Moving from Traditional Software Development Methods to Agile Methods

University of Oulu
Faculty of Information Technology and
Electrical Engineering / M3S
Bachelor's Thesis
Kalle Ollonqvist
30.05.2018

Abstract

In this literature review, using existing research and sources, benefits and challenges of agile software development methods were discussed. Agile methods were also compared to traditional software development methods. Agile was defined and its four points of value were listed. This provided a basis for discussing the benefits and challenges of agile compared to the traditional software development methods. Customer involvement was found to have a positive effect on customer satisfaction in agile. The development team that was using agile as their development method was able to deliver something of value for the customer faster. This was useful if the company had to race to market. However, a software project that was developed with waterfall was found to be more predictable, especially if the development team was experienced. This was in part due to the customer not being able to change the requirements of the product they had ordered, and in part of the progressive or onward-moving development process typical to waterfall. Also, planning is a big part of waterfall development and it affects the predictability and measurability of the project as well. One of the biggest challenges when moving to agile from a traditional development method was changing the fundamental mindset of the people working in the organization. Especially the management-style needed to change from command-and-control management to leadership-and-collaboration. One of the challenges was that customer involvement might become a burden if the customer is continuously changing the requirements. Moving to agile from a traditional development method is not easy and it could lead to adding more sprints to the software development process than was planned.

Keywords
Agile, Scrum, Waterfall

Supervisor
Postdoctoral Researcher, Muhammad Ovais Ahmad

Table of Contents

1. Introduction

Lots of companies are struggling with traditional software development methods and their inflexibility in today's software development environment. The software market is very competitive and, in many cases, releasing valuable software frequently is necessary to stay competitive. Agile development methods may offer a solution to this problem.

Agile is a comparatively new method of software development in the software development scene. It focuses on fast and iterative release or delivery of software. This is done in small increments. (Qumer et.al., 2007). Compared to some traditional software development methodologies, such as waterfall, agile is viewed as a flexible and modern way of developing and delivering software. (Fowler & Highsmith, 2001). All Agile methods are based on the Agile Manifesto. The manifesto is not very long, only a few rows of text, but it clearly states what is valued in agile. The four points of value are;

"Individuals and interactions over processes and tools,

Working software over comprehensive documentation,

Customer collaboration over contract negotiation,

Responding to change over following a plan".

(Beck et al., 2001)

It is important to remember that in agile, the processes and tools, comprehensive documentation, contract negotiation and following a plan are not disregarded. Individuals and interactions, working software, customer collaboration and responding to change are just valued more. All items listed above are valuable to any software development team, but these statements set a course and lay the groundwork for agile development. Agile is also much more than just developing and releasing software. Agile also affects how the software development teams are built and how different tasks are executed, and which tasks are prioritised. Different agile methodologies may handle day to day work differently and have different approaches for following the agile principles and values. Different agile methodologies are for example; eXtreme programming (XP), feature-driven development (FDD), scrum and crystal methodologies. (Dingsøyr, Sridhar, Balijepally, & Moe, 2012).

My research question is: what are the challenges when adopting agile and what are the benefits and challenges of Agile methods? Differences between agile methods and traditional methods will also be compared. I will also be looking at what different challenges appear when moving from a traditional development method to an agile method, for example from waterfall to scrum and how the processes used in these methods differ from one another. I work at a company that has recently moved from conventional waterfall development to agile scrum. This research will also be looking at how a company can adopt agile development methods, and what kinds of benefits that may bring to a project compared to the old method. This thesis is a literature review.

This thesis is organized as follows: Methods used in this thesis are discussed in the Section 2. In Section 3, the findings of this literature review are presented. In Section 4, implication of the results found in this study are discussed. Section 5 concludes the study.

2. Research Method

This research aims to offer a summary of the benefits and challenges of agile found in the scientific articles in this field of study. It also discusses the challenges which may appear when moving to agile development methods.

Scopus and Google Scholar were used to find the articles used as references in this research. Scopus and Google Scholar both offer a good platform to search for academic and scientific papers because they are commonly used to publish academic and scientific articles in this field.

According to Kitchenham (2007), in software engineering, a common reason for undertaking a literature review is to summarize already existing evidence of an area of interest. This thesis offers a summary of benefits and challenges of agile development and compares it to traditional methods.

