Tobias Malm

Requirements Engineering in Agile Projects

– Comparing a sample to requirements-engineering literature

Master's Thesis in Information Systems
Supervisor: D.Sc Anna Sell
Faculty of Social Sciences, Business and Economics
Åbo Akademi University
Åbo 2020

TABLE OF CONTENTS

TABLE OF CONTENTS **1**

1 LIST OF FIGURES **4**

1. INTRODUCTION **1**

1.1. BACKGROUND 1

1.2. AIM AND RESEARCH QUESTIONS 2

1.3. LIMITATION 3

1.4. STRUCTURE 3

2. RESEARCH METHODOLOGY **5**

2.1. EMPIRICAL DATA GATHERING 6

2.2. VALIDITY OF DATA 7

3. AGILE PROJECT MANAGEMENT **9**

3.1. DEFINITIONS 10

3.1.1. Project and project management *10*

3.1.2. Project Management Models *11*

4. AGILE REQUIREMENTS ENGINEERING **16**

4.1. DEFINITION 17

4.1.1. Requirement types *18*

4.2. REVIEW OF LITERATURE 18

4.2.1. Agile Methods for Requirements Engineering *20*

4.2.2. Modelling and Managing Requirements *22*

4.2.3. Gathering and Prioritising Requirements *24*

4.2.4. The Client's Role in Agile Methods *30*

4.2.5. Challenges with Requirements Engineering *31*

5. EMPIRICAL RESEARCH **37**

5.1. REQUIREMENTS ENGINEERING IN THE GREATER HELSINKI AREA 37

5.2. RESULT 40

5.2.1. The Questionnaire *40*

5.2.2. Analysis of Optional Questionnaire Responses *53*

5.2.3. The Interviews *58*

5.2.4. Research Summary *65*

6. CONCLUSION **68**

6.1. FURTHER STUDIES 73

7. SWEDISH SUMMARY 75

REFERENCES 79

Abstract

Subject: Information Systems
Writer: Tobias Malm
Title: Requirements Engineering in Agile Projects
Supervisor: Anna Sell
Abstract: Businesses rely on projects to perform both large and small tasks in today's work environment. Agile has risen as the preferred way to handle projects. A key part of project management is requirement engineering. As agile has matured, more methods for agile requirements engineering has surfaced. This thesis focuses on the impact agile requirements engineering has on project management. To answer the research questions, an empirical research is conducted on a small sample from the Greater Helsinki area. Two methods are used to collect the data. The data is then compared to the findings in literature to observe if similar findings can be seen in the sample as has been seen in the literature. These findings are then used to examine what impact agile requirements engineering has on project management, whether there are benefits to using agile requirements engineering, and what relation the requirements engineering method has to the success of a project. The research indicates that in the sample, agile is a popular way of managing projects. It also highlights that some projects have benefitted from the use of agile models and that agile requirements engineering methods are well adopted.
Keywords: requirements engineering, agile methods, agile project management, Empirical research

Date: 18.5.2020	**Number of pages: 91**

1 LIST OF FIGURES

Figure 1: Järvinen's taxonomy of research methods 5
Figure 2: Implementation steps to develop a large computer program for delivery to a customer 12
Figure 3: The Rapid Development Waterfall Model 13
Figure 4: Evolutionary Development Waterfall Model 15
Figure 5: Participants of a Typical Project in Greater Helsinki Area 46
Figure 6: Methods Used to Manage Requirements in Projects in Greater Helsinki Area 48
Figure 7: Recognised Issues in Projects 50
Figure 8: Percentage of Projects That Finish on Time 53
Figure 9: Percentage of Projects That Fulfil Their Requirements 54

1. INTRODUCTION

1.1.Background

In today's work environment, implementations and new workflows are revolving around projects and agile methods. Businesses rely on projects and their managers to perform both big and small tasks. There are many parts to project management and for each part there is a way to incorporate agile methods. One crucial part to project management is how to manage the requirements that construct the project. In the traditional method of requirements engineering, the Project Manager analyses what the requirements are at the beginning of the project and then the requirements remain the same until the end of the project. Projects that have adopted agile methods can handle changes to the project's requirements. The requirements will be re-evaluated, changed, added and removed throughout the project. This will have an effect on the requirements in a way that the initial requirements will be contrasting from the project's final requirements.

As agile surfaces more in articles and in discussions these days (Bakalova et al., 2011), it is interesting and valuable to research what methods Project Managers use and are those methods considered agile. Project management is an enormous research subject and researching all parts is not the objective. This research will focus on requirements engineering and what methods Project Managers use to complete this part. The research will only encompass a small sample size of the Greater Helsinki area. However, it will be fascinating to see if there are any patterns between the claims that they are using agile models or if they still are stuck using traditional models.

