International Journal of Electronic Engineering and Computer Science
Vol. 3, No. 3, 2018, pp. 39-48
http://www.aiscience.org/journal/ijeecs

Agile Methodologies: Useful Approaches for Software Process Improvements over Traditional Methodologies

Naseer Ahmad*

Department of Computer Science, Virtual University of Pakistan, Lahore, Pakistan

Abstract

Now a days the business market is going to be extremely energetic, and organizations regularly transforming their software requirements in order to be compatible with existing business market. These organizations also give order to the software development teams for rapid delivery of software products in order to fulfil their requirements. The traditional software developments methodologies fail in order to meet these demands of particular organizations or customers. However, traditional developments approaches such as water fall model, spiral model, V model, or object oriented approaches, carry on to control the development of systems a small number of decades and a lot of effort has been done to enhance these traditional methodologies, Agile software development methodology carry its individual set of narrative tests that should be managed to convince the customer or an organization that put order for a particular software product through before time and consistent delivery of the valuable software product. In this paper, we illustrate the main facts and features of agile software development methodology that enhances the development process of software product to fulfil the quick changes of software product in the business market.

Keywords

Agile, Traditional, XP, Scrum, Methodology

Received: May 6, 2018 / Accepted: June 15, 2018 / Published online: August 6, 2018

1. Introduction

There are large numbers of people who have a question in their mind that "What is Agile Software Development Methodology?" And they found different answers about this particular question. The Agile Manifesto is closed under this definition.

In order to develop a software product an evolutionary and incremental approach which is presented in an extremely shared method with the help of self-organizing groups under a suitable control framework with "just sufficient" ceremony that generates tremendous results in a very profitable and sensible mode which fulfil the changing requirements of its stakeholders.

In fact Agile software development methodology is a mixture of various software development methodologies that are based on the incremental and iterative development approaches, in which needs and their results develop throughout teamwork among self-organizing, cross-efficient groups or teams [1]. During the conference of software process methodologists that were seventeen in number in 2001 for the purpose of enhancements in software development, the word "Agile" came into existence. These methodologists observed that there are lot of qualities and features in their methods; therefore they make a decision to give name such processes "Agile". The word "Agile" means in the sense of software that the processes are light and enough. However, the appearance of agile development methodologies came into existence in the mid of 1990s, when

* Corresponding author
E-mail address: ms120400137@vu.edu.pk

the software development methodologies like eXtreme Programming (XP), Scrum, Feature Driven Development (FDD), and also like Adaptive Software Development (ASD) began to come into view [13]. In the field of software development, launching of agile methodologies was done in 2001 in which old and new concepts related to software development was merged in order to get improvements in software processes. Consequently, the word "agile" was selected and the Manifesto for its software development was outlined and also tools and methodologies that would distribute the significances and principles for agile software development.

The significances and standards of Agile Manifesto explain the key components of agility that be supposed to be embedded in any particular technique that declares to be agile. The agile manifesto emphasises the significances or values and items are given below in the table that are still supposed important too:

Table 1. The Agile Manifesto [3].

Agile Values		Agile Items
Individuals and Interactions	over	Processes and tools
Working software	over	Comprehensive documentation
Customer collaboration	over	Contract negoation
Responding to change	over	Following plan

The twelve principles related to agile software development process are described as under:

1. Delivery of valued software in advance and continuous to the particular customer is main objective of agile software development methodologies.
2. Warm welcome to clients when they need changes in software, yet behind schedule in development process. The control of agile processes change for the customer's aggressive improvement.
3. Provide regular delivery of software product, starting a pair of weeks and ending at a pair of months, with a liking to the minimum timescale.
4. A team work of software developers with business people should be done collectively during the entire software project.
5. Development of software project should be around the provoked individual. Must provide support and suitable environment to individuals they need and have expectations from them to fulfil the task.
6. Face-to-face communication is best and suitable technique to express the information to and within a software development team.
7. Key measure of development is the working of software.
8. Agile methodologies encourage maintainable software process improvement. The sponsors, developers as well as customers must be capable to preserve a stable velocity for ever.
9. The enhancement in agility is obtained through regular awareness to the technical superiority and appropriate software design.
10. For exploiting the quantity of work not having to be completed, simplicity is core part of agile manifesto.
11. The suitable and accurate software architectures, requirements, and software designs come into view from self-organising teams.
12. At continuous time periods, the team replicates on how to turn out to be more successful, then refrains and corrects its behaviour as a result.

