ISSN 2047-3338

Comparative Analysis of Two Popular Agile Process Models: Extreme Programming and Scrum

Faiza Anwer[1], Shabib Aftab[2], Syed Shah Muhammad Shah[3] and Usman Waheed[4]
[1-4]Department of Computer Science, Virtual University of Pakistan
[1]faiza.anwer28@gmail.com, [2]shabib.aftab@gmail.com

***Abstract*— Since last two decades, agile software development methodologies have been one of the most debating topics for researchers. These are called light weight development methods because of informal, adaptive and flexible approach. These models are based on the collection of best practices which help to handle problems related to changing requirements, customer satisfaction, and product quality. A number of agile models are available to meet the needs of different projects. However Extreme Programming and Scrum are two most familiar and commonly used models. This study makes a valuable contribution by exploring these models in detail. In this paper a detailed comparison of Extreme programming and Scrum is conducted to find their similarities, differences and explores those features which complement each other.**

***Index Terms*— Extreme Programming, Scrum, Agile Models and Comparison**

I. INTRODUCTION

AGILE software development methodologies provide an iterative and evolutionary development paradigm with more emphasis on changing requirements, customer satisfaction, and team collaboration [1]. These methodologies emerged in 2001 in response to limitations of plan driven methodologies [1]. High rate of failed, cancelled and delayed projects forced software practitioners to reconcile the development principles and practices. Agile models are actually collection of best practices and principles of software engineering. These principles may not be new for software industry but in agile modeling these are used with different approach that makes them more flexible and adaptive during development. These agile principles can accommodate rapid software development needs.

Due to their simplicity, flexibility, and suitability to present needs of software development, agile models are getting popularity from last few decades. Many agile models like Extreme programming (XP), Scrum, Feature driven development (FDD), Dynamic system development method (DSDM), Kanban, Lean software development (LSD), Adaptive software development (ASD) are available.

Extreme Programming (XP) and Scrum are most widely used agile models especially for small scale projects. These are called light weight development methodologies because of excluding formal activities from development process for the sake of simplicity and agility. Both of these models have some common and contrasting features. This study is conducted to explore and compare them in detail. This comparison provides a deep insight about these two methodologies that will greatly helpful for developers and researchers.

Rest of the paper is organized in following sections; Section II and III explain extreme programming and scrum in detail respectively. Section IV provides a detailed comparison of these two methodologies. Section V presents critical analysis and section VI finally concludes this paper.

II. EXTREME PROGRAMMING

Extreme programming (XP) is an agile software development methodology developed by Kent Beck in 1996 while working on a C3 payroll project. Later in 1999, Kent Beck published his book "Extreme Programming Explained" to present a refined form of XP. It is a lightweight, more flexible and low risk disciplined approach of software development with ability to manage vague or rapidly changing requirements [2]. It is considered more suitable for small and medium sized teams [3]. XP is a collection of values, principles and practices that are applied in a disciplined way [4]. It is called "Extreme Programming", because of the fact that it took those practices to extreme which were considered helpful in developing high quality software [5]. XP accentuate greatly on customer satisfaction. Rapid feedback and frequent releases help in managing the defects near to its origin. Lower defect rate reduce the cost of development and result in a more acceptable final product at lower cost.

A. XP Phases

Whole development process consists of six phases: Exploration phase, Planning phase, Iteration to release phase, Productionizing phase, Maintenance phase and Death phase Fig. 1.

Exploration Phase: Exploration phase is first phase of XP life cycle which deals with requirement and architecture modeling of the system. In this phase, user requirements, architecture, tools and technology are defined. A meeting among customer, users and developers is arranged to plan release. Customer writes user stories on stories cards that provide requirement about software. These user story cards comprises of short name, priority of story and one or two text paragraph without technical detail [5]. User story should be detailed enough that help the developers to understand system requirement and also in making estimates. Time estimation means time required to implement a story. If a story require longer implementation time that story can be converted in to small stories by customer. For architecture modeling metaphors are created during architectural spike to consider different alternative solutions. Metaphor is not a complete architecture but a framework with basic objects and their interfaces. Exploration phase can last from few weeks to few months. However in [5] Kent Beck suggests that at the end of exploration phase enough material should be available from user stories that can provide a good start for first product release and developer should have confidence about cost and time estimation of tasks to be implemented.

