International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 01 | Jan 2021 www.irjet.net p-ISSN: 2395-0072

Agile Modeling with Extreme Programming: Values, Principles, and Practices

Manish Kumar[1], Dr. R.K. Dwivedi[2]

[1]University Departments of Computer Applications, Vinoba Bhave University, Hazaribag
[2]University Departments of Mathematics, Vinoba Bhave University, Hazaribag

---

Abstract: Agile is taken into account as one of the foremost practical and versatile software development mechanisms that exist today. It's capable of executing a range of tasks. Agile process make the most of the many benefits that has accelerate product delivery, enhance ability to manage changing priorities, Increase productivity, Enhance software quality, Enhance delivery predictability etc.

Agile software development is a software approach that focuses on continuous processes and adaptability to change. The most common agile methodology is called Extreme Programming. It recommends twelve practices including iterative development practices, automated unit testing, and pair programming.

In this study, we presented an overview of agile software development methodology and mainly focuses on detailed study of the most common agile methodology i.e. called Extreme Programming

Keywords: Agile Software, Extreme Programming, Process, Traditional software development, Kanban, Dynamic Software Development Method (DSDM).

1. INTRODUCTION

Two of the sexiest terms in software development these days are “Agile Software Development” and “Extreme Programming.” Both propose a different approach to software development than the traditional waterfall model, and both promise productivity improvements. The methodologies appeal to many software developers, and many have embraced them. [1][3]

Agile software development (ASD) methodologies [1] are gaining acceptance among mainstream software developers since the late 1990s, after they were first postulated within the sorts of Scrum [2], Extreme Programming [3], Kanban [4] and other methodologies. Agile software development methods are developed and evolved since early 1990s. Because of the short development life cycle through an iterative and incremental process, the agile methods are used widely in business sectors where requirements are relatively unstable. Agile method is a software development method that's people-focused communications-oriented, flexible, speedy, lean, responsive, and learning.

This paper gives overview of agile methodology, also covered in this includes; extreme programming core values, principles and practices. It also summarizes the strengths and weaknesses of extreme programming.

2. OTHER AGILE METHODOLOGIES

2.1. Scrum: Ken Swaber introduced the Scrum methodology in 1995. Scrum is one of the most widely used agile process frameworks in use today. In the Scrum framework, project work is done in a series of iterations, called sprints, which typically last 2-4 weeks. At the end of each sprint, the team must produce software that works to a high enough standard that it can be shipped or delivered to a customer. Within the framework of Scrum there are four formal meetings, known as ceremonies: [2]

•sprint planning,

•the daily scrum (or stand-up),

•sprint reviews, and

•sprint retrospectives.

2.2. Extreme Programming: Development teams started using Extreme Registration (XP) in the mid-1990s. Like other agile approaches, XP is iterative and provides small issues at the end of each iteration.

It defines XP "as a discipline for software development based on the values of simplicity, communication, feedback and courage. It works by bringing the whole team together in a simple practice, with lots of feedback so that the team can see where they are and adapt the practices to their particular situation. The extreme programming philosophy says that it makes software development fun, simple, flexible, predictable, less risky, efficient and more scientific. [3]

2.3. Kanban: Scrum and XP are frameworks that define sets of roles, ceremonies and practices for product delivery. In the context of software development, Kanban is an approach to managing workflows to enable evolutionary change. Rooted in constraint theory and lean product development, Kanban has five key principles: [5]

•visualize the work,

•limit work in process,

•focus on flow,

•make process policies explicit, and

•continually improve.

Unlike the other agile frameworks we discussed, Kanban development does not require fixed iteration. Work moves through the development process as a continuous flow of activity. A key feature of Kanban is to limit the amount of work in progress at any one time (referred to as the Work Frontier on Progress or WIP). In this approach the team only works on a fixed number of items at a time and work can only start on a new item when it is required to keep it flowing and after the previous item has been completed.

2.4. Dynamic Software Development Method (DSDM): DSDM (Dynamic Systems Development Mode) can be a robust project management and agile delivery framework that provides the right solution at the right time. The DSDM philosophy is that any project must be aligned with clearly defined strategic objectives and focus on delivering real benefits to the business at an early stage. DSDM is vendor independent, covers the entire project lifecycle, and provides timely guidance on best practices in project budget delivery, with proven scalability to handle projects of all sizes and with a purpose. [6]

3. EXTREME PROGRAMMING (XP)

Development teams began using Extreme Programming (XP) in the mid-1990s. Like other agile approaches, XP is iterative and provides small versions at the end of each iteration. XP's primary focus is on the engineering aspects of agile software development and is based on 12 practices promoted by Ward Cunningham and Kent Beck. [2]

