Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 4, 2021, Pages. 2990 – 3003
Received 05 March 2021; Accepted 01 April 2021.

Agile Umbrella Methodologies and its Global Impact

Mohit Arora[1], Shivali Chopra[2], Manik Rakhra[3*], Vaishali Minhas[4], Roopcee Walia[5], Raghav Aggarwal[6], Manish Kumar[7], Mohammad shabaz[8]

[1]Department of Computer Science and Engineering Lovely Professional University Phagwara, India.
[2]Department of Computer Science and Engineering Lovely Professional University Phagwara, India.
[3*]Department of Computer Science and Engineering Lovely Professional University Phagwara, India.
E-mail: rakhramanik786@gmail.com
[4]Department of Computer Science and Engineering Lovely Professional University Phagwara, India.
[5]Department of Computer Science and Engineering Lovely Professional University Phagwara, India.
[6]Department of Computer Science and Engineering Lovely Professional University Phagwara, India.
[7]Department of Computer Science and Engineering Lovely Professional University Phagwara, India.
[8]Department of Computer Science and Engineering Lovely Professional University Phagwara, India.

ABSTRACT

Agile methodologies came as a boon in the field of Software Engineering. As per the needs of various stakeholders, "change" is the new change in the market and Agile fits well to this urge. Agile initiates a change in the Software Engineering culture, wherein all the major State-Of-The-Art methodologies viz. Scrum, eXtreme Programming and Kanban has been discussed in this paper. This paper presents current state of these methodologies covering their complete span from basic definitions to global impact in IT industries. The facts are discussed and presented after mining various repositories like Collabnet Version One. It can be inferred that Scrum is widely used and has a major impact amongst all other agile umbrella methodologies. The paper will assist all potential researchers as one stop solution to understand the various dimensions of agile methodologies.

KEYWORDS

Software, Agile Umbrella, Kanban, eXtreme Programming.

Introduction to Agile

The Agile Software Development (ASD) approach has been applied broadly during the mid-nineties of the twentieth century. In spite of the fact that there are just around ten to fifteen years of aggregated experience utilizing the agile technique, it is constantly imagined as one of the standard systems for programming advancement. "Agile" in general, implies that something is adaptable and responsive, so it utilizes its capacity to act aptly in a situation of quick change. Agile is lightweight way to deal with most parts of planning, coding and delivering applications. ASD is an iterative advancement strategy which focuses and recognizes various prerequisites that contribute to change. ASD offers an expert way to deal with programming advancement that includes human, hierarchical and innovative parts of programming improvement forms.

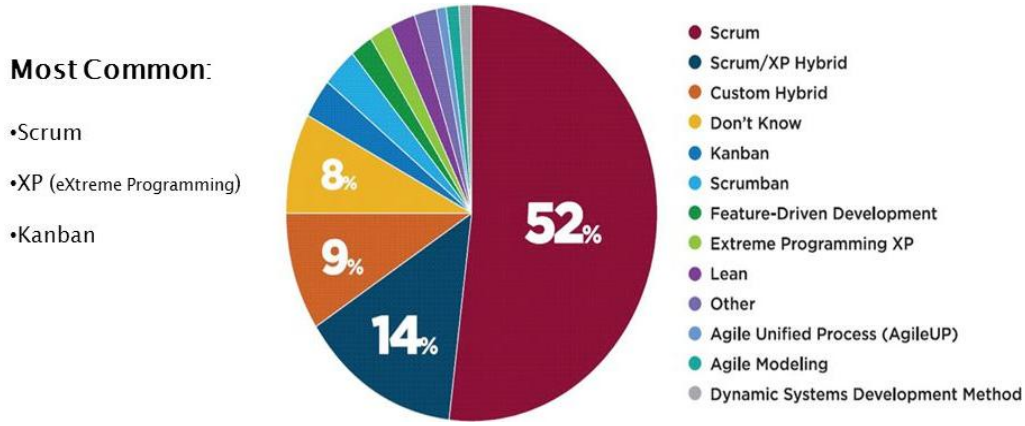

Fig. 1. Popular Agile Methodologies

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 4, 2021, Pages. 2990 – 3003
Received 05 March 2021; Accepted 01 April 2021.

ASD conveys prevalent and top-notch programming items in little and quick cycles with adaptability and flexibility to changing business conditions. In Agile, the prerequisites and results create through relationship among self-sorting out, self-persuading and cross-practical group. A list of various popular Agile Methodologies is given in Fig 1.

Introduction to Scrum

What is Scrum?

Scrum[1]–[3] is an agile framework[4] which takes in user stories as input requirements and accomplished the same in short, fixed-length and time boxed iterations called as sprints. A shippable product is delivered at the end of each sprint. The entire cycle from Sprint Planning to Retrospectives is called a Sprint.

History of Scrum

Jeff Sutherland, John Scumniotales and Jeff McKena developed and implemented Scrum[1] at Easel Corporation in 1993, inspired from the classic HBR article of 1986 titled "The New Product Development Game". The article stated Takeuchi and Nonaka's comparison of an innovative approach in the sport of rugby where the whole team tries to move as a single unit some distance, passing the ball all along within. Jeff and another Ken Schwaber jointly presented a paper, "The SCRUM Development Process"[5] at the 1st OOPSLA Conference of 1995 in Austin, Texas.