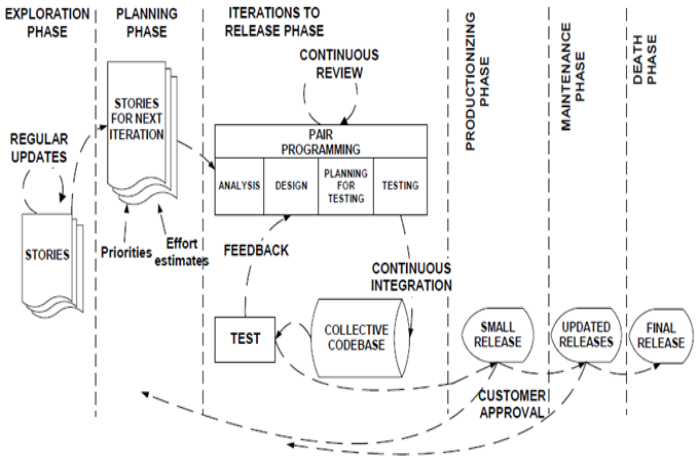

Fig. 1: Life Cycle of Extreme Programming [6]

Planning Phase: After exploration phase, planning phase starts that aimed to find the answers of two questions basically; what can be built within due date that have some business value And what is the plan to do for next iteration? If exploration phase was gone well then planning phase only demand a day or two to complete [5]. During planning phase, task are drawn from user stories and written on task cards.

In XP, planning process is called planning game that further performed in two parts: Release planning and Iteration planning. During planning phase decision about team size, code ownership, schedule, working hours are taken [7].

Release Planning: Basic objective of release planning is to find out the features to be needed in the system and delivery schedule of these features. Both customer and developers participate in release planning meeting. Release planning consists of three phases: exploration phase, commitment phase and steering phase [5]. Customer writes story cards to identify the required features of system. These features are then sorted according to their importance and a smaller set of story cards for the recent release is selected. This is an iterative process that can be adjusted by adding, removing, merging or splitting some stories.

Iteration Planning: Each iteration starts with iteration planning. In this phase, developers prepare a plan of their activities to implement required features of the current release. Like release planning, iteration planning also has exploration, commitment and steering phases but customer is not involved in this step [5], [8]. During iteration planning programmer select tasks to implement and estimates required cost, time and effort for selected task. Tasks can be given to other programmers to balance the workload.

Iteration to Release Phase: This phase incorporate the basic development activities like designing, coding, testing and integration [9]. This is an iterative phase in which each iteration can span over one to four weeks. Each iteration starts with iteration planning. In first iteration, such stories are selected that make overall architecture of the system [5]. Tasks selected for current iteration are actually implemented by a pair of programmers. Programmers select tasks, make a simple design and code it. After coding, functional testing is performed and then code is integrated. Code refactoring is used if developed code does not fulfill requirements. Final development may take several iterations in which coding, testing, listening and designing is performed repeatedly. Standup meeting are used to discuss development progress or any issues that need to be resolved [5]. After final iteration code is ready for production.

Productionizing Phase: Being an iterative and incremental process, XP delivers software in small releases. A release is a small part of planned software that implements some business needs. Frequent releases in XP allow to build required system in increments. A release cycle can consists of a number of iteration that can span from 1 to 4 weeks [10]. Productionizing phase is about deployment of the software in small releases. To check, whether the software is ready for production, acceptance testing, system testing and load testing is performed. During this phase, programmers slow down the rate at which system evolves. As the risk become more important whether a change should go to next release or not [8].

Maintenance Phase: Maintenance is a natural phenomenon for software systems. In XP, software continues to evolve over a period of time. In this phase new functionality is built while keeping the old one running [5]. New architectural design and technologies can be introduced however XP team has to do more care as the system is in production also. The changes that cause production problems are stopped immediately [6].

