Volume 3, Issue 6, June-2016, pp. 321-326 ISSN (O): 2349-7084

International Journal of Computer Engineering In Research Trends

Available online at: www.ijcert.org

Review of Agile Software Development Methodologies

[1]Khushdeep Sharma, [2] Dr. Himanshu Aggarwal

Department of Computer Engineering
Punjabi University, Patiala,India
Khushi.batish@gmail.com , himanshu.pup@gmail.com

Abstract:- During the past forty years, many new software development approaches were introduced to fit the new cultures of the software development companies which aim to produce valuable software in short time period with minimal costs, and within unstable, changing environments and Agile Methodology is one of them. This paper presents a review of agile methodologies, comparison between agile methodologies, how they are divergent from the traditional process methods.

Keywords – Agile Software Development, Extreme Programming, SCRUM.

I. INTRODUCTION

In today scenario software development is expanding and is merging into many diverse fields, and hence becoming more inflexible and complex. Changing and Unpredictable requirements from customers is making it even more difficult. Traditional software development approaches are not able to curb the new changing requirements of the market in the best way, anymore. To solve such problems, new software development approaches are evolved, as agile methodologies which deliver faster, cheaper and better solutions. It includes some modifications to software development processes to make them more efficient, productive and flexible.

II.BACKGROUND

Agility is the flexibility of the software to react expeditiously and ability to fit to various changes in its surround. Main motive is to strip away the project heaviness associated with traditional software development methodologies in order to promote quick response to changing user requirements. Keeping this in mind, a number of software professional individually started using new lightweight software processes and result of their researches leads to set a new methodology called Agile and this term came into existence when seventeen of the developers of the "lightweight" approaches to software development came together in a workshop in early 2001 and they created the Agile Manifesto.

A. Agile Manifesto

The Agile Manifesto gathered representatives from Extreme Programming (XP), Dynamics Systems Development Methods (DSDM), Adaptive Software Development (ASD), Scrum, Crystal Methods, Feature-Driven Development (FDD), and others who saw the need for an alternative to documentation driven, heavyweight Traditional software development processes.

The manifesto reads as follows (Agile Alliance, 2001):

"We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value [3]

Individuals and interactions over Processes and tool Working software over Comprehensive documentation

Customer collaboration over Contract negotiation Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more". [1] The previous four values have been further defined by twelve principles: [2]

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes tackle change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity--the art of maximizing the amount of work not done--is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

B. Difference between Traditional and Agile Approaches:

Traditional Approaches	**Agile Approaches**
The environment is taken as predictable and Stable.	The environment is taken as unpredictable.
Sequential and synchronous process which are rule driven, formal and contains linear ordering of steps.	Concurrent and asynchronous process which are beyond formal rules and are iterative and exploratory in nature.
Optimization is the goal.	Adaption, responsiveness and flexibility are the goal.
Extensive Documentation	Minimal Documentation.
If customer changes the requirements, it becomes difficult to respond.	If customer changes the requirements, it becomes easier to respond.
Restricted access to the architecture. No collective ownership.	The whole team understands the architecture. There is a collective ownership.
Up front planning is maximal. Everything is big before you start.	The focus is on whether customer requirements are met in the current iteration
Extensive Documents and review meetings are needed to solve an issue.	5 minutes discussion per day may solve the problem
No communication and interaction within the team	High level of communication and interaction within the team.
Too slow to respond to user problems.	Provide quick responds to user problems.

Project lifecycle includes phases which contain tasks or activities.	Project lifecycle is guided by product features.
Traditional Approaches	**Agile Approaches**
The whole team is not working at the same time like the developers have to wait until the architecture is ready.	The whole team is working on the same iteration at the same time.
It is work centered process because team will change according to different phases of project cycle.	It is people centered process, as the same team is developing throughout the project.
Return on investment is at end of project.	Return on investment is at early of project.

III. AGILE PROCESSES

Under the umbrella of "Agile" term, there exist more specific methods such as Scrum, Feature-Driven Development (FDD), Extreme Programming (XP), Lean Development, Crystal Methods, Dynamic Systems Development Method (DSDM) and Adaptive Software Development (ASD). Many studies have been conducted on these agile methods. In this paper two agile methods- XP, Scrum will be discussed with more details.

Extreme Programming (XP)

Extreme programming (XP) is lightweight agile methodology which was developed by Beck [2] and whose main aim is customer satisfaction. This methodology focus on developing the end user requirements through incremental planning, evolutionary design, short development cycles and its ability to response to changing business needs. The streamline of development of software is straightforward. The requirements are ordered in terms of priority and are finalized which requirement is to accomplished first with due date and determining what to do next.

XP is built around these four values: [1]

1. Communication: XP emphasized on Communication among team members as many problems in projects can be solved by team members talking to each other.

2. Simplicity: XP states that it is better to start any user story with simplicity that can be changed in the future rather to start something with complexity that might create problems in the future..