3.1. Values, principles and practices of Extreme Programming [6][7]

Extreme Programming (XP) is based on the five values –

- Communication
- Simplicity
- Feedback
- Courage
- Respect

The fundamental principles of Extreme Programming are –

- Rapid feedback
- Assume simplicity
- Incremental change
- Embracing change
- Quality work

Extreme Programming (XP) is based on the 12 practices –

- The Planning Game
- Small Releases
- System Metaphor
- Simple Design
- Continuous Testing
- Refactoring
- Pair Programming
- Collective Code Ownership
- Continuous Integration
- 40-Hour Work Week
- On-site Customer
- Coding Standards

3.2. User Stories

XP uses the concept of user stories as the central mechanism for defining needs. They are created by system users to define attributes and functionalities that will be included in the solution and do not have a high level of detail. Each user story is usually accompanied by a list of acceptance criteria that identifies specific details about the story. [8]

Stories are used to:

•prioritize work into iterations,

•identify risk associated with a request,

•estimate the effort required to deliver the request, and

•establish a conversation between the team and the product owner around the subject of the real business need, in order to confirm a common understanding of what has to be done.

3.3. Release Planning and Execution

XP relies on three levels of planning:

•release planning,

•iteration planning, and

•daily planning.

Release planning identifies the next set of usable characteristics that a release can compose. There are often some job changes before the product is ready for release. Iteration planning plans for each incremental iteration that will ultimately result in an

available product. Finally, when planning daily, the team plans daily activities to ensure the team is on schedule and to identify potential risks that may arise.

In XP, release plans are used to track and describe the features or functionality that will be delivered in each product release. The launch plan is similar to the portfolio concept in the Scrum framework. The adjourned planning meetings are then used as a way of team collaboration in planning the next edition. As the team works to schedule the issue, user stories are sorted based on the most important features for the customer, ensuring that the most important features are always delivered first. The stories are broken down into granular functional requirements in a technique called story decomposition, on a just-in-time basis. Then, by putting the story together, the team identifies the detailed design and acceptance criteria for the story.

While iterations are underway, XP is similar to Scrum in that it uses daily meetings as the primary vehicle for team communication. This daily position meeting is used to facilitate daily planning activities and to review the progress of the previous day's position. In practice, teams using the XP approach often combine elements such as endings, roles, and ceremonies (such as sprint planning and sprint reviews) from the Scrum framework. [8][9][10]

The following diagram provides an overview of the XP model.

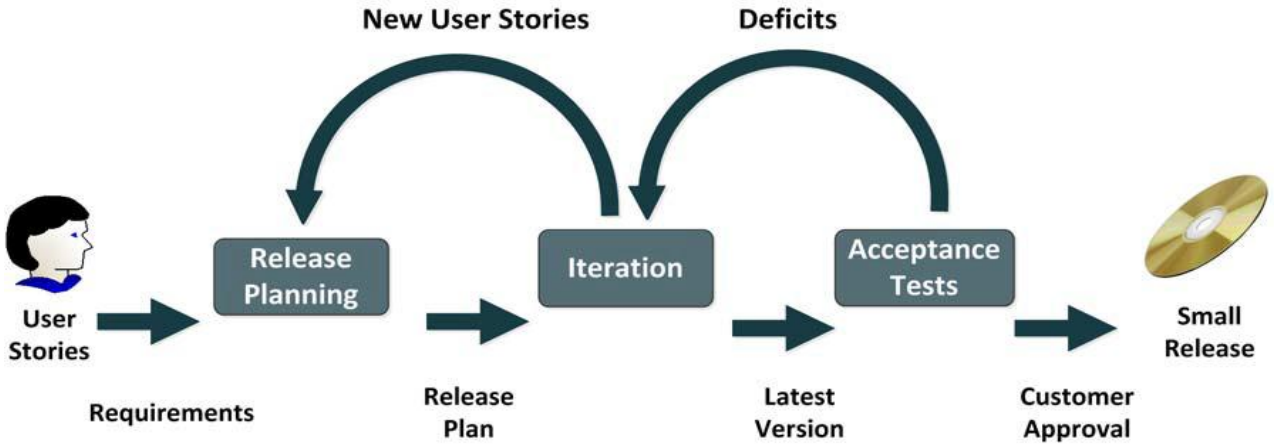

FIGURE1: Extreme Programming Model

3.4. Roles and Responsibilities

In Extreme Programming there are four key roles: [10]

•**Customer**: The customer creates and prioritizes the user stories and performs risk analysis.

•**Developer**: The developer communicates directly with the customer and builds only what is necessary to deliver on each iteration.

•**Tracker**: The tracker keeps track of the schedule and the metrics.

•**Coach**: The coach guides and mentors the team in applying XP practices effectively.

