shabib Aftab, Zahid Nawaz, Faiza Anwer, Muhammad Salman Bashir, Munir Ahmad, Madiha Anwar
Department of Computer Science
Virtual University of Pakistan
Lahore, Pakistan

***Abstract*—Empirical evaluation is one of the widely accepted validation method in the domain of software engineering which investigates the proposed technique via practical experience and reflects its benefits and limitations. Due to various advantages, agile models have been taking over the conventional software development methodologies since last two decades. However besides the benefits, various limitations have been noticed as well by the researchers and software industry in agile family. To achieve the maximum benefits it is vital to fix the limitations by customizing the development structure of agile models. This paper deals with the empirical analysis of modified agile models called Simplified Extreme Programing (SXP) and Simplified Feature Driven Development (SFDD), which are the modified forms of Extreme Programing (XP) and Feature Driven Development (FDD). SXP was presented to eliminate the issues of conventional XP such as, lack of documentation, poor architectural structure and less focus on design. SFDD was proposed to take care of reported issues in FDD such as explicit dependency on experienced staff, little or no guidance for requirement gathering, rigid nature to accommodate requirement changes and heavy development structure. This study evaluates SXP and SFDD through implementing client oriented projects and discusses the results with empirical analysis.**

Keywords—Agile models; SXP; SFDD; Modified XP; modified FDD; empirical evaluation; comparative analysis

I. INTRODUCTION

Conventional software process models are replaced by lightweight agile development methodologies. The reason behind the widely acceptance of agile family by the software industry is the features these models provide such as: light weight approach for development, early delivery of partially working software (module), welcome changes at any stage of development and quick response. Agile models shifted the focus from process to people and valued those factors which were neglected by traditional models [7], [25], [26]. Some of the famous agile models are: Extreme Programming (XP), Scrum, Test Driven Development (TDD), Dynamic System Development Model (DSDM), Crystal methods and Feature Driven Development (FDD), etc. [7], [8]. These models follow the values, principles and practices given by agile manifesto which is considered a parent document of all agile models and contains twelve foundation principles of software development. XP and FDD, both are the widely used agile models in software industry [12], [40]. XP was developed by Kent Beck and mainly focuses to overcome the limitations of traditional software process models. The working of XP consists of certain principles, values and practices, which work together rigorously to develop high quality software [9], [29], [34], [35], [39]. XP provides a flexible and adaptive development approach which can handle the changing business needs in an effective way due to its well-known requirements gathering technique, "story cards". Its 12 practices provide the guidelines to govern the whole development process in an effective and efficient way. Besides the advantages, XP reflects some limitations as well. Drawbacks of XP include poor architecture, weak system design and lack of documentation [29], [32], [36], [37]. Moreover its practices: ‘pair programming’ and ‘on-site customer’ are controversial and cannot be applicable in every situation [38], [39]. Due to these drawbacks, XP is suitable only for small scale and low risk projects. On the other hand FDD follows the process oriented approach [9]-[11]. It is highly adaptive and mainly focuses on design and building aspects of development. As its name reflects, features are the basic building blocks of this model. Feature is considered as a functionality which user wants in the software. Benefits provided by FDD model includes the iterative and incremental approach along with ETVX pattern which ensures the development of high quality software according to client valued features. However along with advantages, some limitations of FDD were also reported such as: little or no guidance for requirement gathering, explicit dependency on experience staff, rigid nature to handle changing requirements and heavy development structure including various activities and team roles. All these issues make it only suitable for medium or large scale projects. SXP [40] and SFDD [12] were proposed to overcome the limitations of XP and FDD respectively. This study empirically valuates the proposed models through empirical case studies conducted in software industry.