These twelve principles, we can assume as main thoughts that should be implanted in the practical work related to software project development that claims to be agile [3]. The rapid delivery of the software product does not term as agility in software development but also the rapid adaptation to modifying requirements. Firstly, the agile methodology is used when the characteristics of the software product incremental changes, mainly those characteristics that have not been contained in the early requirements documents. Secondly, some agile approaches try to reduce the risks by developing the software in small time intervals, known as iterations. Particular tasks are assigned to iterations in order to release working software product at the ending of iteration which fulfils minimal maximum priority requirements. Working software is key measure for making guess of development progress. When all iterations are completed then the remaining requirements are reprioritized. Therefore, the agile methodologies "hold" change, centre of attention is simplicity and also these methodologies highlight the face-to-face communication over the written document [4].

Traditional methodologies are organized by in order sequence of steps such as requirement definition, planning, deployment, building and testing. First, the user requirements are suspiciously acknowledged on fullest level. After this the common software architecture is pictured and the real coding begins and then the different types of testing and the absolute deployments are done. The main thought here is the comprehensive apparition of the resultant project prior to the building begins, and working one sided through the pictured resultant structure. The traditional methodologies need the customer to offer the comprehensive idea of the accurate requirements regarding the proposed software. In this paper I will explain the related work in II section, Agile Software Development Methodologies in section III, Traditional Software Development Methodologies in section IV, finally I will provide conclusion in the section V.

2. Related Work

The business requirements in order to response quickly to the surroundings in a novel, gainful and competent way is convincing the usage of agile methodologies to the developing softwares. Shine Technologies, Cutter Consortium and Standish Group done some surveys and they provide the estimation of usage of agile methodologies every year on the behalf of percentage. Like as the mobile devices have minimized the need for traditional landline telephones. Similarly the agile methodologies take the place of traditional software development methodologies in this period. The agile software development methodologies have bright and leading future. Normally, there are number of characteristics and features of software development project can take advantage from an agile methodology and some can take advantage from a more analytical traditional methodology. Every software project is different when it comes in the environment of methodologies. It is the universal truth that there is no "one-size-fits-all" solution [2]. The growing sum of interest has been compensated to the agile methodologies in the early days by equally researchers and practitioners, therefore, producing an increasing body of experimental data on various characteristics of agile software development. Separately from the personal methodologies and practical works of agile development, challenging issues have occurred, like the scalability issues of agile development for large and multisite projects, and the ability of the agile methodologies with current standards. Additionally, there is some misunderstanding with respect to relationship among ad hoc coding and the agile development. It has been suggested that the main reason for this misunderstanding is slowly technique of agile development. In general, the current agile development methodologies such as XP for software implementation and Scrum for the project management of agile methodology, both seems to suggest a quite restricted approach to accomplishing the tasks of software development. In addition, researches represent that by implementing various agile methodologies and practices, individual agile development groups can achieve a method that fulfils the objectives of CMMI level 2. Therefore, there is still a requirement to expand the agile software development methodologies to meet such CMMI requirements related to additional organisational level performances [3].

3. Agile Software Development Methodologies

As compared to traditional software development methodologies agile methodologies significantly improve the software processes in number of ways because it is extremely shared and evolutionary approach. In these days agile methodologies become much popular in the field of software engineering. Agile methodologies are incremental, iterative, and evolutionary model of software development. In agile methodologies the whole application is dispersed in the incremental elements or units sometime known as iterations and every iteration has a small development time, fixed and rigorously adheres as well. Due to small iterations the agile methodologies proposes an opportunity for quick, visible and inspiring software development.

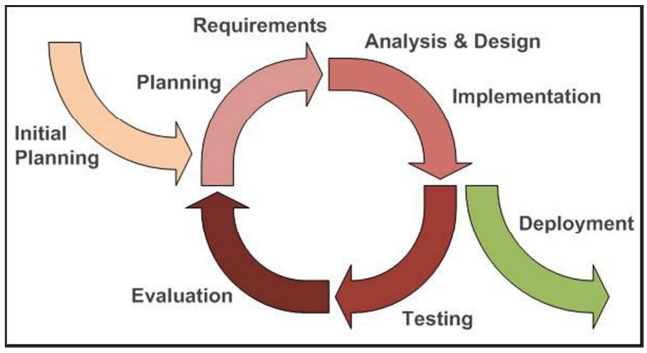

Figure 1. Iterative and Incremental Agile Development Process [1].

