ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 2, Issue 12, December 2013

A Cynical View on Agile Software Development from the Perspective of a new Small-Scale Software Industry

Apoorva Mishra
Computer Science & Engineering
C.S.I.T, Durg, India

Deepty Dubey
Computer Science & Engineering
C.S.I.T, Durg, India

***Abstract*— In the modern era of software development, a large number of life cycle models are available for the development of the software projects in a proper way. SDLC models give a theoretical guide line regarding development of the software. The two basic schools of thought regarding the development of the software are: The traditional approach and the agile approach. In this paper we have explored the limitations of the agile model when applied to the project of a new small-scale software industry.**

***Index Terms*— Software Development Life Cycle (SDLC), agile model, life cycle phases, sprint.**

I. INTRODUCTION

Software development life cycle (SDLC), is a method by which the software can be developed in a proper way. SDLC also increases the probability of completing the software project within the time deadline and maintaining the quality of the software product as per the requirement.

Plan-driven methods work best when developers can determine the requirements in advance . . . and when the requirements remain relatively stable, with change rates on the order of one percent per month.

-- Barry Boehm [1]

As stated by Boehm, there has been a shift in the software industries from the traditional approach to the agile approach. Agile development has been a buzz word in the software industry since 1990s, and the companies perceive it as a magical wand. But there have been many cases where implementing the agile paradigm has resulted in a failure. In this paper a thorough study of the different agile software development models has been done and on the basis of that a list of loop-holes that exist while implementing agile paradigm in a new small scale software industry is presented, which could be the possible reasons, for those failures.

II. PHASES INVOLVED IN TRADITIONAL SDLC MODELS

The phases involved in the traditional software development life cycle model are:

1. Requirements gathering .
2. Requirement analysis
3. high level design
4. low level design
5. Implementation
6. Testing
7. Deployment & maintenance.

Each of the basic activities itself may be so large that it cannot be handled in one step and has to be broken into smaller steps. For example, design of a software is usually broken into multiple, distinct design phases, starting from a very high level design specifying only the components in the system to a detailed design where the logic of the components is specified. The common phases of an SDLC can be represented by the following diagram:

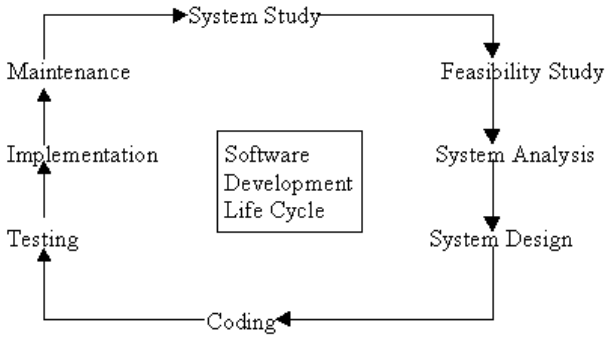

Fig. 2.1 Phases of a traditional SDLC model.

III. AGILE SOFTWARE DEVELOPMENT MODELS

Agile methods are a subset of iterative and evolutionary methods and are based on iterative enhancement[6] and opportunistic development processes. The term “agile”, promotes the professed ability for rapid and flexible response to change of the methodologies.

Agile software development is a conceptual framework for software engineering that promotes development iterations throughout the life-cycle of the project[8].

The Agile Alliance documented its value statement as follows:

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

"Individuals and interactions" over processes and tools

" Working software " over comprehensive documentation

"Customer collaboration" over contract negotiation

"Responding to change" over following a plan

That is, while there is value in the items on the right, we value the items on the left more".

The principles to be followed have also been documented by the Agile Alliance and the agile methods are principle-based, rather than rule-based [2]. These principles are:

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

Some of the famous agile software development methods are: Dynamic Systems Development Method (DSDM), Extreme Programming (XP), and Scrum [3]. In this paper we are going to explore each of these methods in detail.

XP MODEL

In XP the programming is done in pairs[9]. It involves test driven development, continuous planning, change and delivery. No overtime is required. The methodology is based upon five underlying values[7]:

- Communication
- Simplicity,
- Feedback,
- Courage, and
- Respect.

The technical practices of XP are as follows:

- Real customer involvement: The customer is available to clarify doubts regarding requirements, is a subject matter expert, and is empowered to make decisions about the priority of the requirements.
- Incremental deployment: Gradually deploy functionality in a live environment to reduce the risk of a big bang deployment approach.
- Team continuity: Keep effective teams together.
- Shrinking team: As a team grows in capacity (due to experience), keep their workload constant but gradually reduce the size of the team.