The aim for researching requirements engineering in agile projects is to see in what ways projects can be handled better in today's businesses. If a project fails in any number of ways, that has an effect on the business going forward. It will most likely affect the people in that project and also how projects will be handled in the future in that company. Therefore, there is plenty of value to be had from researching this topic in the hopes that it will give insight into project management and how that can be made more effective.

Not only for the financial benefit that it will have for the business but also to handle resources better and increase the motivation inside the business.

1.2. Aim and Research Questions

Research done on project management, agile methodology and on requirements engineering is vast. This research will summarize the literature related to agile requirements engineering up to the date of writing and also about the traditional waterfall method. The reason to also research the traditional method is to specify what methods for requirements engineering are considered agile and which are waterfall. This is so that the research can analyse if projects are truly agile, waterfall or a hybrid between the two.

Once this has been researched and defined, the research will evaluate the situation in businesses and Project Managers in the Helsinki region.

The research questions this thesis will try to answer are:

> *RQ1: Has agile requirements engineering impacted project management?*
>
> *RQ2: Does the product benefit from using agile requirements engineering?*
>
> *RQ3: What relation does the method of requirements engineering have on the success of a project?*

For the analysis and conclusion portion, this thesis will perform a small study to see if similar findings can be observed in my study of the Greater Helsinki area as has been observed in previous studies. An example of an area of interest would be what type of methods Project Managers tend to rely on for requirements engineering and to identify if there is any correlation between methods used and the overall success of the project. The research would be limited to see if similar observations can be found in the small sample size of the Greater Helsinki Region as can be observed in the research material. Another aim for this thesis is to discover what the opinions there are about project work. What do people think works and what do they think does not?

1.3. Limitation

This thesis does not aim to discover new agile methods for requirements engineering because earlier research has already identified many methods used for this activity. It is also not the purpose of this thesis to fix or reinvent requirements engineering. This thesis will suggest further research into specific problematic areas of requirements engineering and suggest steps that could be taken to be more agile. It will also suggest further research into low-documented methods that could be beneficial to requirements engineering. The empirical research for this thesis will all be based on data collected in the Greater Helsinki area. If the research finds a new method used in requirements engineering, it will only try to classify it as agile or waterfall and try to evaluate what impact it has on projects. It is also not the aim of this thesis to try to develop a new method or framework for requirements engineering. If there seems to be a demand for a new method or framework for a particular area, then that will be mentioned in the further studies portion of this thesis.

The research, as mentioned earlier, does not aim to find trends or come to any conclusion on how projects are managed in the area of research. The aim is to see if similar observations can be found in the sample as the observations that have been seen in past studies.

1.4. Structure

The first chapter of this thesis covers the background of requirements engineering and how it relates to projects and project management. Moreover, it also covers what this thesis will try to answer and what the limitations are of this thesis. Chapter two will examine the methodology of the research and why it was chosen. Chapter three will investigate agile project management and how it compares to traditional project management. Chapter four is a literature review about agile requirements engineering. This chapter will include methods in agile requirements and the problematic aspects that have been documented regarding the method. It will be divided based on the type of requirements engineering method that is discussed. Chapter five will be the main research chapter where the thesis will examine why and how data was collected. In chapter six,

the results of the research chapter will be discussed. Chapter six includes the conclusion and discussion part of the thesis as well as suggestions for further research in this area.

2. RESEARCH METHODOLOGY

Research comes in multiple forms. The research method should be derived from what type of output the researcher is looking for. This thesis aims to collect data from a sample of individuals from the Greater Helsinki area that work within projects and compare the observations with research literature about requirements engineering. Therefore, the research approach is this thesis will be about testing how well a sample compares with the research that has been done in the past in this area.

Järvinen (2004) has summarized the research approaches based on the intended output. As this research aims to compare a sample with existing research, this study falls into theory-testing. The theories that the research will be testing in this thesis will be about what methods are used for requirements engineering and also if the problems that are mentioned in the literature can also be seen in the sample.

