ISSN:0975-9646

# Agile Methodologies

Sakshi Sachdeva[1]

trehansakshi21@gmail.com

**Abstract: Agile software development methods have been developed and evolved since early 1990s. This paper explains the differences between traditional software development methods and agile software development methods, and introduces the characteristics of some of the popular agile methods, Scrum and extreme programming.**

**Methods like SCRUM, Extreme programming (XP) etc are increasingly being used to develop software using an adaptation approach rather than a predictive one. This paper basically reviews different agile methodologies, how they are divergent from the conventional process methods and gives an insight of into the current agile methods.**

**Keywords: Agile, Waterfall, Scrum, Extreme Programming, Sprint, Product Owner**

## 1. INTRODUCTION

Scrum is part of the Agile movement. Scrum and other agile methods were inspired by the shortcomings of the dominant software development project management paradigms (including waterfall). It borrows many principles from lean manufacturing. In 2001, 17 pioneers of similar methods met at the Snowbird Ski Resort in Utah and wrote the Agile Manifesto, a declaration of four values and twelve principles. These values and principles stand in stark contrast to the traditional Project Manager’s Body of Knowledge (PMBOK). The Agile Manifesto emphasized on communication and collaboration, functioning software, team self-organization, and the flexibility to adapt to emerging business realities.

The Agile Manifesto doesn’t provide concrete steps. Therefore, Organizations seek more specific methods within the Agile movement which include Crystal Clear, Extreme Programming, Feature Driven Development, Dynamic Systems Development Method (DSDM), Scrum, and others. Scrum is the one that enables initial breakthroughs. Scrum’s simple definitions give the team the autonomy they need to do their best work while helping their boss (who becomes the Product Owner) get the business results he wants. A truly agile enterprise would not have a “business side” and a “technical side.” It would have teams working directly on delivering business value.

Scrum’s early users adapted feedback loops to cope with complexity and risk. Scrum emphasizes decision making from real-world results rather than speculation. Scrum is a simple set of roles, responsibilities, and meetings that never change. Time is divided into short work cadences, known as sprints, typically one week or two weeks long. The product is tried to be kept in a potentially shippable (properly integrated and tested) state at all times. At the end of each sprint, stakeholders and team members meet to see a demonstrated potentially shippable product increment and plan its next steps.

SCRUM defines the systems development process as a loose set of activities that combines known, workable tools and techniques with the best that a development team can devise to build systems. Since these activities are loose, controlling measures are used to manage the process and inherent risk. SCRUM is an also considered as an enhancement of the commonly used iterative/incremental object-oriented development cycle. [2]

New approaches like SCRUM focus on iterative and incremental development, customer collaboration, and frequent delivery through a light and fast development life cycle. Although there are many positive benefits of agile approaches that include shorter development cycle, higher customer satisfaction, lower bug rate, and quicker adaptation to changing business requirements, there have been few empirical field studies on issues and challenges of ASDMs. Therefore, the aim of this research paper was to discover the issues and challenges of one particular agile method in practice, Scrum, through a comparative study of traditional vs agile approaches.[1]

## 2. TRADITIONAL SOFTWARE DEVELOPMENT METHODS (TSDMS)

The most common traditional software development method is the waterfall model. The waterfall model utilizes a structured progression between defined phases: planning, analysis, design, implementation, and maintenance. The planning phase which occupies typically about 15% of total Systems Development Life Cycle (SDLC) is the process in which the scope of the new system is identified, why a system should be built is understood, and how the project team will go about building it is identified through technical, economical, and organizational feasibility analysis. The analysis phase, analyzes the current system, its problems, and then identifies ways to design the new system through requirements gathering. The design phase (35%) decides how the system will operate in terms of hardware, software, and network infrastructure. The implementation phase (30%) is the actual programming and coding. The maintenance phase (5%) focuses on go-live, training, installation, support plan, documentation, and debugging. Figure 1 below show a typical waterfall lifecycle. As we can see in the figure, each phase must be accomplished before the following phase can begin and each phase cannot go back to the previous phase like water in the waterfall cannot climb up once it reaches to a lower position and therefore this model does not accommodates any change.