Scrum in Software Development

The various phases of Scrum are as given beneath.

i. **Product Backlog:** Customer prioritizes requirements as per business value and lists them in the product backlog. Requirements are listed in the form of User Stories (US). While the customer gives the priority, the project team also called the scrum team provides a high-level estimate for each user story.
ii. **Sprint Planning:** At the beginning of each time-box called sprint, team conducts a sprint planning meeting. All the stakeholders participate in the meeting. In the meeting, team picks up stories to be implemented in the sprint from the prioritized product backlog. The number of stories picked up depends on the available capacity in person-hours and the team's productivity. It is very important that the customers prioritize the product backlog. Prioritization ensures that the features developed first are of the highest value. The sprint planning meeting normally takes about half a day.
iii. **Sprint Backlog:** The Scrum team also breaks down product backlog's requirements into sprint tasks. These are the specific development activities needed to implement the requirement. The output of the sprint planning meeting is the sprint backlog. The sprint backlog contains the tasks and task-level estimates of the selected stories. When the Sprint Backlog is complete, you compare the estimated total work with original high-level estimates from the Product Backlog.
iv. **Implementation Cycle:** Once the team is ready with the sprint backlog, implementation of stories commences. The Implementation cycle involves the activities of design, coding and testing. The progress of the team is monitored through visual controls like Story Boards and Effort Burn-down charts.
v. **Daily Scrum:** Every day the daily scrum meeting is conducted at a pre-determined time – typically done at the beginning of the day. This is a short meeting carried out without deviating to technical issues. It is mandatory for the team to "stand-up" during these meetings so that the stipulated time is not exceeded. Each team member shares the status of their work by responding to the three questions: what did I do yesterday, what will I do today, and what obstacles are impeding my progress? At the end of the daily scrum meeting the sprint backlog is updated – with addition, deletion and modification to the planned tasks and the remaining efforts for the same. Figure 2 depicts the Scrum Process.

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 4, 2021, Pages. 2990 – 3003
Received 05 March 2021; Accepted 01 April 2021.

vi. **Sprint Review:** The output of the sprint is a potentially shippable product, which is demonstrated to all the stakeholders and their feedback is sought – this is called the sprint review meeting. All enhancements, bugs or defects identified by the customer are added to the product backlog and are addressed based on their priority.

vii. **Retrospective:** A retrospective is conducted after the Sprint Review. The team assesses what went well, what did not and identifies the changes needed to make the process better.

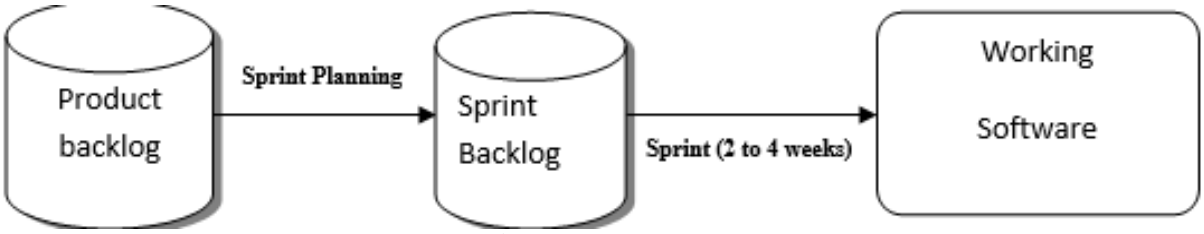

Fig. 2. Scrum Process

It allows team to inspect and adopt.
The Steps followed in Agile Software Development is given below:

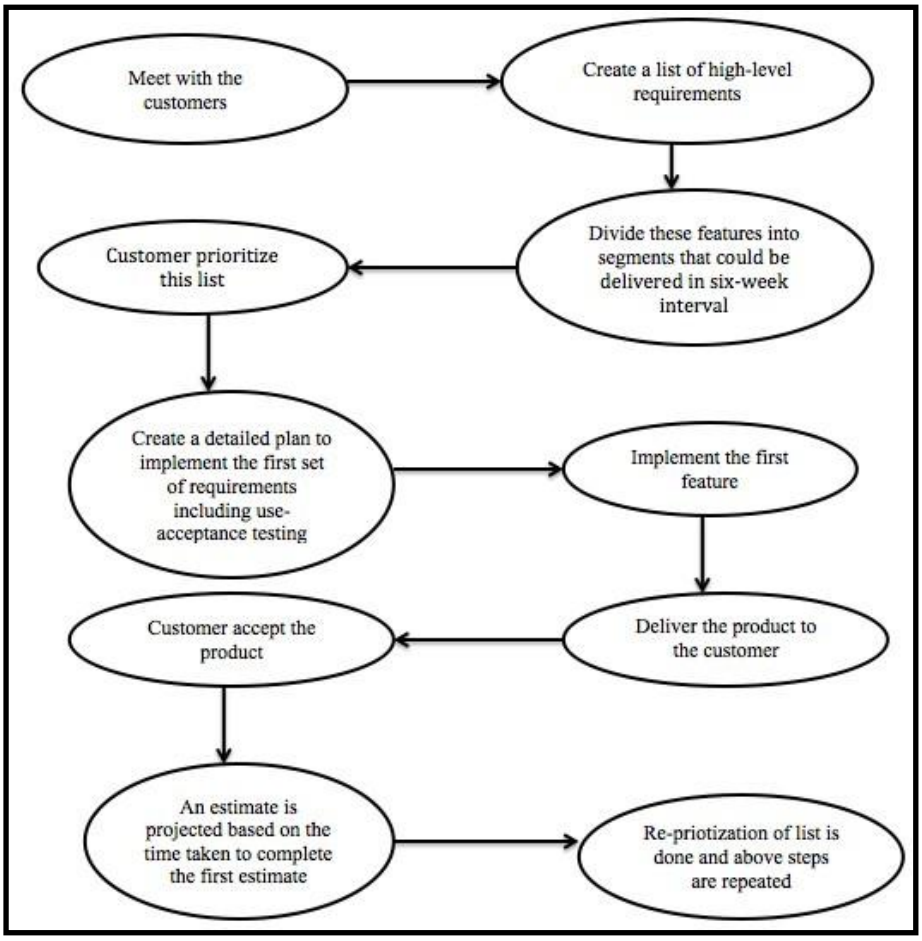

Fig. 3. Steps to be followed in Agile Software Development

a) Advantages of Scrum

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 4, 2021, Pages. 2990 – 3003
Received 05 March 2021; Accepted 01 April 2021.

i. The system of project development using Scrum methodology is transparent. The workers in the team are more accountable to their tasks. This also makes the company maintain their transparency with the client[6].
ii. The team workers are motivated on multiple stages of development[7]. Teams are having defined deadlines so there are some expectations that must be met so team workers are motivated due to the rewards received or will receive for them. This profits the client as they can perceive the stronger set of knowledge base.
iii. Team workers provide continuous feedback as they use daily check-ins for progress reports.