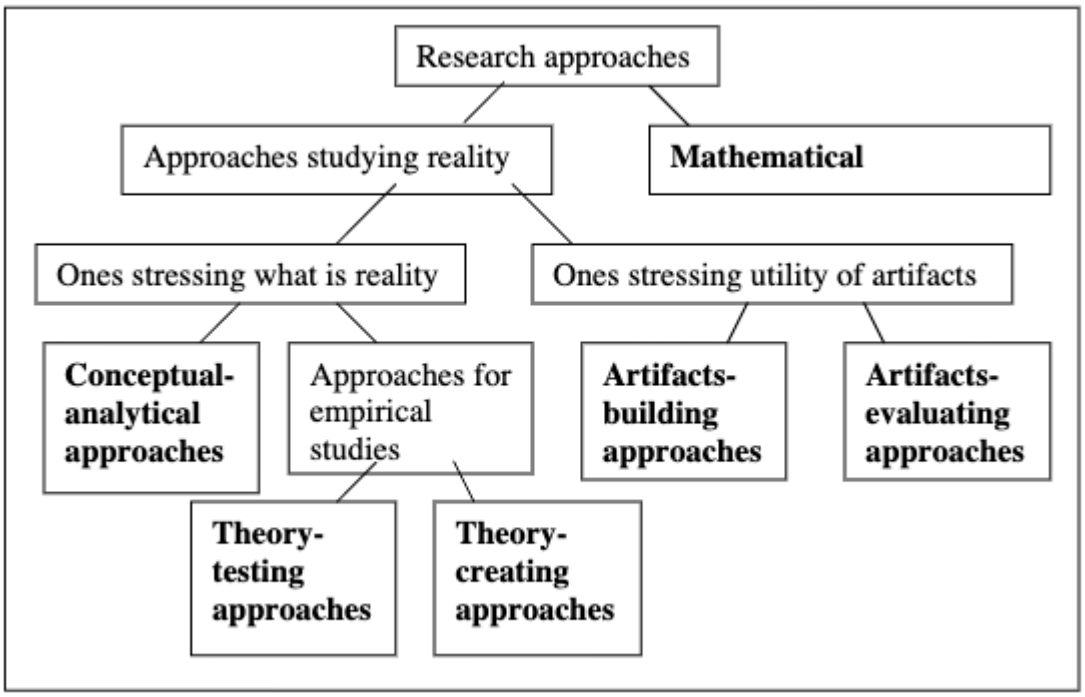


Figure 1: Järvinen's taxonomy of research methods

As a base for the research, this thesis will first examine the research literature regarding agile, project management and requirements engineering. The focus will be on answering

the research questions of this thesis by observing what methods and processes are used for agile requirements engineering. This will include how they compare to traditional requirements engineering methods and also to go through the challenges with both method types. This is to form a picture on the state of agile project management and agile requirements engineering at present. From there forward the research will focus on analysing the situation in project work in the Greater Helsinki area, Finland, what methods they use to handle requirements engineering and what challenges they have encountered. The focus will be on if participants in projects are using agile methods or if they are still using traditional methods. The study will also try to gain an insight into what impact the choice of method has on the projects, the team working on the project and also what opinions there are regarding project management in the area of the study. The information regarding the Greater Helsinki area was gathered by circulating a questionnaire in different media in the hopes of having people involved in projects to answer the questionnaire. The main purpose of the questionnaire was to collect statistical data via questions with predefined answers. The questionnaire also included open-ended questions. The purpose of these questions was to collect information about project work in general as well as to collect opinions from the research demographic. This data will not stand on its own but will be accompanied by data taken from interviews with selected Project Managers and individuals involved in other ways to project management. The main thought behind the empirical research is to observe the status of agile requirements engineering in the Greater Helsinki area and also to see if any patterns can be observed. If any patterns can be observed, this finding will be compared with the literature to see if the sample reflects the findings of the research in the literature. This will relate to both methods used and also the problems that they might accompany.

2.1. Empirical data gathering

To examine the status and patterns of requirements engineering in the Greater Helsinki area, a questionnaire was circulated through different media. The hope was to have as many responses as possible to the questionnaire. In the end, the questionnaire was answered by 39 individuals. Out of the 39 responses, 32 were from the sought-after demographic.

The questionnaire was an electronic form that was circulated in different media in the hopes that potential participants would discover the form. The questions in the form were based on material from requirements engineering literature. The purpose of creating the form was to collect data about what type of methods are used for requirements engineering, problems that are identified and also the opinions of project management. The hope was then that this would give insight into the three research questions of this thesis. As there was no incentive for individuals to answer the questionnaire, the objective when creating the questionnaire was that it should be as short as possible so that as many people will answer it.

The interview questions were formed with the three research questions in mind, the literature and the responses from the questionnaire. The interviews were semi-structured. That means that there were questions to direct the conversation, but it also allowed for the interviewees to talk freely about their experience and opinions regarding requirements engineering. Three individuals that work with projects in different ways were interviewed for this research.

