IJCSC Volume 4 • Number 2 September 2013 pp.153-156 ISSN-0973-7391

Agile Software Development

Rashmi Popli, Dr. Naresh Chauhan
Assistant Professor, YMCA University of Science & Technology
Professor, YMCA University of Science & Technology
rashmimukhija@gmail.com

Abstract: Agile software development represents a major departure from traditional methods of software engineering. It had a huge impact on how software is developed worldwide. Agile software development solutions are targeted at enhancing work at project level. Organizations worldwide are adopting agile software development methods at increasing speed. Little is still known about the current usefulness of agile methods in the complex environment. So lot of research work in area of Agile is still needed. The goal of this survey is to provide first-hand knowledge of the difference between Traditional and Agile methods and to describe various areas of Agile in which research is required. This paper serves as a starting point for creating a common research agenda and enables the generation of more fruitful research results from the field.
Keywords: Agile Software Development (ASD), Extreme Programming (XP)

1. Introduction

Agile Software Development is currently an emerging discipline of Software Engineering, constituting a set of principles initially advocated by a group of seventeen software practitioners, and now practiced by many software professionals. Agile software development methods can be defined as using human- and communication oriented rules in conjunction with traditional methods [1]. These four rules are: individuals and human interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan [2].

Agile Software Development (ASD) - as opposed to traditional, plan-centric software development – refers to a range of lightweight development approaches tailored to:
1) facilitate faster time-to-market and continual integration of new requirements;
2) increase development productivity while maintaining software quality and flexibility; 3) increase the organization's responsiveness while decreasing development overhead[2,3].
Agile methods break tasks into small increments with minimal planning, and don't directly involve long-term planning. Iterations are short time frames that typically last from one to four weeks[5].Each iteration is worked on by a team through a full software development cycle including planning, requirements analysis, design, coding, unit testing, and acceptance testing when a working product is demonstrated to stakeholders. This helps minimize overall risk, and lets the project adapt to changes quickly. Stakeholders produce documentation as required. The Agile life cycle is as shown in Fig. 1.

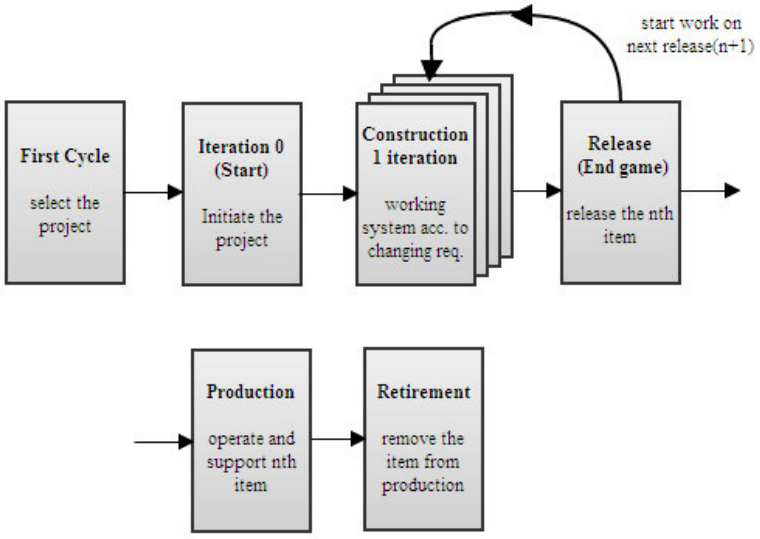

Figure 1: Agile Life Cycle

Agile methods such as XP and Scrum can be viewed as a reaction to plan-based traditional methods which emphasize a "rationalized, engineering-based approach", incorporating extensive planning, codified processes and rigorous reuse.

2. Working with Traditional and Agile Software Development

In the case of a normal heavy weight traditional software Management & development process, the activities will be done as

1. Meet with the customers.
2. Find out the processes required.
3. Get sign-off on the requirements to ensure that the customer not changes their minds.
4. Create a detailed project plan of the entire project, and assign resources to tasks.
5. When project progresses, different individuals working on different pieces may contact customer with similar or different questions. Schedule problems can be addressed by shortening future tasks, like testing.
6. When the project is completed then customer may request many modifications. Project becomes a maintenance project rather than a development project.

