Volume 3, Issue 2, February 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Review of Agile Software Development Methodologies

Kaushal Pathak
USICT, Guru Gobind Singh Indraprastha University, Sector - 16C Dwarka, Delhi - 110078, India

Anju Saha
USICT, Guru Gobind Singh Indraprastha University, Sector - 16C Dwarka, Delhi - 110078, India

Abstract—**Looking at the software engineering principles from a historical perspective, we can see how the software processing methodologies evolved since past 50 years, but probably the most discernible exchange to software business in recent years has been the introduction of evince "Agile". As numerous areas have overblown, there is a requirement to realize the components and narration, as easily as how the agile methodologies are diverse from the traditional one. Nimble practices to develop software projects are rattling hot and are advantageously glorious today. Methods like SCRUM, Extreme programming (XP), Feature driven Development (FDD), Adaptive software development (ASD) etc are increasingly being used to develop software using an adaptation approach rather than a predictive one.**

But, there is a scarcity of the resources which describe on how these resources can be integrated with the agile methodologies. This paper basically reviews different agile methodologies, how they are divergent from the conventional process methods, the pros and cons of applying agile processes to the research projects, and what are the difficulties faced during enforcement of agile methodology in the project.

Keywords— *Agile Software Development, Extreme Programming, SCRUM, Feature driven development.*

I. Introduction

A software process is defined as a set of methods, practices, activities and transformations that are used to acquire and affirm software and its related products [16].The issue of how software development should be organized in order to deliver faster, better, and cheaper solutions has been discussed in software engineering circles for decades. Many remedies have been suggested, from the standardization and assessment of the software process to a multitude of concrete tools, techniques, and practices. Recently, many of the suggestions for improvement have come from experienced practitioners, who have labeled their methods as agile software development. This movement had a huge impact on how software is developed worldwide. However, though there are many agile methods, but there is limited knowledge about how these methods are carried out in practice and what their effects are. The appearance of agile methods has been the most noticeable change to software process development in the last fifteen years. Many reviews, studies, and surveys have been conducted on agile methods. [11]

This paper will focus on what was behind the agile movement, how these methods are distinct from traditional approaches, and what are difficulties in implementation of these approaches. In addition, the paper reviews the existing agile methodologies.

II. Background

Agility, for a software development organization, is the power of software to choose and react expeditiously and fittingly to various changes in its surround and to the demands imposed by this surround. An agile process is one that readily embraces and supports this degree of flexibilty. So, it is not simply about the size of the process or the speed of delivery; it is mainly about flexibility. This term was agreed during a big gathering when seventeen of the developers of the "lightweight" approaches to software development came together in a workshop in early 2001. Previously, circumscribe of assorted groups have independently developed methods and practices to act to the changes they were experiencing in software processing and development [17].

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 [14]

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

B. What is "Agile"?

The understanding of the word agile varies in practice. In addition, it is difficult to define agile methods, as it is an umbrella for well-defined methods, which also vary in practice. This section will show how this word was explained in literature by its defenders, as well as by other researchers.
According to English Dictionary, the word "agile" has two meanings:

- (Mentally quick) able to think rapidly and clearly.
- (Physically quick) able to move your body quickly and fluently.

Although this definition addresses the response to change feature in agile methods, it missed a lot of their real meaning.
Alistair Cockburn is one of the initiators of the agile movement in software development, he defines agile as "*agile implies being effective and manoeuvrable. An agile process is both light and sufficient. The lightness is a mean of staying manoeuvrable. The sufficiency is a matter of staying in the game*" [3]
Barry Boehm described agile methods as "*an outgrowth of rapid prototyping and rapid development experience as well as the resurgence of a philosophy that programming is a craft rather than an industrial process*" [4]

Table I. How Agile is Different from traditional Approaches [7], [8], [9], [12], [15]