2.2.Validity of data

There is always a concern when gathering data for research purposes (LeCompte & Goetz, 1982). Data used in research should be reliable so that the research results have validity. There are multiple factors that might interfere with research. The interference might come from time, situation, observers or the observed (Denzin, 2017). It is therefore important to remove one's own bias about the subject in social research (Denzin, 2017). There are different methods to minimize the interference rival factors have on the research. However, there is no single method that can be used that would prevent interference altogether. Therefore, one way to increase validity is to use triangulations (Denzin, 2017; Torrance, 2012). There are different types of triangulation. For this thesis the focus will be on triangulation of methods. Using triangulation of method means that two or more methods are used to collect data from the sample. The data from the different methods are then compared to validate the findings. If multiple researchers choose the same demography for the same research topic but they differ in method, it can be assumed that based on the method chosen, these researchers will observe different phenomena, ask

different questions and might even come to different conclusions (Denzin, 2017). This is not the only reason why they might come to different conclusions. Their personalities and values must also be considered when observing the conclusions (Denzin, 2017).

The research in this thesis has gathered data by applying two different methods. The first method is the use of the questionnaire and the second method is interviews. The questionnaire was distributed on the internet. To improve the reliability of the data, the questionnaire targeted people that work with projects within the Greater Helsinki area. There also was a question in the beginning of the questionnaire to confirm that the respondent was from the demography. Also, to further validate the data, interviews were held. As the reason for the interviews was to increase the validity of the questionnaire, answers from the questionnaire constructed the basis of the topics that were discussed in the interviews. The interviews were non-standardised as the questions changed depending on the interview subject and the subject's answers to previous questions in the interview. This was done to grant the interviewer more freedom to explore interesting topics that appeared during the interview (Denzin, 2017).

3. AGILE PROJECT MANAGEMENT

To better understand what requirements engineering is and why it is important in the field of project management, it is best to first understand what project management is and how the agile methodology has been applied to project management.

This chapter will also go through other terms used in project management that will be worth knowing to better understand the material in this thesis. The material for project management comes from different sources and not all of them use the same terminology. This thesis will go through material from different methodologies and frameworks, which usually have their own specific terminology. Therefore, it was deemed important to go through the project management terminology in general so that the research can be compared to projects outside of software development and deployment. For this purpose, this research will use the terminology used by the Project Management Institute. One reason for this is that Project Management Institute is a well-established source for research, courses and material in project management. Their book of knowledge about project management is on version 6 at the time of writing. Version 6 came out in 2017. The reason to use Project Management Institute's material as a base for the overall project management terminology is that their material has been refined throughout the years and the latest iteration is quite recent. When material is used that uses differing terminology, this will be translated to use the same terminology as used by the Project Management Institute if it is deemed logical or there is value in doing so. Why it was deemed more clear to use the Project Management Institute's terminology, instead of using terminology from a source related to software implementation or software development, was that it was difficult to find one clear source with general terminology that emerged above the others. Most of the terminology is framework specific. Scrum, which is one the more popular frameworks in agile, has its own roles, processes and tools and their own definition of those as well (Carlson et al., 2012).

3.1. Definitions

The terminology used in this chapter is related to project-management models and methodologies. A project model is the structure of the project. It is built up of stages with the end result of finishing the project. The stages are sometimes referred to as steps or as phases. A methodology is a set of methods or type of methods to achieve a task. This is sometimes referred to as a paradigm. An example of a methodology is the agile methodology where the tools used should focus on being agile and adapt to change (McCormick, 2012).

Framework is the set of tools, roles and processes used to manage projects. Examples of popular agile frameworks are Scrum and Kanban.

In literature, these terms can be seen to be used interchangeably without properly defining them. Therefore, it was deemed important to give a short definition of the most common terms causing confusion.

3.1.1. Project and project management

A project is an undertaking with a clear start and ending. There is nothing that states how short or long a project can be as long as it has a start and an end. Its aim is to create a product. That product can be a physical product, service, system or a result (Project Management Institute, 2017). The project life cycle consists of the phases that a project works through from the start to the end of the project. In project-management literature, this is sometimes the same as project-management life cycle and sometimes it is not. If the project-management life cycle is not the same as the project life cycle, then it usually relates to the different steps a Project Manager takes during a phase. The Project Management Institute used the term project management process group for this sequence of tasks and the term project life cycle for the series of tasks a project goes through from start to finish (Project Management Institute, 2017). The different phases of a project are populated by process groups. Inside each process group are processes that are interlinked logically (Project Management Institute, 2017). You manage a project when you use your knowledge, skills, tools, methods to fulfil the requirements of the project (Project Management Institute, 2017).

In a project that is managed traditionally, a project that follows the traditional project management model, the project team moves from phase to the next phase until they come to the end of the project. The end of the project is when there are no more tasks to be done or requirements to be fulfilled. This is the reason for the name waterfall as you never go backwards to a phase you already have completed. In the planning phase of the traditional model, the requirements have to be known so that the work can be defined and structured. If the planning phase was done thoroughly, then the work set up for the project will encompass the specifications and scope of the project (Wysocki, 2010).

