ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 9, September 2016

A Study on Agile Methodologies

Anupama Kaushik

Assistant Professor, Department of IT, Maharaja Surajmal Institute of Technology, New Delhi, India

Abstract: Now-a-days the software development environment is very challenging. Organizations are constantly changing their software requirements to adjust with new environment. They have also molded themselves to satisfy the demand for fast delivery of software products as well as for accepting changing requirements. In this scenario, traditional software development methods fail to meet up these requirements. Though traditional software development methodologies, such as life cycle models and object oriented approaches, continue to dominate the systems development few decades. Agile software development brings its own set of novel challenges that must be addressed to satisfy the customer through early and continuous delivery of the valuable software. This paper discusses three of the famous agile software development methodologies i.e. Extreme Programming, SCRUM and Feature Driven Development.

Keywords: Traditional software development; Agile software development; Extreme Programming; SCRUM; Feature driven development.

I. INTRODUCTION

A few decades back traditional software development dominated the software industry. But now-a-days due to ever changing customer demand and changing technologies, these traditional software development fails the customer satisfaction. The issue of how software development should be done in order to deliver faster, better, cheaper and customer satisfactory solutions have been discussed in software engineering circles for decades. There have been many studies and suggestion in improving the development process. Recently, this has paved a way to a new software development method called Agile Software Development. This paper discusses the migration towards agile movement and reviews the existing agile methodologies.

II. BACKGROUND

Agile software development (ASD) is a major paradigm, in field of software engineering which has been widely adopted by the industry, and much research, publications have conducted on agile development methodologies over the past decade. Its various principles have emerged from the traditional software development principles and various experiences based on the successes and failures of software projects.

Agile Software Development emerged in February 2001 when a group of software consultants signed the Agile Software Development Manifesto.

This agile manifesto [1] states the main focus of the agile development as the following:

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

The previous four values have been further defined by twelve principles: [2]

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

Agile methods focus on the challenges of unpredictability of the real world by relying on people and their creativity rather than processes [3]. The main theme in agile

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 9, September 2016

methods is to promote and speed up responses to changing environments, requirements and meeting the deadlines.
There are a number of agile software development methods. Methods for agile software development represent a set of practices for software development that have been created by experienced people [3]. The most common methods are extreme Programming (XP) [4], Dynamic Software Development Method (DSDM) [5], Scrum [6], Crystal [7] and Feature Driven Development (FDD) [8].
All these methods focus on customer satisfaction through continues delivery of software. This is achieved by having short iterations in the development process. The iterations focus on timely delivery of working code that provides substantial value to the customer.

III.AGILE PROCESSES

Agile Software Development Methodology is currently widely in use due to its characteristics of rapid software development and accommodation to changing requirement at any level of development [9].

A. Extreme Programming (XP) [11]:

Extreme Programming (XP) has evolved from the problems caused by the long development cycles of traditional development models [10]. The life cycle of XP consists of five phases: Exploration, Planning, Iterations to Release, Productionizing, Maintenance and Death as shown in Fig. 1 [12].

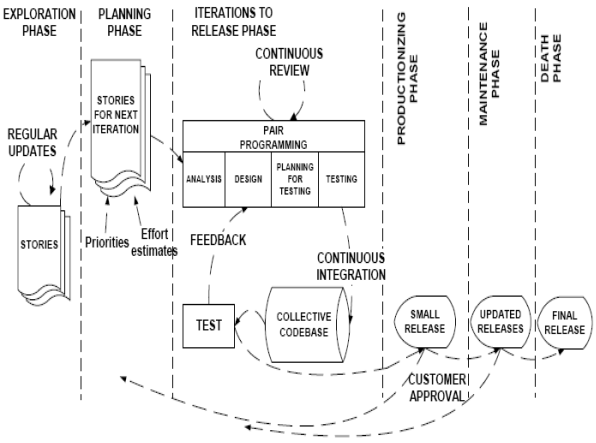

Fig.1 Extreme programming

In the **Exploration phase,** the customers write out the story cards that they wish to be included in the first release. Each story card describes a feature to be added into the program. At the same time the project team familiarize themselves with the tools, technology and practices they will be using in the project. The technology to be used will be tested and the architecture possibilities for the system are explored by building a prototype of the system. The exploration phase takes between a few weeks to a few months, depending largely on how familiar the technology is to the programmers.

The **Planning phase** sets the priority order for the stories and an agreement of the contents of the first small release is made. The programmers first estimate how much effort each story requires and the schedule is then agreed upon. The time span of the schedule of the first release does not normally exceed two months. The planning phase itself takes a couple of days.

The **Iterations to release** phase includes several iterations of the systems before the first release. The schedule set in the planning stage is broken down to a number of iterations and each take one to four weeks to implement. The first iteration creates a system with the architecture of the whole system. This is achieved by selecting the stories that will enforce building the structure for the whole system. The customer decides the stories to be selected for each iteration. The functional tests created by the customer are run at the end of every iteration. At the end of the last iteration the system is ready for production.