Keywords used in this literature review to find the articles are: Agile, Scrum, Waterfall, “Moving to Agile”, “Agile” AND (“Adoption” OR “Adopting”), “Waterfall” AND “Scrum”, “Waterfall to Agile”, “Challenges” AND “Agile”, “Benefits” AND “Agile”

After the first search, 8 relevant articles were found. The search was then expanded, and the rest of the relevant articles were found. The results section considers 15 studies overall. The output of the thesis is to provide a preview of the research into the challenges and benefits of moving from traditional software development methods to agile development methods.

3. Results

3.1 The Benefits of Moving to Agile

One of the reasons why more and more companies are considering adopting agile is that by adopting agile practises to an organisation's software development processes, a company or organisation can gain an advantage on their competitors. For example, adopting agile can mean quicker return on investment. (Sidky et al., 2007). This is because agile software development methods have a shorter release cycle compared to more conventional older methods. Development teams using agile aim to release something valuable in shorter iterations. This means that the company is releasing software that has added value more frequently. This can lead to better software quality overall and better customer satisfaction. (Sidky et al., 2007)

Motivated and empowered software developers who are following agile principles tend to rely on simple designs and technical excellence. The aim of agile development is to release meaningful and valuable software in small increments, but at the core of the agile practices, is the idea of a team which is self-organizing and working at a pace which enables the team to function effectively and creatively. (Dingsøyr et al., 2012). Development teams should not be shying away from changing requirements of the software product in any given phase of the development. Customers are also a big part of the development process and their feedback is valued during the development. This can lead to more satisfying outcomes (Dingsøyr et al., 2012).

Organizations which use agile methodologies in their daily software development work, aim to be able to quickly respond to changes in the software development market or the customer's requirements for the product. Agile teams can creatively find solutions to their business- and technical issues. (Dingsøyr et al., 2012) Agility in agile can be viewed as not being bound by unnecessary bureaucracy. These ideas promote a method which is light and adaptable to different scenarios and situations. Lightness can be viewed as being able to maneuver and react to changes quickly. (Cockburn, 2001)

As mentioned before, having the customer involved in the development process can lead to better results and customer satisfaction. This is especially true for agile, since the customer has frequent opportunities to see the work that the software team is doing. According to Lotz (2013), the customer can also make changes to the requirements of the product they have ordered throughout the development process. Closely working with the software development team translates to a sense of ownership. According to Rigby, Sutherland and Takeuchi (2016), in order to the development team to benefit from agile, the customer needs to be open to close collaboration with the team. (Rigby et.al, 2016). Agile also provides flexibility. If time to market is a big concern for the company, with agile, they can have something of value to release quickly. (Lotz, 2013). The first software release can be built upon in the following releases. Traditional software development methods do not usually allow for iterative release of software. Hence, using agile over traditional methods can provide an edge when competing in a market full of competition.

3.2 Challenges of Agile

Depending on the previous software development methodology, it can be challenging for a company to start using agile development processes. Changing the fundamental way that software is developed can bring up problems not just in the daily work of developers, but also at the organizational and management level as well. When a company or organization adopts agile, they usually must change at least something in their organization structure to support fast and incremental release of software. Changing the structure of the whole organization usually brings up some problems. (Nerur, Mahapatra, & Mangalaraj, 2005) The organization's culture affects the social structure of the organization. It can be very hard to change the core values, assumptions and norms of an organization as these are stabilized and reinforced over time. (Nerur, Mahapatra, & Mangalaraj, 2005) This alone can make moving to agile very difficult, especially for an old organization. Laanti, Salo & Abrahamsson (2011) found out that transformation to agile can be viewed as a change to the organizations development culture, hence difficulties that come from changing the organizations culture are also present when adopting agile. (Laanti et.al, 2011).