Death Phase: This is the last phase of XP. There are two possible situations in which a software system reaches to death phase. In first case, if the developed software has all the needed functionality and customer is satisfied and has no more stories, then it is time to finally release the system. A small

document of five to ten pages is created, about the system for future use. In other case, customer may require a set of features that cannot be developed economically. In such situation, it will be better to close the software development which is called entropic death of system [5].

B. XP Practices

There are twelve XP practices that distinguish XP from other software process models. These practices are used during software development under the guidance values and principles of XP.

Planning Game: System requirements are collected on story cards that are used for further planning. Different team roles, team size, working hours and overall schedule is defined during planning game [11]. Planning game is performed in two parts called release planning and iteration planning.

Small Releases: In each release a set of requirements are developed that have some business and development value [5]. Small releases make the system open and available for evaluation by the customer. Small releases help in getting immediate customer's feedback about system.

Metaphor: It is the architectural design of the system that describes how system should works. For developers, It is very important way to understand the system [11].

Simple Design: Simple design is a great practice of XP that helps to design basic required functionality of the system and avoids unnecessary details. It focuses on currently needed features not on future requirements.

Continuous Testing: Continuous testing provide quick feedback. XP uses unit testing and acceptance testing continuously.

Refactoring: Refactoring is restructuring the system without changing its behavior [11]. It is performed to improve the quality and flexibility of design. It is a routine activity of XP developers to make the code quality better.

Pair Programming: It is very interesting feature of XP that distinguish it from other development approaches. In XP, coding is performed by the two programmers at same machine [5], [28]. The idea behind pair programming is to develop high quality software at lower cost. As most of the errors are captured and corrected within seconds by the companion programmer.

Collective Ownership: Any programmer can access any part of code any time to improve it. This is called collective ownership of code. Code review by number of programmers; enhance the quality of software to be developed.

Continuous Integration: After completing every task, system is integrated and tested. It may happen many times a day. This reduces integration problems and improves software quality.

40-Hour Week: XP discourages extra-long working hours for developers [6]. Tired and bored programmers make more mistakes that's why unnecessary overtimes are avoided in XP. It is a rule of XP, to work 40 hours a week not more than this.

On-Site Customer: A customer's representative is a part of XP team and remains on site all the time [6]. He/ she is usually a domain expert that can decide about system's desired features, answer the questions and can steer the development process. On-site presence help to reduce communication gap between developers and customer. A quick feedback remains available to developers about desired software.

Coding Standards: Coding standards are followed in XP. Code is owned collectively and can be accessed or changed by any programmer. To share the code among programmers, it is necessary to follow some common coding standards [5].

C. XP Values

There are five XP values which are focused while XP practices are applied. These values are simplicity, communication, feedback, courage and respect [5], [10].

Simplicity: XP keeps things simple like simple plan, simple design and simple code. It prefer on designing simple solution of the problem. No extra functionality is added until customer asked for [5]. Simple and small iteration of XP helps to avoid the risk of project distraction.

Communication: Instead of documentation, XP uses active and continuous communication among team members. All the team member and customer present on site and communicate continuously to find more suitable and economical solution of the problem.

Feedback: XP uses feedback that span on different time scale from second to months. Unit testing and integration testing is performed on daily bases, provide quick feedback about system. Feedback and communication help to keep project on the right way. On site presence of customer is a distinguishing feature of XP that helps to get rapid feedback about the developing software.

Courage: XP practices require courage. Sometime it is needed to refactor the code or design that was completed after great effort. It also means that making such decision that never been made before for the system.

Respect is another important value of XP introduced in [12]. Self-respect and respect for other members is equally important that make it possible to implement XP practices (like pair programming, collective code ownership). Showing respect towards work can force the developers to do high quality work.

D. XP Roles

XP define seven roles of team members with their qualities and responsibilities that they must have to perform in team [5], [6].

Programmer: This is most important role in XP team. Coding is main activity in XP which is performed by programmer. There is no analyst, designer or architect in XP team, all these tasks should be performed by programmer.

Customer: Customer is another very important member of XP team who plays an active role throughout the development process. He writes stories, derive functional test and verify these test.