3.1.2. Project Management Models

Project and project management is nothing new (Patanakul et al., 2010; Project Management Institute, 2017), however, it is always evolving to try and satisfy the demands set by the industries. The Waterfall model (seen in figure 2) was introduced in 1970 for managing software development (Petersen et al., 2009; Royce, 1970) and is seen as the oldest traditional model for software-development project management (Ji & Sedano, 2011). However, the use of the waterfall model can be seen outside of the software-development industry (Wysocki, 2010). The idea behind the Waterfall model came from hardware manufacturing and construction practices (McCormick, 2012). The Waterfall model as seen in figure 2, divided the development into seven stages that follow each other (Ji & Sedano, 2011). Royce (1970) expanded on a simple model that was meant for producing simple programs. This model only had two steps and is present in all software development projects. These steps are *Analysis* and *Coding*. The *Analysis* stage was expanded to have two stages for instigating requirements before the *Analysis* stage. Designing, testing and delivering was also included in the model (Royce, 1970).

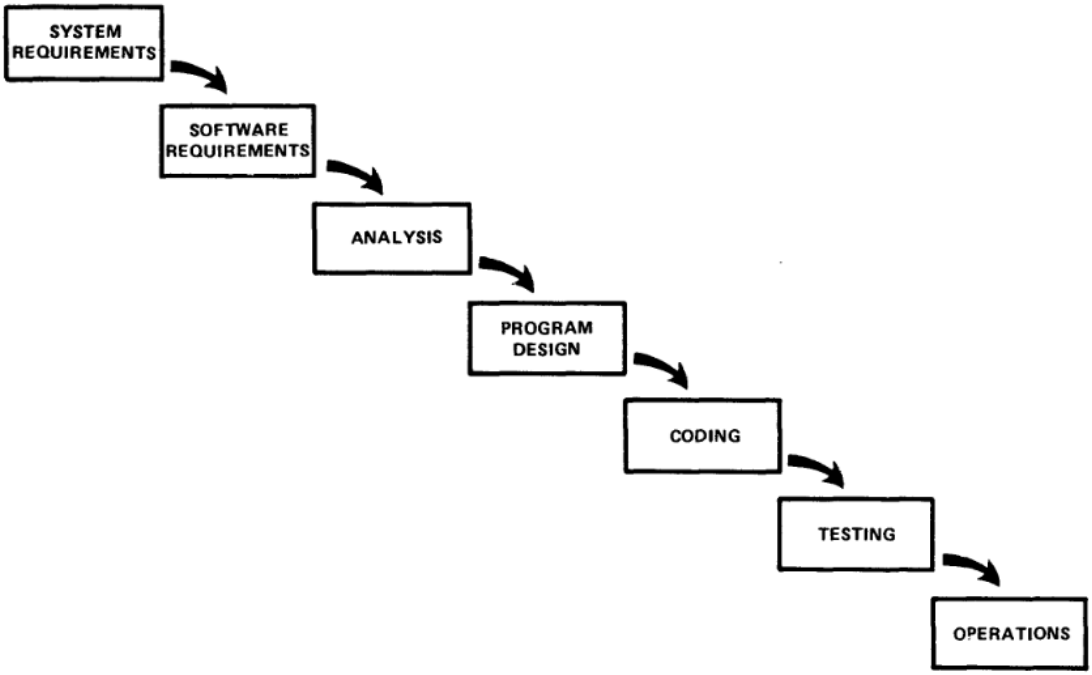
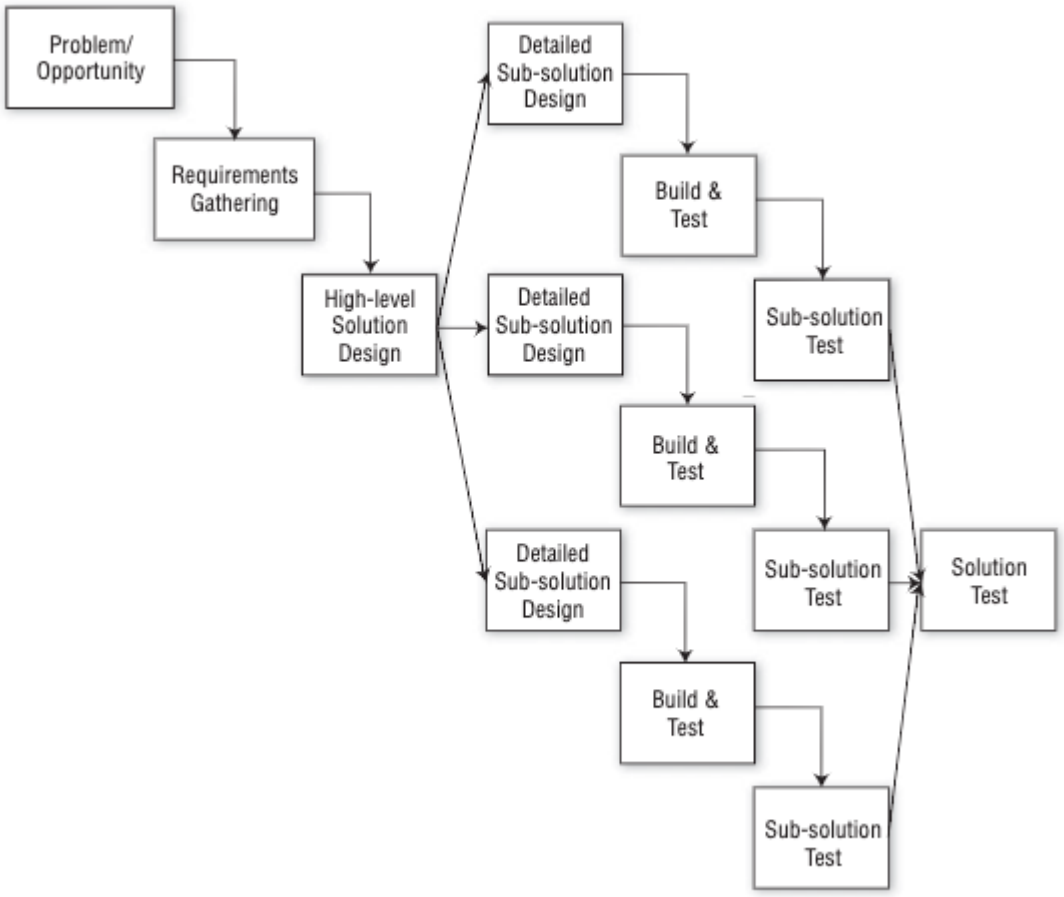