In the case of working with agile software development, the activities will be done as

1. Meet with the customer; create a high level list of features get sign-off on the requirements.
2. Chunk these into features that could be delivered in six-week intervals.
3. Ask the customer to prioritize the list.
4. Create a detailed plan to implement the first feature including a detailed user acceptance test case.
5. Implement the plan for the first feature. Deliver the feature to the customer, involving them with questions whenever necessary.
6. First feature is delivered, and approved by the customer. The time it took to deliver is used for a baseline to predict the sizing of future features. The customer is asked again to reprioritize, and pick the next feature to be delivered in six weeks based on the new estimates, Repeat steps 3, redoing existing features as necessary to add new features until the entire solution is in place.

Traditional methodologies try to be predictive - to create a schedule at the beginning of a project and to conform to this schedule for the life of the project. The reason for failure of the traditional projects is that the users keep changing their minds. Agile software development methods are attempting to solve these problems by incorporating changes even in later stages. Table 1 summarizes the difference between traditional and agile perspective.

Table 1: Traditional and Agile Perspective of Software Development

Task	Traditional Perspective	Agile Perspective
Design Process	**Linear sequence of steps**	**Iterative and exploratory**
Type of environment	**Stable, predictable**	**Turbulent, difficult to predict**
Goal	**Optimization, low tolerance to changes**	**Flexibility, High tolerance to changes**
Development Principle	**Development is based on fixed order**	**Principle of freer co-operation of development team**
Problem-solving Process	**Selection of best means to a given and through well planned and formalized activities**	**Learning through experimentation and introspection**
Type of learning	**Single-loop/adaptive**	**Double-loop/generative**
Type of management	**Directive management**	**Emphasis on team communication**
Team-size	**Large teams**	**Small teams(2-10 developers)**
Customer role	**Customer role is only at the initial and final stages of the project**	**Customer is involved at each and every stage**
Testing	**Testing is done at the end of development**	**Testing is done throughout the course of development**

3. Benefits and Problems with Agile Development

Agile Software development has had a deep impact on the software industry in recent years. Serious limitations have also been identified in Agile. The benefits of Agile Software Development are significant and include:

1. Software development life-cycle time of a project is reduced up to 75%.
2. Work-loads of team members are stable.
3. Software systems are developed with a fixed number of developers.
4. ASD is highly flexible to change of Management & development plans
5. Quality of the project is improved by earlier feedback from the customer.

The limitations of Agile Management methods are:

1. It is hard for upper-management to understand exactly where the project stands. Due to the various iteration steps, it can be hard to understand if the project is on track.
2. Agile management methods do not handle large teams well. The approach only works for small to medium-sized teams.
3. Agile development requires highly skilled and highly motivated individuals, which may not always be existent.

4. Related Work

In this section, we provide a brief review of the relevant research, identifying some key Agile Publications. The first review of the existing literature on agile software development was done in a technical report published by Abrahamsson et al. at VTT in 2002 [7]. The report discusses the concept of agile development, presents processes, roles, practices, and experience with 10 agile development methods, and compares the methods with respect to the phases that they support and the level of competence that they require. Abrahamsson et al. provide evidence that agile methods are ‘‘effective and suitable for many situations and environments”,

Alan MacCormack [8, 9] argued that the stage-gate model (sequentially doing one phase of development at a time, i.e., the waterfall model) does not work well when the market or technology advances at a faster pace than the project can respond. In those cases where speed is important, an agile approach provides a more flexible means of responding to change. MacCormack studied over 160 software projects at several major software companies. His major findings showed that success in using Agile depends on multiple, early releases to customers, rapid feedback to the development team, and a software architecture that allows independence of component teams. He also found that market uncertainty produced late changes (regardless of whether Agile or waterfall was used).

Larman[10] reviewed the problems with the waterfall approaches in his classic book on Agile targeted at managers, identifying a number of issues which are summarized here:

1. Waterfall works best for projects with little change, little novelty, and low complexity
2. Waterfall pushes high-risk and difficult elements to end of the project
3. Waterfall aggravates complexity overload
4. Waterfall is poorly suited to deal with changing requirements
5. Waterfall encourages late integration
6. Waterfall produces unreliable up-front schedules and estimates

Williams et al. [11] investigated the usage of a subset of XP practices at a group in IBM. The product developed at IBM using XP was found to have significantly better pre-release and post-release quality compared to an older release. The teams using XP reported an improvement in productivity and morale. In addition, customers were more satisfied with the product developed using XP because the teams delivered more than what the customers had originally asked for.

In 2005, Erickson et al. [12] described the state of research on XP, agile software development, and agile modeling. With respect to XP, they found a small number of case studies and experience reports that promote the success of XP. The XP practice of pair programming is supported by amore well-established stream of research, and there are some studies on iterative development.