Agile development methodologies are lightweight processes that are based on small iterative phases or cycles. By this characteristic these methodologies dynamically occupy the users to create, prioritize, and investigate the requirements and also rely on implicit knowledge of team as opposed to documentation. The practice of agile methodologies is customer oriented, changing requirements, suitable for small development teams in projects in indistinct and lightweight software development model. While traditional software development methodologies such as waterfall model and spiral models usually known as heavyweight software development methodologies which we will discuss in upcoming section.

Agile software development methodologies emphasize rapid delivery to the customers. These methodologies are iterative based and perform incremental development of the software products, and on every successful delivery of the software product iteration, it also deliver the software product augmentation to the customer, therefore agile methodologies are fulfilling the customer satisfaction via before time and nonstop delivery of valuable software. While the lifecycle traditional development methodologies based on delivery of the software product only after the whole completion of the software development process and before that customers have no any accurate idea about the software product that is being developed.

The key variation among traditional development

methodologies and agile methodologies is the approval of change. This is the skill to reply to change that often decides the success and failure rate of the software project. The traditional methodologies do not allow change and freeze the functionality of the software product. Agile methodologies essentially welcomes the changing requirements also addition or elimination of characteristics throughout the development lifecycle. Since agile methodologies are iterative based so it is easy too accept requirement changes during the development of the software product. The framework of agile development allows both clients and software developers to change the requirement throughout the software project, on the other hand the authority is only in the hands of clients to accept, reject, and prioritize the ever changing needs.

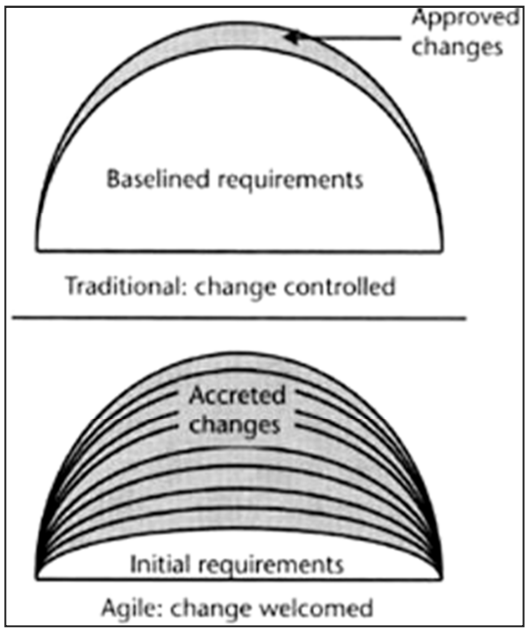

Figure 2. Requirement Change Management in Agile and Traditional Development Methodologies [1].

Agile development methodologies allow active customer participation and feedback too. The customers make participation and achieve higher priority in agile development techniques instead of any traditional approaches. The face to face communication among customers and development teams happen in agile development approach. The customers say warm welcome for active involvement in software projects as it permits to manage the project and process of development is much clear to customers, also they kept advanced. This customer participation reduces or overcome one of the most frequent issues on software projects: "What the customers will accept at the end of the software project varies from what customers told the developers at beginning". This participation of customers helps them to form a reasonable vision of the software product. Agile development methodologies also help to reduce cost and time as compared to the traditional development methodologies. Now we will discuss different methodologies in this section that falls in the category agile software development methodologies.

3.1. Extreme Programming (XP)

The best known first agile software development methodology is Extreme Programming (XP) (Beck 2000) that holds the property of fundamental process model for its parent approach that is agile development approach and it has been approved and adapted by its descendants. Additionally, the foundation of XP begun in 1990s when Kent Black tried to discover a enhanced approach of doing software development when he was managing a software project at Daimler Chrysler and this enhanced approach called XP methodology and confirmed to be a booming methodology. The main variation in this approach is that it focuses on adaptability instead of predictability and the reason behind this methodology is that the development of software is much flowing practice in which requirements cannot be completely predicted from starting but will forever change as projects progress. Therefore, it is essential to have a methodology in software development that is proficient to adapt to changing requirements at any stage of the software development life cycle. Kent Black in his experiments he found four dimensions which latterly became the philosophies of XP and if we implement these dimensions effectively then we can improve the processes of software project. These four dimensions are:

1. You need to improve your communication among stakeholders.
2. You need to look for simplicity.
3. You need to get feedback of customers and project manager on how effectively you are performing.
4. You need to constantly carry on with courage.