II. RELATED WORK

Drawbacks of agile models have to be eliminated in order to achieve the maximum benefits, for this purpose many researchers have proposed the modifications in agile models. XP and FDD were discussed and optimized in many studies from which some of are discussed here. In [13], researchers presented the Tailored Extreme Programming (TXP) model which was specifically designed for small scale projects where requirements have fewer or no tendencies to change. In [14], researchers proposed the feature of reusability in XP model. They introduced a framework to add the ability of component based architecture refinement reusability in traditional XP. The used framework provided a way to develop simple and loosely coupled design which can be modified easily in future. Researchers in [15] customized the XP by introducing parallel refinement iteration to the development activities in order to enhance the quality; however the proposed model is not suitable for software projects having a lot of inter dependencies among modules. In [16], authors customized the software maintenance model by using many XP practices such as: on-site customer, planning game, small releases, pair programming, metaphor, test driven development and refactoring. In [18], researchers integrated Personal Software Process (PSP) with XP. The proposed model introduced "Personal Planning Phase" in which developer can plan the activities by using PSP practices. Six important practices from each model (XP and PSP) are integrated in proposed model. In [19], XP was customized to develop medium scale projects with large team by eliminating its drawbacks such as weak design and lack of documentation. Moreover a phase named "Analysis and Risk Management" was introduced to handle the failure risks. In [30], Analytical Hierarchy Process (AHP) was used with CRC cards during designing phase of XP. AHP was used to design a systematic approach of CRC cards prioritization. AHP is a hierarchal model consists of five steps which reflect the human thinking process. By using AHP the developers can select, design and implement the most important classes first. In [31], XP was customized for medium to large scale projects. The research highlighted the drawbacks of classical XP such as weak design, poor architecture, lack of risk management and lack of documentation. These issues of XP make it suitable only for small scale projects. To eliminate these issues, new phases were introduced in modified proposed model. Author in [1] proposed Feature Driven Reuse Development (FDRD), an enhanced version of FDD which considered re-useable feature-sets for development along with the new requirements. Author in [2] presented Competitor Driven Development (CDD), a hybrid process model which integrated the practices of Extreme Programming (XP) and Feature Driven Requirement Reuse Development (FDRD). The proposed model is a self-realizing requirement generation model which keeps track of market trends as well as competitor's next product launch to extract requirements. Moreover CDD considers the market orientation of product to guess the product's success rate. In [3], authors proposed a hybrid model SCR-FDD, an integration of Scrum and FDD. The proposed model covered the imitations of both models by taking the schedule related aspects from Scrum and quality related aspects from FDD. In [4], researchers presented Feature-Driven Methodology Development (FDMD), a modified version of FDD which integrated the features of object oriented approach with Situational Method Engineering (SME). In the proposed model requirements are represented as features, which are based on object oriented principles. The feature is defined by using action, result and object. Authors in [5] proposed Secure Feature Driven Development (SFDD), an enhanced version of FDD which introduced some changes in classical FDD to cover security related issues. The proposed model introduced two phases in classical FDD named "Build security by feature" and "Test security by feature" along with the "In-phase Security" element in each phase. Moreover, a new role is also added called security master to ensure the secure software development. Authors in [6] proposed an ontology based approach in FDD for semantic web application. The proposed model used the concepts of domain ontology from domain knowledge modeling. Ambiguity and inconsistency regarding Language is handled by RDF and OWL however the agility of FDD can be compromised by adding the concepts of domain ontology in each phase.

III. Modified Agile Models

The proposed Simplified Extreme Programming (SXP) is focused to overcome the limitations of classical XP. It provides more flexible and simple approach for small to medium scale projects. The issues of pair programming and on-site customer are handled in an effective way. On the other hand, SFDD [12] was proposed to overcome the limitations of FDD such as explicit dependency on experienced staff, little or no guidance for requirement gathering, rigid nature to accommodate changes in requirements, heavy development structure. SFDD focused on small to medium scale projects along with an effective requirement elicitation technique of story cards which simplified the requirement change process. Both the proposed models are briefly explained below.