b) Hybrid software development using Scrum

Scrum is the widely used agile umbrella methodology. Developing and using hybrids of Scrum is must for fitting the purpose of several complex software projects that require two different methodologies for completing the tasks easily[8], [9]. Fig 3 depicts the Steps to be followed in Agile Software Development As Scrum encourages transparency and personal deadlines as motivation, a lot of work can be completed if the advantages of different methodologies are used. Some of the Scrum hybrids that have been proposed are as follows:

i. Water-Scrum-fall
ii. Scrummerfall
iii. Water Scrum
iv. Scrumban
v. Scrum XP hybrid

We have presented two Scrum hybrids in our paper – Scrumban (Scrum and Kanban) and Scrum XP hybrid.

c) Scrum adoption challenges and mitigation strategies

i. There is no project deadline. Scrum methodology uses personal deadlines to complete tasks but it provides no definition for project deadline. It only requires the team workers to meet their expectations. Projects can be completed in a set time if the team works cordially by completing tasks in their required time frame and not lag behind in development [10].
ii. Smaller teams adopt Scrum methodology easily but it becomes a problem for larger teams (10+ members) to adopt it. Teams can be divided into small clusters and then integrated a larger umbrella using Scrum hybrid [11] [12].
iii. Dependence on Scrum Master is high hence if he does not have a strong vision of the project, it may lead to the downfall of the team and the project. The choice of Scrum Master [13]. Hence, needs to be well thought of.
iv. Team members are required to be highly experienced. This limits the Scrum development process to high level members and excludes the newer subordinates of the team. New members can be asked to follow the tasks assigned to them and gain experience to easily delve into this methodology [14] [15].

Introduction to Kanban

a) What is Kanban?

Kanban system of scheduling is used for lean[16] and Just-In-Time (JIT) processes[17][18]. There are cards, either virtual or physical that are named “Kanban” which move throughout the process from starting to the end for the purpose of keeping a constant flow of Kanban such that the inventory at the start remains the same at the end.

b) History of Kanban

In the late 1940s, Toyota, under the leadership of Taiichi Ohno, Father of Toyota Production Systems, decided to study the supermarkets as they have a unique in-store system with in-store stocking methods so as to optimize their engineering processes. Supermarkets have enough products on the shelves without needing to store excessive quantity, on hold, as the supply equals the demand. The supermarkets don’t need to hold excessive products which make for significant efficiency in inventory management while always having stock of products for consumers.

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 4, 2021, Pages. 2990 – 3003
Received 05 March 2021; Accepted 01 April 2021.

Toyota applied the same system to its factory floors with the objective that a product's demand guided their ordering and their placement on the shelves. Kanban is a Japanese term which stands for "visual sign" or "card" and signifies the concept of essential communication through visual management. The Kanban system utilizes visual cues such as cards by transferring them to teams so that the material transfer process is robust and time efficient. The Kanban system was used first in machine plant shop of Toyota's main plant and gradually moved to being used on their various systems for different types of products[17].

c) Kanban system in Software Development

The use of Kanban system for software development revolves around a Kanban board, a tool of work visualization and workflow optimization for the teams[19]–[21]. The board can be physical or digital. Kanban boards contain information about the phases of the project's stage transition which depends on the team and the type of software projects[22]. Usually it covers "To-Do", "Work in Progress" & "Completed"[23] [24].

As Kanban is focused on getting tasks completed, Kanban is based on a set of 4 principles[25].

i. Begin with the current knowledge
ii. Adapt to incremental, evolutionary alterations
iii. Respect the current responsibilities, abilities and roles
iv. Encourage leadership at all stages

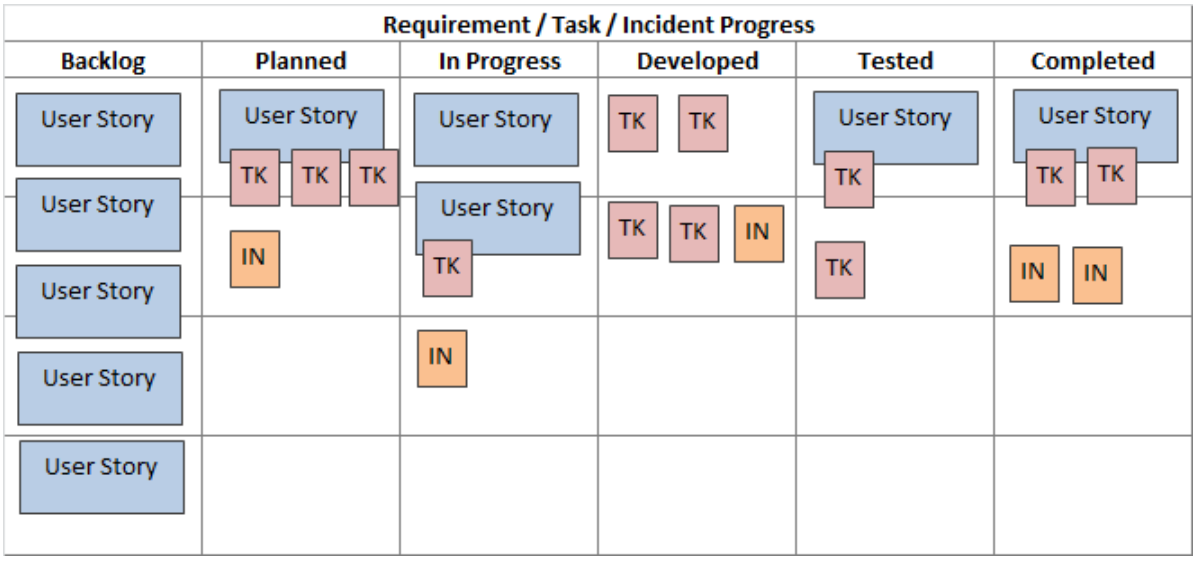

Fig. 4. Kanban System of Software Development

Figure 4 represents the kanban system of software development.Kanban is a philosophy of working rather than a system so there ought to be some steps or practices[26]. There are 6 practices that are followed when working using Kanban system –

