A SYSTEMATIC STUDY ON AGILE SOFTWARE DEVELOPMENT METHODOLOGIES AND PRACTICES

Dr.C. Thiyagarajan,

Assistant Professor,

Department Of Computer Technology,

PSG College of Arts & Science,

Coimbatore-14.

Abstract :

Software engineering techniques have been employed for many years to create software products. The selections of appropriate software development methodologies for a given project, and tailoring the methodologies to a specific requirement have been a challenge since the establishment of software development as a discipline. In the late 1990's, the general trend in software development techniques has changed from traditional waterfall approaches to more iterative incremental development approaches with different combination of old concepts, new concepts, and metamorphosed old concepts.

Keywords: Agile, Software development, Waterfall Model, Extreme Programming, Scrum, Kanban.

The aim of most software companies is to produce software in short time period with minimal costs, and within unstable, changing environments that inspired the birth of Agile. Agile software development practice have caught the attention of software development teams and software engineering researchers worldwide during the last decade but scientific research and published outcomes still remains quite scarce. This paper explains the values and principles of ten agile practices that are becoming more and more dominant in the software development industry. Agile processes are not always beneficial, they have some limitations as well, and this paper also discusses the advantages and disadvantages of Agile processes.

1.Introduction:

The Waterfall Model is a sequential development approach, in which development flows downwards through the phases of requirements analysis, design, implementation, testing and maintenance. Agile Development Model is based on iterative and incremental development approach in a highly collaborative manner to produce high quality software in a cost effective and timely manner which allows the project to adapt the changes quickly.

Agile methodologies stresses on delivering the smallest working piece of functionality as early as possible and constantly improving it and adding additional functionality throughout the project lifecycle. Agile helps in minimizing and mitigating the overall risk, and allows the project to adapt to changes quickly and does not require a requirements freeze upfront unlike waterfall model.

1.1 Agile Manifesto:

In 2001, the Agile manifesto, established the approach now known as agile software development process created by 17 influential figures, a guiding force for Agile practitioners which details four core

values for enabling high-performance, efficiency and outputs:

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

1.2 Agile Principles:

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 timescale.

4. Business people and developers must work together daily throughout 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. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

9. Continuous attention to technical excellence and good design enhances agility.

10. Simplicity, the art of maximizing the amount of work not done is essential.

11. The best architectures, requirements, and designs emerge from self-organizing teams.

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

2. Agile Methodologies:

Agile methodologies share common principles, but differ in practices:

- Extreme Programming (XP)
- Scrum
- Lean Software Development (LSD)
- Kanban
- Adaptive Software Development (ASD)

2.1EXTREME PROGRAMMING (XP):

A. Background:

It was introduced in 1998 by Kent Beck, Ron Jeffries, Ward Cunningham based on experience from C3 project.

B. Philosophy and Scope:

Extreme Programming (XP) is a well-known and a light weight discipline of software development that focuses on engineering practices. XP aims at enabling successful software development despite unclear or constantly changing software requirements. It is a system of practices which is intended to improve software quality and quickly addresses the changing customer requirements to meet business needs.

C. Features and Benefits:

XP Values: It is based on 4 values.

Communication: It is definitely a key factor to the success of any project as most projects fail because of poor communication.

Simplicity: It means to develop the simplest product that meets the customer's needs.

Feedback: It means that developers must obtain and value feedback from the customer, from the system, and from each other.

Courage: It means to be prepared to make hard decisions that support other principles and practices.

XP Activities: Coding, Testing, Listening, Designing

D. Limitations:

- Not suitable for large, difficult or complex projects.
- It requires great amount of coordination between the programmers while doing pair programming and any small conflict may damage the objective of collective code ownership and hence impact the iterations.
- Development of 'metaphor' is required to be shared within team carefully to ensure the common understanding of the terminology.
- Pair programming is a very important practice in XP. However, it cannot be applied to one-developerprojects. Customer collaboration is not very strong. Testing and code development is done by the same person. All the possible problems may not be found because the developer tests from the same perception the product is built.

2.2 SCRUM:

A. Background:

It was first described by Jeff Sutherland, Ken Schwaber, MikeBeedle in 1996. It focuses on Agile project management technique rather than development aspect of projects. Scrum is more focused on managerial skills of both managers and developers. Thus, theoretically, scrum can be applied to any industry**.**

B. Philosophy and Scope:

Term 'Scrum' is taken from the game 'Rugby' where player passes the ball in steps to hit the goal, which is similar as moving project in iterations to meet the core objective. Scrum is iterative, incremental process to develop any project/product or managing any work. It is a light weight software development process consisting of implementing a small number of customer requirements in two to four week sprint cycles. The continuous integration using small sprint reduces the risks and help gaining client confidence. Daily stand-up meeting is very powerful approach to manage and drive the sprint and hence project teams should be small comprising about seven to ten people.