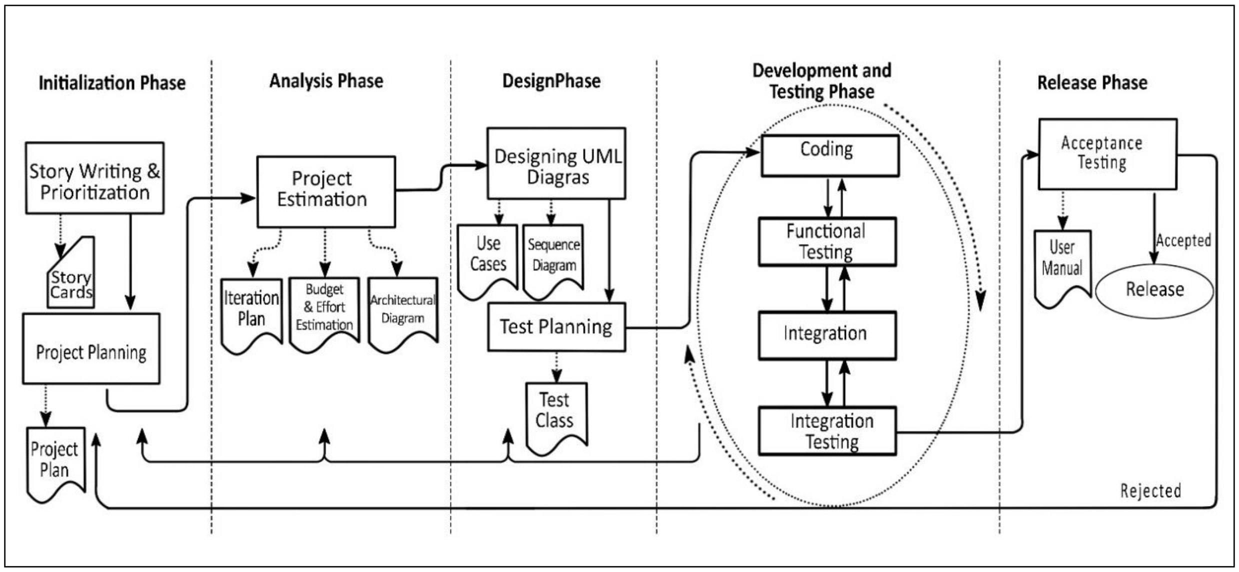

Fig. 1. SXP.

A. SXP

Simplified Extreme Programing (SXP) consists of five phases; Initialization, Analysis, Design, Development & Testing and Release as shown in Fig. 1. In the proposed model, customer involvement is restricted to initialization and release phase only and all other phases are executed by development team with the complete coordination. Necessary documentation is produced during each phase that helps to resolve change management issues. "Initialization" is the first phase of SXP and is responsible to extract and manage the requirement as well as to create an overall plan for project. Requirements are extracted and managed through story cards, a story card consists of following features: functionality name, type, priority and the short description without any technical detail. Type defines whether the functionality is functional and nonfunctional and priority is assigned with number so that higher priority features can be developed in early iterations. Project planning includes the decisions regarding project scope, cost and tools to be used for the development. "Analysis" is the second phase and deals with budget and schedule related activities which are performed by development team only. In this phase required budget is estimated and documented. An iteration plan is also formed which includes the detail about number of iterations, number of stories implemented in each iteration and the time of each iteration. A training session is also conducted to make the development team familiar with the tools and technology (if the team members are not already familiar). "Design Phase" is third phase of SXP which deals with two activities: “Designing UML Diagrams” and “Test Planning”. Conventional XP does not include any documentation which makes requirement change management very difficult. This issue is effectively solved by SXP by focusing on system design with use case diagrams and sequence diagrams. Test cases are also developed in this phase. Writing tests prior to code help the development team to understand different design opportunities. "Development and Testing" is the fourth phase and works in an iteratively. Activities of this phase include coding, functional testing, integration and integration testing. Developer writes the code for selected stories by keeping in view the design document which was developed during design phase. Functional testing is performed by using test cases, developed during test planning activity. Coding activity is repeated if any issue is reported in functional testing. These tests are performed by programmers and results are noted to keep the track of defects. Code is integrated with previous developed module in case of successful functional testing followed by another testing known as integration testing. "Release" is the last phase in which customer performed acceptance testing. The developed workable product is released after the customer’s approval along with the User manual. If the customer is not satisfied with the developed product then whole development process can be repeated again with changed or modified set of requirements.

