Eastern Mediterranean University

Department of Computer Engineering

Software Engineering Program

Famagusta, North Cyprus

Final Report

CMSE 491 – Selected Topics in Software Engineering

Research of Recent Developments in Agile Development Methodology

Students – Group 4:

1. Talal H B Mahdy - 147139
2. Abdoulgwad Hussien Elsheredi - 147597

Instructor: Prof. Dr. Işık Aybay

Lab Coordinator: Ms. Begum Koru

December 2017

Outline: **Page**

1. List of Figures:

2. List of Tables:

3. Introduction:

The aim of this project is to prepare a research document about the recent developments and methods in Agile Development. An Intermediary report was prepared prior to this document in order to assist us in organizing the research about Agile Development Methodology. Agile development is quiet a broad topic with a variety of different ways of implementation. In this document, we will refer to three different research documents to understand and simplify Agile Development Methodologies.

4. Research Planning:

To start off, we focused on completely understanding the principles of Agile Software Development and how it has improved the Software Project Management. In the Intermediate phase, we almost completely focused on this aspect. In the final phase, we did some more research on the recent trends and practices in Agile Development and its Methodologies, how and why they've become successful and the future of these Methodologies. To do this, we are going to review three different publications about Agile Development.

5. Collected Information:

This is the main section of the report. In this section, we will begin by describing an overview of the Software Development Lifecycle. Then, we will explain about Agile Development principles and values, and the researched Methodologies of Agile Software Development by referring to our chosen publications. Finally, we will explain about the advantages of agile compared to some other Software Development Methods and about the future of Agile Development Methodologies.

5.1. Software Development Life Cycle (SDLC):

A Software Development Life Cycle is a framework that describes the activities performed at each stage of a software development project. There are many types of activities that are performed during the development of Software, some of them are:

- Requirements Specification
- Architectural Design
- Component Design
- Detailed Design
- Implementation
- Unit Testing
- Integration Testing
- Acceptance Testing
- Installation
- Maintenance

A well-known example of a Software Development Model is the traditional Waterfall Model. It is called as such because the model develops systematically from one phase to another in a downward fashion, like a waterfall. In the Waterfall Model, each stage of development is dependent on the previous stage as can be shown in Figure 1. To move on the next stage in Waterfall Model, the previous stage should be complete and approved by the stakeholder.

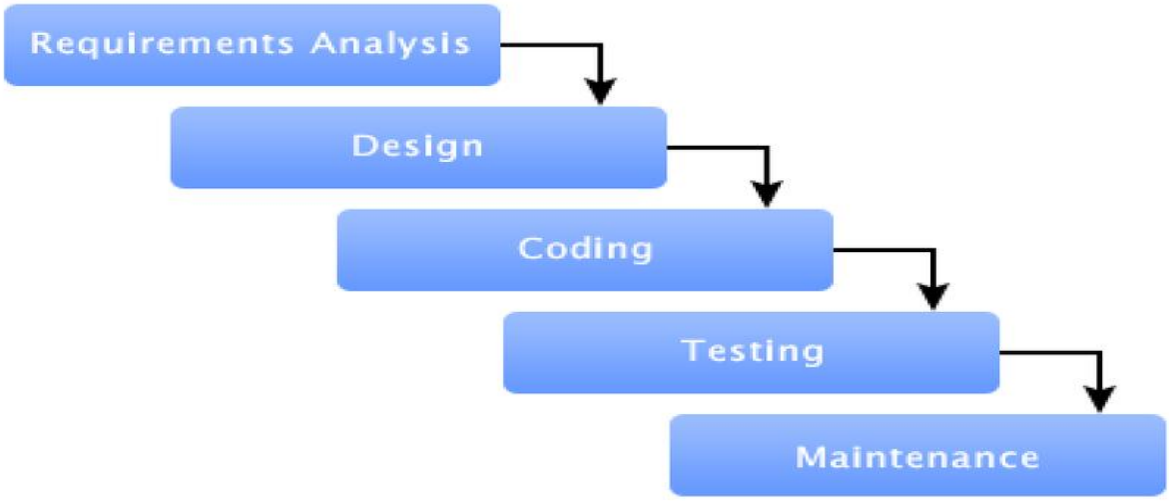

Figure 1. Watefall Model (Calikus, 2015)