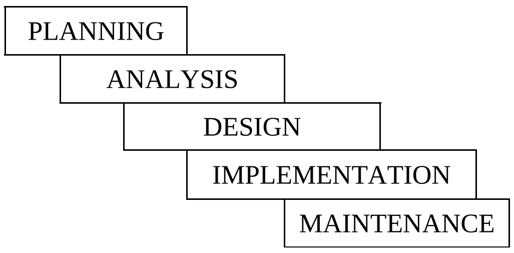

Figure: 1 Waterfall model lifecycle

Over the past four decades, traditional waterfall-style software development methods have been widely used for large-scale projects in the software industry and in the government sector due to their predictability, stability, and high assurance. As mentioned earlier, however, TSDMs have a number of key shortcomings, including slow adaptation to constantly changing business requirements, and a tendency to be over budget and behind schedule with fewer features and functions than specified. TSDMs are unable to respond to change and it is difficult to define all the requirements at the beginning of the project.

Many researches show that only a small percentage of projects that used traditional methods were completed on-time and on-budget with all features and functions specified. However, major percentage of the projects were completed either over-budget, over the time estimate and/or offering less features and functions; some of the projects were canceled at some point during the development cycle.

To overcome these shortcomings and issues, several practitioners developed agile software development methods including Scrum, extreme Programming (XP), Crystal, and Adaptive Software Development (ASD). The next section discusses the characteristics and principles of agile software development methods. [1]

## 3. AGILE SOFTWARE DEVELOPMENT METHODS (ASDMS)

The manifesto for agile software development which was created by seventeen practitioners in 2001 (http://www.agilemanifesto.org), reveals which items are considered valuable by ASDMs. ASDMs concentrate more on

1) Individuals and interactions than processes and tools,
2) Working software than comprehensive documentation,
3) Customer collaboration than contract negotiation, and
4) Responding to change than following a plan.[1]

The twelve principles behind the agile manifesto also present the characteristics of ASDMs.

1) Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2) Welcome changing requirements, even late in development. Agile processes tackle change for the customer's competitive advantage.
3) Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4) Business people and developers must work together daily throughout the project.
5) Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6) The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7) Working software is the primary measure of progress.
8) Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9) Continuous attention to technical excellence and good design enhances agility.
10) Simplicity--the art of maximizing the amount of work not done--is essential.
11) The best architectures, requirements, and designs emerge from self-organizing teams.
12) At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.[3]

Agile values and principles emphasize to focus on the people involved in a project and how they interact and communicate with each other within a team. Agile values and principles helps enhances team morale and better velocity than a team with poor functioning of talented individuals.[4]

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.

This definition addresses the response to change feature in agile methods. Alistair Cockburn, one of the initiators of the agile movement in software development, defines agile as “agile implies being effective and maneuverable. An agile process is both light and sufficient. The lightness is a mean of staying maneuverable. The sufficiency is a matter of staying in the game”

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” [3]

## 4. HOW AGILE IS DIFFERENT FROM TRADITIONAL APPROACHES

The Agile users claim that ASDMs have the potential to provide higher customer satisfaction, lower bug rates, shorter development cycles, and quicker adaptation to rapidly changing business requirements. Boehm believes that the primary focus of ASDMs is on rapid value whereas the primary focus of TSDMs is on high assurance. He also believes that ASDMs should be used for small teams and projects. If the size of the team and projects are large he suggests TSDMs.

| Agile Approaches | Traditional Approaches |
|---|---|
| The approach while using agile methods is adaptive. | The approach while using traditional methods similar to waterfall is generally predictive. |
| Flexibility and responsiveness is the goal. | Optimization is the goal. |
| It is an iterative and exploratory technique which is followed beyond formal rules. | It is deliberate and formal technique in which linear ordering of steps is followed and is typically rule-driven. |
| Concurrent and asynchronous process. | Sequential and synchronous process. |
| As the same team is developing throughout, it is people centered process. | As people will change according to different phases, it is work centered process. |
| Documentation is minimal. | Documentation is substantial. |
| Provide quick responds to user feedback. | Too slow to provide fixes to user. |
| Can respond to customer requests and changes easily. | Change requirements is difficult in later stages of the project. |
| 5 minutes discussion or scrum meetings (which are too short) may solve the problem. | Documents and review meetings are needed to solve an issue. |
| High level of communication and interaction is there. | There is no or very little communication within the team. |
| As the work is done in sprints, therefore the releases are out fast for customer review. | Normal releases take a lot of time to be formally out for the customer to test. |

**Table 1.** How Agile is Different from traditional Approaches [3], [1]

## 5. 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. Many studies have been conducted on agile methods. Also, many books and articles analyzed and compared agile methods in details. Two agile methods, XP and Scrum will be discussed with more details.