In a traditional software development method like waterfall the customer is only involved with the development process at the start when going through the requirements. In agile, customers should be more involved with the development team (Lotz, 2013). Some customers might not be used to this or they simply do not have that big of an interest to be a part of development. Customer involvement is very beneficial for the software project, because it ensures that the software product is going to end up as the customer wants. However, the conditions for agile can be unfavorable in a situation where customers are unavailable or cannot collaborate when needed. Conditions can also be unfavorable for agile in a project where the requirements for the software project are clear to everyone right from the start. (Rigby et.al, 2016)

A team that has been working with a traditional method for years are most likely used to having a clear plan for development. Adopting the time-boxed delivery model in an agile development method like Scrum, which includes short development cycles called sprints, usually lasting about 2-4 weeks, could prove difficult. (Lotz, 2013). This can lead to requiring additional sprints to get all the development work done, especially early in the agile adoption process. Also having the customer involved in the development process can lead to changes in requirements which can lead to adding more sprints than intended. (Lotz, 2013). Additional sprints might also be required if the development team is inexperienced with sprint planning or agile in general. Adding sprints and extending the project development time leads to the increase of the cost of the project. This should be taken into consideration when planning to use agile.

3.3 Moving to Agile from a Traditional Software Development Method

Understanding the main differences between traditional software development methods and agile software development methods is fundamental for a successful introduction of agile. Differences between the core points in software development with agile and traditional software development models are described in Table 1.

Table 1

Traditional versus agile software development according to Nerur, Mahapatra & Mangalaraj, 2005.

	Traditional	**Agile**
Fundamental Assumptions	Systems are fully specifiable, predictable, and can be built through meticulous and extensive planning.	High-quality, adaptive software can be developed by small teams using the principles of continuous design improvement and testing based on rapid feedback and change.
Control	Process centric	People centric
Management Style	Command-and-control	Leadership-and-collaboration
Knowledge Management	Explicit	Tacit
Role Assignment	Individual—favors specialization	Self-organizing teams—encourages role interchangeability
Communication	Formal	Informal
Customer's Role	Important	Critical
Project Cycle	Guided by tasks or activities	Guided by product features
Development Model	Life cycle model (Waterfall, Spiral, or some variation)	The evolutionary-delivery model
Desired Organizational Form/Structure	Mechanistic (bureaucratic with high formalization)	Organic (flexible and participative encouraging cooperative social action)
Technology	No restriction	Favours object-oriented technology

Fundamental differences between traditional software development methods and agile methods come apparent on the first row of table 1. Both methodologies have their benefits. For example, the end-product is more predictable with traditional methodologies, than with agile. (Lotz, 2013). When using agile methodologies, the

customer is an important part of the development process. Change requests are frequent, and the customer can also be involved in testing the product. Requirements may change, which can lead to the product being different from the original presumption. However, if a traditional method such as waterfall is used, a lot of time is spent on perfecting the plan. Planning is usually one of the most important aspects of traditional development. Following a plan brings stability and predictability to the development process, but it also tends to stiffen the process which can be a challenge in today's software scene. Using a traditional method also includes extensive documenting which can be useful. In contrast, agile advocates value working software more than extensive documentation. (Beck et al., 2001).

Many of the differences between traditional software development methods and agile development methods are also true for waterfall and scrum. Scrum is the most popular agile development method and waterfall is the most popular traditional development method. The entire software development process can be divided into seven tasks: conception, initiation, analysis, design, construction, testing and deployment. (Lotz, 2013). In waterfall, these tasks are executed in sequence from top to bottom. Moving back to previous tasks is not prohibited. Work done in each stage can be evaluated and approved before moving on to the next stage. This also means that only one stage in the development is worked on at a time. Two significant benefits of using waterfall over agile are that the software product is more predictable, and that the development process is easier to measure. (Lotz, 2013). These are both due to early and thorough planning. Also, all the required software deliverables are known right from the start of the project. This reduces the possibility of producing code that is not compatible with the rest of the projects software modules (Lotz, 2013).

