The International Journal Of Engineering And Science (IJES)
||Volume|| 3 ||Issue|| 01 || Pages || 36-40 || 2014||
ISSN (e): 2319 – 1813 ISSN (p): 2319 – 1805

Agile Software Development: Novel Approaches For Software Engineering

Sk. Safikul Alam[1] ,Sourabh Chandra[1]
Department of Computer Science & Engineering, Calcutta Institute of Technology, Uluberia, Howrah-711316, West Bengal, India

----------ABSTRACT ----------
Incremental software development methods have been traced back to 1957. In 1974, it was introduced an adaptive software development process. So-called "lightweight" software development methods evolved in the mid-1990s as a reaction against "heavyweight" methods, which were characterized by their critics as a heavily regulated, regimented, micromanaged, waterfall model of development. Proponents of lightweight methods (and now "agile" methods) contend that they are a return to development practices from early in the history of software development. Early implementations of lightweight methods include Scrum (1995), Crystal Clear, Extreme Programming (1996), Adaptive Software Development, Feature Driven Development, and Dynamic Systems Development Method (DSDM) (1995). These are now typically referred to as agile methodologies, after the Agile Manifesto published in 2001.

KEYWORDS: *Agile, Lightweight, Heavyweight, Regimented, Micromanaged, Waterfall model.*

Date of Submission: 05 December 2013 Date of Acceptance: 25 January 2014

I. INTRODUCTION

Agile software development is a group of software development methods based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development and delivery, a time-boxed iterative approach, and encourages rapid and flexible response to change [1]. It is a conceptual framework that promotes foreseen interactions throughout the development cycle. The *Agile Manifesto* introduced the term in 2001. Agility has become today's buzzword when describing a modern software process. Everyone is agile. An agile team is a nimble team able to appropriately respond to changes. Change is what software development is very much about. Changes in the software being built, changes to the tem members, changes because of new technology, changes of all kinds that may have an impact on the product they built or the project that creates the product. Support for changes should be built-in everything we do software. An agile team recognizes that software in teams and that the skill of these people, their ability to collaborate is at the core for the success of the project [2].

The Agile Alliance [AG103] defines twelve principles for those who want to achieve agility:

- 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 changes 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 top 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 behaviour accordingly.

Agility can be applied to any software process. However, to accomplish this, it is essential that the process be designed in any way that allows the project team to adapt tasks and to streamline them, conduct planning in a way that understands the fluidity of an agile development approach, eliminate all but the most essential work products and keep them lean, and emphasize an incremental delivery strategy that gets working software to the customer as rapidly as feasible for the product type and operational environment [3].

II. AGILE MANIFESTO

In February 2001, 17 software developers met at a ski resort in Snowbird, Utah, to discuss lightweight development methods. They published the "Manifesto for Agile Software Development" to define the approach now known as agile software development. Some of the manifesto's authors formed the Agile Alliance, a non-profit organization that promotes software development according to the manifesto's principles [4].
Twelve principles underlie the Agile Manifesto, including:

- Customer satisfaction by rapid delivery of useful software.
- Welcome changing requirements, even late in development.
- Working software is delivered frequently (weeks rather than months).
- Working software is the principal measure of progress.
- Sustainable development, able to maintain a constant pace.
- Close, daily co-operation between business people and developers.
- Face-to-face conversation is the best form of communication (co-location).
- Projects are built around motivated individuals, who should be trusted.
- Continuous attention to technical excellence and good design.
- Simplicity.
- Self-organizing teams.
- Regular adaptation to changing circumstances.

In 2005, a group headed by Alistair Cockburn and Jim Highsmith wrote an addendum of project management principles, the Declaration of Interdependence, to guide software project management according to agile development methods.

III. SOFTWARE DEVELOPMENT PROCESS

A set of activities that leads to the production of a software product is known as software process. Although most of the softwares are custom build, the software engineering market is being gradually shifted towards component based. Computer-aided software engineering (CASE) tools are being used to support the software process activities. However, due to the vast diversity of software processes for different types of products, the effectiveness of CASE tools is limited. There is no ideal approach to software process that has yet been developed. Some fundamental activities, like software specification, design, validation and maintenance are common to all the process activities.

Software Development Life Cycle

The agile methods are focused on different aspects of the software development life cycle. Some focus on the practices (extreme programming, pragmatic programming, agile modelling), while others focus on managing the software projects (the scrum approach). Yet, there are approaches providing full coverage over the development life cycle (dynamic systems development method, or DSDM, and the IBM Rational Unified Process, or RUP), while most of them are suitable from the requirements specification phase on (feature-driven development, or FDD, for example). Thus, there is a clear difference between the various agile software development methods in this regard. Whereas DSDM and RUP do not need complementing approaches to support software development, the others do to a varying degree. DSDM can be used by anyone (although only DSDM members can offer DSDM products or services). RUP, then, is a commercially sold development environment.

Software Model

A software process model is an abstraction of software process. These are also called process paradigms. Various general process models are waterfall model, evolutionary development model and component-based software engineering model. These are widely used in current software engineering practice. For large systems, these are used together.

