Agile Software Development Methodology – Framework for Efficient & Quick Delivery of Software Product

Mihir Mehta[1] Rekha Shah[2]
[1,2]Lecturer
[1,2]Department of Computer Engineering
[1,2]GP Ahmedabad India

Abstract— Agile management, or agile process management, or simply agile refers to an iterative, incremental method of managing the design and build activities of engineering, information technology and other business areas that aim to provide new product or service development in a highly flexible and interactive manner. The paper will attempt to bring together concepts and practices of 'agile development', 'lean product development', 'choric systems', 'leadership studies' and concepts of the 'learning organization', together with 'the Model of Concurrent Perception' to suggest a new Paradigm of Software Development and Project Management. This paper also focuses on fundamentals of Agile Process, its characteristics, scope of agile process with its advantages and various methodologies for agile software development XP & Scrum.

Key words: Agile Software Development Methodology, Agile SDLC Model

I. INTRODUCTION

Agile software development methods have been developed and evolved since early 1990s. Due to the short development life cycle through an iterative and incremental process, the agile methods have been used widely in business sectors where requirements are relatively unstable. [1] Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Every iteration involves cross functional teams working simultaneously on various areas like planning, requirements analysis, design, coding, unit testing, and acceptance testing. At the end of the iteration a working product is displayed to the customer and important stakeholders. Agile method is a software development method that is people-focused commutations-oriented, flexible (ready to adapt to expected change at any time), speedy (encourage rapid and iterative development of the product in small releases), lean (focuses on shortening timeframe and cost and on improved quality), responsive (reacts appropriately to expected and unexpected changes), and learning (focuses on improvement during and after product development). [1]

In agile software development approach the tasks are divided to small time frames to deliver specific features for a release with in rigid and short time span. For each specific task iteration is taken and working software build is delivered after each iteration. Each build is incremental in terms of features; functionality and same step will be continuing till the required final product with all functionality; features we get. And the final build holds all the features required by the customer. Here graphical representation for agile software development approach is drawn for better understanding the proposed approach.

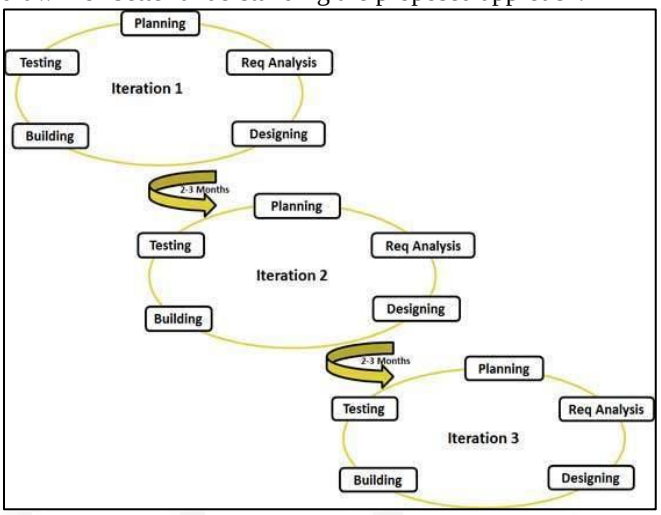

Fig. 1: Graphical Model for Agile Methodology

In February 2001, seventeen software developers met at the SnowBird resort in Utah to discuss lightweight development methods, among others Jeff Sutherland, Ken Schwaber, and Alistair Cockburn. And they had published the Manifesto for Agile Software Development.

They had focused and suggested following features of Agile Software Development:

- Individuals and interactions - in agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.
- Working software - Demo working software is considered the best means of communication with the customer to understand their requirement, instead of just depending on documentation. Because of small working software module communication becomes too easy and it will give clear idea to customer too at every increment about the required their features are perfectly matched or not.
- Customer collaboration - As the requirements cannot be gathered completely in the beginning of the project due to various factors, as in starting customers exactly sometimes can't explain about their fully desired features & functionality because of various reasons like poor technical knowledge. So continuous customer interaction is very important to get proper product requirements.
- Responding to change - agile development considers iterative approach for product development so at any stage of customer requirement changes then we can give him/her quick response to satisfy his/her requirement change.

II. AGILE PRINCIPAL & COMPARISON OF AGILE SOFTWARE DEVELOPMENT & TRADITIONAL SOFTWARE DEVELOPMENT

	Traditional development	Agile development
Fundamental hypothesis	Systems are fully specifiable, predictable and are developed through extended and detailed Planning.	High quality adaptive software is developed by small teams that use the principle of continuous improvement of design and testing based on fast feedback and change
Management style	Command and control	Leadership and collaboration
Communication	Formal	Informal
User requirements	Detailed and defined before coding/implementation	Interactive input
Cost of restart	High	Low
Testing	After coding is completed	After Every iteration
Client involvement	Low	High
Appropriate scale of the project	Large scale	Low and medium scale
Requirements	Very stable, known in advance	Emergent, with rapid changes
Size	Large teams and projects	Small teams and projects
Primary objectives	High safety	Quick value

Table 1: Comparison of Agile SDLC & Traditional SDLC [3]

A. Agile Principles

The Agile Alliance also documented the principles they follow that underlie their manifesto. As such the agile methods are principle-based, rather than rule-based. Rather than have predefined rules regarding the roles, relationships, and activities, the team and manager are guided by these principles [4]:

- 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 harness 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 time scale.
- Business people and developers must work together daily through 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.

III. EXAMPLE OF AGILE DEVELOPMENT METHODOLOGY

This section provides a brief introduction about two agile methodologies. The two were chosen to demonstrate the range of applicability and specification of the agile methodologies. For each methodology we provide an overview.