As seen in Table 1, moving from a traditional software development method to agile requires a shift in the mindset of the people working in the organization. The management style needs to change from command-and-control management to leadership-and-collaboration. (Nerur, Mahapatra & Mangalaraj, 2005). This change also allows for the software development team to become more self-organized. Being self-organized is one of the core principles of agile as mentioned in the agile manifesto (Beck et al., 2001). Especially project managers need to change their management style from commanding to collaborating. (Gandomani et.al, 2013) In agile, they will have much less authority, which can lead to problems when moving from a traditional method to agile (Nerur, Mahapatra & Mangalaraj, 2005). Dikert, Paasivaara and Lassenius (2016) found out that in a lot of cases, the managements unwillingness to change made changes above the team level impossible. (Dikert et.al, 2016). This is especially problematic because moving to agile requires a shift in the mindset in the whole organization. (Nerur, Mahapatra & Mangalaraj, 2005). According to Sureshchandra & Shrinivasavadhani (2008) changing the management style in an organization is one of the most difficult tasks when moving from a traditional development method to an agile method. They found themselves commanding and controlling the team they were coaching to adopt agile instead of making them self-driven or self-organized. (Sureshchandra & Shrinivasavadhani, 2008).

According to Hoda and Noble (2017) in practice organizations go through the agile adoption process in phases. It is almost impossible for an older organization to adopt all the agile practices and drop the old traditional practices overnight. Usually organizations first change their development process to be a sort of a hybrid between traditional and agile practices. This is especially true for bigger organizations. (Hoda & Noble, 2017). Being self-organized is one of the desirable attributes of an agile development team. (Dingsøyr et al., 2012). In practice, this change in the management

style of the team happens gradually over time. The team practices gradually become more and more manager assisted than manager driven as the team gains experience in practicing autonomy. As the team gains and accepts more autonomy, the whole organization starts to change from hierarchical to open. These changes in team practices and management approaches reflect on each other pushing the organization towards being more self-organized. (Hoda & Noble, 2017).

3.4 The State of Agile in Organizations

According to West and Grant (2010), agile has changed to be a part of the mainstream of development practises. It is common that parts of agile and parts of traditional development practices are implemented together to suit bigger organizations. According to West and Grant (2010), organizations with less than 1000 people prefer Agile over traditional methods. 29 percent of application developers or programmers in the study stated that they use Agile as their development method when only 7 percent of developers and programmers stated that they use waterfall. In organizations with more than 1000 employees, waterfall is used more in relation to smaller organizations: 30 percent of developers and programmers in these bigger organizations stated that they are using Agile and 10 percent of developers and programmers stated that they are using waterfall.

According to Murphy, Bird, Zimmerman, Williams, Nagappan and Begel (2013) a survey done at Microsoft during six years between 2006-2012, there was no notable trends to adopt agile inside the company. However, the agile and non-agile practitioners in the company agreed on the possible benefits agile could bring. (Murphy et.al, 2013). This could indicate that there is not such a big push in some bigger companies like Microsoft to adopt agile.

3.5 Choosing the Correct Agile Development Method

Choosing the correct Agile development method is fundamental for the future success of the development work done by the development team. Mainly the same project structure has been used for hundreds of years in building and product development projects. The traditional linear way of seeing a project through, for example building a bridge, offered a good plan and predictability. These were essential for funding the project and assuring that everything was built in the correct way and in the correct order. However, this kind of project structure may not be optimal for developing a product or building an architecture in software. All the following development methods have, at least in part, been formed to combat the problems that occur in developing software in a traditional way. They all offer different advantages and disadvantages. Most widely used agile methods are described and their benefits and challenges evaluated.

3.5.1 Scrum

Scrum is the most widely used agile methodology. (Lotz, 2013). There are multiple different factors in a process of developing software and these factors can change rapidly. These factors can be for example: resources that are available for a project, the time-frame of the project and the requirements. Scrum tries to adapt to these changes as flexibly as possible. (Abrahamsson et.al 2002)

The scrum process consists of three different phases: the pregame phase, the development phase and the postgame phase. The pregame phase is divided in two main phases: planning and high-level design. In the planning phase, the product backlog is created. It includes all the known requirements to be implemented in the final product. This backlog needs to be maintained also in the future. After the backlog has been drawn up and all the requirements are listed in it, the architecture High-level design phase may begin. In the high-level design phase, the basic architecture of the system is planned based on the requirements in the product backlog. The development phase is when the actual implementation (programming) and testing are done. This phase is divided into sprints, which essentially are short time periods normally ranging from 1 to 8 weeks in length. Works that are taken up in each sprint are planned out beforehand before each sprint starts. Works in each sprint are listed in the sprint backlog. Each sprint ends with a sprint review where the works that were done in a sprint are presented and the capacity of the development team is evaluated. (Abrahamsson et.al 2002). For example, if the development team underestimated the work, it is taken into consideration in the future sprints when they are planned in sprint planning meetings. In the postgame phase, the release is closed. No more items or issues can be found, and the requirements have been fulfilled. Integration, system testing and documentation are also done in this phase. (Abrahamsson et.al 2002).

