9

Lightweight Project Management Methodologies

G P Sudhakar

The traditional project management methodologies are process oriented and need much documentation. Hence, the lightweight (also known as Agile) methodologies are coming into the market. These methodologies are being used in IT, Telecom and Manufacturing industries worldwide. This article gives an introduction to Agile methodologies like Scrum, Extreme Programming, Crystal, Lean, Adaptive Software Development (ASD), and Feature Driven Development (FDD).

1. Introduction

This article gives an introduction to the lightweight project management methodologies, so called "*Agile* methodologies". These days, the lightweight methodologies are being used widely in IT, Telecom, Banking and Finance, etc., industries. The methodologies discussed in this article are Scrum, Extreme Programming (XP), Lean Software Development, Crystal Methodologies, Adaptive Software Development (ASD), and Feature Driven Development (FDD).

As we know very well, traditional project management methodology, so called "Heavyweight methodologies" need huge documentation, more process concentration and less customer interaction. Hence, there is a need for lightweight methodologies, which gives liberty to programmers and allows programmers to do what they are good at with less documentation burden, more customer interaction during development and can sustain to frequent customer's requirements changes.

With this intention, in 2001, *Agile Alliance,* was formed with a group of 12 industry experts including Kent Beck, Jim Highsmith, and Robert C. Martin, etc.

2. Agile Philosophy

The *Agile* methodologies are based on four values (Agile Manifesto), which are given below.

- Individual interaction over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan.

According to this manifesto, communication and interaction are more important than the programming talent (Robert C. Martin, 2001). According to Robert (2001), a team of average programmers who interact well are more likely to succeed than a team of superstars who fail to communicate well.

These agile methodologies suggest us to produce the documents on need basis as and when it is urgent and important. Otherwise do not produce unnecessary documentation, which is an overhead on programmers. Customer interaction is very much important in complex software systems development. Agile methodologies recommend the availability of customer representative on full time basis to the programmers. This makes the systems and requirements clear for the programmers. Change management becomes easy if the customer representative is readily available to the programmers. According to this manifesto, responding to customer change requests quickly is important rather than just following the plans initially devised. This agile manifesto comes along with 12 principles.

The first principle of agile manifesto is *"Our highest priority is to satisfy the customer through early and continuous delivery of valuable software"*.

This indicates that the deliverables are to be continuous and should be delivered quickly to the customer. Hence, the agile methodologies suggest the short release cycles with less features or scope. Usually Agile teams are self-organizing teams. Only the responsibility is given to the team. They plan and execute the tasks to complete the deliverables on time.

The other features of Agile teams include place people physically closer, replace documents with talking in person and at whiteboards (Alistair and Jim, 2001). According to Alistair and Jim (2001), Agile philosophy concentrates on people factors in the projects such as amicability, talent, skill and communication and agile processes are designed to capitalize on each individual and each team's unique strengths.

"*Agility requires that teams have common focus, mutual trust, and respect; a collaborative, but speedy, decision-making process; and the ability to deal with ambiguity*" (Alistair and Jim, 2001).

Agile development better suits the small teams. Usually the agile team consists of two to eight team members in Agile projects. Agile methodologies can also be implemented for bigger team sizes as well. Now, let us see an agile methodology known as Scrum in the next section.

3. Scrum

Every Scrum project starts with a *Product backlog*. This backlog consists of functionality, technical architecture, known bugs, etc. Anybody can add anything to the Product backlog. However only the Product Owner prioritizes them (Darrell, 2005). Once Product Owner, ScrumMaster, Scrum team are comfortable, they will start a *Sprint*. A *Sprint* of 30 days is created from the Product backlog. This Sprint is nothing but a release of deliverables. Every Sprint in scrum project will have it's Sprint backlog. Sprint backlog is derived from the product backlog. Based on this backlog, team members get the tasks to execute. Team members estimate the effort based on the tasks assigned to them.

During Sprint, external world does not interfere in team's work. Sprint team meets daily for 15 minutes under the leadership of ScrumMaster. The ScrumMaster asks every team member following three questions.

i) What did you achieve after last Scrum meeting?

ii) What are the obstacles faced in achieving that?

iii) What are you planning to do before next Scrum meeting?

By asking the above three questions the ScrumMaster gets the status of work and their plans from team members. These meetings happen everyday during Sprint. *Sprint* is complete with the release of deliverables, or cancellation of the project. At the end of Sprint, stakeholders' meeting happens and the ScrumMaster updates the progress to the stakeholders.

According to Linda (2001), at the end of the Sprint, stakeholders' meeting is held, increments are delivered, surprises are reported, anything can be changed, plans and assignments are made for the next Sprint, or project can be cancelled.

According to SE Project 2003/2004 group E (2004) report, theoretically Scrum can be applied to any group of people who should work together for a common goal. Its focus is on team empowerment and adaptability. In ideal case, all team members should be collocated for optimal communication among team members otherwise they should meet with teleconferences (SE Project 2003/2004 group E report, 2004). Scrum basically concentrates on short, regular releases.