The **Productionizing phase** requires extra testing and checking of the performance of the system before the system can be released to the customer. At this phase, the team may find new changes and the decision will then be made if they should be included in the current release. During this phase, the iterations may take from three weeks to one week. The postponed ideas and suggestions are documented for later implementation during the maintenance phase.

After the first release is productionized for customer use, the XP project must both keep the system in the production running while also producing new iterations. In order to do this, the **Maintenance phase** requires an effort also for customer support tasks. Thus, the development velocity may decelerate after the system is in production. The maintenance phase may require incorporating new people into the team and changing the team structure.

The **Death phase** is when the customer needs are all satisfied and no more stories are left to be implemented. This is the time in the XP process when the necessary documentation of the system is finally done and no more changes to the architecture, design or code are made. Death may also occur if the system is not delivering the desired outcomes, or if it becomes too expensive for further development.

B. SCRUM [11]

The term 'scrum' originally derives from a strategy in the game of rugby where it denotes "getting an out-of play ball back into the game" with teamwork [13].
The Scrum approach has been developed for managing the systems development process. It is an empirical approach applying the ideas of industrial process control theory to systems development resulting in an approach that reintroduces the ideas of flexibility, adaptability and productivity [13]. It does not define any specific software development techniques for the implementation phase.

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 9, September 2016

Scrum concentrates on how the team members should function in order to produce the system flexibly in a constantly changing environment. Scrum process includes three phases: pre-game, development and post-game as shown in Fig.2.

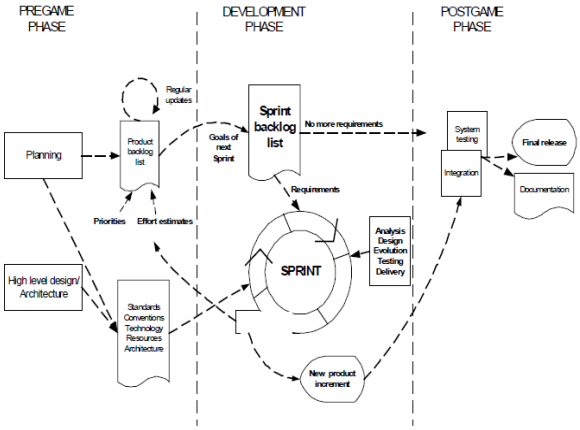

Fig. 2 Scrum Process

In the following, the Scrum phases are introduced according to Schwaber [11] [13] [14].

The pre-game phase includes two sub-phases: Planning and Architecture/High level design.
Planning includes the definition of the system being developed, about the project team, tools and other resources, risk assessment and controlling issues, training needs and verification management approval. A **Product** Backlog list is created containing all the requirements that are currently known. The requirements can originate from the customer, sales and marketing division, customer support or software developers. The requirements are prioritized and the effort needed for their implementation is estimated. The product Backlog list is constantly updated with new and more detailed items, as well as with more accurate estimations and new priority orders. At every iteration, the updated product Backlog is reviewed by the Scrum Team(s) so as to gain their commitment for the next iteration.

In the architecture phase, the high level design of the system including the architecture is planned based on the current items in the Product Backlog. In case of an enhancement to an existing system, the changes needed for implementing the Backlog items are identified along with the problems they may cause. A design review meeting is held to go over the proposals for the implementation and decisions are made on the basis of this review. In addition, preliminary plans for the contents of releases are prepared.

The development phase (also called the game phase) is the agile part of the Scrum approach. This phase is treated as a "black box" where the unpredictable is expected. The different environmental and technical variables (such as time frame, quality, requirements, resources, implementation technologies and tools, and even development methods) identified in Scrum, which may change during the process, are observed and controlled through various Scrum practices. Rather than taking these matters into consideration only at the beginning of the software development project, Scrum aims at controlling them constantly in order to be able to flexibly adapt to the changes.

C. Feature Driven Development (FDD)[11]
Feature Driven Development (FDD) is an agile and adaptive approach for developing systems. The FDD approach does not cover the entire software development process, but rather focuses on the design and building phases [8]. However, it has been designed to work with the other activities of a software development project [8] and does not require any specific process model to be used. The FDD approach embodies iterative development with the best practices found to be effective in industry. It emphasis quality aspects throughout the process and includes frequent and tangible deliveries, along with accurate monitoring of the progress of the project.

FDD consists of five sequential processes and provides the methods, techniques and guidelines needed by the project stakeholders to deliver the system. Furthermore, FDD includes the roles, artifacts, goals, and timelines needed in a project [8]. Unlike some other agile methodologies, FDD claims to be suitable for the development of critical systems [8].

FDD consists of five sequential processes as shown in Fig. 3.The iterative part of the FDD processes i.e. design and build supports agile development with quick adaptations to late changes in requirements and business needs. Typically, an iteration of a feature involves a one to three week period of work for the team.