Scrum tries to control and respond to possible changes in recourses, timeframe and project. This happens through different Scrum practices which are: the product backlog, effort estimation, sprint, sprint planning, sprint backlog, daily scrum meetings (standup meetings) and sprint reviews. (Abrahamsson et.al 2002). This results in a very flexible way of developing software. However, it can be difficult for a new team to plan the workload for each sprint. This is a skill that improves with every new completed sprint, but it takes time. In Scrum the team is not managed by anyone, it manages itself. This means that the team makes its own decisions on what to do. As stated in the section 3.1, according to Dingsøyr et.al (2012), self-organizing is a very desirable quality for a development team to have because the team can work at a pace which enables it to function effectively and creatively. (Dingsøyr et al., 2012). Scrum can be best adopted by small teams of less than 10 engineers. If the organization has more than 10 engineers, more teams should be formed. (Abrahamsson et.al, 2002).

3.5.2 Extreme Programming

Extreme Programming (XP) came to be from the problems of traditional development. The idea of XP initially was to get the job done as easily and efficiently as possible. According to Abrahamsson, Salo, Ronkainen and Warsta (2002) even when XP was firstly introduced, the practices included in XP were nothing new. However, the practices were combined in a novel way. According to Abrahamsson et.al (2002) XP consists of five different phases: exploration, planning, iterations to release, productionizing, maintenance and death. XP tries to make successful software possible even though the requirements are constantly changing. It fits best small to medium sized teams. The software product is done in multiple iterations and the customer is involved heavily in the development and testing process.

In the exploration phase the customer writes down story cards which include wanted features for the product. At the same time, the development team creates a prototype of the future product to explore different architecture possibilities. In the planning phase stories are prioritized and an agreement is made regarding the features included in the

first product release. In the Iterations to release phase, different iterations of the product are created. The customer specifies which features are included in which iteration. Functional tests are also run, and they are created by the customer. In the productionizing phase, extra testing and performance tests are done. If there are still some features that are not yet included in the product but are not that crucial, they can be moved to be implemented in the maintenance phase. In the maintenance phase, the development speed of new iterations is slowed down and the focus is moved to different customer support tasks. In the death phase, there are no longer any features or stories to be implemented. All the necessary documentation is written as the architecture of the system will not change anymore. (Abrahamsson et.al, 2002).

XP can be a good fit for a development team to use as the agile methodology. It is meant for small to medium sized teams so this way of developing is not suited for a bigger development team. Also, the physical space where the team does its work should be tailored to support easy communication and collaboration in XP. Pushback from team members to use XP might already deem the adoption of the new development methodology not successful as XP requires all team members to be invested in using it for it to work properly. (Abrahamsson et.al, 2002)

3.5.3 Feature Driven Development

Feature Driven Development (FDD) is an adaptive way of development which emphasis on design and building phases of the development. (Abrahamsson et.al, 2002). Similarly, like all the other agile methods mentioned, it also uses an iterative way of development and aims to deliver good software frequently. It provides all the guidelines, methods and techniques needed for delivering the software product. FDD consists of five consecutive processes where the designing and building is done. These five processes are: develop an overall model, build a feature list, plan by feature, design by feature and build by feature. In the develop an overall model phase, the whole project is split into different domains. The experts of each domain inform each team member of the overall design and description of the system to be developed. This is also called walkthrough. Then the teams in each domain create object models for the domain. In the build a features list phase, all the client valued functions are combined in to a list. These features form feature sets, which are present in a specific domain of the system. In the plan by feature phase, these feature sets are prioritized, and the work is delegated. In the design by feature and build by feature phases feature teams are formed to complete each feature set. Both design by feature and build by feature are done iteratively. Iteration lengths range from a few days to two weeks.