1. Visualize the workflow
2. Limit Work-In-Progress
3. Manage Flow
4. Explicative the process policies
5. Provide feedback in loops
6. Improve by collaborating

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 4, 2021, Pages. 2990 – 3003
Received 05 March 2021; Accepted 01 April 2021.

d) Advantages of Kanban

i. Flexibility

As the philosophy of Kanban system lies in visual mode of communication, it is applicable to any professional domain from engineering to administration and is rather easy for the teams to implement with seamless transition across different functions.

ii. Incremental Improvements

Kanban system asserts improvement by streamlining the process and eradicating unnecessary functions.

iii. Responsiveness

Since Kanban system provides incremental improvements, the responsiveness of the team increases and tasks are handled better. This gives birth to Just-In-Time processes.

iv. Increased Output

Kanban encourages teams to limit their pending work at any current time, commonly called "Work-In-Progress". Limiting WIP decreases stress on any member of a team thus encouraging members to move forward equally and allows for intense focus on the tasks at hand. This prevents multitasking which yields low efficiency of work.

v. Empowered Teams

Team members are able to share and collaborate on any tasks to move forward when following the Kanban system. Hence, team members equally share their responsibilities.

vi. Quality Control

The Kanban system of software development makes the process so smooth and convenient by eradicating unwanted process and measure and increasing team focus that the product obtained is of much higher quality with little work on fine-tuning it so greater quality control follows.

e) Hybrid Software Development of Kanban

Scrumban is an agile methodology combining the principles of Scrum and Kanban[27]. Scrumban is used by teams to adjust to client and developer needs without being burdened by their project methodology[20] [28]. The prescriptive nature of Scrum for agility and adaptability of Kanban for incremental improvements in the process combines the best of both worlds thus making the workflow exceptionally flexible[29] [30].

The properties of Scrum used in Scrumban are –

i. Planning in iterations at regular intervals, synchronized with reviews and perceptions.
ii. Decision to handle work into the sprint based on project's complexity and sprint length.
iii. On demand prioritization of tasks helps teams to work on the next big thing and not less.
iv. Assert the required analysis before initiating development.
v. Using ready queue as a means of organization of tasks.

The properties of Kanban incorporated in Scrumban are given below

i. Continuous workflow allows teams to pull their work into Doing.
ii. Work-In-Progress Limits.

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 4, 2021, Pages. 2990 – 3003
Received 05 March 2021; Accepted 01 April 2021.

iii. Flexibility and respect of individual roles.
iv. Emphasis on Just-In-Time planning.
v. Using process buffers and flow diagrams for exposing process weakness and detecting opportunities for improvements.
vi. Focus on cycle time rather than burndown.
vii. Policies of process transitions are clearly developed.

f) Kanban Adoption Challenges and Mitigation Strategies

i. Opportunity versus Confusion

Kanban is straightforward. Truth be told, it is most likely the least difficult strategy for this kind. This makes it exceptionally light to begin, yet additionally snappy to fizzle with. When you have no standards, no unmistakable job definition, no meeting structure, zero required exercises, etc., you can rapidly slip into the disordered universe of "what the hell is going on?" Consequently, nearly everybody who rehearses Kanban goes with it with a lot of methods acquired from increasingly prescriptive frameworks or processes. I will share what we do at Kanbanize in a different article, however my point is that each group or organization should think of an increasingly organized approach to approach Kanban. It might sound odd; however, the individuals will really request that. They need a person or thing to depend on until they figure out how to utilize the opportunity and the adaptability that Kanban gives.

ii. Humans versus Robots

This one is presumably the riskiest of the three entanglements that I am touching on today. Kanban began in manufacturing as a basic signaling instrument between workers. It wants to give the perfect measure of parts at the correct time and doesn't really think about your feelings. Truth be told, this heartless grouping of endless undertakings might be very damaging to certain individuals. In the event that somebody continues finding "squander" in your work, you would likely despise them, wouldn't you? In the event that somebody continues pushing you to advance you might be disappointed, I get that. To moderate the dangers referenced above – be an extraordinary administrator. Regard the individuals in your group, converse with them a ton, sympathize with their pain and be their help. This will be hard on occasion since they will be basically whining about stuff, however in the event that you make them feel hopeless by applying the "moronic manufacturing techniques" to their innovative work, it is additionally your business to comfort them. In the event that I must be gruff, I need to concede that a few people just don't get it, regardless of how hard you attempt. They will presumably never get it, however don't abandon them. Just recall that we are largely human and being treated as robots make us despondent.

iii. Starting Small versus Big Bang

This is the place a great many administrators, groups, and organizations flop each day. There's a principle in Lean to begin with something little that is sufficient to be appeared to clients, yet it is entirely outlandish, in light of the fact that you might not have any desire to show a bit of your work that is a long way from finished and most likely very monstrous. In one of the organizations that I worked for we had the thought of a Minimal Marketable Feature, which is fundamentally the least you can do that would sound good to a client.

Introduction to eXtreme Programming

a) What is eXtreme Programming?

Extreme Programming (XP) is the most important agile development methodology. Extreme Programming is used to improve the quality of software products and is very responsive to the requirements of customer. It involves several software engineering practices for development of software [31]. Each practice of software development is very basic and complete. The grouping of various practices produces intricate performance. The main motive behind this methodology is to reduce the cost required to change the requirements of customers by introducing small

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 4, 2021, Pages. 2990 – 3003
Received 05 March 2021; Accepted 01 April 2021.

development iterations instead of long cycles. Changes in requirements are common and extreme programming methodology is providing solution to handle the changes in an efficient manner[32].

b) History of eXtreme Programming

Extreme Programming (XP) is a product improvement strategy grew essentially by Kent Beck. XP was one of the primary agile strategies; for sure XP was the prevailing agile technique in the late 90s and mid 00s before Scrum got predominant as the noughties passed. Numerous individuals (counting myself) consider XP to be the essential impetus that got consideration regarding agile strategies, and better than Scrum as a base for beginning in agile improvement [33].