Traditional Approaches	**Agile Approaches**
Deliberate and formal, linear ordering of steps, rule-driven.	Emergent, iterative and exploratory, beyond formal rules.
Optimization is the goal.	Adaption, flexibility, responsiveness is the goal.
In this type the environment is taken as stable and predictable.	In this type the environment is taken as turbulent and difficult to predict.
Sequential and synchronous process.	Concurrent and asynchronous process.
It is work centered process because people will change according to different phases.	It is people centered process, as the same team is developing throughout.
Project lifecycle is guided by tasks or activities.	Project lifecycle is guided by product features.
Documentation is substantial.	Documentation is minimal.
Developers do waiting until the architecture is ready.	The whole team is working at the same time on the same iteration. Good coordination between team members
Too slow to provide fixes to user.	Provide quick responds to user feedback
Change requirements is difficult in later stages of the project	Can respond to customer requests and changes easier
More time is spent on design so the product will be more maintainable. The "what ifs" arise earlier	There is no time for the what ifs
No communication within the team, novices stay in their rooms and try to understand things	High level of communication and interaction, reading groups, meetings
Restricted access to architecture	The whole team influences and understands the architecture. Everybody will be able to do a design presentation
Documents and review meetings are needed to solve an issue	5 minutes discussion may solve the problem
Everything is up front, everything is big before you start	The focus is on whether customer requirements are met in the current iteration

Normal releases take 18 months	After 10 months the first release was out

III. Agile Processes

Under the name of "Agile" term, there are more specific approaches such as Extreme Programming (XP), Scrum, Crystal Methods, Dynamic Systems Development Method (DSDM), Feature-Driven Development (FDD), Adaptive Software Development (ASD), and Lean Development. As mentioned earlier many studies have been conducted on agile methods. In addition, many books and articles analyzed and compared agile methods in details. Three agile methods, XP, Scrum and FDD will be discussed with more details.

A. Extreme Programming (XP)

Extreme programming (XP) is one of the first agile processes that have been proposed. In general, XP consists of a set of individual practices that when put together yield a successful software Practice. Further the focus of XP is on the business aspect of a project resulting in increased productivity. Below are the core practices and values followed by Extreme Programming:

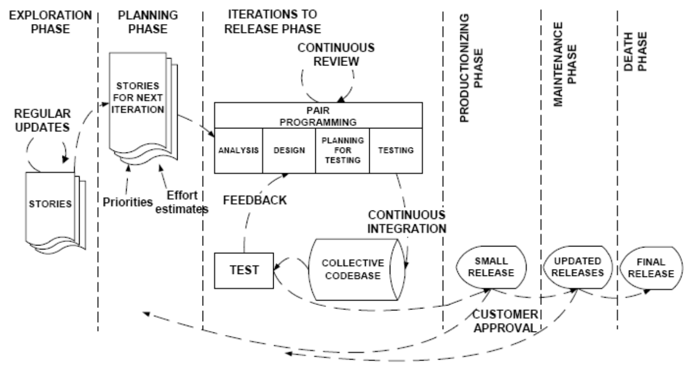


Fig.1 Extreme programming (taken from [6])

Whole Team: The team is considered to be very important in XP. The team may consist of developers, who develop the software, testers who are responsible for providing Quality Assurance, and analysts who help in design and the customer representative who provides the feedback. The customer representative may be the actual end user of the system. [10]

Planning: Planning is needed for calculation of estimate of how much effort and cost is needed to develop project. These planning are very effective because the product is visible all time. There are two types of planning in XP methodology.

a) **Release Planning:** The customer presents features that are expected by him in the software to developer. The developer reviews these features and find out the difficulties constrained in it. Based on all these constraints and technical challenges an initial release is decided.

b) **Iteration Planning:** Individual story iterations have shorter time spans which span over a few weeks. The customer presents the features which need to be developed over the next iteration. Based on the features presented, the team estimates the time and cost that may be involved in it. Also each iteration helps in learning about the product.[10]

Small Releases: The customer defines the features that have to be in project, represented as stories, which need to be developed. Every story represents the smallest increment to which new features of the system can be added, which usually takes only a few weeks to be developed.
The team releases the running and tested software to the customer after every iteration. The customer evaluates the software or releases of the product. The release cycles are very frequent and the customers are always presented first with a software release often. The releases which are delivered frequently undergo continuous integration and thorough testing.[10]
Simple Design: The design shows the functionality of the system. To achieve simple design, XP puts an emphasis on using refactoring techniques such as removing duplicated code, improving the existing design. Programmers must verify that the system is still operational after refactoring activity takes place. The XP process requires that all the phases of software development viz. design, implementation, and testing of the system should be carried out by a pair of programmers sharing one computer. This helps programmers to spend more time on finding solutions to challenging problem and less time doing routine debugging.
Pair Programming: By pair programming we mean that, the code is written by two programmers on a single machine. This ensures that the programs are reviewed by at least another programmer. This will lead to a better design, testing and code.

Also pair programming helps in spreading knowledge of the product across the unit. The pair programmers often switch their work and this leads to all the members in the team participating in pair programming. This helps in raising their skills and understanding of the system.