Coach: Coach is a person that should have both managerial and technical skills. Good communication and decision power help the coach to keep the team members together and on right track.

Tracker: Duty of tracker is to gather metrics like load factor and functional test scores about the project. Tracker collect data from each developer after two or three days and record how much time is spent on a task and how much is still required to complete it. It is tracker's responsibility to check that iteration and commitment schedule are realistic and can be meet [5].

Tester: The responsibility of tester is to guide and help customers to write functional tests and verify them. As in XP, unit testing is performed by the programmers so tester has a very little to do.

Consultant: XP team has no specialist but in some cases team needs technical guidance from an expert, in that case a consultant can be hired for a time being. Two or more developers discuss with consultant in a meeting to learn about solution of the problem.

Big Boss: He is a coordinator of the project that has responsibilities of team building, providing necessary resources, equipment and tools. Big boss has to show courage while supporting team's decision that is never experienced before.

III. SCRUM

Scrum is most widely used agile software development model. It provides an iterative and incremental framework for software development that is based on best practices used in Japanese industry. In 1995 Jeff Sutherland and Ken Schwaber introduced the Scrum methodology that was later presented by Ken Schwaber and Mike Beedle in a Book named "Agile Software Development with Scrum" in 2001. The idea behind scrum was to handle drawbacks of traditional development methodologies. In scrum each product release is planned according to customer requirements, time pressure, competition, product quality and available resources.

Scrum is an empirical approach that is based on process control theory to add flexibility, adoptability and productivity in the development process [13], [28]. The strength of scrum lies in three points transparency, inspection and adaptation. Transparency means that every aspect of process that affect the result should be visible to all members involved in product development. Inspection means keep eye on process to detect any unacceptable deviation. Finally adaptation helps in adjusting the process in case of any unacceptable deviation [13], [14]. Scrum provides an iterative and incremental framework of development that builds software product in small cycles called Sprints. Sprints have one month or less duration. Scrum framework consists of three roles, four ceremonies and three artifacts Fig. 2.

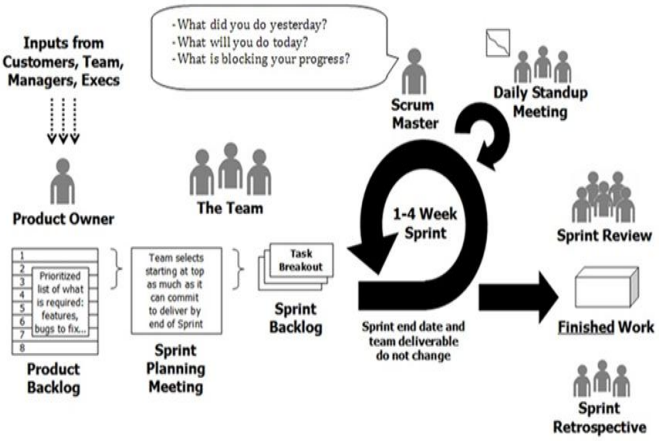

Fig. 2: Scrum Framework [14]

A. Scrum Phases

Scrum activities can be grouped in three phases called Pregame, Game and Postgame [15], [16].

Pregame: This phase starts by defining the vision of the project which may be unclear initially but can be refined further in later sprints. Product owner is responsible to define the vision and prepared a prioritized list of functional and nonfunctional requirements for the software. This prioritized list of required features is called Product Backlog [13]. A plan that includes the time and cost estimation is also prepared in this phase with final product delivery date and number of releases in which final product will be delivered. A high level architectural design is developed that tells how to implement different tasks, defined in product backlog. Some other important task completed in this phase includes risk assessment, definition of development team, validation of development tools and verification of approval and funds.

Game: This is actual development phase of the scrum which is performed in small iterations called sprints. Sprint is a time boxed development period ranges from one week to four weeks based on complexity and risk involved. Each sprint incorporates activities like develop, wrap, review and adjust [15].