Figure 2: Implementation steps to develop a large computer program for delivery to a customer (Royce, 1970)

However, this type of project management was argued to be inadequate for software development nor did it fit very well into other parts of IT. Royce (1970) argued when he proposed the model that it *"[...] is risky and invites failure."* as it does not handle change effectively and might require major redesign if the requirements were not managed well enough. More modern models were developed to be able to produce software faster. One such example is the Rapid Development Waterfall model which can be seen in figure 3.

Figure 3: The Rapid Development Waterfall Model (Wysocki, 2010)



This model proposes to split the workflow into parallel swim lanes that can be developed separately from the others in an attempt to shorten the schedule for the development of the program. In the end, this can be seen as a riskier model than the waterfall model and might easily cause delays (Wysocki, 2010).

Both the Waterfall model and the Rapid Development Waterfall model are linear models as they have a set number of phases that the project follows until the completion of the product (Wysocki, 2010). Traditional project management also has incremental models. Example models are Staged Delivery Waterfall model and Feature-Driven Development model. The incremental models do not differ much from the linear ones (Wysocki, 2010), however, it is important to mention that there are different models that go under the umbrella term traditional model. In this thesis, waterfall and traditional model will be used interchangeably and they will refer to the type of models that were just mentioned.

The traditional models were not effective enough for software development and the products became expensive and they also suffered in quality (Carlson et al., 2012). Practices were implemented to better manage these fast changing projects. During the 90s many of the frameworks that are popular today, were introduced (McCormick, 2012). Since *The Agile Manifesto* was introduced in 2001, the way software and other IT projects were managed started changing fast as these agile values were being adopted throughout the industry (Bjarnason et al., 2011a; Carlson et al., 2012; Hass, 2007). The agile manifesto does not have any strict guidelines. It only suggests to value certain things over others so that developers can handle changes more efficiently (Fowler et al., 2001). Agile spawned new tools and methods for managing projects. Two of the main reasons why agile methods were adopted is to give the clients value from the project as soon as possible and also to reduce risks (Bakalova et al., 2011). In the software engineering industry, agile is used to such a degree that if you are not familiar with the agile methodology then you are probably not suitable for the industry (Lopez Lorca et al., 2018).

The project models for agile follow the agile methodology. Agile models come in two forms. They can be iterative or they are adaptive (Wysocki, 2010). An iterative model is suggested when most of the requirements are known about the product, however, some features still need to be discovered during development. Evolutionary Development Waterfall model is a good example of such a model (Wysocki, 2010).