C. Features and Benefits:

Scrum Artifacts:

a. Product backlog: It is the complete list of requirements – including bugs, enhancements requests, and usability and performance improvements – that are not currently in the product release.

b. Sprint backlog: It is the list of backlog items assigned to a sprint, but not yet completed in common practice; no sprint backlog item should take more than two days to complete.

c. Burn down chart: This chart is updated every day, shows the work remaining within the sprint. The burn down chart is used to track sprint progress and to decide when items must be removed from the sprint backlog and deferred to the next sprint.

d. Release backlog: Requirements pulled from the product backlog and identified and prioritized for an upcoming release. The release backlog contains more details about the requirement and low level estimate which are usually estimated by the team performing the work.

D. Limitations:

- Usually it works well with small team therefore growing team may require extended coordination.
- The project is highly dependent on cohesiveness of the team and the individual commitments of the team members, a minor lack in coordination/communication may cause major impact in the sprint.
- Scrum does not explicitly address the issue of criticality.
- Customer is offsite and tight customer collaboration is not possible.

2.3 LEAN SOFTWARE DEVELOPMENT (LSD):

A. Background:

It was adapted from Lean manufacturing of TOYOTA production system and Bob Charette's Lean development in the 1980s. Lean Software Development Poppendieck & Poppendieck in 2003. It focuses on the project management aspects of a project and specifies no technical practices; it integrates well with other agile methodologies, such as XP, that focus more on the technical aspects of software development.

B. Philosophy and Scope:

Lean Software Development (LSD) is an iterative methodology that focuses on reducing waste and optimizing the entire process to achieve the maximum possible gain. Lean has a rich history in manufacturing and has gained popularity in software development in recent years**.**

C. Features and Benefits:

Lean Software Development promotes seven Lean principles.

a. **Eliminate Waste**: Eliminate anything that does not add customer value.

b. **Build Quality In**: Validate and re-validate all assumptions throughout the process. If a metric or practice no longer has value, discard it.

c. **Create Knowledge**: Use short iterative cycles to provide quick, constant feedback to ensure the right things are being focused on.

d. **Defer Commitment**: Don't make decisions until enough is known to make the decision—a sound understanding of the problem and the trade-offs of potential solutions is required.

e. **Deliver Fast**: Minimize the time it takes to identify a business problem and deliver a system or feature that addresses it.

f. **Respect People**: Empower the team to succeed.

g. **Optimize the Whole**: Use cross-functional teams to keep from missing important, possibly critical aspects of the problem and of the system designed to solve it.

D. **Limitations**:

- The project is highly dependent on cohesiveness and the individual commitments of the team members therefore team building is critical factor.
- A missing or inappropriate involvement from appropriate business analyst could result in scope creep.

2.4 KANBAN:

A. Background:

The Kanban method as formulated by David J. Anderson. It focuses on continuous flow of work instead of sprinting; starting and stopping the delivery of work every 2 to 4 weeks.

B. Philosophy and Scope:

Kanban is taken from Japanese term which means 'signboard'. Kanban is a visual process management system that tells what to produce, when to produce it, and how much to produce. It is a method for managing the creation of products with an emphasis on continual delivery while not overburdening the development team.

Like Scrum, Kanban is a process designed to help teams work together more effectively. It uses the concept of 'signboard' using workflow status (such as TBD, WIP, Done) that provides a comprehensive view of the project and promotes the concept of 'wide communication'.

C. Features and Benefits:

Kanban Principles: The 6 principles of Kanban are Visualize the work flow, Limit WIP (Work in Progress), Manage the work flow, Make processes/policies explicit, implement feedback loops, Improve collaboratively.

Kanban Practices: Start with what you do now, Agree to pursue incremental, evolutionary change, Respect the current processes, roles, responsibilities & job titles, and Encourage acts of leadership at all levels.

E. Limitations:

- A small breakdown in Kanban system's process can result in entire line shutting down and recovery requires an additional effort.
- The throughput of the Kanban system is not managed instead it comes as a result of controlled WIP and known cycle time.

2.5 ADAPTIVE SOFTWARE DEVELOPMENT (ASD):

A. Background:

It was described by Jim Highsmith and Sam Bayer in 2000. ASD is part of rapid application development and focuses on rapid creation and evolution of software systems.

B. Philosophy and Scope:

ASD is a method for creating and developing software systems. It offers solutions for the development of large and complex systems by incremental and iterative development, with constant prototyping. It involves product initiation, adaptive cycle planning, concurrent feature development, quality review, and final quality assurance and release. It relies on team members' work at peer level. Usually customer doesn't have all the requirements in the beginning and act as a member with the concept of progressive elaboration.