According to Abrahamsson et.al (2002) FDD is fit for use in projects that are especially critical and that need an emphasis on quality. For an organization that is considering adopting this agile method, it is suggested that the adoption process is done in small increments. (Abrahamsson et.al, 2002).

3.5.4 Crystal

Crystal methodologies have been created by Alistair Cockburn. It encapsulates multiple different development methodologies and the team which uses the method, should choose the best fit for each project they take up. It also includes different principles which are used to tailor the chosen methodology within Crystal to fit the project even better. The different methodologies inside Crystal have their own criticality level. The

criticality levels are Comfort, Discretionary Money, Essential money and Life. (Abrahamsson et.al, 2002) If the system produced by the team has a failure, its impact is measured in this scale. For example, if the system being developed runs in a hospital setting, a system failure can cause loss of life. The method inside Crystal should be chosen using this scale by evaluating the criticality level if the system fails. There are some common features between the different methodologies a development team could choose. For example, all projects done with Crystal follow incremental development cycles of maximum 4 months. It also allows for adoption of Scrum or XP practices in its projects. According to Abrahamsson et.al (2002) currently Crystal consists of three main methodologies that are currently in use: Crystal Clear, Crystal Orange and Crystal Orange Web. Crystal Clear is for very small projects (6 members) and Crystal Orange is for medium-sized projects (10-40 members). Crystal Orange Web is optimized for web development. (Abrahamsson et.al, 2002)

Crystal methodologies can be a fit for a development team as an agile methodology if their team size is quite small and the team is physically located in the same building or office space. It is not currently fit for developing life-critical systems. (Abrahamsson et.al, 2002)

4. Discussion

Based on the literature found in this research, agile offers a flexible way of developing software which fits the needs of many organizations. The software market is highly competitive, and agile methods can offer faster time to market compared to traditional methods. Traditional project structure has been used for example in construction projects for a long time. Making a plan and following it strictly was instrumental for the success of the projects. However, many organizations have found that this inflexible project structure might not be optimal in the quickly developing and changing software market. Development of some critical systems may benefit from the strict planning and onwards moving project structure of traditional methods, but for most modern software houses, agile is the way to go. Requirements tend to change in software projects all the time and agile methods are better fit to answer these changes.

Customer involvement is important when developing with agile methods. It can offer its own benefits and challenges, customer involvement usually translates to better customer satisfaction, but it can strain the development process if the customer changes requirements frequently. Moving from traditional methods to agile methods comes with many challenges. The challenges described in this research should be noted when starting the transition from traditional methods to agile methods. Changing the management style is one of the biggest challenges that companies face during the transition.

In practice, no company changes from traditional development methods to a perfect agile way of doing things. It happens gradually. Many bigger organizations have found that traditional development methods suit their needs better, despite the current trend to adopt agile methods. It is also very common that an organization implements different practices from both agile and traditional methods. Some of the most widely used agile methods were discussed and their basic project structure was described. These give a basic understanding of what different agile methods there are, and which method might fit the development team planning to move to agile.

5. Conclusions

This thesis aimed to provide an overview of the research on moving from traditional software development methods to agile methods and the challenges and benefits of agile in comparison to traditional development methods. Using agile over traditional methods can give a company an advantage when moving into the market. Agile methods use short release cycles. This means that, the development team is able to release something of value to the customers faster than if they were using traditional methods. The fast release of software can lead to better customer satisfaction. Agile teams are also self-organized. They can work in a pace that suits them best which can lead to happier and better performing workers. The lack of bureaucracy in Agile methods enable the organization to react to changes quickly and flexibly. Software projects executed with agile were not found to be as predictable and measurable as projects executed with traditional methods.

One of the biggest challenges for a company moving to use agile was changing the mindsets of its employees, especially changing the management style from command-and-control to leadership-and-collaboration. In agile, the management has less authority than in the hierarchical management model found in organizations using traditional development methods. This can cause problems if the transition was not unanimously found to be beneficial or desirable. At the start of adopting agile practices, the team might need more sprints in order to deliver the software product. In the end, this can increase the costs of the software project. However, planning the sprints gets easier overtime. Main differences found between agile and traditional methods were management style, the importance of documentation, roles, customers participation, project cycle, and organization structure. Four different agile development methods were introduced and their fit for adoption discussed. Scrum was found to be the most prominently used one.