5.2. Agile Development:

Agile development is a term used to describe iterative software development. Iterative software development shortens the software development lifecycle. Agile development teams execute the entire software development lifecycle in smaller increments, usually called sprints. Sprints are typically 1-4 weeks long. Agile development is often contrasted with traditional or waterfall development, where larger projects are planned up front and executed against that plan. Agile requires a cultural shift in many companies because it focuses on the clean delivery of individual pieces or parts of the software and not on the entire application. Agile has replaced Waterfall as the development methodology of choice in most companies. In 2001, 17 software development professionals gathered to discuss concepts around the idea of lightweight software development and ended up creating the Agile Manifesto. The four core values of agile software development as stated by the Agile Manifesto emphasize:

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

In the 2001 meeting concerning Agile Software Development, 12 fundamental principles describing Agile Development were formed. The 12 principles laid down in the Agile Manifesto have been adapted for managing a variety of business and IT-related projects. These principles are:

1. Satisfying 'customers' through early and continuous delivery of valuable work.
2. Breaking big work down into smaller components that can be completed quickly.
3. Recognizing that the best work emerges from self-organizing teams.
4. Providing motivated individuals with the environment and support they need and trust them to get the job done.
5. Creating processes that promote sustainable efforts.
6. Maintaining a constant pace for completed work.
7. Welcoming changing requirements, even late in a project.
8. Assembling the project team and business owners on a daily basis throughout the project.
9. At regular intervals, having the team reflect upon how to become more effective, then tuning and adjusting behavior accordingly.
10. Measuring progress by the amount of completed work.
11. Continually seeking excellence.
12. Harnessing change for competitive advantage.