Even though, XP uses common practices that are commonly used in other development methodologies, XP goes further by implementing these common practices at extreme level and these practices are given below in the table.

Table 2. Why it's called eXtreme [6].

Good Practices	Pushed to Extreme
Code Reviews	Pair programming, code review all the time
Testing	Provide unit testing as well as functional (customer) testing
Design	Refactoring. Making it part of everyone's daily business
Simplicity	Always provide simplest design

Good Practices	Pushed to Extreme
Architecture	Metaphor, defining and refining architecture continuously
Integration Testing	Provide continuous integration
Short Iterations	Provide small iterations (the planning game)

XP provides 28 rules and best practices and can be packed together into simple twelve rules that are:

1. User Stories (Planning): User stories are like use cases but are not the similar as use cases. User stories are used for time estimation for the purpose of release planning meeting. This technique is helpful for customers because they define the specifications of new application and will be the baseline of the project team for management and cost estimation of the project as well.
2. Small Releases (Building Blocks): At early stages the XP team put a very simple system into the production, and perform implementation continuously on every small cycle.
3. Metaphor: The XP team often use a common set of names as well as common description of the system that is helpful for communication and development.
4. Collective Ownership: It means that whole code belongs to all XP programmers. This lets the XP team to perform rapid development, because when a change is required, the change is done without any delay.
5. Coding Standard: XP team perform development in pairs to share the ownership of the whole code, all XP programmers need to perform software development or coding in the same way, with particular rules that give guarantee about code communicates clearly.
6. Simple Design: The program that is developed with XP methodology should provide simplest design that fulfils the customer's current requirements means that always look for the implementation of the system which is simple and easy as possible yet covers all the required functionality of the system.
7. Refactoring: The application that is built with XP methodology should be consistently regulated and improved by all XP team members. For effective implementation good communication among XP team members is necessary to avoid duplication.
8. Testing: The main focus of XP team members is on validation of the application at all times means that every small release of the application must pass validation tests before being final release.
9. Pair Programming: XP programmers do their programming in pairs. All code implementation by two programmers is done on a single computer or machine but they work together. The key advantage of pair programming is better software at lower cost as compared to the programming working alone.
10. Continuous Integration: The application builds are completed number of times a day. This maintains the all XP programmers on the same page, and performs rapid development; also programmers can avoid the fragmentation since they perform continuous integration of the code together.
11. 40-hour Workweek: XP programmers work 40 hours only in a week because tired programmers make much errors and faults. XP team members do not work much overtime, which keeps the programmers fresh, active and healthy.
12. On-site Customers: The customers should be the essential part of the software project. The customers must be available all the times at development site because he/she realizes that the project progress is going on right track [6].

XP process or practices can be represented with the help of diagram given below:

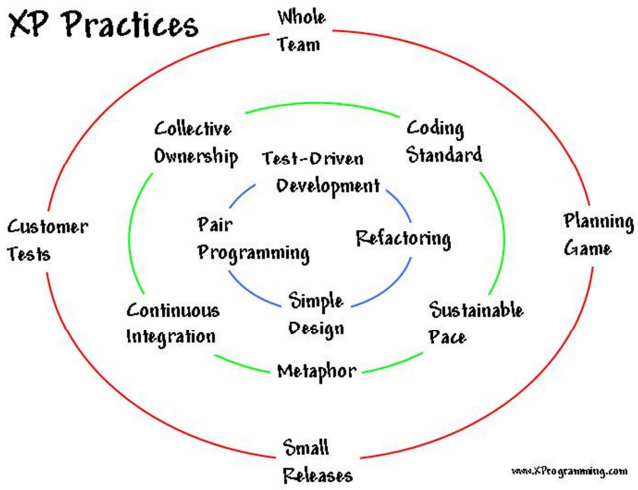

Figure 3. XP practices or process [7].