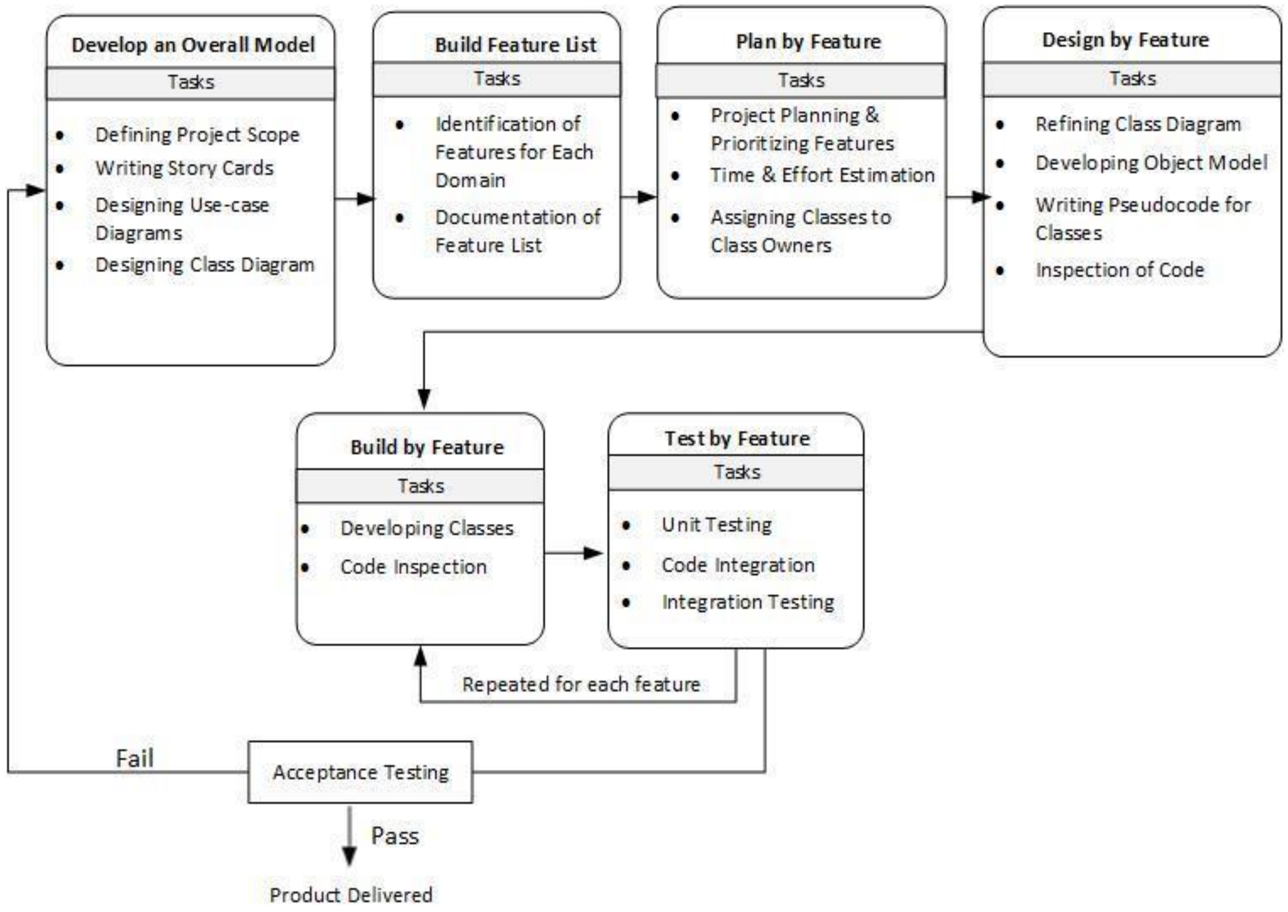

Fig. 2. SFDD.

B. SFDD

Simplified Feature Driven Development (SFDD) consists of six phases and various activities as shown in Fig. 2. "Develop an Overall Model" is the first phase which deals with the identification of requirements and scope of project. Domain expert and chief programmer are the main roles of this phase. Domain expert provides the project requirements through story cards and chief programmer finalizes the project scope by keeping in view the provided requirements moreover use case diagrams and class diagrams are also developed in this phase. "Build Feature List" is the second phase of SFDD and deals with the extractions of features from the documents developed in first phase. Features are basically the functions which a customer wants in the software. Related features are collected in a list called feature list. Chief programmer converts the requirements in to feature lists in this phase. "Plan by Feature" is the third phase which deals with the project planning activities and starts with a meeting where domain expert and chief programmer finalize the budget and time frame of the project. Chief programmer further finalizes the number of iterations and assigns features to iterations by keeping in view the priorities. This phase also includes the estimation of effort (resource persons) and hardware/software resources which are needed for the project. At the end of the phase classes are assigned to class owners (developers). "Design by Feature" is the fourth phase and deals with the process of refining the class diagrams developed in the first phase. Object model is finalized in this phase and class owner completes the pseudo code for the assigned classes. To ensure the quality, a role of QA manager is introduced in this phase. "Build by Feature" is the fifth phase of model and first phase of iteration. Development actually starts in this phase according to the pseudo code, written in previous phase. QA manager makes sure that the developing module is according to the features. Test by feature is the last phase of model and second phase of iteration which deals with the testing activities and starts with unit testing to make sure that the developed module is bug free and working properly, if passed then integrated with already developed module.