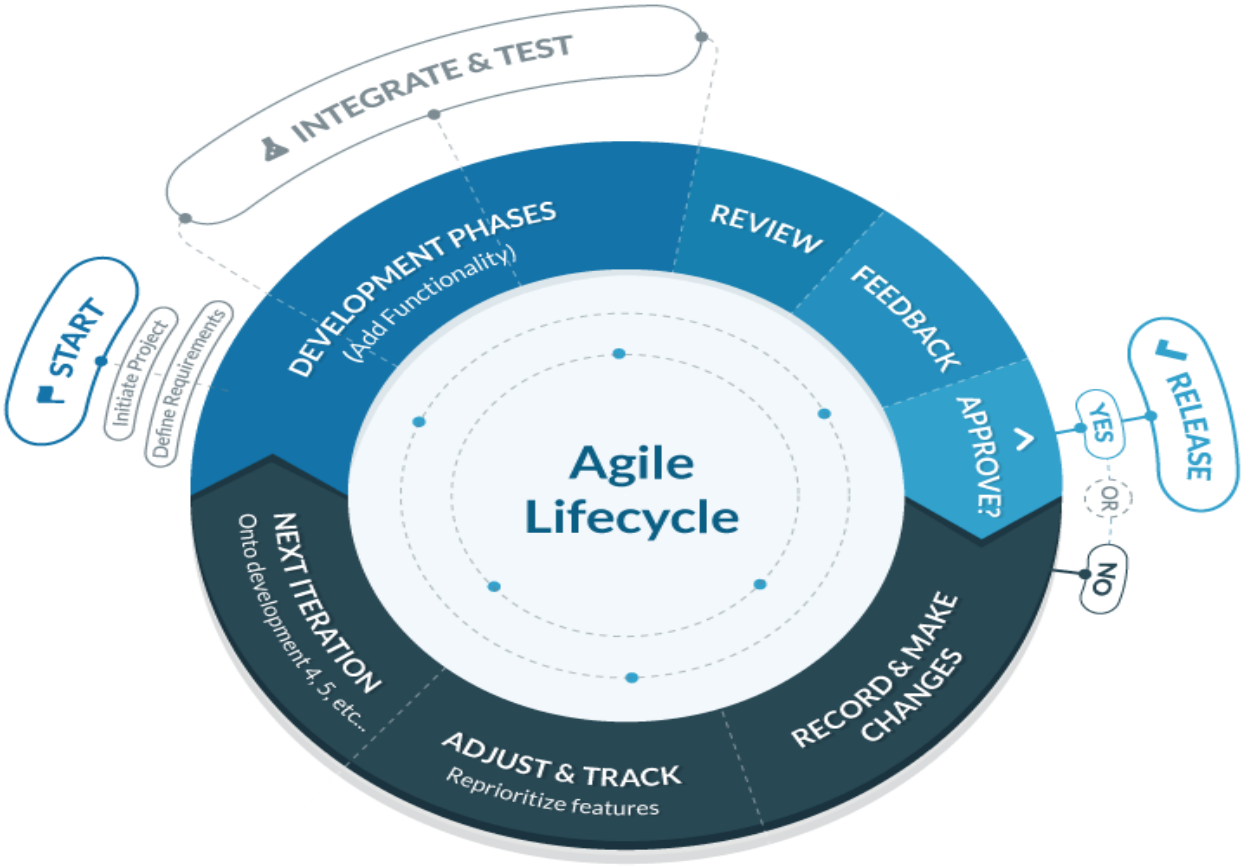

Figure 2. Agile Development Model (Burger, 2016)

5.3. Methods in Agile Development:

Agile Development is a broad and can be considered as a general term for many different techniques for software development management. Some of the Agile development methods are:

- Adaptive Software Development (ASD)
- Feature Driven Development (FDD)
- Agile Unified Process (AUP)
- Crystal Clear
- Dynamic Software Development Method (DSDM)
- Scrum
- Extreme Programming (XP)

In the following sections, we are going to explain two of the most popular Agile Development Methods used nowadays which are Scrum and Extreme Programming.

5.4. Scrum Framework:

Scrum is an agile project management framework with an emphasis on Software Development. Scrum is lightweight, simple to understand but difficult to actually master. It is suitable for the type of projects where the requirements can be quiet unpredictable. The core values of the Scrum Framework are:

- Courage:
  Scrum Team members have courage to do the right thing and work on tough problems
- Focus:
  Everyone focuses on the work of the Sprint and the goals of the Scrum Team
- Commitment:
  People personally commit to achieving the goals of the Scrum Team
- Respect:
  Scrum Team members respect each other to be capable, independent people
- Openness:
  The Scrum Team and its stakeholders agree to be open about all the work and the challenges with performing the work