The advantages of Scrum (Darrell, 2005) include

- Productivity increases
- Continuous improvement
- Team communication improves
- Product becomes manageable
- Customer relationship improves
- Cultural change towards the success of project
- Delivery on time
- Transparent.

4. Extreme Programming (XP)

Extreme Programming (XP), originated by Kent Beck, is based on four values. They are Simplicity, Communication, Feedback and Courage (Linda, 2001). In XP, the customer representative should be available full-time to the programmers and the development team. Any XP project starts with *User stories*. XP project is made up of releases. Each release is divided into iterations.

Each release may have one to three months duration. At the starting of the iteration, customer splits the user stories specific to the release and assigns the specific iteration. Each iteration may stay for one to two weeks (Everette, 2002). Once user stories are decided for the iteration, these stories are further divided into tasks, which can be estimated and executed by the team members. A team member may take number of tasks.

XP stresses on collective ownership of artifacts. XP is a good choice when requirements are not clear (SE Project 2003/2004 group E report, 2004). XP is meant for small to medium size teams. Customer plays an integral part in XP project. Kent Beck in his book *Extreme Programming Explained* introduced the XP Philosophy and XP Practices.

XP is based on the following practices:

- **Planning Game**: The customer sets the priorities and gives the user stories. The developers does the each task estimation and execution, so that the developers buy in the plans devised.
- **Small Releases**: XP believes in small and quick releases. It believes in providing immediately whatever customer wants. Each release should have the running and tested software.
- **Metaphor**: XP teams should have a common vision and they can use common terminology in the project.
- **Simple Design:** XP uses simple design. Because requirements may change tomorrow, design whatever is needed to meet today's requirements (Ganesh, 2004).

- **Testing:** Unit test cases should be written before code is written. Testing should happen till all unit test cases pass otherwise fix the bugs found immediately.
- **Continuous Integration:** XP practices in continuous integration. Integration happens at least once a day in each iteration. All unit tests and functional tests must pass (Linda, 2001).
- **Refactoring:** Once unit testing is done on the code, refactor, if needed, based on the duplication of the code**.**
- **Pair Programming:** In XP projects, programmer is not isolated. Two programmers sitting before one computer write code. One may think about the program to write and other may think about the strategy (Linda, 2001).
- **Collective Ownership:** No single person is owner of specific code. Collective ownership applies to all artifacts. Anybody can modify others code so that entire team gains knowledge of the system.
- **No Overtime**: XP programmers go home on time. In peak time maximum one week of overtime is allowed. Otherwise it indicates something is wrong in the process or schedules (Ganesh, 2004).
- **Coding Standards:** Because everybody follows coding standards in XP projects, anybody can modify others' code easily. It is readable and can be modified easily.
- **On-Site Customer**: A customer representative is available full-time with the XP project team. He/She communicates with the programmers and the team so that he/she is aware of the status of the XP projects without any surprises.

5. Lean Software Development

For the past two decades, lean philosophy has changed the way we do things in many domains like manufacturing, logistics, and product development. When the customer delays in telling what he/she wants as long as possible, and when he/she asks for something, give it to them so fast that they don't even have time to change their mind (Mary, 2003). This is the idea behind lean software development. It is based on the concepts of lean manufacturing.

Lean software development is based on the following seven principles.

i) **Eliminate Waste:** Lean thinking starts with identifying the waste in the process and deliverables and removes the waste immediately. The wastes in the software development include partially done work, extra processes, extra features, task switching, waiting, handoffs, and defects (Mary, 2003).

ii) **Amplify Learning**: Amplify the learning of the programmers because they have to learn both the technical and requirements related concepts from the users/customers (Darrell, 2005).

iii) **Delay Commitment**: Delay the critical decisions as long as possible. It is better to delay critical decisions because of lack of adequate information in hand. It is better to take decisions with facts and figures in hand.

iv) **Deliver Quickly:** Deliver as soon as the customer asks for, quickly and before customer changes his/her mind.

v) **Empower the Team:** This is very much important because software development happens in teams and programmers should be motivated and empowered to get the results from the team.

vi) **Build integrity:** Integrity is of two types. They are perceived integrity, and conceptual integrity. Perceived integrity makes sure that the information flows from the users to the programmer level through master designers, architects, etc. Conceptual integrity makes sure that all parts of the software product work together well as a whole (Mary, 2003).

vii) **See the whole:** Traditionally developing bigger software systems is breaking it into manageable pieces and optimize the sub systems, which leads to the sub-optimized overall system.

6. Crystal

Crystal is a family of methodologies, distinguished by different colors (Clear, Yellow, Orange, Red, Maroon, Blue, Violet). The rigorousness of the methodologies increases with the darkness of the color. Crystal Clear is the lightest color and it is suited for the small projects of team size ranging from two to eight. Crystal is a family of adoptive, human powered, ultralight and shrink to fit software development methodologies (Linda, 2001). These methodologies suggest collaborating closely, observing processes closely and iterating.