Collective Code Ownership: In XP, the code is owned by the entire team. No "module" is owned by a single person. In XP, any developer can work on the code base at any given time. This means that the code is seen by many people and this increases quality and reduces defects. If code is owned by an individual, the probabilities of errors or mistrusts are greater.

Sustainable Pace: The team develops the project at constant pace. The focus is on pace that can be sustained indefinitely. Work is carried out in a way such that productivity lasts. Working overtime means more pressure on the members and does not guarantee quality code.

Customer Tests: The XP customer checks the software by stating automated acceptance tests. This helps in validating the features of the product. The team treats the customer – customer tests in the same way as they do programming tests. The story is not successfully implemented until it passes the acceptance test, which is written by the customer.

The advantage of this method is to speed up the development process such that if programmers detect a fault in the code he has the right to fix it. A coding standard is specified globally, to make sure that the development team use the same design and coding conventions. To keep the development team motivated, XP discourages team members from working more than 40 hours a week. In addition, overtime weeks are usually limited to no more than two weeks in a row [1].

B. SCRUM

Scrum is another light weight method victimize for the development of software. Its principle lies in the fact that small teams working cross functionally produce good results. Scrum is more revenue centric with attention on improving revenue and quality of the software. Since being lightweight it can adapt to changing requirements and releases the software in small release cycles called sprints.

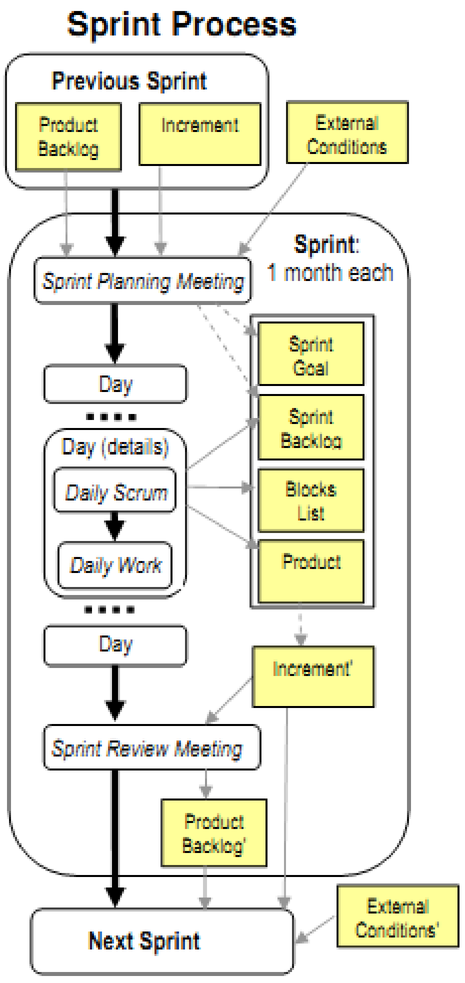


Fig. 2: Scrum Sprint Process [5]

This method has a framework which needs to be followed during development. The team can choose the amount of work, staff and how to get the work done. This ensures to give the scum team great flexibility and gives a productive work

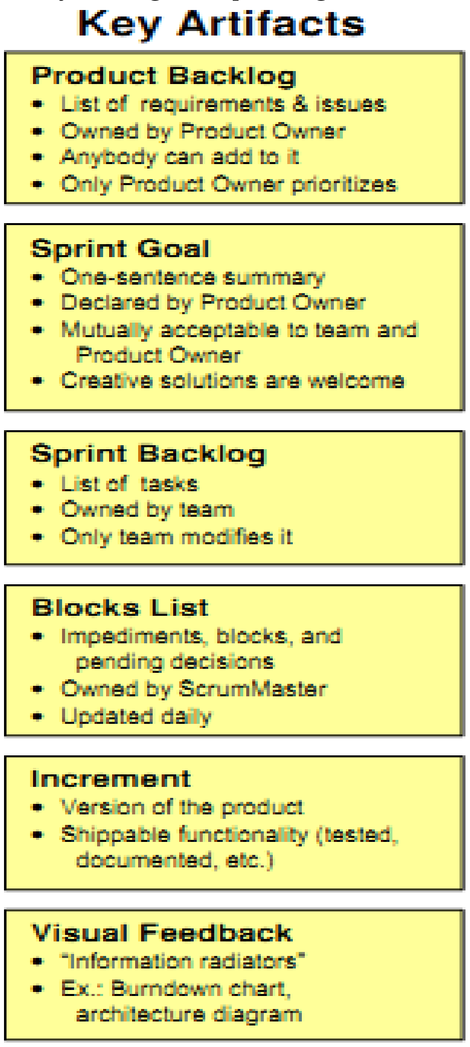
place. Three important things in scrum are the product owner, scrum master and team. The product owner specifies the various features of software, the release date and priorities. The scrum master makes sure that the team is functioning properly, productively and enables cooperation across all roles and functionality.
In addition, Scrum has a set of ceremonies associated with it. They include the sprint planning meeting, Daily Scrum Meeting and Sprint review meeting.