3.5. Business Analysis in XP

While XP focuses on value-based development, it does not explicitly address business analysis activities. XP is built on the basic assumption that the role of the customer is played by a small number of people who know what the most valuable roles are. When XP is implemented on a larger scale, or with customers who do not have a clear view of the incremental value of features, business analysts add significant value through facilitation and negotiation with stakeholders to reach a common

understanding of what the most valuable deliverables will be. Business analysts can also contribute by facilitating story mapping.

In traditional XP, clients create and manage user stories directly. This can lead to non-detection requirements that run the risk of restricting resolution without considering root cause or applicability to other customer groups. Business analysis skills can be used to ensure that underlying issues are addressed in a way that works for the majority of project stakeholders, if not, as well as to ensure that full acceptance criteria have been collected for all user scenarios. They often carry out writing and storytelling activities on projects that involve dedicated business analysts.[9]

3.6. Techniques

•**User Story**: User stories identify which roles in the story provide value and therefore identify the stakeholders who can elaborate on that value.[9]

•**Story Mapping**: Story maps show relationships between user stories and larger activities that the user must be able to accomplish.

•**Story Decomposition**: Epics, features, or minimally marketable features (MMF) tie groups of user stories together into larger packages that can be discussed with stakeholders.

•**Story Elaboration**: Defines the detailed design and acceptance criteria for a user story on a just-in-time / just-enough basis.

3.7. Extreme programming Life Cycle

Five stages of XP software development life cycle: [8][9]

Planning

This is the first step in the Extreme Programming development life cycle. Its main task is to establish the objectives of the entire project and certain iterative cycles. At this point, the team meets with the customer and asks about all future aspects of the software. The customer formulates vision of the product in the form of user stories. The developers evaluate them and prioritize them in the release plan. After that, work begins to turn them into tasks.

Designing

At this stage of the project, the team must define the main characteristics of the future code. The main thing is to create a simple design, because simplicity is one of the fundamental principles of the XP methodology. Extreme Programming developers often share responsibilities at the design stage. Each developer is responsible for the design of a certain part of the code.

Coding

The developers of Extreme Programming believe that good code should be simple. That's why they constantly refactor it. The refactoring procedure allows them to simplify the code or its parts without affecting the functionality of the final product.

Testing

In Extreme Programming, the test procedure is generally performed not after the final or intermediate product is made, but in conjunction with the code writing procedure.

Listening

In the final stage of the life cycle, the XP team must obtain customer feedback. He is the only person who estimates the final and intermediate products.

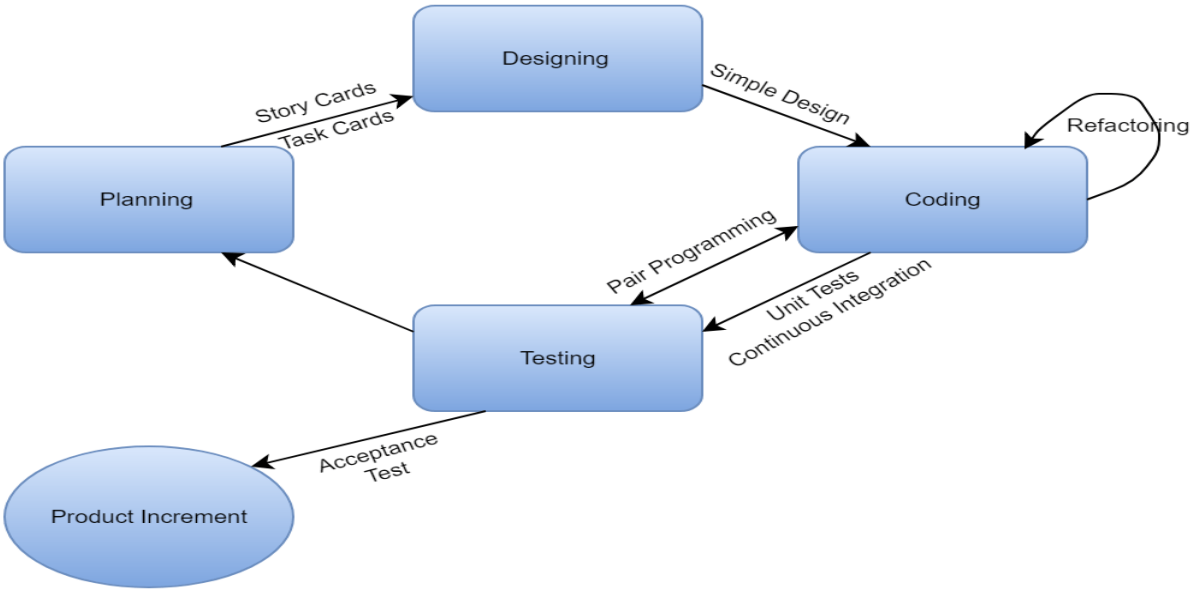

FIGURE2: Extreme Programming Life Cycle

3.8. Comparisons of Extreme programming and Traditional approach

Extreme Programming is different from traditional approaches in many ways: [1]

1. XP involves short, tight iterations of building and releasing software.

2. The customer is on site and is a component of the event team. Once the iteration has started, developers add pairs. No development is finished by one individual. All development is jointly owned. This can be a surprisingly element of XP.