- Root cause analysis: Examine the cause of a discovered defect.
- Shared code: Once code and its associated tests are checked into the code base, the code can be altered by any team member.
- Code and tests: Maintain only the code and tests as permanent artifacts.
- Daily deployment: Put new code into production every day.
- Negotiated scope contract: Fix the time, cost, and required quality of a project but call for an on-going negotiation of the scope of the project.
- Pay-per-use: Charge the user every time the system is used.

XP Strengths

1. Lightweight methods suit small-medium size projects
2. Produces good team cohesion
3. Emphasises final product

XP Weaknesses

1. Difficult to scale up to large projects where documentation is essential
2. Needs experience and skill if not to degenerate into code-and-fix
3. Pair programming is costly.

SCRUM

In the Scrum process [3,4] puts a project management "wrapper" around a software development methodology.

Scrum teams are self-directed and self-organizing teams. The team commits to a defined goal for an iteration and is given the authority, autonomy, and responsibility to decide how best to meet it.

- There are three main artifacts produced by Scrum teams, the Product Backlog,
- the Sprint Backlog, and
- the Sprint Burn down chart.

All of these are openly accessible and intentionally visible to the Scrum Team. An overview of the Scrum process is provided in Fig.2.

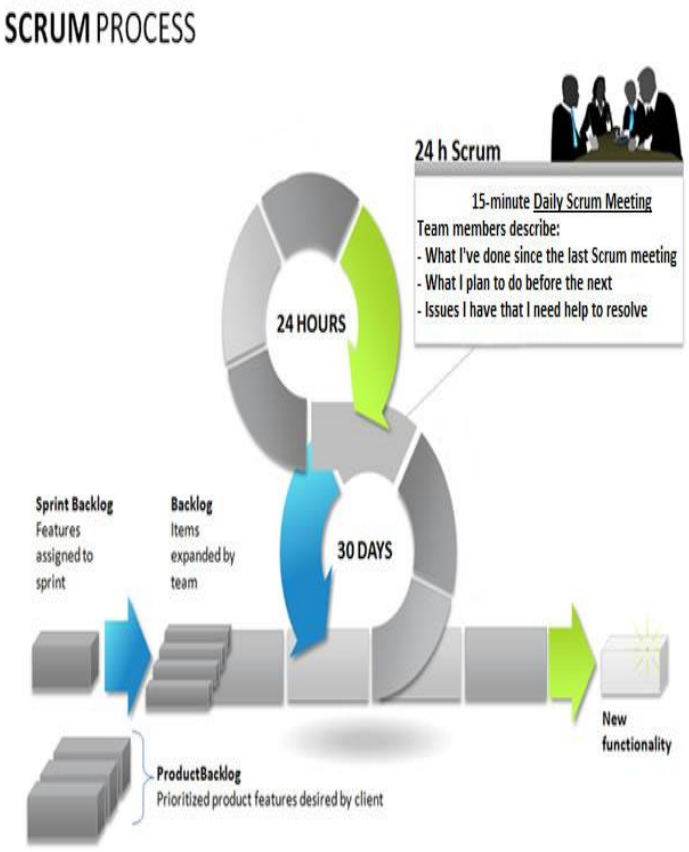

Fig. 3.1 Overview of the scrum process

As represented by the figure the Scrum process is composed of the following steps[5]:

- A Sprint Planning meeting is held with the development team, management, and the Product Owner. The Product Owner creates and prioritizes the Product Backlog. In the planning meeting, the Product Owner chooses which features are included in the next 30-day increment (called a Sprint) usually driven by highest business value and risk.
- During a Sprint, the code integration and regression testing is done daily.
- Short, 15-minute Scrum Meetings are held daily. The meeting is held in a room with a whiteboard so that tasks and blocks can be written down. While others) may attend the Sprint Meeting, only the team members and the Scrum Master can speak. Each team member answers the following questions:
  - What have you done since the last Scrum?
  - What will you do between now and the next Scrum?
  - What got in your way of doing work?

  The Scrum Meeting is an important part of the methodology.
- At the end of a Sprint, a Sprint Review takes place to check the progress.

- The loop continues with a Sprint Planning meeting that takes place to select the features for the next Sprint.

Dynamic Systems Development Method (DSDM)

DSDM applies a framework for RAD and short time frames. The Principles governing the DSDM are:

1. Active user involvement.
2. DSDM teams are empowered to make decisions.
3. Focus on frequent product delivery.
4. Product acceptance is fitness for business purpose.
5. Iterative and incremental development to converge on a solution.
6. Requirements are initially agreed at a higher level.
7. All changes made during development are reversible.
8. Testing is integrated throughout the life cycle.
9. Collaborative and co-operative approach among all stakeholders is essential.