c) Process Flow of Extreme Programming

The life cycle of extreme programming is the time required to complete the software project using practices of XP. XP Life Cycle starts with the planning stage and ends with the delivery of software product to the customer. XP is based on the principle of cycles or iterations and the whole product is made with the iterations and repeats until the project is done [34]. There are total five stages of XP life cycle which are described as follows:

i. Planning

Planning is the first stage of XP life cycle. In this stage, the goals and cycles are decided for the whole project. The developers of the company meet the customers and ask about the project details. The customer explains his/her hallucination of the project which is known as **user stories**. After collecting the details from the customer, the developers estimate and sorts the tasks and convert them into tasks for actual implementation [35].

ii. Designing

At this stage of the XP life cycle model a simple design is made where the main features of project are defined. Responsibilities of different developers are also described in this stage where every developer explained certain part of the design which will be implemented in the code part.

iii. Coding

In this stage simple code is written for the software programs. Refactoring of the code is done to make the code simpler.

iv. Testing

In the extreme programming, testing is done during code writing instead of after completion of project.

v. Listening

This is the last stage of the XP life cycle where regular feedback is taken from the customers.

d) Extreme Programming Practices

There are 12 core practices in extreme programming that needs to be followed in software development. Good skills are necessary to follow these practices.Figure 5 represents extreme programming process flow. Developers can tailor these practices as per the project needs. The practices that we follow in extreme programming are as follows:

i. Planning Game

XP Planning helps to plan that which part of the project needs to be completed by the last date and what is the next to

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 4, 2021, Pages. 2990 – 3003
Received 05 March 2021; Accepted 01 April 2021.

be done in the project. Tw types of planning is done- One planning is Release Planning where customer explains the modules that they need in the project and developers identify the difficulties in achieving the outcomes of the customer. Other type of planning is Iteration Planning where head of the team gives direction to the team members every week [36].

ii. Small Releases

The team releases running part of the software according to the customer requirements after each iteration. The iteration is of 3-4 weeks.

iii. Simple Design

The basic design should be created before the actual implementation. Design part gets changed throughout the iterations. Good Design is very important in an iterative or incremental methodology like Extreme Programming.

iv. Pair Programming

Pair Programming is a practice where two people work on the complete code at the same computer. One may be good in the logic building and the other in programming. This will increase the productivity and also give the knowledge of the released project.

v. Testing

XP methodology works on regular feedbacks from customer. So before getting the feedback on different modules, every part of the software needs to unit tested. Unit testing is done frequently to check any error present in the code. This is also known as 'Test Driven Development'.

vi. Continuous Refactoring

This practice involves change or modification in code or design frequently as per change in needs of customer. This helps in developing a quality product which will satisfy customer requirements.

vii. Continuous Integration

Integration of different modules is a continuous process of extreme programming. Programmers generally use the same programming language to prevent the integration issues.

viii. Team Code Ownership

Pair programming practice is followed in extreme programming where more than one programmer works on the code or design. Every member of the work is responsible for the code that they are writing for the software development.

ix. Coding Standard

For writing the code, proper standards and rules need to be followed in the extreme programming so that complete code looks simply. The simpler code is easy to understand and maintain.

x. Metaphor

XP team provides the complete description of working of the project in understandable form. It is same like use cases in UML diagram.

xi. Sustainable Pace

Excessive speed in programming can cause problems like errors or bugs in code. Cost in debugging will also be

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 4, 2021, Pages. 2990 – 3003
Received 05 March 2021; Accepted 01 April 2021.

increased if lot of errors is in the code. So optimum speed is needed in programming.

xii. On-Site Customer

With the developers, customer should be present all the time so that he or she can answer to all the questions.

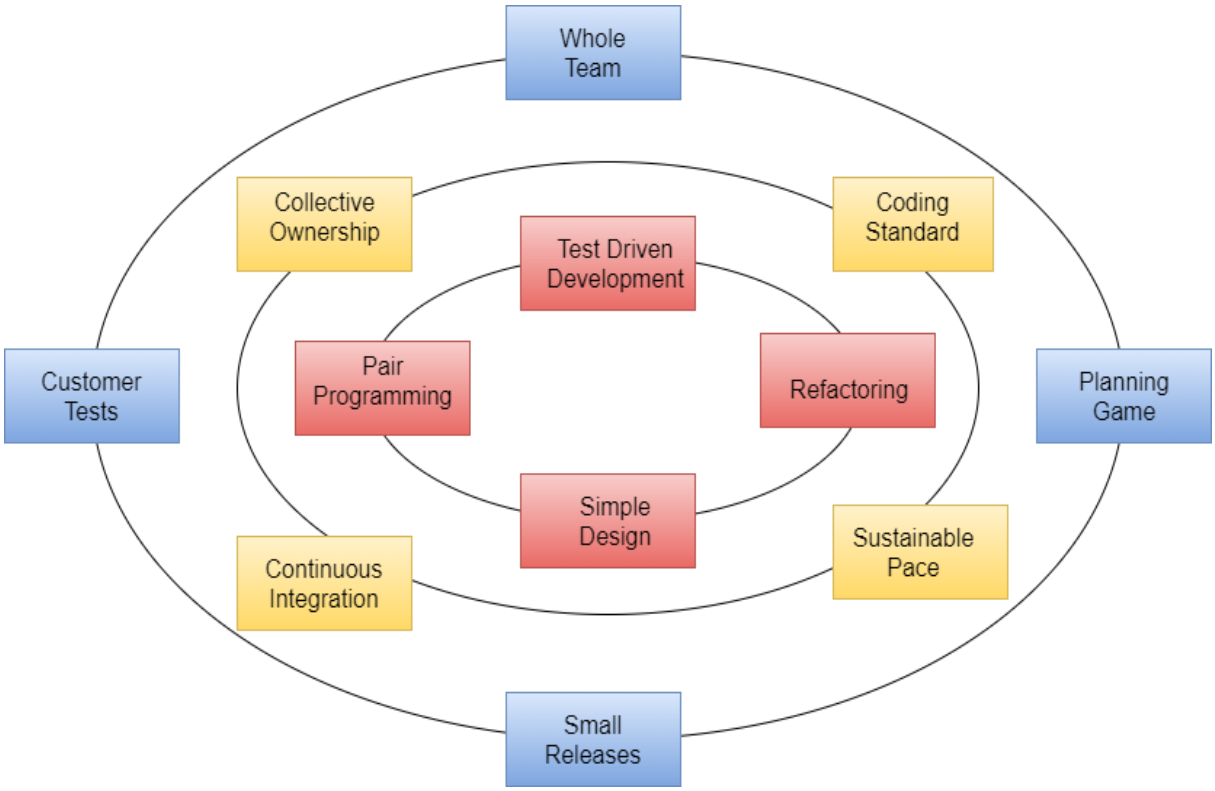

Fig. 5. eXtreme Programming process flow

e) Advantages of eXtreme Programming

i. The best bit of leeway of Extreme Programming is that this strategy permits programming improvement organizations to spare expenses and time required for venture acknowledgment. XP wipes out inefficient exercises to decrease expenses and disappointment of everybody included. It enables engineers to concentrate on coding.