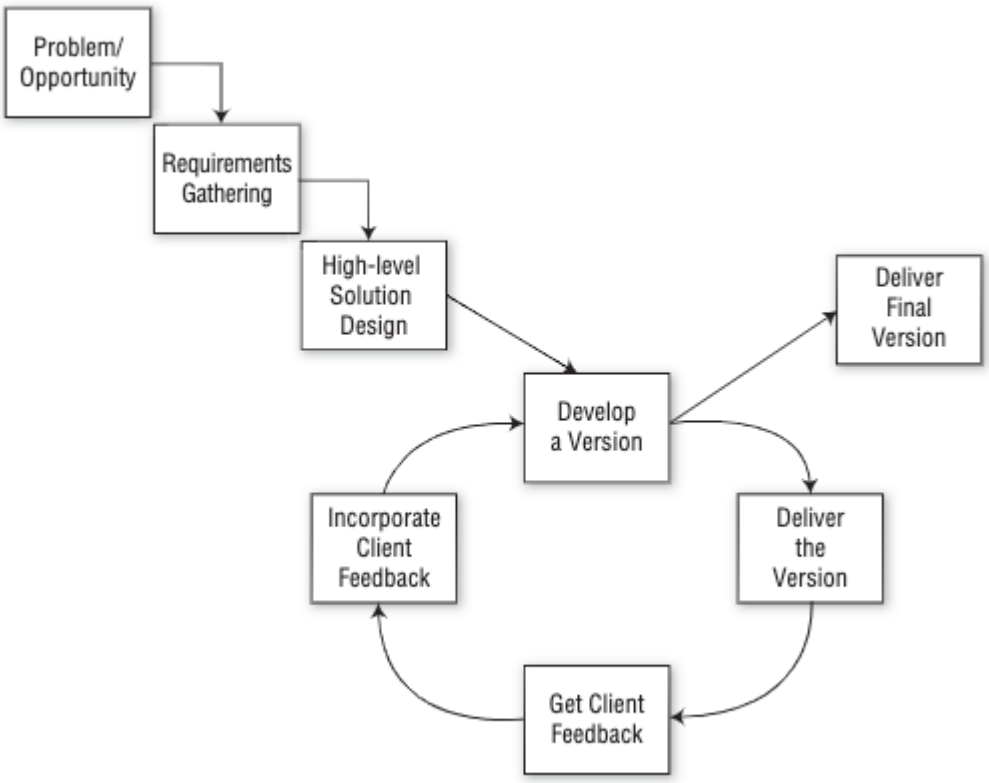


Figure 4: Evolutionary Development Waterfall Model (Wysocki, 2010)

The other model category is the adaptive-model category. These models are intended for projects where the intention is to discover the requirements during development. Popular frameworks that make use of adaptive models are Scrum and Adaptive Software Development (Wysocki, 2010).

Even as agile methods are used more often in projects, it is also good to know that traditional methods are not yet gone. The reason why traditional methods are still used are plenty. Not all projects or businesses are suitable for using agile methods. If the requirements cannot be flexible, there are legal reasons or the business processes are based on contracts, there might not be room for the use of agile methods (Coram & Bohner, 2005) This mindset has changed over time though. The change from traditional, to hybrid and forward to using agile methods has been a struggle for businesses, however, changes in the market forced businesses to change (Ketterer et al., 2016).

4. AGILE REQUIREMENTS ENGINEERING

The previous chapter went through what a project is and what components projects usually contain. One such component is requirements engineering. This is the process of collecting, documenting, analysing and managing the requirements (Paetsch et al., 2003). For all types of projects, requirements engineering is an important aspect of how the project will be implemented. To effectively collect and communicate requirements is a problematic part of projects (Lopez Lorca et al., 2018).

As requirements engineering is the main topic of this thesis, this chapter will contain a thorough summary from the literature regarding requirements engineering. The literature summary will include material of both traditional and agile requirements engineering but the main focus will be on agile and how agile compares to traditional requirements engineering. A significant portion of the source material will be about software project management as that industry was the first area to adopt agile methods (Carlson et al., 2012). The software industry saw a strong adoption of agile methods over more traditional ones (Bjarnason et al., 2011a). As agile methods have become more mature, the methods have been adopted into other types of projects as well (Carlson et al., 2012). The amount of research that goes into those areas is more limited though.

The material about agile is rooted in the Agile Manifesto (Fowler et al., 2001). Many of the articles and books used as source material in this thesis, directly cites the Agile Manifesto, others mention tools that have been developed with the Agile Manifesto in mind. The Agile Manifesto was not the start of agile, however, since the Agile Manifesto was published, an ever increasing number of projects have started using agile throughout the industry (Carlson et al., 2012). Today, agile development has become mainstream (Shim & Lee, 2019).