### 5.1 Extreme programming

Extreme programming (XP) is one of the first agile processes that have been proposed. It works by bringing the whole team together in the presence of simple practices, with enough feedback to yield a successful software Practice. The focus of XP is on the business aspect of a project resulting in increased productivity.

#### 5.1.1 Basic Extreme Programming

Whole Team: The team is considered to be very important in XP. In Extreme Programming, every contributor to the project is an integral part of the "Whole Team". The team forms around a business representative called "the Customer", who sits with the team and works with them daily. The team may consist of developers, who are responsible for the development of 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.

Planning: Planning is needed for the estimations of effort and cost required to develop project. These estimates during the planning are very effective because the product is visible all time. There are two types of planning in XP methodology.

1) Release Planning: The customer presents features that are expected by him in the software to developer. The developer reviews these features and then estimate their difficulty. With the cost estimates in hand, and with knowledge of the importance of the features, the Customer lays out a plan for the project. Initial release plans are necessarily imprecise and XP teams revise the release plan regularly.
2) Iteration Planning: is the practice whereby the team is given direction every couple of weeks. The customer presents the features which need to be developed over the next iteration. XP teams build software in two-week "iterations", delivering running useful software at the end of each iteration. The programmers break down the required features into tasks, and estimate their cost (at a finer level of detail than in Release Planning). Based on the amount of work accomplished in the previous iteration, the team signs up for what will be undertaken in the current iteration. Also each iteration helps in learning about the product.

These planning steps are very simple, yet they provide very good information and excellent control in the hands of the Customer. Every couple of weeks, the amount of progress is entirely visible. There is no "ninety percent done" in XP: a feature story was completed, or it was not. This focus on visibility results in a nice little paradox: on the one hand, with so much visibility, the Customer is in a position to cancel the project if progress is not sufficient. On the other hand, progress is so visible, and the ability to decide what will be done next is so complete, that XP projects tend to deliver more of what is needed, with less pressure and stress.

Customer Tests: As part of presenting each desired feature, the XP Customer defines one or more automated acceptance tests to show that the feature is working. The team builds these tests and uses them to prove to themselves, and to the customer, that the feature is implemented correctly. Automation is important because in the press of time, manual tests are skipped. That's like turning off your lights when the night gets darkest.

The best XP teams treat their customer tests the same way they do programmer tests: once the test runs, the team keeps it running correctly thereafter. This helps improving the system, always notching forward, never backsliding.

Small Releases: The customer defines the features that have to be in project, represented as stories. 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. The releases which are delivered frequently undergo continuous integration and thorough testing.
Simple Design: The design shows the functionality of the system. To achieve simple design, XP emphasize on using refactoring techniques such as removing duplicated code, improving the existing design. The system is required to be verified to be 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 code is reviewed by at least another programmer. This will lead to a better design, testing and code.

## 5.2 Scrum

Scrum is a light weight method for the development of software whose principle lies in the fact that small teams working cross functionally produce good results. It is more revenue centric with a focus on improving revenue and quality of the software. Since it is lightweight it can adapt to changing requirements. Scrum releases the software in small release cycles called sprints.
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 enhances a productivity of the work done. [7]
Scrum is a simple set of roles, responsibilities, and meetings that never change. It emphasizes decision making from actual results rather than speculation. Short time work cadences, known as sprints are typically one week or two weeks long. The product is kept in a potentially shippable (properly integrated and tested) state at all times. At the end of each sprint, stakeholders and team members meet to see a demonstration of a potentially shippable product increment and plan its next steps.[2]
Three important things in scrum are:
A) The Product Owner,
B) The scrum master and
C) The team.