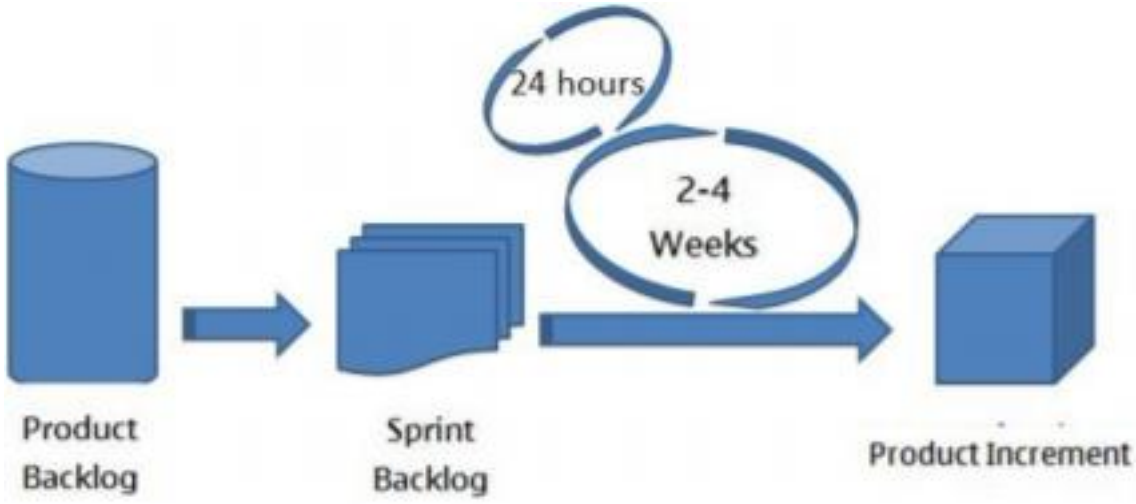

Figure 3. Scrum Process (Li, 2017, p. 7)

A typical Scrum team has 3 different types of roles which are:

1. Scrum Master:

A scrum master should not be confused with a team leader or a project manager role. A scrum master's duty is to help the scrum team handle the things besides development, such as arranging and hosting meetings with product owners and the management. The scrum master ensures that the Scrum framework and the agreed processes in the scrum framework are followed. A Scrum master guides the team to success rather than control the team. He also acts as an interface between the development team and the stakeholders.

2. Development Team:

A typical Scrum team consists of around 5-9 developers. The team is self-organized and that means each member of the team can discuss which task is done by which team member. The team members carry all tasks required to build the product such as analysis, design, development, testing, etc.

3. Product Owner:

The product owner represents the product stakeholders and is responsible for all the requirements, project objectives, and the entire project. A scrum team should have only one product owner. The product owner should focus on the business side of the product development and should not interfere with the technical aspects.

Scrum Process:

The scrum development process is usually an iterative cycle of around 2-4 weeks as can be depicted in Figure 3. Each iterative cycle is called a sprint. The main sprint events in the scrum process are:

- Sprint
- Sprint Planning
- Daily Scrum
- Sprint Review
- Sprint Retrospective

The development team should make every effort to deliver results in each cycle. Each day, the team holds daily scrum meetings of around 15 minutes to track each member's progress, to understand the difficulties faced and re-plan what's to be

done next. Each sprint starts with a sprint planning meeting that aims to define a sprint backlog. At the end of each sprint, a working product should be demonstrated. If some work is not done at the end of sprint, then they should be written back to the Product Backlog in order to discuss them in the next Sprint Planning Meeting. A Sprint Planning Meeting is divided into two parts. In the first part the Scrum master, Scrum team, management, Product owner discuss what and how will the task be developed in the next Sprint. Only the Scrum master and Scrum team participate in the second part of the meeting. Participants refine the selected task into several small steps, and then write the Sprint Backlog. At the end of the sprint, there is a Review meeting for the Product Owner where the scrum team presents the completed work to the stakeholder. After the review meeting, a retrospective meeting is conducted where the team shall answer the questions: What worked well in the previous Sprint and what could be improved in the next Sprint?

5.5. Extreme Programming:

Extreme Programming is a lightweight software development methodology. It is considered as a plan for the project management practices. It is a methodology where there are multiple short development cycles instead of a long development cycle. This allows the developers to adapt to the rapid change in the environments and requirements. Extreme Programming is committed to reduce the software project risk, improve responsiveness to business changes and increase the productivity during development. It is called Extreme programming because it takes the effective principles, values and practices to extreme levels.