The seven properties of Crystal Clear are frequent delivery, reflective improvement, osmotic communication, personal safety, focus, easy access to expert users and technical environment with automated tests, configuration management and frequent integration (SE Project 2003/2004 group E report, 2004).

7. ASD (Adaptive Software Development)

Adaptive Software Development is based on Complex Adaptive Systems (CAS) theory. This software development methodology is based on the life cycle "Speculate->Collaborate->Learn". Speculate is nothing but planning ahead. Collaborate talks about collaborating in the complex systems development. Collaborative activities build products (Jim, 1997). Learning activities expose the products to all the stakeholders. According to Jim (1997), it is difficult to collaborate without learning or learn without collaboration.

The Adaptive Software Development life cycle, as given by Linda (2001) is as follows:

- Speculate
- Conduct project initiation phase
- Determine project time box
- Determine number of cycles for time box
- Write objective statement for each cycle
- Assign primary components to each cycle
- Assign technology and support components to each cycle
- Develop task list
- Collaborate
- Learn.

8. FDD (Feature Driven Development)

Feature Driven Development (FDD) is the modern concept being used in software product development. The following best practices are followed in FDD (Stephen and John, 2002):

- Domain Object Modeling
- Developing by feature
- Individual Class ownership
- Feature Teams
- Inspections
- Regular Builds
- Configuration Management.

9. Mixing the Methodologies

Scrum can be mixed with Extreme Programming (XP) for engineering purposes. Scrum can also be combined with Crystal Clear (SE Project 2003/2004 group E report, 2004). Crystal Clear can be coupled with XP and Scrum.

10. Comparison with Other Methodologies

The traditional project life cycle methods include waterfall, iterative and incremental method, cowboy method and test-driven approach, etc. In waterfall method, requirements gathering followed by analysis, followed by design, followed by coding and implementation, followed by testing and deployment happens. In iterative and incremental method, the above-mentioned phases will undergo iterations and increments to next phase. With this, the difference is evident in how we execute projects using the agile project management methodologies.

11. Conclusion

We have seen the *Agile* and lightweight methodologies such as Scrum, Extreme Programming, Crystal, Lean, Adaptive and FDD in this article. This is an introduction to these methodologies widely used in IT and Telecom industries. These can be complemented with PMBOK and PRINCE2, which are used mainly as heavyweight methodologies.

Many IT organizations are using these agile methodologies in their projects. The main advantage with these methodologies is reduced documentation and process related activities and concentration on the customer interaction and project deliverables. Also these methodologies stress on continuous short releases of the product. The methodologies mentioned in this article are widely used in the industry.

(G P Sudhakar is a Consulting Editor at Icfai Research Centre, Hyderabad. He can be reached at purna24@hotmail.com).

11. References

1. Alistair Cockburn & Jim Highsmith (2001), "Agile Software Development: The People Factor", Computer, IEEE Computer Society, vol. 34, no. 11, pp. 131-133, Nov., 2001.
2. Darrell Norton (2005), "Lean Software Development Overview", Available online at *http://codebetter.com/blogs/darrell.norton/pages/50341.aspx*
3. Darrell Norton (2005), "Scrum Overview", Available online at http://codebetter.com/blogs/darrell.norton/pages/50339.aspx
4. Everette R. Keith (2002), "Agile Software Development Processes: A Different Approach to Software Design", Available online at *http://www.agilealliance.com/system/article/file/1099/file.pdf*
5. Ganesh Sambasivam (2004), "Extreme Programming (XP)", Available online at *http://agilealliance.com/system/article/file/1376/file.pdf*
6. Jim Highsmith (1997), "Messy, Exciting, and Anxiety-ridden: Adaptive Software Development", *American Programmer*, Vol. X, No. 1, January 1997, Available online at *http://www.jimhighsmith.com/articles/messy.htm*
7. Linda Rising (2001), "Agile Methods: What's it All About?", DDC-I Online News, November 2001, Available online at *http://www.ddci.com/NewsArchive/news_vol2num9.php*
8. Mary Poppendieck, "Lean Software Development", *C++ Magazine*, Fall 2003, Available online at *http://www.poppendieck.com/pdfs/Lean_Software_Development.pdf*
9. Robert C. Martin (2001), "Agile Processes", Available online at *http://www.objectmentor.com/resources/articles/agileProcess.pdf*
10. SE Project 2003/2004 group E, "Software Project Management: Methodologies & Techniques", Software Engineering Project (2IP40), Dept. of Mathematics & Computer Science, Technische Universiteit Eindhoven, September 2004, Available online at *http://paul.luon.net/writings/essays/SEP-SPManagement.pdf*
11. Stephen R. Palmer and John M. Felsing, "A *Practical Guide to Feature-Driven Development*", Pearson Education, 2002.