A. Extreme Programming (XP)

Extreme programming (XP) is a software development methodology which is intended to improve software quality and responsiveness to changing customer requirements [2]. As a type of agile software development, Other elements of extreme programming include: programming in pairs or doing extensive code review, unit testing of all code, avoiding programming of features until they are actually needed, a flat management structure, code simplicity and clarity, expecting changes in the customer's requirements as time passes and the problem is better understood, and frequent communication with the customer and among programmers. It advocates frequent "releases" in short development cycles, which is intended to improve productivity and introduce checkpoints at which new customer requirements can be adopted. Extreme Programming (XP) [6] originators aimed at developing a methodology suitable for "object-oriented projects using teams of a 12 or fewer programmers in one location." [3]. The methodology is based upon five basic principles: communication, simplicity, feedback, courage, and respect.

- Communication. XP has a culture of oral communication and its practices are designed to encourage interaction. The communication value is based on the observation that most project difficulties occur because someone *should have* spoken with someone else to clarify a question, collaborate, or obtain help. "Problems with projects can invariably be traced back to somebody not talking to somebody else about something important." [3]
- Simplicity. Design the simplest product that meets the customer's needs. An important aspect of the value is to only design and code what is in the current requirements rather than to anticipate and plan for unstated requirements.
- Feedback. The development team obtains feedback from the customers at the end of each iteration and external release. This feedback drives the next iteration. Additionally, there are very short design and implementation feedback loops built into the

methodology via pair programming and test-driven development [3].

- Courage. The other three values allow the team to have courage in its actions and decision making. For example, the development team might have the courage to resist pressure to make unrealistic commitments.
- Respect. Team members need to care about each other and about the project.

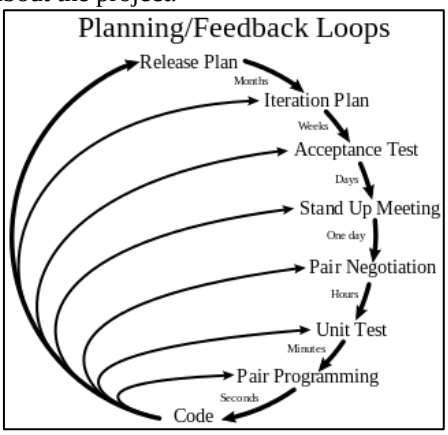

Fig. 2: Graphical Model For Extreme Programming

B. Scrum

Scrum is an iterative and incremental agile software development framework for managing product development. It defines "a flexible, holistic product development strategy where a development team works as a unit to reach a common goal", challenges assumptions of the "traditional, sequential approach" to product development, and enables teams to self-organize by encouraging physical co-location or close online collaboration of all team members, as well as daily face-to-face communication among all team members and disciplines involved [2].

A key principle of Scrum is its recognition that during product development, the customers can change their minds about what they want and need and that unpredicted challenges cannot be easily addressed in a traditional predictive or planned manner. As such, Scrum adopts an evidence-based empirical approach —accepting that the problem cannot be fully understood or defined, focusing instead on maximizing the team's ability to deliver quickly, to respond to emerging requirements and to adapt to evolving technologies and changes in market conditions.

1) Scrum vs Extreme Programming [6]

- Scrum teams typically work in iterations (called sprints) that are from two weeks to one month long. XP teams typically work in iterations that are one or two weeks long.
- Scrum teams do not allow changes into their sprints. Once the sprint planning meeting is completed and a commitment made to deliver a set of product backlog items, that set of items remains unchanged through the end of the sprint. XP teams are much more amenable to change within their iterations. As long as the team hasn't started work on a particular feature, a new feature of equivalent size can be swapped into the XP team's iteration in exchange for the un-started feature.
- Extreme Programming teams work in a strict priority order. Features to be developed are prioritized by the customer and the team is required to work on them in that order. By contrast, the Scrum product owner prioritizes the product backlog but the team determines the sequence in which they will develop the backlog items.

IV. CONCLUSION

Agile software development describes a set of principles for software development under which requirements and solutions evolve through the collaborative effort of self-organizing cross-functional teams. It advocates adaptive planning, evolutionary development, early delivery, and continuous improvement, and it encourages rapid and flexible response to change. In Section I we have explained in detail with graphical diagram about working of Agile paradigm. Section II gives idea about comparisons of Agile Vs. Traditional Software development Paradigm. Where as in Section III we have presented overview about various examples (frameworks) for Agile Methodology.

REFERENCES

[1] Manish Kumar, Santosh Kumar Singh, Dr. R. K. Dwivedi, "A Detail Study of Agile Software Development with Extreme Programming", IJARCSSE Vol. 5 Issue 10, Pg. Num 719-725.
[2] Lauri Williams, "A survey of Agile Development Methodologies", 2007.
[3] BODJE N'Kauh Nathan-Regis and Dr G.M Nasira, "Agile Methods and Quality A Survey" CS & IT-CSCP 2013, PG. Num 167-182.
[4] Roy Morein, "An Agile Software Project Management Manifesto-A Reference Disicipline framework for Agile Development" Int. J. Advance Soft Compu. Appl, Vol. 6, No. 1, March 2014.
[5] Kuda Nageswara Rao, G. Kavita Naidu, Praneeth Chakka ,"A Study of the Agile Software Development Methods, Applicability and Implications in Industry" International Journal of Software Engineering and Its Applications Vol. 5 No. 2, April, 2011
[6] Marin Stocia, "Sofyware Development: Agile vs Traditional" Informatics Economical Vol. 17, n0. 4/2013.