ii. In the extreme programming methodology, generally XP team writes simple code which can be modified at any point of time.

iii. Developers work in pairs or team while doing the coding. They help each other which increase the productivity.

iv. Frequent feedback is the key point of extreme programming. According to the feedback from the customer, the team members do the changes wherever required. This helps to make the product according to customer satisfaction.

v. One of the major advantages of extreme programming is the reduction of risks. This helps the customer to get whatever they want without any error.

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 4, 2021, Pages. 2990 – 3003
Received 05 March 2021; Accepted 01 April 2021.

f) Disadvantages of eXtreme Programming

i. Extreme programming mainly focuses on coding instead of designing. Good design is very important for development of software products. If the design is not good then customer will not be satisfied.

ii. XP projects need interaction with the customers time to time. Customer should present near to the development team so that team can interact with the customer and can take regular feedbacks for the improvement. XP methodology can be applied to the projects where the customers are near to the development team.

iii. In XP methodology, requirements in documentation get changed frequently as per change in requirements of customer. If the documentation is not proper then high failures are possible in the software.

iv. Submission of project modules to the customer should be done within close-fitting deadlines. Stress levels of the developers also increases because they have to submit the projects in tight deadlines. Because of stress they make mistakes in the project which degrades the quality of product.

g) Scrum XP Hybrid

Agile manifesto and qualities give the edge, and every procedure and system apply its practices dependent on the spry standards and qualities. So as to get the most extreme profit by these strategies, a few systems or approaches began to be utilized together. We have exhibited the most recent Agile Survey before in our past posts. Scrum XP Hybrid is utilized by 10% and Scrumban is utilized by 6% of the associations took part in the Agile study [37].

Scrum and Extreme Programming (XP) are two agile procedures that function admirably couple. Truth be told, in the event that you strolled in a group doing one of these procedures you may have hard time rapidly choosing whether you had strolled in on a Scrum group or a XP group, both in light of the fact that their structure is comparable and on the grounds that Scrum groups frequently embrace XP practices, and the other way around [38].

h) Extreme Programming Adoption Challenges and Mitigation Strategies

Adopting extreme programming (XP) is a troublesome endeavour. On the off chance that you are in a group that is mid-project with a current codebase, the trouble and dangers are considerably higher. The present task stuff, for example, keeping up the current codebase, conveying the product on schedule, and group elements, alongside the test of embracing XP, for example, learning another range of abilities, defying current feelings of trepidation and issues, and disguising the procedure, isn't something to go into daintily. Notwithstanding the way that bringing changes into forms is dangerous, showing individuals new practices and approaches backs them off [39].

If any company is thinking to adopt Extreme Programming methodology in an organization then company has to choose a suitable project for extreme programming and team. Choose an experienced mentor who can lead his/her team. Manager has to choose a team who is known to extreme programming practices and communication with the team members. Start the undertaking with the insignificant basic Extreme Programming rules for the task. Enable the guidelines to develop for better usage. Enable adequate time for the group to scale the expectation to absorb information. Deal with the group culture and change [40].

i) XP Global Impact

There is a need of modern contextual analyses of XP as a rule, and of investigations of XP in Global Software Development (GSD) specifically. Be that as it may, some experimentally based XP contextual investigations do exist. Abrahamson directed a controlled contextual investigation of four programming engineers utilizing XP to actualize information the executives programming. Correlation between the first and second arrivals of the task yielded increments in arranging estimation exactness and efficiency while the deformity rate stayed consistent. Correspondingly, Maurer and Martel detailed a contextual investigation of a nine-developer web application venture. The group demonstrated solid efficiency increases in the wake of changing from a report driven advancement procedure to XP [41].

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 4, 2021, Pages. 2990 – 3003
Received 05 March 2021; Accepted 01 April 2021.

The XP-EF is a benchmark for communicating the context of XP contextual investigations, the XP rehearses an association has chosen to receive and additionally alter, and the business-related consequences of this reception. In the XP-EF, scientists and experts record basic context data of their undertaking by means of the XP Context Factors (XP-cf). These contextual analysis results recommend that the use of XP can help improve the business-related results, (for example, quality and profitability) of groups working inside specific contexts. In any case, there has been close to nothing explore on the utilization of XP in a GSD domain where the casual correspondence required in XP may turn into stressed [42-48].