Waterfall Model

The waterfall model was one of the first published models for the software process. This model divides software processes in various phases. These phases are:

- Requirements analysis
- Software design
- Unit testing
- Component testing
- System testing
- Maintenance

Theoretically the activities should be performed individually but in practice, they often overlap. During the maintenance stage, the software is put into use. During this, additional problems might be discovered and the need of new feature may arise. This may require the software to undergo the previous phases once again.

Agile Model

"Agile Development" is an umbrella term for several iterative and incremental software development methodologies. Well-known agile software development methods include:

- Agile Unified Process (AUP)
- Crystal Methods (Crystal Clear)
- Dynamic Systems Development Method (DSDM)
- Extreme Programming (XP)
- Feature Driven Development (FDD)
- Graphical System Design (GSD)
- Kanban or Lean
- Lean software development
- Scrum
- Scrum-ban
- Software Development Rhythms

Agile Practices

Agile development is supported by a bundle of concrete practices. Some notable agile practices include:

- Acceptance Test Driven Development (ATDD)
- Agile Modelling
- Backlogs (Product and Sprint)
- Behaviour-driven development (BDD)
- Continuous integration (CI)
- Information radiators (Scrum board, Kanban board, Task board, Burn down chart)
- Iterative and incremental development
- Pair programming
- Planning poker
- Refactoring
- Scrum meetings (Sprint planning, Daily scrum, Sprint review and retrospective)
- Story-driven modelling
- Test-driven development (TDD)
- Use case
- User story
- Velocity tracking

The Agile Alliance has provided a comprehensive online collection with a map guide to the applying agile practices.

Method Tailoring

In the literature, different terms refer to the notion of method adaptation, including 'method tailoring', 'method fragment adaptation' and 'situational method engineering'. Potentially, almost all agile methods are suitable for method tailoring. Even the DSDM method is being used for this purpose and has been successfully tailored in a CMM context. Situation-appropriateness can be considered as a distinguishing characteristic between agile methods and traditional software development methods, with the latter being relatively much

more rigid and prescriptive. The practical implication is that agile methods allow project teams to adapt working practices according to the needs of individual projects [5]. Practices are concrete activities and products that are part of a method framework. At a more extreme level, the philosophy behind the method, consisting of a number of principles, could be adapted.Extreme Programming (XP) makes the need for method adaptation explicit. One of the fundamental ideas of XP is that no one process fits every project, but rather that practices should be tailored to the needs of individual projects. Partial adoption of XP practices, as suggested by Beck, has been reported on several occasions. Mehdi Mirakhorli proposes a tailoring practice that provides a sufficient road-map and guidelines for adapting all the practices. RDP Practice is designed for customizing XP. This practice, first proposed as a long research paper in the APSO workshop at the ICSE 2008 conference, is currently the only proposed and applicable method for customizing XP. Although it is specifically a solution for XP, this practice has the capability of extending to other methodologies. At first glance, this practice seems to be in the category of static method adaptation but experiences with RDP Practice say that it can be treated like dynamic method adaptation.

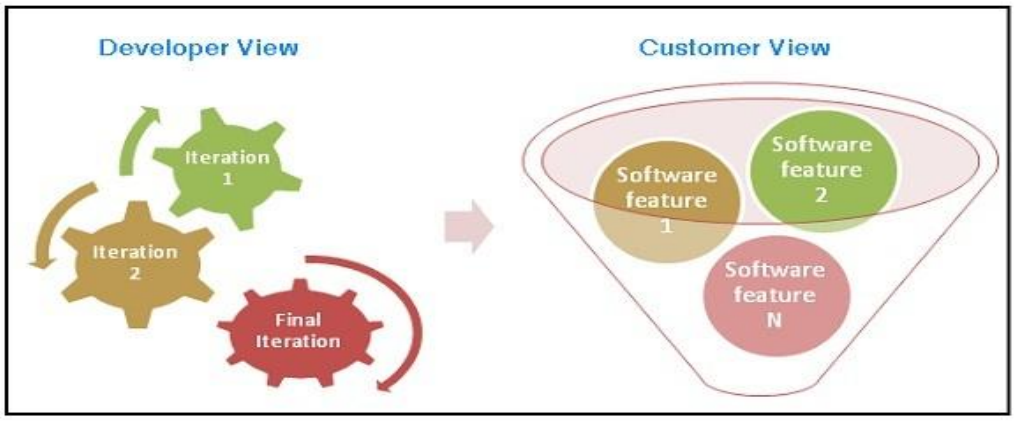

Figure 1: Customisation of Software to create an agile form

Measuring Agility

While agility can be seen as a means to an end, a number of approaches have been proposed to quantify agility. *Agility Index Measurements* (AIM). Score projects against a number of agility factors to achieve a total. The similarly named *Agility Measurement Index*, scores developments against five dimensions of a software project (duration, risk, novelty, effort, and interaction). Other techniques are based on measurable goals. Another study using fuzzy mathematics has suggested that project velocity can be used as a metric of agility. There are agile self-assessments to determine whether a team is using agile practices (Nokia test, Karlskrona test, 42 points test). While such approaches have been proposed to measure agility, the practical application of such metrics is still debated. There is agile software development ROI data available from the CSIAC ROI Dashboard [6].