Postgame: This is a closure phase. After implementing the desired features during development phase, final release occurs. A release is declared closed when all the goals defined during pregame phase are met. In closure phase final integration testing is performed, user manuals and training materials are prepared for the final release.

B. Sprint Cycle

Scrum works in sprints that are a time boxed duration in which team actually develop the software according to product backlog [15], [17]. During sprint, scrum team works on product backlog under the guidance of scrum master to develop functional software which will be delivered at the end of sprint. Following activities are performed during sprint.

Sprint Planning: Each sprint starts from sprint planning that is completed in two phases. In part one, product owner and scrum master review the product backlog tasks that are most important. They decide the objectives and context of the high priority tasks which help the team members to understands the product needs clearly. Part one of this meeting mainly focuses on the aspect of product.

In second part of meeting focus is shifted on how to build task till the end of sprint. Team reviews the probability of task completion irrespective to product owner decision. Then team commit to complete the work in decided time period. Scrum teams are self-organizing that divide the tasks and responsibilities according to their interest.

Daily Scrum: Scrum team member daily conduct a 10 to 15 minutes meeting called daily scrum. This meeting helps the team members to know about the project progress. Team members can find the cause of any speed interruption and take corrective action accordingly. In this meeting every member tries to answer the following three questions [14].

What did I do yesterday to achieve the sprint goal?
What will I do today to achieve the sprint goal?
Is there any hindrance in doing what I planned to do?
First two questions help to understand the project progress and last question helps to find the solution of problem that is causing delay in the project progress.

Sprint Development: During this phase activities like design, development and testing is carried out for each tasks in product backlog. These tasks are implemented according to their priority defined by product owner.

Sprint Review: At the end of each sprint, a review of the developed product is conducted. This is inspected and adopt phase of product. In this review meeting product owner judge whether the development is going according to needs. Detailed conversation among product owner, scrum master and team members help to get feedback about product which may change the development directions [13].

Sprint Retrospective: Sprint retrospective is inspect and adopt phase for the process. During this phase, scrum master and team members discuss what is working and what is not working in the process. This helps in deciding what practices should be carried out in next sprint and what should be changed in next sprint. This meeting greatly helps in improving the process.

C. Scrum Roles

There are three roles in scrum called product owner, scrum master and team [14], [17].

Product Owner: Product owner is a customer's representative who has overall responsibility of product. He creates and prioritizes the list of required features to be developed in the form of product backlog. He can reposition the item in product backlog according to changing business needs. He decides the project schedule and is responsible of providing finance accordingly. He negotiates with scrum team to convey the interests of all stakeholders. Product owner is a person accountable for the profit or loss of the product. A scrum team can have only one product owner. To fulfill his duties, product owner must have clear understanding of business, engineering and marketing. Good communication skills are very important to deal with different stakeholders having different interests.

Scrum Master: Scrum master is a team facilitator who makes sure that team members are following scrum practices, rules and values to gain the business value. His role is different from traditional project manager. He conducts a brief meeting with team daily, called daily scrum to watch the progress. He is responsible of protecting team form outside intervention and provides good circumstance to work. At the end of each sprint, an evaluation meeting called scrum retrospective is conducted. In this meeting all the members share their experience and lesson learned during sprint. This greatly helps in enhancing team knowledge and deciding what should be done in next sprint.

Team: Scrum teams are self-organizing which consists of 3 to 9 members. In scrum team, specific roles are not assigned to members. They can divide tasks among them according to their interest. The entire team should have skills in designing, developing, testing or documenting the product. These are the people who are responsible of delivering a working product after each sprint.

IV. COMPARISON OF XP AND SCRUM

Both XP and Scrum are well-known, widely used agile models with some similarities and differences. To explore them from different perspectives a detailed comparison is conducted by considering different factors and features in Table I. This comparison can be very useful for researcher and developers to make a good choice according to project needs. For the sake of comparison we have consulted a number of research papers and studies which include [1], [3], [5], [6], [15]–[25], [27]-[32], [33].

V. CRITICAL ANALYSIS

Detailed comparison of XP and Scrum reveals very interesting facts about both models. Both of these have some common and some distinguishing aspects. It is observed that both models are focused towards building fully functional software using an adaptive approach. Both have incremental and iterative nature however iteration duration is different.