Global Share of various Agile Methodologies

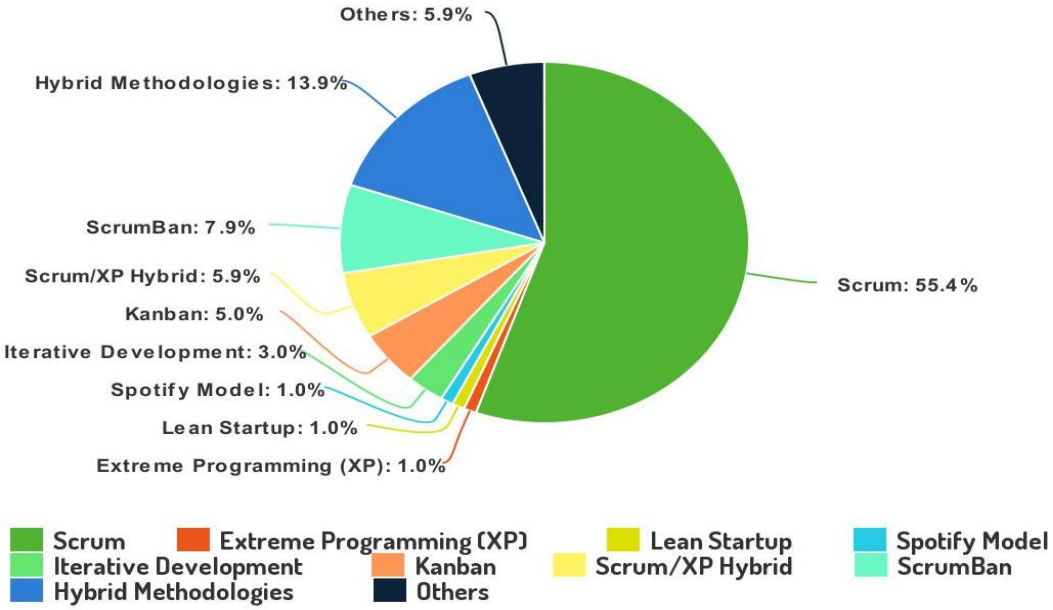

Fig. 6. Global impact of agile methodologies

References

[1] K. Schwaber and J. Sutherland, "The Scrum Papers: Nut, Bolts, and Origins of an Agile Framework," *SCRUM Inc.*, p. 224, 2010.

[2] M. Beedle, M. Devos, Y. Sharon, K. Schwaber, and J. Sutherland, "Scrum- A Pattern Language for Software Development," *Addison-Wesley*, vol. 4, 1999.

[3] A. Srivastava, S. Bhardwaj, and S. Saraswat, "SCRUM model for agile methodology," *Proceeding - IEEE Int. Conf. Comput. Commun. Autom. ICCCA 2017*, vol. 2017-Janua, pp. 864–869, 2017.

[4] F. Verbruggen, J. Sutherland, J. M. van der Werf, S. Brinkkemper, and A. Sutherland, "Process Efficiency - Adapting Flow to the Agile Improvement Effort," *Proc. 52nd Hawaii Int. Conf. Syst. Sci.*, vol. 6, pp. 6981–6987, 2019.

[5] K. Schwaber, "SCRUM Development Process," *Bus. Object Des. Implement.*, no. April 1987, pp. 117–134, 1997.

[6] F. U. Rehman, B. Maqbool, M. Q. Riaz, U. Qamar, and M. Abbas, "Scrum Software Maintenance Model: Efficient Software Maintenance in Agile Methodology," *21st Saudi Comput. Soc. Natl. Comput. Conf. NCC 2018*, pp. 1–5, 2018.

[7] S. Downey and J. Sutherland, "Scrum metrics for Hyperproductive Teams: How they fly like fighter aircraft," *Proc. Annu. Hawaii Int. Conf. Syst. Sci.*, pp. 4870–4878, 2013.

[8] J. Sutherland and D. Ph, "Future of Scrum : Creating a Scrum Company with a Type C All-At- Once Scrum," *Cambridge Innov. Cent.*, no. Type C, 2011.

[9] J. Sutherland, "Future of scrum: Parallel pipelining of sprints in complex projects," *Proc. - Agil. Confernce 2005*, vol. 2005, pp. 90–99, 2005.
[10] J. Sutherland, N. Harrison, and J. Riddle, "Teams that finish early accelerate faster: A pattern language for high performing scrum teams," *Proc. Annu. Hawaii Int. Conf. Syst. Sci.*, pp. 4722–4728, 2014.
[11] S. Sharma and N. Hasteer, "A comprehensive study on state of Scrum development," *Proceeding - IEEE Int. Conf. Comput. Commun. Autom. ICCCA 2016*, pp. 867–872, 2017.
[12] A. Mundra, S. Misra, and C. A. Dhawale, "Practical scrum-scrum team: Way to produce successful and quality software," *Proc. 2013 13th Int. Conf. Comput. Sci. Its Appl. ICCSA 2013*, pp. 119–123, 2013.
[13] J. Sutherland and N. Ahmad, "How a Traditional Project Manager Transforms to Scrum: PMBOK vs. Scrum," *Agil. Congr.*, pp. 1–7, 2011.
[14] J. Sutherland, A. Viktorov, J. Blount, and N. Puntikov, "Distributed scrum: Agile project management with outsourced development teams," *Proc. Annu. Hawaii Int. Conf. Syst. Sci.*, pp. 1–10, 2007.
[15] J. Sutherland and D. De Waard, "Scrum in Sales," *Enterp. Scrum*, pp. 3–4, 2011.
[16] N. A. A. Rahman, S. M. Sharif, and M. M. Esa, "Lean Manufacturing Case Study with Kanban System Implementation," *Procedia Econ. Financ.*, vol. 7, no. Icebr, pp. 174–180, 2013.
[17] Y. Sugimori, K. Kusunoki, F. Cho, and S. Uchikawa, "Toyota production system and kanban system materialization of just-in-time and respect-for-human system," *Int. J. Prod. Res.*, vol. 15, no. 6, pp. 553–564, 1977.
[18] C. Ani, M. Norzaimi, Kamaruddin, Shahrul, A. Azid, and Ishak, "Analysis of the effective production Kanban size with triggering system for achieving just-in-time (JIT) production," *Proc. - 2018 4th Int. Conf. Control. Autom. Robot. ICCAR 2018*, pp. 316–320, 2018.
[19] S. Nakazawa and T. Tanaka, "Prototype of Kanban Tool and Preliminary Evaluation of Visualizing Method for Task Assignment," *Proc. - 2015 Int. Conf. Comput. Appl. Technol. CCATS 2015*, pp. 48–49, 2016.
[20] S. P. Patil and J. R. Neve, "Productivity Improvement of Software Development Process Through Scrumban: A Practitioner's Approach," *2018 Int. Conf. Adv. Commun. Comput. Technol. ICACCT 2018*, pp. 314–318, 2018.
[21] S. Nakazawa and T. Tanaka, "Development and application of kanban tool visualizing the work in progress," *Proc. - 2016 5th IIAI Int. Congr. Adv. Appl. Informatics, IIAI-AAI 2016*, pp. 908–913, 2016.
[22] S. Shafiq and I. Inayat, "Towards studying the communication patterns of Kanban teams: A research design," *Proc. - 2017 IEEE 25th Int. Requir. Eng. Conf. Work. REW 2017*, pp. 303–306, 2017.
[23] H. K. Raju and Y. T. Krishnegowda, "Kanban pull and flow - A transparent workflow for improved quality and productivity in software developmet," *IET Conf. Publ.*, vol. 2013, no. 645 CP, pp. 44–51, 2013.
[24] H. Gong, B. Liu, and D. Shao, "A Simulation Model of Kanban Software Process," *Proc. - Asia-Pacific Softw. Eng. Conf. APSEC*, vol. 2017-Decem, pp. 745–746, 2018.
[25] N. Oza, F. Fagerholm, and J. Munch, "How does Kanban impact communication and collaboration in software engineering teams?," *2013 6th Int. Work. Coop. Hum. Asp. Softw. Eng. CHASE 2013 - Proc.*, vol. 68, pp. 125–128, 2013.
[26] M. O. Ahmad, J. Markkula, and M. Oivo, "Kanban in software development: A systematic literature review," *Proc. - 39th Euromicro Conf. Ser. Softw. Eng. Adv. Appl. SEAA 2013*, pp. 9–16, 2013.
[27] K. Terlecka, "Combining Kanban and Scrum - Lessons from a team of sysadmins," *Proc. - 2012 Agil. Conf. Agil. 2012*, pp. 99–102, 2012.
[28] C. Plengvittaya and D. Sanpote, "Scrumban for teaching at undergraduate program: A case study from software engineering students, University of Phayao, Thailand," *3rd Int. Conf. Digit. Arts, Media Technol. ICDAMT 2018*, pp. 109–114, 2018.
[29] N. Nikitina, M. Kajko-Mattsson, and M. Strale, "From scrum to scrumban: A case study of a process transition BT - 2012 *International Conference on Software and System Process, ICSSP* 2012, June 2, 2012 - June 3, 2012," pp. 140–149, 2012.
[30] D. I. K. Sjøberg, A. Johnsen, and J. Solberg, "Quantifying the effect of using Kanban versus scrum: A case study," *IEEE Softw.*, vol. 29, no. 5, pp. 47–53, 2012.
[31] L. Layman, L. Williams, and L. Cunningham, "Exploring extreme programming in context: An industrial case study," *Proc. Agil. Dev. Conf. ADC 2004*, pp. 32–41, 2004.
[32] F. Dumitriu, D. Oprea, and G. Mesnita, "Issues and strategy for agile global software development adoption," *Proc. World Multiconference Appl. Econ. Bus. Dev.*, no. January 2011, pp. 37–42, 2011.
[33] J. Erickson, K. Lyytinen, and K. Siau, "Agile modeling, agile software development, and extreme