The **Product Owner** should be a person with vision, authority, and availability. The Product Owner is continuously communicating the vision and priorities to the development team.
As the scrum values self-organization among teams, a Product Owner must fight the urge to micro-manage. At the same time, Product Owners must be available to answer questions from the team. He is responsible for getting initial and on-going funding for the project by creating the project’s overall requirements, return on investment (ROI) objectives, and release plan.

**Scrum Master** (SM) is responsible for ensuring that Scrum values, practices, and rules are enacted and enforced. The SM acts as an intermediary between the management and the team and therefore is responsible for enabling cooperation across all roles and functionality.
The Scrum Master acts as a facilitator for the Product Owner and the team. The Scrum Master does not manage the team. The Scrum Master works to remove any impediments that are obstructing the team from achieving its sprint goals. He makes sure that the team is functioning properly and is productive at all times. This helps in making sure the team's successes are visible to the Product Owner. The Scrum Master also works to advise the Product Owner about how to maximize ROI for the team.

The **Team** is responsible for the actual implementation of the functionality described in the requirements. Scrum teams should be self-managing, self-organizing, and cross-functional to maximize performance. All of the team members are responsible for both the success and the failure of sub-systems and of entire systems. A Scrum development team contains about seven fully dedicated members (officially 3-9), ideally in one team room protected from outside distractions. For software projects, a typical team includes a mix of software engineers, architects, programmers, analysts, QA experts, testers, and UI designers. In each sprint, the team is responsible for determining how it will accomplish the work to be completed and meet the goals of the sprint. [1]
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. The product owner prepares an artifact called the Product Backlog has a list of features of the product which are still not implemented including functionality and technical architecture. This artifact is discussed and the timelines related to the features included in it are negotiated as well.
- The Daily Scrum meeting is a fifteen minute session initiated by the scrum master. The master reviews the work that is done regarding development, discusses the obstructions if any and also assigns the work to be done too.
- The sprint review meeting is 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.
The sprints or iterations are around 30 days in length. [3]
Scrum fits well into small projects. Requirements can be prioritized in a well-structured manner. But, a disadvantage of using Scrum methodology is that the customer is offsite and therefore tight customer collaboration is not possible. Also improved team dynamics enabled by Scrum are not available in one-developer project. [6]

## CONCLUSION

This article described a new type of model for software development i.e. agile software development. In this paper also introduces the roles, ceremonies, and artifacts of Scrum, which is one of the most well-known agile software development methods in the industry and discussed some key artifacts of another agile methodology called as extreme programming.

Agile software development methods were developed to provide more customer satisfaction, to shorten the development life cycle, to reduce the bug rates, and to accommodate changing business requirements during the development process. This paper presents characteristics of traditional software development methods and agile software development methods, and the differences between them.

Agile approaches are meant to increase fastness and flexibility in the software projects. We understood that agile methods are a highly practical oriented field. The application of different practices of XP and Scrum differ from company to company. So to say, sound judgment on agile methods can be done by doing rigorous case studies, using the historical record of the companies and their projects.

## REFERENCES

[1] Juyun Cho, "ISSUES AND CHALLENGES OF AGILE SOFTWARE DEVELOPMENT WITH SCRUM" http://iacis.org, 2008. [Online]. Available: http://iacis.org/iis/2008/S2008_950.pdf.

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

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

[4] Sakshi Sachdeva, "Agile testing", Vol. 6 (2), 2015.[Online]. Available: http://www.ijcsit.com/docs/Volume%206/vol6issue02/ijcsit20150602188.pdf.

[5] Kaushal Pathak, "Review of Agile Software Development Methodologies" Volume 3, Issue 2, February 2013, [Online]. Available: http://www.ijarcsse.com/docs/papers/Volume_3/2_February2013/V3I2-0251.pdf.

[6] Kuda Nageswara Rao, G, "A Study of the Agile Software Development Methods, Applicability and Implications in Industry", Vol. 5 No. 2, April, 2011.[Online]. Available: http://www.sersc.org/journals/IJSEIA/vol5_no2_2011/4.pdf.