3.2. Scrum

Scrum belongs to the family of agile software development methodologies and it can be implemented to almost any project; however, this is often used in software development. For rapid changes in requirements or in highly emergent requirements, Scrum is much suitable for these conditions. It suggests every sprint starts with a concise planning meeting and also contains the review facility for better development of the software products. The model of this methodology proposes that the progress of the projects through a series of

sprints and these sprints are time boxed that are no longer than one month; these sprints are often of two weeks long.

Every team member of the Scrum methodology figure out that how many elements they can assign to for planning the meeting at the beginning of sprint, and after this a spring backlogging is maintained that is the list of activities to perform during each sprint. The Scrum team members take a small set of elements from initiative to code and functionality test during the sprint of Scrum methodology. Finally elements are done, meaning coded and functionality tested as well as integrated into the developing system. To synchronize the performance of team member the Scrum model visualizes the daily sprints as they communicate the performance of the sprint. Finally, The Scrum review is conducted by the Scrum team to expresses the new functionality to any other stakeholder who desires to offer the feedback that could affect the upcoming sprint [14].

The product itself is primary artefact in the Scrum development methodology. The Scrum model supposed the team members to get the product to a potentially shippable state after final completion of every sprint. Another artefact of Scrum methodology is product backlog. Product backlog provide the complete list of activities that stay to be additional to the software product. The owner of the product set priorities of the activities and team members perform their tasks according to the desired priorities of the product owner. Other additional artifacts of the agile development methodology are sprint burn down chart and release burn down chart. These charts show the remaining work of the sprints.

Scrum methodology also includes roles of different stakeholders or members related to the Scrum. One role is *ScrumMaster* and it perform the role of coach of the Scrum team, and responsible for to train the practitioners in order to achieve highest level of performance. The *ScrumMaster* is unlike the traditional project managers in number of ways in Scrum process. *ScrumMaster* does not permit day-to-day indications to the team members and does not commit task to the Scrum individuals. A good *ScrumMaster* allows the team to focus maniacally throughout the sprint on the objective they have chosen. On the other hand, the focus of the *ScrumMaster* to help the team members to do the best that they can be, the owner of the product works to guide the team members to the right objective. It is the duty of the *Product Owner* to prioritize the backlog throughout the Scrum development because he/she is the second role of the Scrum methodology that has its own importance in Scrum development. Another duty of the *Product Owner* is to guarantee it's up to balance as much is studied about the product to be built, the team, it users and so on. The *Scrum team* itself is final and main role of the Scrum methodology project management. Even though, the Scrum individuals may join the Scrum team with different job titles, and these titles are unimportant. This methodology describes that every member gives in what the approach they implement to complete the task of every sprint. This does not indicate that the tester will be probable to architect the system again; the Scrum individuals will use their time performing in what the regulation they performed before implementing the agile Scrum model. But with Scrum, the Scrum individuals are supposed to perform away from their favourite rules and regulations at any time performing so probable for the good quality of the Scrum team. One approach to supposition of the interlocking life of these three illustrated roles in the agile development methodology is as a race car. The team of Scrum methodology itself act like a car, ready to speedup the car in which direction you want. The product owner play the role of the driver of Scrum car, that make sure the he/she is driving the car in right direction. The ScrumMaster Play the role of chief mechanic that keeps the car well tuned and working at it best it can be [8].

3.3. Feature Driven Development (FDD)

Feature Driven Development (FDD) belongs to the family of agile methodologies and it is appeared in last 15 years as a substitute to the waterfall development methodology. This methodology is introduced by the Project Manager Jeff De Luca and Peter Coad 1997 and Jeff De Luca was a brilliant project manager of a software development organization of in Singapore. Jeff De Luca recognized that the given task is not possible to complete in time in the presence of available resources by using the traditional approach of software development because the problem domain was so much complicated and difficult. Jeff De Luca discovered the concept of FDD and modelling in color technique with the help of his partner Peter Coad. After discovering the concept of FDD and modelling in color technique Peter Coad publish his book "*Java Modeling in Color with UML*" in 1999.

FDD is an agile approach that is highly and short iterative, offers precise and significant improvement and status information with smallest overhead and interruption for software developers, delivers regular and concrete operational results at all stages, emphasizes product quality at all stages and is liked by customers, project managers and software developers.