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 4, 2021, Pages. 2990 – 3003
Received 05 March 2021; Accepted 01 April 2021.

programming: The state of research," *J. Database Manag.*, vol. 16, no. 4, p. 88, 2005.
[34] P. Abrahamsson and J. Koskela, "Extreme programming: A survey of empirical data from a controlled case study," *Proc. - 2004 Int. Symp. Empir. Softw. Eng. ISESE 2004*, pp. 73–82, 2004.
[35] 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.
[36] N. Harrison, "A study of extreme programming in a large company," *Avaya Labs*, no. January 2003, 2003.
[37] 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. 35–46, 2011.
[38] T. Dingsøyr, T. Dybå, and P. Abrahamsson, "A preliminary roadmap for empirical research on agile software development," *Proc. - Agil. 2008 Conf.*, pp. 83–94, 2008.
[39] J. Noll, "A Survey of Empirical Studies of Extreme Programming," no. 1, p. 7, 2007.
[40] I. Ghani, "Software security engineering in extreme programming methodology: a systematic literature review," *Sci. Int.*, vol. 25, no. 2, pp. 215–221, 2013.
[41] S. Sohrabi, "Challenges of user Involvement in Extreme Programming projects Shahriar Mohammadi Bahman Nikkhahan Information Technology Engineering Group, Department of Industrial Engineering," vol. 3, no. 1, pp. 19–32, 2009.
[42] R. Ferdiana, L.E. Nugroho, P.I. Santoso, and A. Ashari, "Global eXtreme Programming, a Software Engineering Framework for Distributed Agile Software Development," vol. 1, no. 3, pp. 106–112, 2010.
[43] M. Rakhra, R. Singh, T.K. Lohani, and M. Shabaz, *"Metaheuristic and Machine Learning-Based Smart Engine for Renting and Sharing of Agriculture Equipment,"* vol. 2021, 2021.
[44] M. Rakhra, N. Verma, and S. Bhatia, "Structural, Morphological, Optical, Electrical and Agricultural Properties of Solvent/ZnO Nanoparticles in the Photodegradation of DR-23 Dye," *J. Electron. Mater.*, vol. 49, no. 1, pp. 643–649, 2020, doi: 10.1007/s11664-019-07760-z.
[45] M. Rakhra and R. Singh, "Internet Based Resource Sharing Platform development for Agriculture Machinery and Tools in Punjab, India," *ICRITO 2020 - IEEE 8th Int. Conf. Reliab. Infocom Technol. Optim. (Trends Futur. Dir.*, pp. 636–642, 2020, doi: 10.1109/ICRITO48877.2020.9197908.
[46] M. Rakhra Asst, *"Phishing Educational Framework Davneet kaur,"* no. Icisc, pp. 832–836, 2018.
[47] M. Rakhra and R. Singh, "Materials Today : Proceedings Smart data in innovative farming," *Mater. Today Proc.*, no. xxxx, 2021, doi: 10.1016/j.matpr.2021.01.237.
[48] D. Venkatesh and M. Rakhra, "Agile adoption issues in large scale organizations: A review," *Mater. Today Proc.*, no. xxxx, 2020, doi: 10.1016/j.matpr.2020.11.308.