3. A 'test-first' philosophy prevails. In XP you write the test to point out that the code will work before you write the code. Testing and coding are performed together. Before any new software is released, the entire test suite is run to make sure that it doesn't break the other component of the system.

4. Some principles underlying the people aspects of XP are: work no over 40 hours every week, be courageous, take responsibility and share ownership.

5. Some principles underlying code production are: keep it simple, have one shared metaphor to guide system development, regularly restructure the system to boost it (refactoring), continuously integrate and test, and follow coding standards.

4. EXTREME PROGRAMMING STRENGTHS AND WEAKNESSES

Extreme is an adaptive process and is also highly flexible, allowing software development to keep up with rapidly changing business needs for global competition, this practice enables organizations to meet the needs of a changing product with frequency. Scheduling reduces administrative and overhead costs, improves staff productivity, and meets customer needs, compared to the traditional plan-based development process. [10][11]

On the other hand, extreme programming has weaknesses; it is a process for a small development team of 10 to 12, it is also not scalable, which makes it difficult to take real large-scale projects. It also lacks artifacts and poor documentation, making it difficult to maintain the system developed with a great methodology, unlike the plan-driven process that practices great advanced design with artifacts and extensive documentation in each development life cycle. . Due to the lack of initial planning, according to the schedule compared to the traditional plan-based process, this safety application system is not recommended for a critical system. Experienced developers are required to implement them successfully. [10][11]

Extreme Programming	
Strengths	**Weaknesses**
Quick prototype delivery	Not scalable
Iterative approach to development	Too much emphasis on early results delivery
Rapid respond to changing requirements needs.	Pair programming expensive to practice
Create room for experimental designs	Test-drive approach extends development time.
Enhanced system reliability	Undefined requirements
Refactoring enhances software quality	Unstructured approach to development
Quality code	Lacked planning
Access to dedicated users	It lacked documentation
Lower overhead	Higher overhead
Suitable for medium size team	Not suitable for a large size team.

Table: Strengths and weakness extreme programming

5. CONCLUSIONS

Extreme Programming is a software development approach based on values of simplicity, communication, feedback, and courage.

Companies that build their workflow on XP principles and values create a competitive yet motivational atmosphere within and between teams. Programmers appreciate each other's project input; deliver software quickly because they can distinguish relevant tasks from unnecessary ones. They react quickly to feedback realizing it's a reasonable criticism aimed at making a product better.

XP teams don't consider each technical challenge as a problem but think of it as a way to develop skills.

REFERENCES

[1] W. S. Humphrey, A discipline for software engineering. Reading, Mass.: Addison Wesley, 1995.

[2] L. Williams and A. Cockburn, "Agile software development: It's about feedback and change," IEEE Software, vol. 20, pp. 39-43, 2003.

[3] K. Beck, Extreme programming explained: Embrace change. Reading, MA.: Addison Wesley Longman, Inc., 2000.

[4] G. Succi and M. Marchesi, "Extreme Programming Examined: Selected Papers from the XP 2000 Conference," presented at XP 2000 Conference, Cagliari, Italy, 2000.

[5] R. M. a. J. Newkirk, Extreme Programming in Practice: Addison-Wesley, 2001.

[6] R. Jeffries, A. Anderson, and C. Hendrickson, Extreme Programming Installed. Upper Saddle River, NJ: AddisonWesley, 2001.

[7] P. Schuh, "Recovery, Redemption, and Extreme Programming," IEEE Software, vol. 18, pp. 34-41, 2001.

[8] G. Melnik, L. Williams, and A. Geras, "Empirical Evaluation of Agile Processes," presented at XP/Agile Universe 2002, Chicago, USA, 2002.

[9] V. R. Basili, "The role of experimentation in software engineering: Past, present and future," presented at Keynote address in 18th International Conference on Software Engineering (ICSE18), Berlin, Germany, 1996.

[10] N. Fenton, "Viewpoint Article: Conducting and presenting empirical software engineering," Empirical Software Engineering, vol. 6, pp. 195-200, 2001.

[11] P. McBreen, Questioning extreme programming. New York: Addison-Wesley, 2001.