DSDM Lifecycle

The DSDM life cycle consists of the following phases.

- Feasibility study
- Business study – prioritized requirements
- Functional model iteration
- risk analysis
- Time-box plan
- Design and build iteration
- Implementation

The paradigm is the 80/20 rule. In this model, majority of the requirements can be delivered in a relatively short amount of time.

IV. PROBLEMS WITH AGILE APPROACH

Based on the above discussion regarding the agile methods, problems associated with the agile approach can be identified.

Some problems with the agile approach from the perspective of a new small scale software industry are:

1. It requires a serious commitment from the user for the duration of the project as full time user involvement is necessary.
2. There is a potential for scope creep.
3. There is a risk of never ending projects.
4. If the teamwork and communication are not excellent, potential misunderstandings can be created.
5. It is very difficult for the team members (especially testers) who are used to everything being well defined from the beginning.
6. The users need to be ready and available for prompt testing of the features as they are delivered and throughout the entire duration of the project.
7. It is difficult for small-scale companies which are new in the market to hire extraordinarily efficient employees as demanded by the agile manifesto.
8. The tight schedule for project development is stressful for most of the employees.
9. For a new small-scale industry, it is very difficult to have self-organizing teams with excellent communication and understanding among them.
10. It is very difficult to proceed with the agile approach, when the staff attrition rate is higher, as in agile methodology, the concentration is more on the verbal communication and the documentation is minimum [10]. So when new employees are hired, it is very difficult for them to understand the project without proper documentation.

V. CONCLUSION

There are many SDLC models such as, Waterfall, RAD, spiral, incremental, Prototype etc. of traditional type and scrum, XP, DSDM etc. of agile nature used in various organizations depending upon the conditions prevailing in it. All these different software development models have their own advantages and disadvantages. In this paper, the work has been done to find out the different loop-holes that are there in the agile methodology from the perspective of a new small-scale software industry. The loop-holes that have been found as a result will help these industries to perform better by trying to avoid those drawbacks, when implementing the agile methodology.

REFRENCES

[1] B. Boehm, "Get Ready for Agile Methods, with Care," IEEE Computer, vol. 35, no. 1, pp. 64-69, 2002.
[2] C. Larman, Agile and Iterative Development: A Manager's Guide. Boston: Addison Wesley, 2004.

[3] K. Schwaber and M. Beedle, Agile Software Development with SCRUM. Upper Saddle River, NJ: Prentice-Hall, 2002.
[4] J.Highsmith, Agile Software Development Ecosystems.Boston, MA: Addison-Wesley, 2002.
[5] Roger Pressman, titled "Software Engineering - a practitioner's approach".
[6] V. R. Basili and A. J. Turner, "Iterative Enhancement: A Practical Technique for Software Development," IEEE Transactions on Software Engineering, vol. 1, no. 4, pp. 266 – 270, 1975.
[7] Naresh Kumar Nagwani, Pradeep Singh, "An Agile Methodology Based Model for Change- Oriented Software Engineering", International Journal of Recent Trends in Engineering, Vol 1, No. 1, May 2009.
[8] Fowler, M. (2000), "Put Your Process on a Diet", Software Development".
[9] Vishwas Massey, K.J Satao, " Comparing Various SDLC Models And The New Proposed Model On The Basis Of Available Methodology".
[10] Adel Hamdan Mohammad et al, "Agile Software Methodologies: Strength and Weakness", International Journal of Engineering Science and Technology (IJEST), Vol. 5 No.03 March 2013.

AUTHOR(S) PROFILE

Apoorva Mishra, received the B.E (hons) degree in computer science and engineering from R.C.E.T, Bhilai in 2011, then worked at Tata Consultancy Services, Mumbai till May 2012. Now he is pursuing M.Tech from C.S.I.T, Durg. His area of interest includes: software engineering, database management, cryptography, operating system and data structure. He has published various research papers in journals and conferences.

Deepty Dubey, received the B.E degree in computer science and engineering from S.S.C.E.T, Bhilai in 2005, then completed her M.Tech from R.C.E.T, Bhilai, in 2010. Now she is working as an assistant professor at C.S.I.T, Durg. Her area of interest includes: networking, cryptography, algorithms, software engineering etc-. She has published many research papers in international journals and conferences.