3. Feedback: Feedback from end customer is required to gather concrete information about the state of the system and to determine the progress of the system.

4. Courage: courage adds an open-minded attitude on top of the other values.

Lifecycle of an XP project

The lifecycle of an XP project, shown in Figure, is divided into six phases: Exploration, Planning, Iterations to release, Production, Maintenance and Death. In the Exploration phase, the customer writes out the story cards which describe the target to be achieved. In planning phase a priority order is set to each user story and a schedule of the first release is determined. Next in the Iterations to Release phase, the development team creates a sub system and with continuous integration and testing, the end results are developed. In the Production phase performance of the system with extra testing is done before the system can be released to the customer .In Maintenance phase, all the postponed ideas and suggestions found at earlier phases are documented for implementation in this phase. Finally the Death Phase comes when the customer have no more stories to be executed. [5][6].

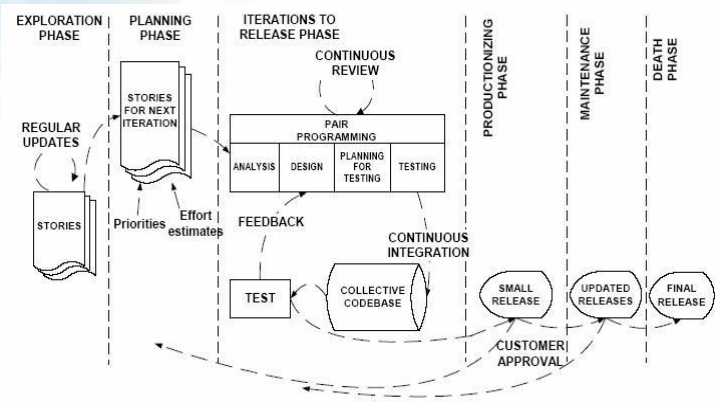

Figure 1 : Lifecycle of the XP process

XP presents following practices which are used during the lifecycle: [4]

1. The Planning game: Based upon customer requirements also known as user stories, the programmer estimates the effort and time needed for their implementation and first release date is also decided.

2. Small and frequent releases: In each story, some features of the project are developed and the running and tested functionality is released to the customer in a few weeks and is verified by the customer. Then the next iteration start which represents the smallest increment to the existing story in which new features of the system can be added and then again through continuous integration and thorough testing, the next functionality is released.

3. System metaphors: The system is defined by a set of which describes how the system works and these metaphors are set by both customer and the programmers.

4. Simple Design: Simple design is achieved by using refactoring techniques such as removing duplicated code, increasing the "cohesion" of the code and lowering the "coupling" of the code. After refactoring is done, the system is required to be verified to be still operational.

5. Customer Tests: To validate that the feature is implemented correctly, customer test are conducted. These tests help in achieving the quality oriented end results.

6. Pair programming: In this one machine is used by two programmers to write code which makes the system more efficient as the one developer writes the code while the other reviews each line of code. This will lead to improved design, testing and code.

7. Collective code ownership: Accessibility of the code is collectively taken by all team and each member has the ability to make changes wherever needed. Hence a liberal working atmosphere is provided to team.

8. Continuous integration: Continuous integration of the written code is done several times a day to ensure that all the components work together correctly.

10. Test driven development (TDD): All code must pass all tests before it can be released i.e. automated unit tests and acceptance tests.

11. Coding standards: Coding rules exist and are followed by the programmers so as to bring consistence and improved quality oriented results.

12. On Site Customer: A customer's representative is always available to check the streamline of the system to be developed.

Scrum:

SCRUM is another light weight method for the development of software which was initiated by Ken Swaber in 1995. It's features are derived from game Rugby in which each team acts as a whole to reach the common goal.It was exercised before the commencement of Agile Manifesto. Later, it becomes one of the methods of agile methodology since it has the same underlying approach as of agile development.

Characteristics of SCRUM :

- SCRUM only defines a group of rules and management practices that must adopted for the success of the project and doesn't require or provide any specific technique for the development phase.
- Its principle lies in the fact that good results can be produced by small teams working cross functionally.
- Scrum releases the software in small release cycles called sprints.
- Scrum ensures to give the team great flexibility as the team can choose the amount of work, staff and how to get the work done and it enhances a greater productivity of the work done. [9]
- It underlines the decision making from actual results rather than speculation.
- The end product is kept in a potentially shippable (properly integrated and tested) state at all times.

- At the end of each sprint, customer and team members meet to see a demonstration of a potentially shippable product increment and plan its next steps.[9]

The Scrum management practices are:
• Product backlog
• Daily scrum
• Sprint
• Sprint planning meeting
• Sprint backlog
• Sprint review meeting.

Product Backlog: It is considered the one responsible for demand gathering from the end customer. [11]. It is a list of activities which will be developed during the project. Meetings are conducted with the end customer, stakeholders, project partners and team members to extract out the technical demands and business needs. Product backlog consists of requirements pending for a project based on the parameters of complexity, the costs of the project, priority, days initial risks analysis, the work tools or some other unit of measure that the team decides.