Integration testing is then performed to check the integrated working of modules. Finally domain expert performs the acceptance testing. Proposed model simplified the structure of FDD through effective customization.

IV. Empirical Evaluation

This research aims to perform the empirical evaluation of proposed modified agile models. For this purpose two case studies are conducted in which both models, SXP and SFDD were used to develop small scale web based projects. The selected case studies were part of an empirical research project in which multiple agile models were used to develop various client oriented applications in a software house, situated in Islamabad, capital of Pakistan. The software house consists of experienced staff with dominating knowledge of software development along with higher degrees in computer science disciplines. The developers were using agile methods for most of the projects. Both case studies were implemented in same working environment but with different teams. Most of the characteristics of applications are same such as size of project, no of iterations, no of team members, and the tools used in development. The detail regarding the characteristics of developed projects is given in Table I. The case study of SXP is implemented by the team which had significance experience of agile development. On the other hand, to implement the SFDD, the chosen team had less or no experience of agile development however training session of 10 days was organized.

For SFDD, less experienced team was selected as the authors of proposed model (SFDD) claimed that the issue in classical FDD regarding the dependency on experienced staff has been eliminated. The detailed empirical results collected during the development are shown in Table II. Partial and aggregated results of selected case studies are discussed in [39], [33]. However this paper demonstrates the complete results of empirical experiment including all the iterations by keeping in view the guidelines extracted from [17], [27], [28], [19]. Both case studies are implemented with four iterations. After each iteration, partial working software (module) was released for the client.

TABLE I. Case Studies Detail

Characteristics	**SXP**	**SFDD**
Product Type	Human Resource Management	Human Resource Management
Size	Small	Small
Iterations	4	4
Programming Approach	Object Oriented	Object Oriented
Language	C#, ASP.NET	C#, ASP.NET
Documentation	MS Office	MS Office
Testing	Browser Stack	Browser Stack
Web Server	IIS	IIS
Project Type	Average	Average
Team Size	5 Member	5 Member
Feedback	Weekly	Weekly
Development Environment	Visual Studio 2012	Visual Studio 2012
Other Tools	MS Visio	MS Visio
Reports	Crystal Report	Crystal Report

TABLE II. EMPIRICAL RESULTS

Sr. No	Software Metric	Release 1		Release 2		Release 3		Release 4		Total	
		SXP	SFDD	SXP	SFDD	SXP	SFDD	SXP	SFDD	SXP	SFDD
1	Completion Time (weeks)	1	0.9	0.9	0.8	0.9	0.8	1	0.7	3.8	3.2
2	Number of Modules	2	1	1	1	1	1	2	1	6	4
3	No of User Stories	8	21	4	20	3	15	6	9	21	65
4	Budgeted Work Effort (h)	200	180	180	160	180	160	200	140	760	640
5	Actual Work Effort (h)	180	180	165	147	175	140	175	125	695	592
6	Number of User Interfaces	6	3	3	3	3	2	2	2	14	10
7	No of Classes	4	7	3	5	2	4	2	4	11	20
8	Lines of Code	820	4300	734	3450	860	2760	646	2600	3060	13110
9	KLOC	0.820	4.3	0.734	3.4	0.860	2.7	0.646	2.6	3.060	13.1
11	No of Code Integrations	10	7	8	5	12	3	7	3	37	18
12	Post Release Defects	2	2	4	1	6	1	3	1	15	5
13	Post Release defects / KLOC	2.4	0.465	5.45	0.294	6.97	0.37	4.64	0.38	4.902	0.381
14	Productivity (= line of code/ actual time spent in hours)	4.56	23.88	4.44	23.46	4.91	19.71	3.69	20.80	4.4	22.14
16	No of Pre-release Change Requests	2	6	3	3	4	1	1	2	10	12
17	Total Change requests/KLOC	2.44	1.395	4.09	0.882	4.65	0.370	1.55	0.769	3.27	0.916
18	Time to Implement Changes (h)	3	4	2	3	4	3	2	1	11	11

The second column of Table II represents the attributes/metrics which are measured in each release for both the models and the last column contains the cumulative/average values of metrics from all four releases. The remaining columns (release 1 to release 4) present the values of metrics (column 2) in each release for SXP and SFDD. Metrics are used to measure the software in terms of development, cost, working, productivity, quality, effectiveness and efficiency from various aspects [20]-[24].