XP has set of twelve principles that provide a concrete guidance about whole development process whereas in Scrum selection of development practices is left on team members

[6], [21]. Scrum provides a framework rather than concrete development practices. That's why Scrum greatly depends upon developers' skill and experience [3]. XP mainly focus on engineering aspects of software projects whereas Scrum deals with management related issues. These contrary practices are also complementing each other. Joint application of Scrum and XP can give positive impact on team productivity, product quality [26].

VI. CONCLUSION

XP and Scrum are renowned agile models that are widely used for small projects especially. These models used best practices in agile fashion to accommodate rapid application development needs. In this study a complete description explaining their different phases, practices and roles is provided. A detailed comparison is also conducted to get deep understanding about these models. This can be very helpful for developers, researchers and scholars interested in these agile models. This comparison reveals that these models have common and contrasting features. Some of contrasting feature complement each other that encourage researchers to experiment with combination of XP and Scrum for software development.

Table I: Comparison of XP and Scrum

Features	Extreme Programming	Scrum
Development Approach	Iterative and incremental	Iterative and incremental
Project Size	Small	All
Team Size	2 to 10	Multiple teams of less than 10 members
Team Activities	Yes; Planning game, Pair programming, Collective code ownership etc.	No
Iteration/Sprint Duration	1 to 3 weeks	4 weeks
Stakeholder's Involvement	Throughout the process	Not defined
Communication Style	Oral, through standup meetings	Oral, through Scrum meeting
Project Management	No	Yes; Practices for project management are available
Physical Environment	Co-located teams	Not defined
Abstraction Mechanism	Object oriented	Object oriented
Focus	Towards engineering aspects	Towards management and productivity aspects
Response to Change	Quick	Quick
Requirement Elicitation	User stories and on-site customer practices are used	Not defined
Distinction Among Different Requirements (Functional, Non-functional)	Not defined	Not defined
Documentation	Less	Less
Upfront design Document	No	Not defined
Design Flexibility	Start from Simple design that can be changed using refactoring.	Focus on simple design
Development order defined by	Customer	Scrum Team
Development Style	Adaptive	Adaptive
Code Ownership	Whole team	Not defined
Changes During Iteration	Allowed	Not allowed
Acceptance Criteria	Defined	Defined
Feedback	Span from minutes to months	Span over a month
Testing	Unit testing, integration testing, acceptance testing	Not defined
Structured Review meetings	No	No
Validation Technique	Functional Testing and Acceptance Testing	Not defined
Quality Assurance Activities	Test first approach	Not defined
Coding Standards	Properly defined	Not defined
Software Configuration Practices	Not defined	Not defined
Support for Distributed Projects	No	Not defined
Process Management	No	No

REFERENCES

[1] L. Williams, "Agile software development methodologies and practices," in *Advances in Computers*, vol. 80, Elsevier Inc. 2010, pp.1-44.

[2] J. Newkirk, "Introduction to agile processes and extreme programming," in *Proc. 24th Int. conf. Software engineering*, May 2002, pp. 695-696.

[3] E. Mnkandla, and B. Dwolatzky, "A survey of agile methodologies," The transactions of the SA institute of electrical engineers, vol. 3, pp.236-247, Dec. 2004.

[4] 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.

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

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

[7] 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.

[8] T. Dudziak, "eXtreme programming an overview," Methoden und Werkzeuge der Software produktion WS, 2000/1999, pp. 1-28.

[9] M. C. Paulk, "Extreme programming from a CMM perspective*," IEEE softw*., vol. 18, no. 6, pp.19-26, 2001.

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

[11] O. Kobayashi, M. Kawabata, M. Sakai and E. Parkinson, "Analysis of the interaction between practices for introducing XP effectively," in *Proc. 28th Int. conf. Softw. Eng*, May 2006, pp. 544-550.

[12] K. Beck, and C. Andres "Extreme Programming Explained: Embrace Change," Addison-Wesley Professional, 2004.