All the five processes of FDD is described according to Palmer [8].

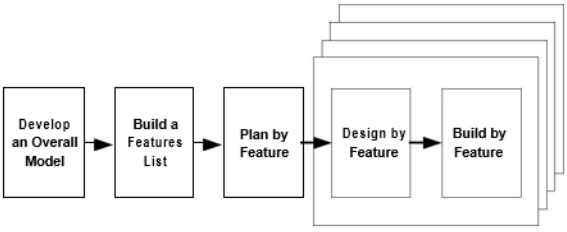

Fig 3. Processes of FDD

Develop an Overall Model
When the development of an overall model begins, the domain experts are already aware of the scope, context and requirements of the system to be built [8]. The documented requirements such as use cases or functional specifications already exist at this stage. However, FDD does not explicitly address the issue of gathering and managing the requirements. The domain experts present a so called "walkthrough" in which the team members and

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 9, September 2016

the chief architect are informed of the high-level description of the system. The overall domain is further divided into different domain areas and a more detailed walkthrough is held for each of them by the domain members. After each walkthrough, a development team works in small groups in order to produce object models for the domain area at hand. The development team then discusses and decides upon the appropriate object models for each of the domain areas. Simultaneously, an overall model shape is constructed for the system [8].

Build a Features List

The walkthroughs, object models and existing requirement documentation give a good basis for building a comprehensive features list for the system being developed. In the list, the development team presents each of the **client valued functions** included in the system. The functions are presented for each of the domain areas and these function groups consist of so-called major feature sets. In addition, the major feature sets are further divided into feature sets. These represent different activities within specific domain areas. The feature list is reviewed by the users and sponsors of the system for their validity and completeness.

Plan by Feature

Planning by feature includes the creation of a high-level plan, in which the feature sets are sequenced according to their priority and dependencies and assigned to Chief Programmers. Furthermore, the classes identified in the "developing of an overall model" process are assigned to individual developers, i.e. class owners. Also schedule and major milestones may be set for the feature sets.

Design by Feature and Build by Feature

A small group of features is selected from the feature set(s) and class owners form the feature teams needed for developing the selected features. All the selected features are produced by the design by feature and build by feature iterative processes. In this one iteration takes from a few days to a maximum of two weeks. There can be multiple feature teams concurrently designing and building their own set of features which include tasks such as design inspection, coding, unit testing, integration and code inspection for different features. After a successful iteration, the completed features are promoted to the main build while the iteration of designing and building begins with a new group of features taken from the feature set.

IV. CONCLUSION

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 flexibility. In this paper a study on three of the commonly used agile methodologies i.e. Extreme Programming, SCRUM and Feature Driven Development is done.

ACKNOWLEDGMENT

The author is thankful to all the authors whose references are included and especially to **Pekka Abrahamsson, Outi Salo and Jussi Ronkainen.**

REFERENCES

[1] Information and Software Technology, 2008, vol. 50, pp. 833-859.
[2] Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Humt, A., Jerries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J., Thom, D.: Manifesto for agile software development. Website (2001) http://agilemanifesto.org/.
[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] T. Dyba, "Improvisation in small software organizations", IEEE Software, Vol.17, No. 5, pp. 82–87, 2000.
[5] P. A .Gerfalk, B. Fitzgerald, "Flexible and distributed software processes: old petunias in new bowls?", Communications of the ACM , Vol. 49, No. 10, pp. 27–34, 2006.
[6] S. Nerur, R. Mahapatra, G. Mangalaraj, "Challenges of Migrating to Agile Methodologies", Communications of the ACM,Vol. 48, No. 5, pp. 72–78, 2005.
[7] Schwaber K. and Beedle M., Agile Software Development with Scrum. Prentice Hall, 2001.
[8] Palmer, S. R. and Felsing, J. M., A Practical Guide to Feature-Driven Development. Upper Saddle River, NJ, Prentice-Hall, 2002.
[9] M. Aoyama, " Web-based agile software development", IEEE Software, Vol. 15 , No. 6 ,pp. 56–65 , 1998.
[10] Dyba T., and Dingsoyr T., "Empirical Studies and Agile Software Development: A Systematic Review", Information and Software Technology, 2008, Vol. 50, pp. 833-859.
[11] Pekka Abrahamsson, Outi Salo & Jussi Ronkainen,, Agile software development methods, Review and Analysis, VTT publications 478.
[12] Beck K. "Embracing change with Extreme programming", IEEE computer, Vol. 32, No.10, pp. 70-77, 1999a.
[13] Schwaber, K. and Beedle, M., Agile Software Development with Scrum. Upper Saddle River, NJ, Prentice-Hall, 2002.
[14] Schwaber, K. Scrum Development Process. OOPSLA'95 Workshop on Business Object Design and Implementation. Springer-Verlag, 1995.