- The sprint planning meeting is between the customer and the team. An artifact called the Product Backlog prepared by the product owner has a list of features of the product including functionality and technical architecture.[6]

- The Daily Scrum meeting is a fifteen minute session initiated by the scrum master. The meeting reviews the work that is done regarding development.

- The sprint review meeting held with the customer to discuss the code developed over the last sprint or release cycle. All the stakeholders involved with the product can participate in the meeting and provide inputs for the next sprint. This meeting makes use of two artifacts called the Sprint Backlog and Burn down Chart which record the activities involved in the sprints The sprint Backlog is a subset of the product backlog.[6]

The sprints or iterations are around 30 days in length. A sprint begins with creating a Sprint Backlog.



Key Artifacts

Product Backlog
- List of requirements & issues
- Owned by Product Owner
- Anybody can add to it
- Only Product Owner prioritizes

Sprint Goal
- One-sentence summary
- Declared by Product Owner
- Mutually acceptable to team and Product Owner
- Creative solutions are welcome

Sprint Backlog
- List of tasks
- Owned by team
- Only team modifies it

Blocks List
- Impediments, blocks, and pending decisions
- Owned by ScrumMaster
- Updated daily

Increment
- Version of the product
- Shippable functionality (tested, documented, etc.)

Visual Feedback
- "Information radiators"
- Ex.: Burndown chart, architecture diagram

Fig. 3: Scrum Key Artifacts [5]

The team works on the backlog and resolves all the problems mentioned in the backlog. The sprint ends with the system being demonstrated to all the stakeholders.[6] Some Scrum teams have recorded a 10-20% increase in productivity by employing the practices of Scrum. Also progress is made even when the requirements are not stable. Scrum is more of a management process rather than a method meant for developing software.[6] Figure 2 depicts the scrum sprint process which involves the sprint process meeting, daily scrum meeting and sprint review meeting using the sprint backlog. Each iteration is called a sprint which is of duration 30 months. The output of each sprint acts as the input for the next sprint.[6]

C. Feature-Driven Development (FDD)

The Feature-Driven Development (FDD) approach focuses on the software features of the system. They are the main driver of the entire development process. It differs significantly from the other agile processes because they put a strong emphasis on planning and upfront design. As shown in Figure 3, the first step of the FDD process is to build a detailed model of the system, which captures all the stakeholders' assumptions and requirements. Once the domain model is built, the team members print a list of the features of the system.

Each feature has to be developed in a few hours or days, but no longer than 2 weeks. Using FDD, development teams are divided according to design and to implement a particular feature. The development work is performed in parallel on all the features. Each team is headed by a feature owner, who is responsible for the code segment that implements those features. This is contrasted with the XP approach where the ownership of the code belongs to the whole development team and not to a specific member.[13]

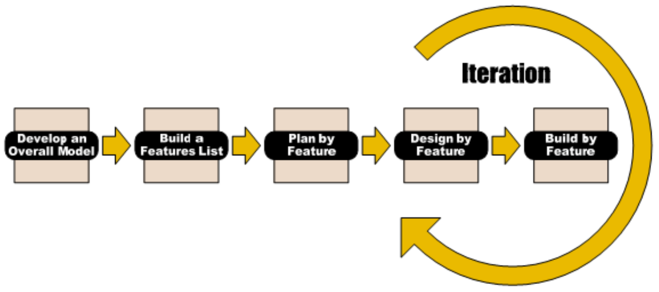


Fig. 4: The simple steps of FDD (taken from [6])

The FDD process enforces rigorous guidelines in order to find defects in the system. It also enforces coding standards and encourages regular builds on a daily or weekly basis in order to add freshly designed features to the base system. Since the features are developed in parallel, it is mandatory to have a configuration management system that allows calmly integrating of the changes that are made to the system. In FDD approach there is a tracking and papering mechanism that is used to show the project status based on the number of features that have been implemented as well as the overall progress of the design, coding, and testing activities. Each feature is scored using a value ranging between 0 (for a feature that has not yet been worked on) and 1 (a completed feature) and anything in between refers to a feature in progress.[13]

IV. Difficulties faced during implementation of agile methods

- Developer's Fear of Skill-Deficiency Exposure:

In a review of 17 companies, in all these companies, developers feared that the agile process could bring their deficiencies to light. So, always there is a pressure on the person.