[13] K. Schwaber and M. Beedle, "Agile software development with Scrum," vol. 1, Upper Saddle River: Prentice Hall, 2002.

[14] J. Sutherland, K. Schwaber, and C. J. Sutherl, "The scrum papers: Nuts, bolts, and origins of an agile process," 2007.

[15] D. Cohen, M. Lindvall and P. Costa, "An introduction to agile methods," in *Advances in Computers*, vol. 62, 2004, pp.1-66.

[16] K. Schwaber, "Scrum development process. In Business Object Design and Implementation," Springer London, 1997, pp. 117-134.

[17] P. Deemer, G. Benefield, C. Larman and B. Vodde, "A lightweight guide to the theory and practice of scrum," Ver, 2, p.2012.

[18] G. Ahmad, T. R. Soomro and M. N. Brohi, "Agile Methodologies: Comparative Study and Future Direction," *Eur. Acad. Res*., 2014.

[19] A. Ahmed, S. Ahmad, N. Ehsan, E. Mirza and S. Z. Sarwar, "Agile software development: Impact on productivity and quality," in *IEEE Int. Conf. Management of Innovation and Technology*, Jun. 2010, pp. 287-291.

[20] A. I. Khan, R. J. Qurashi and U. A. Khan, "A comprehensive study of commonly practiced heavy and light weight software methodologies," *Int. J. Comput. Sci. Issues*, vol. 8, no. 4, pp. 441-450, 2011.

[21] P. Abrahamsson, J. Warsta, M.T. Siponen and J. Ronkainen, "New directions on agile methods: a comparative analysis," in *Proc. 25th Int. Conf. Softw Eng*., May. 2003, pp. 244-254.

[22] A.M.M. Hamed and H. Abushama, "Popular agile approaches in software development: Review and analysis," in *Int. Conf. Computing, Electrical and Electronics Engineering*, Aug. 2013, pp. 160-166. IEEE.

[23] A. Qumer and B. Henderson-Sellers, "Comparative evaluation of XP and Scrum using the 4D Analytical Tool (4-DAT)," in *Proc. Eur. Mediterr. Conf. Inf. Syst.*, 2006, pp. 1-8.

[24] J. M. Fernandes and M. Almeida, "Classification and comparison of agile methods," in *7th Int. Conf. Quality of Information and Communications Technology*, Sep. 2010, pp. 391-396, IEEE.

[25] U. S. Shah, "An Excursion to Software Development Life Cycle Models: An Old to Ever-growing Models," *ACM SIGSOFT Softw. Eng. Notes*, vol. 41, no. 1, pp.1-6, 2016.

[26] K. Mar and K. Schwaber, "Scrum with XP," 2002, Available: http://www.informit.com.

[27] L. Lindstrom and R. Jeffries, "Extreme programming and agile software development methodologies," *Inf. Syst. Manag*., vol. 21, no, 3, pp.41-52, 2004.

[28] G. Rasool, S. Aftab, S. Hussain and D. Streitferdt, "eXRUP: A Hybrid Software Development Model for Small to Medium Scale Projects," *Journal of Software Engineering and Applications*, vol. 6, no. 9, p.446. 2013.

[29] A. Dalalah, "Extreme Programming: Strengths and Weaknesses," *Computer Technology and Application*, vol. 5, no. 1, 2014.

[30] M. Huo, J. Verner, L. Zhu and M. A. Babar, "Software quality and agile methods," in *Proc. 28th Annu. Int. Conf. Computer Software and Applications Conference*, Sep. 2004, pp. 520-525, IEEE.

[31] J. Cho, "Issues and Challenges of agile software development with SCRUM," *Issues in Information Systems*, vol. 9no. 2, pp.188-195, 2008.

[32] L. Rising and N. S. Janoff, "The Scrum software development process for small teams," *IEEE software*, vol. 17, no. 4, pp.26-32, 2000.

[33] K. N. Rao, G.K. Naidu and P. Chakka, "A study of the Agile software development methods, applicability and implications in industry," *Int. J. Softw. Eng. its Appl.*, vol. 5 no. 2, pp.35-45, 2011.