V. CRITICAL ANALYSIS

From the detailed empirical results (Table II), significant differences can be seen among the performances of both the models. Even though the working environment as well as the size and nature of both the applications were same, but SFDD performed much better than SXP. KLOC of the application developed using SXP are 3.069 with the implementation of 21 user stories however on the other hand SFDD implemented 65 user stories with 13.1 KLOC (Fig. 3 and 4).

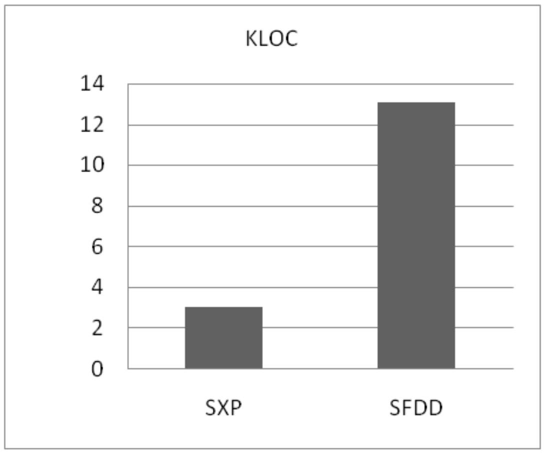

Fig. 3. KLOC.

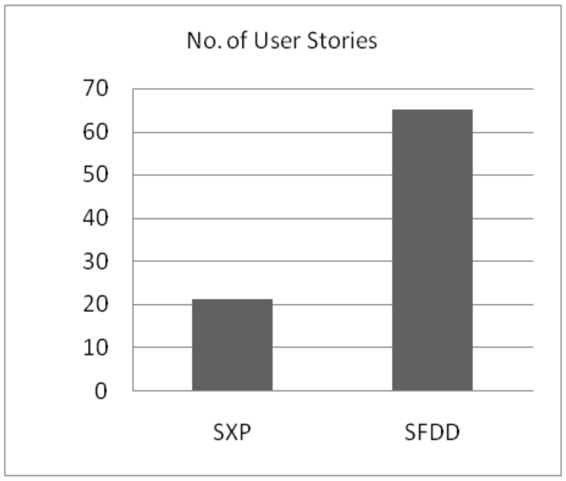

Fig. 4. Implemented user stories.

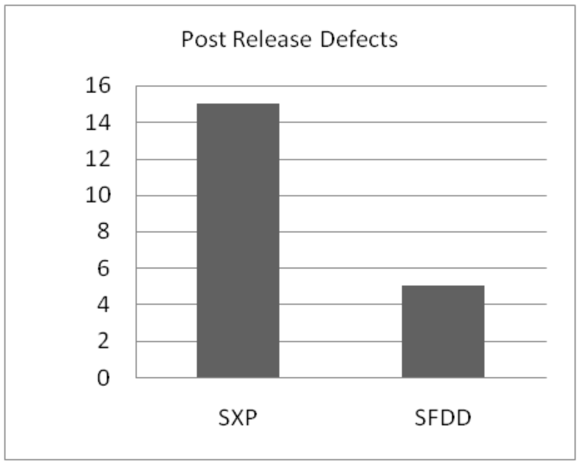

Fig. 5. Post release defects.

No. of post release defects is an important software metric which reflects the quality of developed application as well as the satisfaction of customer. After the release 15 defects were reported in the application developed using SXP however only 5 defects were reported in the application developed using SFDD (Fig. 5).

Time to implement pre-release change requests is also considered as one of the important quality metric which reflects the change management feature of software process model. 10 pre-release changes were proposed during SXP case study which took 11 hours to implement however no. of pre-release change requests in SFDD case study were 12 which took the same time for the implementation (Fig. 6) as in SXP (11 hours).

Software productivity reflects the team effort during the application development. Productivity of the application developed by SXP was far lower than the application of SFDD (Fig. 7). During SXP case study, 3060 lines of code were written in 695 hours (Actual Work Effort) with the productivity of 4.4 however during the implementation of SFDD, 13110 lines were written in 592 hours and reflected productivity of 22.14. As compared to SFDD, SXP showed very poor performance by keeping in view the empirical results. SFDD performed very well according to all software parameters (Table II) even with the team having less experience with agile methodologies. There might be various reasons of poor performance in SXP case study. Complexity level of the developed application in SXP case study may be higher than the application of SFDD however according to best of our knowledge the nature and complexity level of both the application were same. As the performance of SXP is lower in every release so, there might be issues in code integrations as there were total of 37 integrations in SXP and only 18 in SFDD. Moreover the issue of communication among the team members can also be a reason of lower performance. The issue of awareness with agile development cannot be considered as the team of SXP was experienced with agile and team of SFDD had less or no experience with agile development.