5. Research Areas in Agile

Organizations worldwide are adopting agile software development methods at increasing speed. These methods are effective and suitable, but empirical studies are scarce. Little is still known about the current usefulness of agile methods in the

complex environment. So lot of research work in area of Agile is still needed. Instead of inventing a new method proposing a model for selection is more required. Various areas of research in Agile are:

Pair Programming: Pair programming, or areas that connect well to existing streams of software engineering research, might be described as being at an intermediate, or even a mature, state.

Estimation: The agile managers use many different methods to estimate and schedule their work efforts. The organizations use different methods depending on the type of projects, the inherent risks in the project, the technologies involved etc. Most of the time, effort estimations are clubbed with the development estimates and no separate figures are available. In a complex environment with frequent changes that cannot be anticipated, estimation must be done in an incremental and adaptive way.

Tools and resources: To improve communication, quality and scheduling and estimation, tools and resources are needed to be developed.

Scaling: In prior reported results on the use of ASD, teams have typically between 2-20 members. Teams at Microsoft and other companies can be much larger, between 500-5000. How Agile can be adapted to work for these large teams is need to be investigated.

Coordination: In large companies Agile is not adopted simultaneously by all teams. How Agile teams coordinate dependencies and deliverable with non-agile teams.

Empirical body of knowledge: An empirical body of knowledge about the various facets of ASD is needed to be developed to replicate various studies about ASD in industry and academia.

6.Conclusion

In this study the concept of Agile software development as a new approach to Software development is explored. The need for an empirical research framework for agile methods has been identified. In this paper Agile and traditional methods are compared. It has been find that agile can be useful for small teams of domain experts, who are able to communicate well with customers and are very good designers and implementers. In future work more qualitative studies with an explicit focus on issues and problems need to be conducted

References

[1] Alistair Cockburn, “Agile Software Development”, Pearson Education, 2002.
[2] Beck, “K. Manifesto for Agile Software Development”, WWW page of the Agile Manifesto: **http: //agilemanifesto.org/.** Oct 2007.
[3] Beck, “K. Principles of the Agile Manifesto”, WWW page of the Principles of the Agile Manifesto: **http://agilemanifesto.org/principles.html**., Oct 2007.
[4] Pekka Abrahmson, Outi Salo, Jussi Ronkainen, Juhani Warsta, “Agile Software Development Methods”, VTT Publications 478, ESPOO 2002.
[5] Scott W.Ambler, “The Agile System Development Life Cycle”, Ambysoft, Managing Agile Projects, 2005
[6] B. Boehm “Get Ready for Agile Methods, with Care,” Computer, vol. 35, no. 1, 2002, pp. 64–69.
[7] P. Abrahamsson, O. Salo, J. Ronkainen, J. Warsta, , “Agile software development methods: review and analysis” VTT Technical report, 2002.
[8] MacCormack, Alan, “Agile Software Development: Evidence from the Field”, June 2003,” Agile Development Conference.
[9] MacCormack, Alan,"Product-Development Practices That Work: How Internet Companies Build Software." Winter 2001, Sloan Management Review, v42, n2, pp. 75-84.
[10] Larman, Craig, “Agile & Iterative Development: A Manager’s Guide” 2004, Addison-Wesley, pp. 58-62.
[11] L. Williams, W. Krebs, L. Layman, A. Antón, and P. Abrahamson , "Toward a Framework for Evaluating Extreme Programming",Proceeding of Empirical Assessment in Software Eng. (EASE) 2004, Edinburgh, Scot., pp. 11-20, 2004.
[12] J. Erickson, K. Lyytinen, K. Siau, “Agile modeling, Agile software development, and extreme programming: the state of research”, Journal of Database Management 16(4) (2005) 88–100.

Agile software development is a modern project management method that breaks a large project into smaller chunks to receive customer feedback at each stage. This helps customers experience the product at different stages and share their impressions and inputs. The result is a final product that customers truly love! Companies follow agile software development methodology to accelerate software delivery yet be flexible to changes. According to GoodFirms, 61.5% of companies follow Agile methodology because it lets them change priorities fast. In this article, weâ€™re talking about Agile software development methodologies and practices. Youâ€™ll also learn what phases the Agile life cycle consists of when it comes to software development. Table of contents. Show all. What Is Agile Software Development? Benefits of Agile. What emerged was the Agile Software Development Manifesto. Representatives from Extreme Programming, SCRUM, DSDM, Adaptive Software Development, Crystal, Feature-Driven Development, Pragmatic Programming, and others sympathetic to the need for an alternative to documentation driven, heavyweight software development processes convened.Â In traditional software development processes, important decisions are made far in advance of implementation.