While developing the software process, at each level communication is essential part of the project and no progress can be done without effective communication among clients, managers as well as developers. If we assume that the developers are communicating each other through communication channels then we can realize that the project is going on right track. As the size of the software product

increases the complexity also increased. We can say that the size of the software product is directly proportional to the complexity of the software product. FDD has the ability to decompose the larger problem in very small problems that can be addressed in a minimum time scenario, usually of 2 weeks. The decomposed problems which are independent to each other minimize the demand of communication. FDD also decomposes the project into small iterations so that space in time among test and analysis is minimized. If we discover the bugs earlier then we can minimize the cost of fixing bugs.

Different people have different ideas about the quality of the software product. An end user can talk about the software quality in terms of response time, software reliability, ease of use of the product. Software developers can talk about the quality of the software product in terms of quality parts of the design, ease of maintenance, ease of improvement and enhancement, patterns, compliance to standards, and meetings.

The project managers look the software product quality in sense of available resources, product delivery in time, delivery of quality product within budget and ease of maintenance and improvement. They also look at the product that how well it meets the user's requirements. Does this permit them to fulfil a regular business requirement and be proactive in meeting the risks that are always present in the business market? This makes essential to see the quality as a variety, with inner quality at one point and outer quality at other point. The concept of product quality in FDD is enlarged so as not to test the code only, but also contain the elements such as coding style, coding standard, measuring the audits and metrics in the code.

There are six key roles defined in FDD. First key role in FDD is of *Project Manager* who leads the project and is responsible for reporting progress, managing the budget and equipment, resources, and space, etc.

The key responsibility of overall designing the system is of *Chief Architect* (CA). His/her responsibility is to run the workshop design sessions in which the FDD team communicates in the system design. The facilitation, technical and modeling skills are required for work. CA pushes the project with the help of technical barriers tackling the project.

The day-to-day developing tasks are the duties of *Development Manager* (DM). It is the duty of DM to address daily conflicts for resources when chief programmers do not perform well among themselves to ease the role needing efficient technical skill.

The *Chief Programmers* are well trained and experienced software developers and they involve themself in designing the project activities, high level requirement analysis and also they are responsible for leading the team of three developers to six developers with the help of low level analysis, design and develop the novel software features.

Under the supervision of *Chief programmers*, *Class Owners* perform as developers of small teams; they design, code, test and specify the requirement features by the new systems.

Final role in FDD is of *Domain Experts*, they can be end users, business analysts, sponsors, or any mixture of these persons. They have effective knowledge about application and the developers depend on them to deliver the accurate system. They require efficient verbal, written and presentation skills. Their skills and presence in the project are extremely serious to the progress or success of the software being built.

There are some supporting roles in FDD, such *Language Guru* who is responsible for to be skilled in particular language. To setup, maintain and to run the consistent build process is the responsibility of *Build Engineer*. The responsibility of *Tool smith* to create the small development tools for software development teams, data conversion team and also for test teams. The responsibility of the System Administrator is to configure, manage, troubleshoot, and networking of the work stations particular to the project team. In additional roles the role of *Tester* is to verify the software product functions meets the user requirements or not. The *Deployers* convert the current data into the new design required by the new software system. The *Technical Writer* document and organize online and written user documentation [9].

3.4. Adaptive Software Development (ASD)

Adaptive Software Development (ASD) is an approach software development process that developed beyond rapid application development. It represents the principle that regular adaptation of the process to performance at hand is the common status of concerns. ASD has four phase's process model that are communication and planning, analysis, design and development, and testing and deployment. In the communication and planning phase, the project documentation and specification is take place, which are composed of probability and risk assessment, are organized. The analysis phase starts in the condition that the customer gives approval on the behalf of acceptance of first phase. The software quality will be agreed in analysis phase with the help of documentation. In this phase system analysts elicit detailed information and also user's requirements. In design and development phase ASD model make use of prototype approach to validate the design and development

requirements. The test cases for every increment are arranged at the start of the testing and deployment phase. Unit testing is carried out in this phase followed by the integration testing between different modules. After this the complete system test is carried out, followed by system acceptance test which is the final test to validate increment from client. In this phase the main deployment actions are installation, training and security [15].