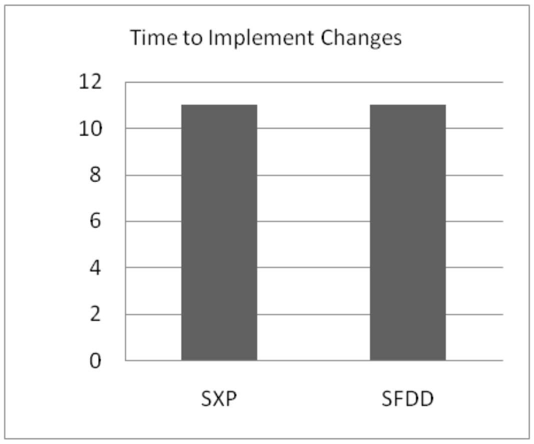

Fig. 6. Time to implement pre-release change requests.

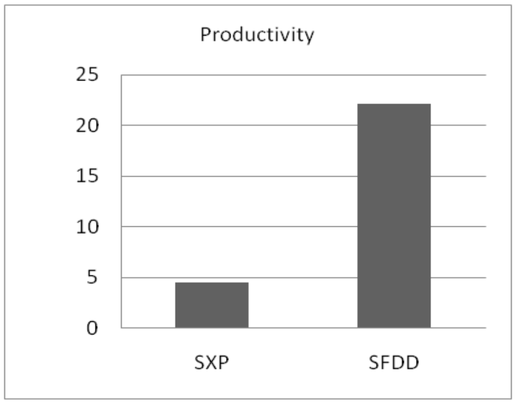

Fig. 7. Productivity.

VI. Conclusion and Future Work

This paper evaluated the proposed modified agile models, SXP and SFDD through empirical case studies. SXP focused to reduce the reported issues of conventional XP such as: Lack of documentation, poor architectural structure and less focus on design. Due to these issues, XP is only suitable for small scale and low risk projects. SFDD has taken care of the issues reported in FDD, such as explicit dependency on experienced staff, no guidance for requirement gathering, rigid nature to accommodate requirement changes and heavy development structure. Empirical analysis was performed via development of client oriented projects by using SXP and SFDD. Both projects were related to Human Resource Management (HRM) and also were same in nature as well as in size and complexity level. The development team for SXP case study was experienced in agile development however for SFDD case study the chosen team had less experience of agile as the proposed SFDD eliminated the dependency on experienced staff. According to empirical results, SFDD performed much better than SXP even with the less experienced team. In comparison of SFDD, SXP performance was very poor in each metric such as lines of code, implemented user stories, post release defects, productivity and time required to implement pre-release change requests. There might be various reasons of poor performance of SXP model such as complexity level, integration issues and communication problems within the development team. It is suggested that both the models should be further tested with large and complex projects.

REFERENCES

[1] S. Thakur and H. Singh, “FDRD: Feature driven reuse development process model,” in Proceedings of 2014 IEEE International Conference on Advanced Communication, Control and Computing Technologies, ICACCCT 2014, 2015, pp. 1593–1598.

[2] V. P. Doshi and V. Patil, “Competitor driven development: Hybrid of extreme programming and feature driven reuse development,” 1st Int. Conf. Emerg. Trends Eng. Technol. Sci. ICETETS 2016 - Proc., no. Cdd, p. 7602985, 2016.

[3] S. Ali, S. S. Tirumala, and A. Babu G, “A Hybrid Agile model using SCRUM and Feature Driven Development,” Int. J. Comput. Appl., vol. 156, no. 5, pp. 1–5, 2016.

[4] R. Mahdavi-Hezave and R. Ramsin, “FDMD: Feature-Driven Methodology Development,” Proc. 10th Int. Conf. Eval. Nov. Approaches to Softw. Eng., pp. 229–237, 2015.

[5] A. Firdaus, I. Ghani, and S. R. Jeong, “Secure Feature Driven Development (SFDD) Model for Secure Software Development,” Procedia - Soc. Behav. Sci., vol. 129, pp. 546–553, 2014.