C. Features and Benefits:

Phases: Speculate (initiation and planning), Collaborate (concurrent feature development) and Learn (quality review).

Lifecycle Characteristics: Mission focused, Time boxed, Risk driven, iterative, Change tolerant, Feature based.

3.ADVANTAGES OF AGILE SOFTWRE DEVELOPMENT:

The key benefits of adopting agile methodology in software development processes explained in detail as follows:

Rapid, iterative and incremental delivery

Project delivery is divided into small functional releases to check functionality, to manage risk and to get early feedback from customer and end users. These new and small features releases are delivered quickly and frequently on a fixed schedule iterations of 1-4 weeks each, with a high level of predictability. Project plans, requirements, design, code and tests are created initially and updated incrementally as required to adapt to project changes.

Increased performance

Daily stand-up meetings provide an opportunity to exchange valuable information and to fine tune improvements continuously.

Flexibility of design

Flexibility is based on the development process used for the project and is defined as ability to change directions quickly. The main feature of Agile approach is to adapt to changing requirements quickly which enables the design to be made flexible that can handle changes easily.

Adaptive to the changing environment

Using agile software development approach, software is developed over several iterations. Each iteration is characterized by analysis, design, coding and testing.

Reduces risks of development

As the incremented mini software is delivered to the customers after every short development cycle and feedbacks are taken from the customers, it warns developers about the upcoming problems which may occur at the later stages of development.

4. DISADVANTAGES OF AGILE SOFTWARE DEVELOPMENT:

Besides many advantages of using agile software development method and like traditional methods, the agile methods also have some limitations and have also been criticized by some practitioners and researchers, mainly focusing on following aspects:

Not suitable for large projects

An agile development method is suitable for small teams, but does not scale well to larger projects, as numerous iterations are needed to complete the desired functionality.

Customer Interaction

Agile requires active user involvement and close collaboration with the project team throughout the software development cycle. This practice of Agile is very rewarding and ensures delivery of the right product.

Insufficient and unclear requirements

Agile requirements are usually insufficient and unclear which eliminates wasted effort on deliverables that don't last which controls budget and schedule.

Changing requirements

Requirements emerge and evolve throughout the development lifecycle which means flexibility to makes change as needed to ensure delivery of the right product.

Difficulty in integration testing

Testing is integrated throughout the product development lifecycle which ensures the quality throughout the project without the need for an extensive test phase at the end of the project. This practice of Agile demands testers throughout the project which increases the cost of resources on the project.

5. CONCLUSION:

Traditional software development approaches have a potential to provide straightforward, systematic, and organized process in the software development. However, the traditional approaches have limitations, which include slow adaptation to rapidly changing business requirements, a tendency to be over budget and behind schedule, a lack of dramatic improvements in productivity, reliability, and simplicity.

Agile development methodology is a conceptual framework for undertaking any software engineering project. In general agile approach can provide a shorter development cycle, higher customer satisfaction, and quicker adaptation to rapidly changing business requirements. It also attempts to minimize risk and maximize productivity by delivering working software in short iterations that increases the internal and external practices of the software development. A selection and implementation of appropriate process is crucial as it ensures the organization to maximize their chance to deliver their software successfully with long term implications.

References:

1.Agile Alliance. "Principles behind Agile Manifesto". www.agilemanifesto.org/principles.html, 2006.

2. Beck, Kent, "Manifesto for Agile Software Development". Agile Alliance, http://agilemanifesto.org, 2001.

3. Beck, K. and C. Andres, "Extreme Programming Explained" (2nd Edition), Addison-Wesley Professional, 2004.

4. Beck, Kent, "Extreme Programming Explained: Embrace Change", Addison-Wesley, ISBN 0-201-61641-6, 2000.

5. Beck, Kent, "Extreme Programming Explained: Embrace Change", second ed., Addison-Wesley, ISBN 978-0321278654, 2004.

6. D. Cohen, M. Lindvall, P. Costa, "An introduction to agile methods, in: M.V. Zelkowitz (Ed.)", Advances in Computers, Advances in Software Engineering, vol. 62, Elsevier, Amsterdam, 2004.

7. Schwaber, K. and M. Beedle, "Agile Software Development with Scrum", Prentice Hall PTR, 2001.

8. K. Schwaber, M. Beedle, "Agile Software Development with Scrum", Prentice Hall, Upper Saddle River, 2001.

9. M. Poppendieck, T. Poppendieck, "Lean Software Development: An Agile Toolkit for Software Development Managers", AddisonWesley, Boston, ISBN 0-321-15078-3, 2003.

10. Highsmith, J., "Adaptive software development: a collaborative approach to managing complex systems", Dorset House Publishing Co., Inc, 2000.