To address this challenge, developers need an environment where they feel safe to expose their weaknesses like they could document any fears, issues or concerns due to which they didn't feel comfortable in an open forum, or there may be separate meetings for junior or new staff.[11]

- Broader Skill Sets for Developers

Generally in all the companies, agile environments seem to blur the boundaries among developers' roles and require competence in a broad range of skills, as opposed to specialization in one.

To address this challenge, organizations must strike a balance between team members becoming "masters of all" or "masters of none." It is to be advisable that developers must have broad knowledge on all aspects of software development but should also specialize and hone their skills in certain areas.[11]

- Increased Social Interaction

Agile practices such as pair programming, collocation, meetings, and retrospectives increase social interaction, communication, and your presentation skills. In most of the cases there are people who were technically very talented but had inherently weak communication and presentation skills, while all managers saw the benefits of constant face-to-face communication, the degree of communication in agile environment.

Social-skills training are an obvious solution to this challenge. It can help new people to get into the company values easily.

- Understanding Agile Principles

Some projects can or some can't implement agile values and principles. Some implemented agile methods "on paper," but they didn't achieve agility's ultimate goals. It may be due to many reasons like intangible combination of staff personality, management style and other factors.

Formal training is a typical solution to teach agile practices. Some companies included a provision for training and attendance at agile conferences focusing on its values and principles. Continuous training was preferable to one-off training in helping developers absorb and retain agile values and principles.[11]

V. Conclusions and future work

This article described a new type of model for software development i.e. agile software development. In this paper, we presented our analysis of three agile software processes. We also describe what are the problems faced during implementation of agile software development. The objective is to help software engineers to understand the key characteristics of these processes and therefore select the most suitable process with respect to the type of software projects they develop.

As a future work, there is a need to review other agile processes not covered in this paper such as the Lean Software Development (LSD) and the Dynamic Systems Development Method (DSDM) etc.

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] Agile Documentation: A Pattern Guide to Producing Lightweight Documents for Software Projects.

[4] Agile Processes in Software Engineering and Extreme Programming: 9Th International Conference, XP 2008, Limerick, Ireland, June 10-14, 2008 : Proceedings.

[5]. Scrum Process Mechanics, William C Wake, Available at http://xp123.com/xplor/xp0507/Scrum-dev.pdf , extracted on 3/2/2007

[6] D-B.Cao, "An Empirical Investigation of Critical Success Factors in Agile S'oftware Development Projects", PhD thesis, Capella University, USA, 2006.

[7] S. Nerur and V. Balijepally, "Theoretical Re -fl ections on Agile Development Methodolo-gies," Comm. ACM, vol. 50, no. 3, 2007, pp. 79–83.

[8] S. Nerur, R. Mahapatra, and G. Mangalaraj, "Challenges of Migrating to Agile Methodolo-gies," Comm. ACM, vol. 48, no. 5, 2005, pp. 72–78.

[9] P. Schuh, Integrating Agile Development in the Real World, Charles River Media, 2004.

[10] What is Extreme Programming, Ron Jeffries, Available at http://www.xprogramming.com/xpmag/whatisxp.htm, extracted on 11/02/2006

[11] Kieran conboy and Sharon coyle,"People over process: Key Challenges in agile development" IEEE Software.

[12] Mikio Aoyama, "Agile Software Process Model", IEEE Software, pp 454-455.

[13] Qasaimeh, M., "Comparing Agile Software Processes Based on the Software Development Project Requirements" IEEE Computer Society, 2008

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

[15] Aoyama, M., "Agile Software Process model", IEEE Software, 2002

[16] G. Cugola and C. Ghezzi, "Software Processes: a Retrospective and a Path to the Future", In Proc. of the Software Process Improvement and Practice Conference, 1998, pp. 101-123.

[17] T. Dybå and T. Dingsøyr, "Empirical Studies of Agile Software Development: A Systematic Review," Information and Software Technology, vol. 50, nos. 9–10, 2008, pp. 2-4.