[6] F. Siddiqui and Alam, M. Afshar, “Ontology Based Feature Driven Development Life Cycle,” Int. J. Comput. Sci. Issues, vol. 9, no. 1, 2010.

[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, 2017.

[8] G. Rasool, S. Aftab, S. Hussain, and D. Streitferdt, “eXRUP: A Hybrid Software Development Model for Small to Medium Scale Projects,” J. Softw. Eng. Appl., vol. 6, no. 9, pp. 446–457, 2013.

[9] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, “Agile Software Development Methods: Review and Analysis,” 2017.

[10] S. R. Palmer and M. Felsing, A Practical Guide to Feature Driven Development. 2002.

[11] D. Ph, “Major Seminar On Feature Driven Development Agile Techniques for Project Management Software Engineering By Sadhna Goyal Guide : Jennifer Schiller Chair of Applied Software Engineering,” p. 4, 2007.

[12] 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.

[13] 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.

[14] N. Swamy, L. M. Rao, and K. S. Praveen, “Component Based Software Architecture Refinement and Refactoring Method into Extreme Programming,” vol. 5, no. 12, pp. 398–401, 2016.

[15] M. R. Jameel Qureshi and J. S. Ikram, “Proposal of Enhanced Extreme Programming Model,” Int. J. Inf. Eng. Electron. Bus., vol. 7, no. 1, pp. 37–42, 2015.

[16] 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.

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

[18] N. Iqbal, M. ul Hassan, A. Rehman Osman, and M. Ahmad, “A framework for partial implementation of PSP in Extreme programming,” Int. J. Eng. Res. Appl. www.ijera.com, vol. 3, no. 2, pp. 604–607, 2013.

[19] 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.

[20] N. E. Fenton, and S. L. Pfleeger, "Software Metrics: A Rigorous and Practical Approach: Brooks," 1998.

[21] S. H. Kan, Metrics and models in software quality engineering. Addison-Wesley Longman Publishing Co., Inc. 2002.

[22] C. Jones, "Applied Software Measurement", McGraw Hill, 1991.

[23] N. Fenton and J. Bieman, “Software Metrics: Roadmap,” It Prof., vol. 2, pp. 38–42, 2014.

[24] S. Ashraf and S. Aftab, “Scrum with the Spices of Agile Family: A Systematic Mapping,” Int. J. Mod. Educ. Comput. Sci., vol. 9, no. 11, pp. 58–72, 2017.

[25] 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, pp. 12–22, 2017.

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

[27] S. U. Nisa and M. R. J. Qureshi, “Empirical Estimation of Hybrid Model: A Controlled Case Study,” Int. J. Inf. Technol. Comput. Sci., vol. 1, no. July, p. 8, 2012.

[28] M. 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, 2012.

[29] 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. April, pp. 1–10, 2017.

[30] S. Alshehri and L. Benedicenti, “Prioritizing CRC cards as a simple design tool in extreme programming,” Can. Conf. Electr. Comput. Eng., pp. 13–16, 2013.

[31] M. R. J. Qureshi, “Agile software development methodology for medium and large projects,” IET Softw., vol. 6, no. 4, p. 358, 2012.

[32] 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.

[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 (IJCSNS)., vol. 18, no. 1, pp. 63–70, 2018.

[34] E. Mnkandla and B. Dwolatzky, “A Survey of Agile Development Methodologies,” no. December 2004, pp. 209–227, 2007.

[35] I. Journal et al., “Extreme Programming : Newly Acclaimed Agile System,” vol. 3, no. 2, pp. 699–705, 2010.

[36] R. Crocker, “The 5 reasons XP can’t scale and what to do about them,” Proc. 2nd Int’l. Conf. Extrem. Program. Agil. Process. Softw. Eng., pp. 62–65, 2001.

[37] A. Dalalah, “Extreme Programming: Strengths and Weaknesses,” î Comput. Technol. Appl., vol. 5, no. 1, 2014.

[38] 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,” Proc. - Agil. 2007, pp. 37–48, 2007.

[39] 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 (IJCSNS)., vol. 18, no. 3, pp. 161–167, 2018.

[40] F. Anwer and S. Aftab, “SXP: Simplified Extreme Programing Process Model,” Int. J. Mod. Educ. Comput. Sci., vol. 9, no. 6, pp. 25–31, 2017.