The *Daily Scrum* also known as *Stand Up* Meeting is a quick daily meeting of fifteen minute session in which all team members defines the daily plan of action and results of the previous day's tasks. Three questions must be answered by every member about their responsibilities in this meeting [10]

- Tasks which were accomplished yesterday.
- Tasks which will be done today.
- Obstacles which were faced during the accomplishment of tasks?

Sprint is considered as implementation phase in which all work tasks articulated in the Product Backlog are implemented by the Scrum team and it is one to four weeks long.

Sprint Planning Meeting is the meeting in which the team with the help of customer plans its Sprint.

Sprint Backlog is a subgroup of Product Backlog which consists of list of activities that must be carried out during the Sprint

Sprint Review Meeting is the reflection of things that happens after each Sprint. Complete analysis and discussion is held about what went wrong or right and lessons learned from them. It is held with the customer to discuss the tasks developed over the last sprint or release cycle.
Three important Roles in scrum are:
A) The Product Owner,
B) The scrum master and
C) The team.

Scrum Master reviews the progress of the work that is done and discusses and solves the barriers which comes during the development of project. He assigns the work to be done to the team members and initiate the Daily Scrum meeting. He is responsible for keeping the team concentrate on the specific goals [7][8].

Product Owner represents the external client and frames the customer requirements with the vision of the product to development team. He is responsible for getting the funding for the project by creating the project's overall requirements, return on investment (ROI) objectives, and release plan.

The *Team* is responsible for the actual implementation of the Sprint[9]. A team includes a mix of software engineers, architects, programmers, analysts, QA experts, testers, and UI designers. Scrum teams should be self-managing, self-organizing, and cross functional to maximize performance.

Working of SCRUM :

Firstly the product backlog is built which describes all the requirements of customer. Based upon priorities the teams make a list of the tasks that must be accomplished in Sprint (Sprint Backlog) and the tasks are handed out. The phases of analysis, coding and testing are done for the implementation of the sprint. At the end of each Sprint, a new executable version of the product is presented to the client for feedback and the identified pitfalls are again added to the Project Backlog.

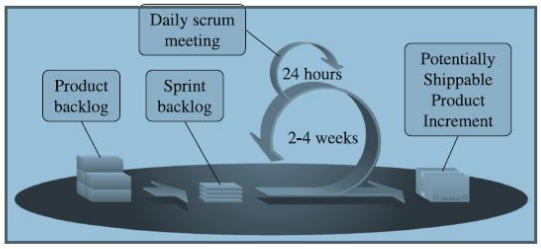

Figure 1. General idea of the scrum process dynamics. Source: Cohn (2008).

IV.CONCLUSIONS AND FUTURE WORK

Anecdotal evidence shows that agile software development methodologies which are based upon incremental and evolutionary concepts are very suitable and effective, but evidence of practical adoption of theses methodologies is very limited. That is empirical studies are urgently needed to have a deeper understanding of these methods for evaluating the possibilities and effectiveness of using agile methodologies.

In this paper, we discuss about agile methodologies, how they are divergent from traditional methodologies and the analysis of two agile software processes is also carried out. For a future work, a case study can be carried out on various small scale IT industries to check the practical adoption level of agile methodology and problems encountered while using these methodologies.

REFERENCES

[1] Beck, Kent; et al. (2001). "Manifesto for Agile Software Development". Agile Alliance. Retrieved 14 June 2010.

[2] Beck, Kent; et al. (2001). "Principles behind the Agile Manifesto". Agile Alliance. Archived from the original on 14 June 2010. Retrieved 6 June 2010

[3].Highsmith, , "Agile software development: the business of innovation", IEEE Software, 2001.

[4] K. Beck, "Embracing Change with Extreme Programming", IEEE Computer, vol. 32 no. 10, pp. 70-77, October 1999 [doi:10.1109/2.796139]

[5] From Wikipedia, the free encyclopedia, Available at:http://en.wikipedia.org/wiki/Extreme_programming

[6] C. Schwaber and R. Fichera, Corporate IT leads the second wave of agile adoption. Forrester Research,Inc, 2005.

[7] M. Cristal, D. Wildt and R. Prikladnicki, Usage of SCRUM Practices within a Global Company. Global Software Engineering, 2008. ICGSE 2008. IEEE International Conference on, 2008, 222-226.

[8] M. Singh, U-SCRUM: An Agile Methodology for Promoting Usability. In Ag. AGILE '08. Conference,Toronto, 2008, 555-560.

[9] "Scrum Methodology," [Online]. Available: http://scrummethodology.com/.

[10] RISING, L.; JANOFF, N. S. The Scrum Software Development Process for Small Teams. IEEE Software, v. 17, v. 4, p. 26-32, 2000.

[11] MAR, K.; SCHWABER, K. Scrum With XP. 2001. Available from: <http://www.controlchaos.com/XPKane.htm>. Access in: dez. 2008.