Suitability

Large-scale agile software development remains an active research area.

Agile development has been widely seen as being more suitable for certain types of environment, including small teams of experts.Positive reception towards Agile methods has been observed in Embedded domain across Europe in recent years.

Some things that may negatively impact the success of an agile project are:

- Large-scale development efforts (>20 developers), though scaling strategies and evidence of some large projects have been described.
- Distributed development efforts (non-collocated teams). Strategies have been described in *Bridging the Distance* and *Using an Agile Software Process with Offshore Development*.
- Forcing an agile process on a development team.
- Mission-critical systems where failure is not an option at any cost (e.g. software for air traffic control).

The early successes, challenges and limitations encountered in the adoption of agile methods in a large organization have been documented [7]. Agile methods have been extensively used for development of software products and some of them use certain characteristics of software, such as object technologies [8]. However, these techniques can be applied to the development of non-software products, such as computers, motor vehicles, medical devices, food, and clothing. Risk analysis can also be used to choose between adaptive (*agile* or *value-driven*) and predictive (*plan-driven*) methods. Scientists [9] suggest that each side of the continuum has its own *home ground*, as follows:

Agile home ground	Plan-driven home ground	Formal methods
Low criticality	High criticality	Extreme criticality
Senior developers	Junior developers	Senior developers
Requirements change often	Requirements do not change often	Limited requirements, limited features
Small number of developers	Large number of developers	Requirements that can be modelled
Culture that responds to change	Culture that demands order	Extreme quality

Table 1. Suitability of different development methods

IV. CONCLUSION

Agile methodologies can also be inefficient in large organizations and certain types of projects. Agile methods seem best for developmental and non-sequential projects [10]. Many organizations believe that agile methodologies are too extreme and adopt a hybrid approach that mixes elements of agile and plan-driven approaches. The term "agile" has also been criticized as being a management fad that simply describes existing good practices under new jargon, promotes a "one size fits all" mindset towards development strategies, and wrongly emphasizes method over results [11]. Alistair Cockburn organized a celebration of the 10th anniversary of the Agile Manifesto in Snowbird, Utah on February 12, 2011, gathering some 30+ people who'd been involved at the original meeting. A list of about 20 elephants in the room ("undisguisable" agile topics/issues) were collected, including aspects: the alliances, failures and limitations of agile practices and context (possible causes: commercial interests, decontextualization, no obvious way to make progress based on failure, limited objective evidence, cognitive biases and reasoning fallacies), politics and culture. Agile development paradigms can be used in other areas of life such as raising children. Its success in child development might be founded on some basic management principles; communication, adaptation and awareness. The basic Agile Development paradigms can be applied to household management and raising children. [12]. In some ways, agile development is more then a bunch of software development rules; but it can be something more simple and broad, like a problem solving guide.

REFERENCES

[1] Abran, Alain; Moore, James W.; Bourque, Pierre; Dupuis, Robert; Tripp, Leonard L. (2004). Guide to the Software Engineering Body of Knowledge. IEEE. ISBN 0-7695-2330-7.
[2] Sommerville, Ian (2008). Software Engineering (7 ed.). Pearson Education. ISBN 978-81-7758-530-8. Retrieved 10 January.
[3] Manifesto. Principles behind the Agile Manifesto. (2001) [cited 2011 August]; Available from:http://agilemanifesto.org/.
[4] Kar, N.J.(2006). Adopting Agile Methodologies of Software Development ; Available from:http://www.infosys.com/infosys-labs/publications/Documents/adopting-agile-methodologies.pdf.
[5] Ilieva, S., P. Ivanov, and E. Stefanova (2004). Analyses of an agile methodology implementation. in 30thEUROMICRO Conference. Rennes, France: IEEE Computer Society.
[6] Beck, K. (2000). Extreme Programming Explained: Embrace Change Reading, MA: Addison-Wesley.
[7] Cohn, M. (2009). Succeeding with Agile: Software Development Using Scrum. Redwood City, CA:Addison-Wesley Professional.
[8] Gall, N. and A. Bradley. (2009) Best Practices for Dividing Developer and Architect Responsibilities to Achieve SOA-Based Agility. Gartner.
[9] Ghezzi, Carlo; Mehdi Jazayeri, Dino Mandrioli. (2003). Fundamentals of Software Engineering (2nd (International) ed.). Pearson Education @ Prentice-Hall.
[10] Jalote, Pankaj (2005) [1991]. An Integrated Approach to Software Engineering (3rd ed.). Springer. ISBN 0-387-20881-X.
[11] Pressman, Roger S (2005). Software Engineering: A Practitioner's Approach (6th ed.). Boston, Mass: McGraw-Hill. ISBN 0-07-285318-2.
[12] Sommerville, Ian (2007) . Software Engineering (8th ed.). Harlow, England: Pearson Education. ISBN 0-321-31379-8.