6. References

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). Agile software development methods review and analysis*, VTT Electronic*. Retrieved from http://www.otalib.fi/cgi-bin/thw/trip/?${BASE}=vttjure&${HTML}=wwwrecorden&${OOHTML}=wwwrecorden&${TRIPSHOW}=form=wwwabstracten&${FREETEXT}=R%3D44278

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Kern, J. (2001). Manifesto for agile software development.

Cockburn, A. (2001). Software development as a co-operative game*, Agile Software Development 2nd ed (additions)*, Retrieved from http://alistair.cockburn.us/

Dikert, K., Paasivaara, M., & Lassenius, C. (2016). Challenges and success factors for large-scale agile transformations: A systematic literature review. *Journal of Systems and Software*, *119*, 87-108. doi:10.1016/j.jss.2016.06.013

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

Gandomani, J., Zulzalil, H., Ghani, A. & Sultan, A. (2013). Obstacles in moving to agile software development methods; at a glance. *Journal of Computer Science, 9*(5), 620-625. doi:10.3844/jcssp.2013.620.625

Hoda, R., & Noble, J. (2017). Becoming agile A grounded theory of agile transitions in practice. *IEEE/ACM 39th International Conference on Software Engineering*. doi: 10.1109/ICSE.2017.21

Kitchenham, B. (2007). Guidelines for performing Systematic Literature Reviews in Software Engineering. *EBSE Technical Report*. Retrieved from https://www.researchgate.net/publication/258968007_Kitchenham_B_Guidelines_for_performing_Systematic_Literature_Reviews_in_software_engineering_EBSE_Technical_Report_EBSE-2007-01

Laanti, M., Salo, O., & Abrahamsson, P. (2011). Agile methods rapidly replacing traditional methods at nokia: A survey of opinions on agile transformation. *Information and Software Technology*, 53(3), 276. Retrieved from https://search.proquest.com/docview/852928647?accountid=13031

Lotz, M. (2013). Waterfall vs. agile: Which is the right development methodology for your project? Retrieved from https://www.seguetech.com/waterfall-vs-agile-methodology/

Martin Fowler, & Jim Highsmith. (2001). The agile manifesto. *Software Development, 9*(8), 28. Retrieved from https://search.proquest.com/docview/222190361

Murphy, B., Bird, C., Zimmermann, T., Williams, L., Nagappan, N., & Begel, A. (2013). Have agile techniques been the silver bullet for software development at Microsoft? *IEEE International Symposium on Empirical Software Engineering and Measurement.* 75-84. doi:10.1109/ESEM.2013.21

Nerur, S., Mahapatra, R., & Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. *Communications of the ACM.* doi:10.1145/1060710.1060712

Qumer, A., Henderson-Sellers, B., & McBride, T. (2007). Agile adoption and improvement model. *Proceedings European and Mediterranean Conference on Information Systems 2007*. Retrieved from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.7417&rep=rep1&type=pdf

Rigby, D. K., Sutherland, J., & Takeuchi, H. (2016). Embracing agile. *Harvard Business Review*, 94(5), 40-50. Retrieved from https://systematic.com/media/1649532/Embracing-Agile.pdf

Sidky, A. (2007). A structured approach to adopting agile practices: The agile adoption framework. Retrieved from https://vtechworks.lib.vt.edu/bitstream/handle/10919/27889/asidky_Dissertation.pdf

Sidky, A., Arthur, J., & Bohner, S. (2007). A disciplined approach to adopting agile practices: The agile adoption framework. *Innovations in Systems and Software Engineering, 3*(3), 203-216. doi:10.1007/s11334-007-0026-z

Sureshchandra, K., & Shrinivasavadhani, J. (2008). Moving from waterfall to agile. *Agile 2008 Conference*. doi:10.1109/Agile.2008.49

West, D., Grant, T., Gerush, M., & D'silva, D. (2010). Agile development: Mainstream adoption has changed agility. *Forrester Research*, 2(1), 41. Retrieved from: https://www.osp.ru/netcat_files/18/10/h_d8eddd303b6cf0c38c23601c4363bee4