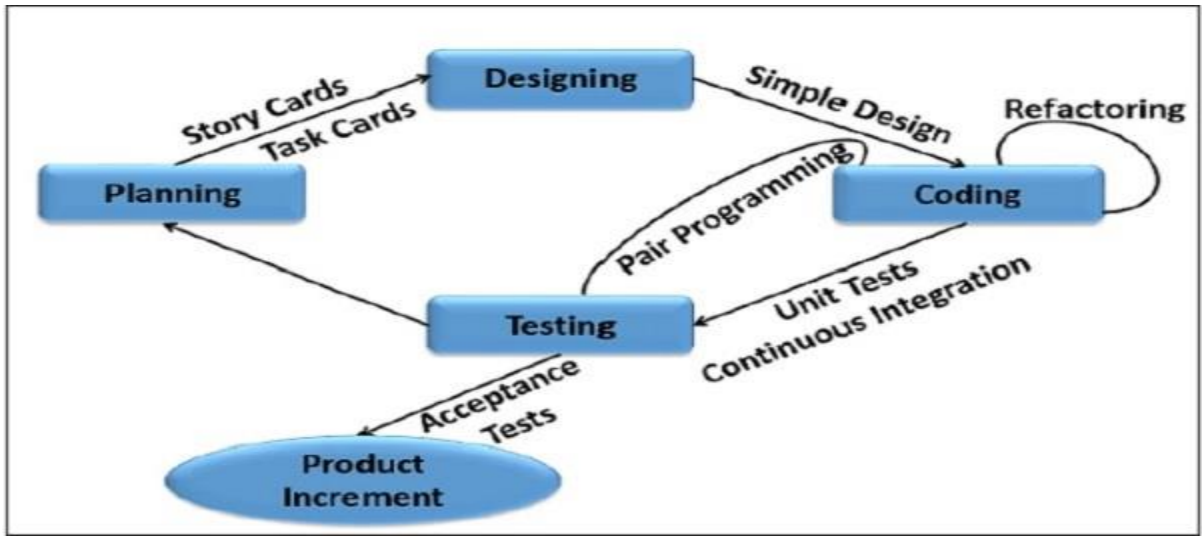

Figure 4. Extreme Programming ("Extreme Programming Introduction", n.d.)

There are 5 main values concerned with Extreme Programming. An extreme programming approach cannot be achieved unless these values are followed:

- Communication: A well understanding of the problem is the biggest challenge that the development team has to face. The most effective way to solve this problem is by strengthening the communication between team members.
- Simplicity: Extreme programming encourages developers to start with the simplest solution and expand it as the functionality increases. Focus on the design and the code of today instead of planning in advance for the designs and code of next week or next month.
- Feedback: Stakeholders and developers should communicate constantly and through continuous, clear feedback, whenever they have a problem in an Extreme Programming project. At the same time, the software developers need to quickly implement some functionality based on the acquired feedback, and present the work to the customers, and get their quality and accuracy feedback on the progress done.
- Courage: If an error or changes in the requirements occur, developers may need to drop some previous work and then re-implement what they have done. In this case, the developers undoubtedly need the courage to do so. Many developers are hesitant to change an already completed work and feel that it is unnecessary but they should follow the needs of the stakeholders.
- Respect: Programmers should never commit changes that break compilation, that make existing unit-tests fail, or that otherwise delay the work of their team members. The respect value includes respect for others as well as self-respect. Nobody on the team should feel unappreciated or ignored. This ensures a high level of motivation and encourages loyalty toward the team and toward the goal of the project.

In the next section we will talk about the Extreme Programming activities that are followed.

The 4 main activities concerned with Extreme programming are as follows:

- Listening: Extreme Programming is almost completely based on communication between team members and the stakeholders. It does not depend a lot on formal written documents. The developers not only need to listen to customer requirements, but should also listen to opinions and suggestions from other people. The developers need very excellent communication skills.
- Designing: Designing in Extreme programming is not a task that a developer should perform. A developer may think that if listening, coding and testing are done well, then the system will work as need all the time. But that is not true at all because as time goes on the system becomes more and more complex and the developer may get stuck in development.
- Coding: In Extreme Programming, many practices of coding are followed such as pair programming, code refactoring, test driven development to get a high quality and a well performing program. There are always 2 programmers working together side by side. One of the roles of the two programmers is called the navigator and the other is a driver. Code refactoring is a process that improves the code after it has been written. Extreme programming emphasizes that testing plans should be done before development actually begins which is known as test-driven development.
- Testing: In Extreme Programming, test is not easy to perform and should not be taken lightly. A good test, same as a good quality of code, is the key to ensure the quality of a software product. Extreme Programming testing is run through the entire development process. From the beginning of the development, the code should be continually tested and most of the issues of the software should be resolved at an early stage rather than to wait until a later stage. This way, the software's quality is guaranteed and the development costs will be reduced.