ASD is the combination of three steps, Speculation, Collaboration and Learning, and each step is revolving around program coding. In the speculation phase the programmer try to understand the precise nature of the software product and customer's requirements. The speculation phase depends on bug and customer reporting to facilitate the project. If reporting is not provided by the customer then the software developers make use of fundamental requirements outlined by the customer. The collaboration phase happened or made when the individual software developers coagulate what they are every performing and how to attach their parts. There is no need of additional information and external input to the software developers to address this phase of the software. In the learning phase, the latest version of the product is released to the customers. This creates the bug and customer reporting used while speculation phase of the project, and cyclic repetition is done itself [10].

4. Traditional Development Methodologies

Traditional methodologies are often called heavyweight software development methodologies and these are based on in order sequence of steps, such as initial requirements definition, solution building, testing and employment. Traditional methodology requires defining and documenting an established requirements bundle at the start of the project. There are number of traditional development methodologies but in this paper I will discuss only two methodologies: Waterfall model and Spiral model.

4.1. Waterfall Model

The software engineers during the 1960s, were used the "code and fix" method. The software developer named Christophe Thibuat said that "one year for slamming code, one year for debugging". In 1970 the waterfall methodology was proposed by Winston Royce. There are defined phases in waterfall methodology that accentuate a structured series among these phases and these phases are consists of number of activities and deliverables. These activities and deliverables necessarily completed before the subsequent phase can start. The phases in waterfall methodology are named differently to each other and the main thought about first phase to capture that *what* the system will perform or do, customer's requirements related to the system and software. The second phase illustrates that how it will be implemented or designed. The third phase illustrates that from where the software developers start their coding. The fourth phase of waterfall methodology illustrates system testing. The fifth phase focused on task implementation like training and documentation. On the other hand, in engineering domain, the term waterfall is used as common name to all sequential development methodologies. The waterfall model lifecycle is as under:

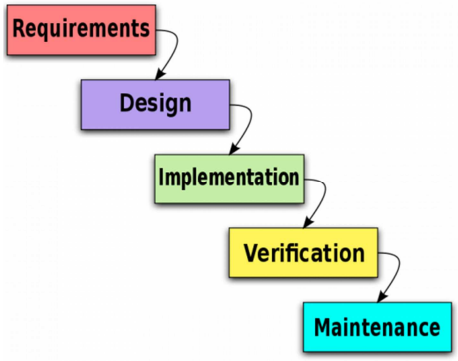

Figure 4. Waterfall Model Lifecycle [11].

4.2. Spiral Model

Spiral model also belongs to the category of traditional development methodologies and was defined by Barry Boehm that was based on experience with number of refinements of waterfall model as implemented to the large software projects. Spiral model combines the elements of design and prototyping-in-stages, in an attempt to together the benefits of top-down as well as bottom-up approaches. Spiral model consists of four main phases that are:

1. Determine Objective: In this phase the precise goals for the phase of project are recognized.
2. Identify and Resolve Risks: In this phase main risks are identified, analysed and information is collected to minimize the risks.
3. Development and Test: In this phase a precise model is selected for upcoming phase of software development.
4. Planning the next Iteration: In this phase the project review is take place and new plans are drawn up to the upcoming round of the spiral [2].

The Barry Boehm's spiral model is given below:

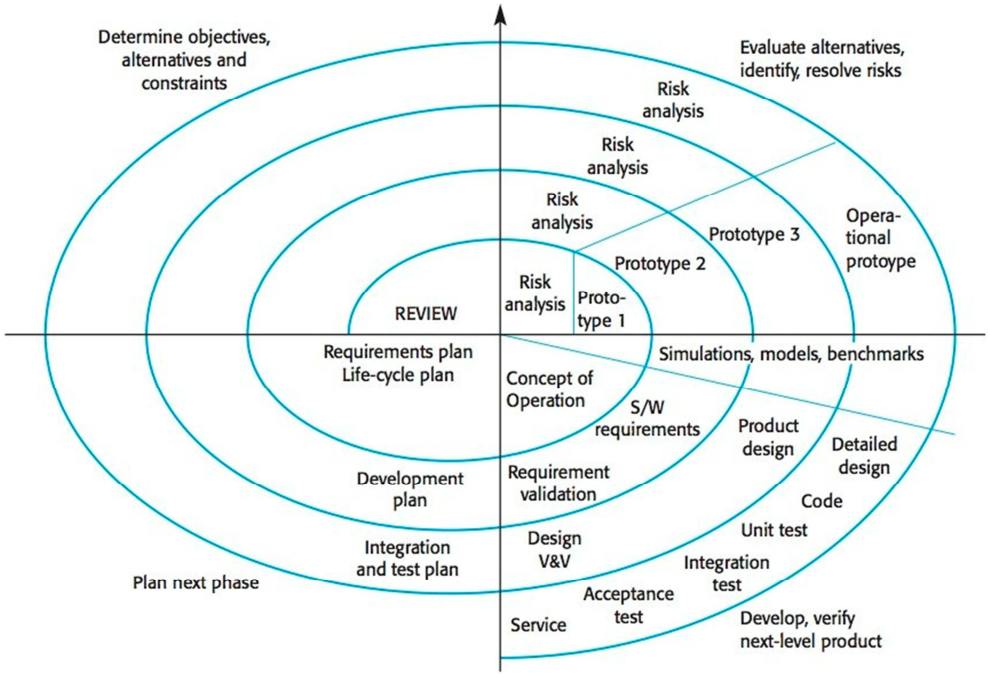

Figure 5. Barry Boehm's Spiral Model [12].

5. Conclusion

Agile methodologies are based on iterations and increments and these are nowadays much famous in the software development industries. Through, in number of software industries, at the adoption stage the agile development methodologies must begin to well-established processes of small, medium and large industries. There is growing requirement to have a reasonable knowledge of agile methodologies in using in the software development organizations; also the enough knowledge – the advantages of agile approaches for the purpose to accept the agile approaches into there software development techniques and for covering their active business requirements [1]. Traditional methodologies use predictive approach and phases instead of iterations. The traditional methodologies are process oriented as well as tool oriented. Comprehensive documentation is carried in traditional methodologies while in agile the main focus is on development instead of documentation.

References

[1] A B M Moniruzzaman and Dr Syed Akhter Hossain, "Comparative Study on Agile software development methodologies", Global Journal of Computer Science and Technology (c) Volume 13 Issue 7 Version I 12 Jul 2013.

[2] M. A. Awad, "A comparison between Agile and Traditional Software Development Methodologies", The University of Western Australia, 2005, pp. 2-47

[3] Outi Salo, "Enabling Software Process Improvement in Agile Software Development Teams and Organizations", University of Oulu, for public discussion in the Auditorium L10, Linnanmaa, January 12th, 2007, pp. 24-26

[4] Ngoc Tuan Nguyen, "How software process improvement standards and agile methods co-exist in software organizations?", Thesis M.Sc. Business Information Technology, Enschede, August 2010, pp. 18-21

[5] caslino, "Traditional Vs Agile Software Development", Feb 04, 2014 http://www.optimusinfo.com/blog/2014/02/04/traditional-vs-agile-software-development.html

[6] Wilfrid Hutagalung, "Extreme Programming", 2006 http://www.umsl.edu/~sauterv/analysis/f06Papers/Hutagalung/#waterfall

[7] http://cachesys.com/cachesys/methodology.html

[8] "Scrum Methodology" http://www.mountaingoatsoftware.com/agile/scrum

[9] Sadhna Goyal, "Feature Driven Development, Agile Techniques for Project Management and Software Engineering", Technical University Munich, 2008, pp. 3-7

[10] http://www.umsl.edu/~sauterv/analysis/6840_f09_papers/Nat/Agile.html#ASD

[11] Chong Chang, "Selecting an Appropriate Software Development Lifecycle (SDL) Model in an Agency Environment", 16 Aug 2012. http://www.metia.com/seattle/chong-chang/2012/08/sdl-model-in-an-agency-environment/

[12] http://leansoftwareengineering.com/2008/05/05/boehms-spiral-revisited/

[13] Bret Rudnick, "Agile Versus Traditional – A Tale of Two Methodologies", Sheraton System Architectures Workshop, 2013, pp. 8-16

[14] Booch Grady, "Object Solutions: Managing the Object-Oriented Project", Addison Wesley, 1995.

[15] Pittman Mathew. "Lessons Learned in Managing Object-Oriented Development", IEEE Software, January 1993, pp. 41-52.