5.6. Comparing Agile to other Software Methodologies and the Future of Agile:

Up to this date, some companies are hesitant to switch to an Agile development environment due to the fear of change and because they want to stick to what works and has been working for them since years. But many companies on the other hand decided to use Agile Software Development as their primary development strategy. Now it is absolutely clear that agile is turning into a mainstream methodology, and soon only a very few companies will use the Waterfall model. Many companies like Microsoft and IBM are working to improve and promote the use of Agile Development because they believe that the traditional Waterfall model is not fast enough and it is not innovative. But at the same time, Agile is not suitable to develop all kinds of projects and it cannot solve all the problems of software development.

Table 1. Difference between Waterfall and Agile

Project Factor	Waterfall	Agile
Customer Availablity	Required customer involvement only at milestones.	Customers preferably available throughout the project
Scope/ Features	The contract limits changes to requirements. Works well when all requirements are previously known.	Works well when scope is not well known in advance. Welcomes changes at a cost.
Feature Prioritization	Everything asked for by stakeholders should be exactly done. All or nothing approach increases risk of failure.	Prioritize by value ensures most important features are implemented first. Decreases risk of complete failure.
Team	Team coordination is very limited. It only happens during handoff points from one stage to another.	Prefers smaller teams with very high level of communication and synchronization.
Funding	Reduces risk of over prices software by agreeing on the complete contract beforehand.	May increase stress since the actual value is unknown.

6. Conclusion:

In conclusion, we started off this project by doing some important and essential research about Agile and its various methodologies. The concept of Agile Development began with a manifesto that was written in 2001, outlining 4 core values and 12 main principles of Agile Development. The concept of Extreme programming and the Scrum framework improved a lot in recent years and big companies like Microsoft and IBM support them. The future of Agile development looks very promising and it is very widely used nowadays as an alternative to the old Waterfall Model. Finally, if there's anything we've all learned in the last decade of "agile transformations" it's that there is no one-size-fits-all approach to adopting or implementing an agile approach. Every organization has different needs, constraints, and requirements.

References

Burger, R. (2016). *The Ultimate Guide to Agile Software Development - Capterra Blog*. *Blog.capterra.com*. Retrieved 27 November 2017, from https://blog.capterra.com/the-ultimate-guide-to-agile-software-development/

Calikus, E. (2015). *Waterfall Model*. *ResearchGate*. Retrieved 27 November 2017, from https://www.researchgate.net/figure/Figure-6-Waterfall-Model_283048282_fig5

Dingsoyr, T., Nerur, S., Balijepally, V., & Moe, N. (2012). A decade of agile methodologies: Towards explaining agile software development. *Journal Of Systems And Software*, *85*(6), 1213-1221. http://dx.doi.org/10.1016/j.jss.2012.02.033

Extreme Programming Introduction. *tutorialspoint*. Retrieved 24 December 2017, from https://www.tutorialspoint.com/extreme_programming/extreme_programming_introduction.htm

Highsmith, J., & Cockburn, A. (2001). Agile software development: the business of innovation. *Computer*, *34*(9), 120-127. http://dx.doi.org/10.1109/2.947100

Li, J. (2017). *Agile Software Development*. *Snet.tu-berlin.de*. Retrieved 27 November 2017, from https://www.snet.tu-berlin.de/fileadmin/fg220/courses/WS1112/snet-project/agile-software-development_li.pdf