Compared to the traditional methods, agile methods incorporate flexibility into the workflow. This ability to change the project during its life cycle is connected strongly to requirements engineering. As the project matures, more about the actual desired outcome will be known. This new information will result in new requirements and reprioritization of already known requirements (Ji & Sedano, 2011; Sedeño et al., 2017; Wysocki, 2010). Which in theory might offset some shortcomings that arise with traditional requirements

engineering methods, however, Fernández et al. (2016) suggested that agile might not solve the problems of traditional methods but it might make them more obvious.

4.1. Definition

Requirements engineering is the process where requirements are determined, documented and managed (Paetsch et al., 2003). A requirement is something that needs to be fulfilled for a project to reach one of its objectives (Project Management Institute, 2017). There are different types of requirements. What these different types are and how they play a role in agile requirements engineering, will be explained later in this chapter. The requirements will be based on the stakeholders desired outcome of the project (Project Management Institute, 2017). To state it in another way is to say that the requirements are there to fulfil the goals of the project - goals that the stakeholders have placed on the project. The term requirements engineering is defined in the systems and software engineering standard (ISO/IEC, 2018). In project management, requirements engineering can go under different terms depending on which project management framework or process was used. The Project Management Institute's material refers to this process simply as collecting requirements (Project Management Institute, 2017). This process can also be referred to as requirements analysis (IEEE, 1990). This thesis will mainly talk about requirements engineering, however, it is good to keep in mind that depending on the source, this part of a project can go under other terms. The choice for using requirements engineering is mainly because the majority of research papers used as sources for this thesis uses the term requirements engineering.

The requirements of a project is an important component for the work done in a project (Racheva, Daneva, Herrmann, et al., 2010). In traditional project management projects, the requirements are the components that build up the work breakdown structure. The work breakdown structure is a vital part of the initial stage of a project and the work breakdown structure is used to plan the rest of the project (Wysocki, 2010). I.e. without collecting all of the requirements for the project to be finalized, the project cannot receive a budget, schedule, resources or be completed. For projects following agile models, the requirements will change during the project's life cycle, however, at the start of the project there will be a list of requirements collected based on the objectives of said

project. The list of requirements at the beginning of development is not going to be perfect (Shim & Lee, 2019). Once development has started, requirements will be added, removed and reprioritized during the project's life-cycle (Cao & Ramesh, 2008).

4.1.1. Requirement types

The type of requirement is based on what objective that requirement was determined to fulfill. E.g. Colour coding interface buttons so that a user of a system can more easily identify which button is for accepting and which is for declining. The objectives can address something functional or non-functional. Functional objectives are related to what the product of a project is supposed to do and non-functional objectives are related to the constraints and technical objectives of said product such as usability, performance and reliability (Miller et al., 2015). A functional objective would be for example that a system has to show the Terms of Agreement to the end user before they are able to press Agree. A non-functional would be that the system has to be able to handle a minimum of 1000 users simultaneously.

A common criticism for agile is that its ability to identify non-functional requirements is subpar to traditional methods (Farid & Mitropoulos, 2012; Inayat et al., 2015; Ramesh et al., 2010). The non-functional requirements often need to be discovered early in the development process because identifying a non-functional requirement in the later stages of the development might require enormous changes to the plan that will take time and money to account for (Goetz, 2002). However, some of the more recent articles do suggest that agile has matured to be able to handle some non-functional requirements (Miller et al., 2015). New processes are being developed and researched to better accommodate discovering non-functional requirements. One such process is the motivational model where emotions are used to elicit requirements from stakeholders (Lopez Lorca et al., 2018; Miller et al., 2015). This type of model will be examined more closely later in this chapter.

4.2. Review of Literature

This subchapter will review what research has been done in the area of agile project management and in particular requirements engineering. This is to form a baseline of the

state for agile methods used in projects at present, so that we can then compare that to the empirical research done in the Greater Helsinki area. The articles that are included in this chapter have done their research in different parts of the world.

Agility was introduced to reduce cost and to increase both productivity and quality already before the Agile Manifesto was released. However, after the release of the Agile Manifesto, the software industry was fast to adopt the methods that originated from the agile methodology (Carlson et al., 2012). Not all companies were eager to adopt it though. It seems that bigger and older companies were sceptical. However, these companies seemed to be scared that they would fall behind on progress. Therefore, they were curious to start using the agile methods to see how well it fit them. In The Boston Consulting Group's (BCG) article 'The End of Two-speed IT' a hybrid solution was proposed by BCG so that companies could with lower risk venture into agile methods (Ketterer et al., 2016). This was a compromise that, according to them, was a necessary one. Companies needed to support faster and more flexible ways of handling new digital initiatives. As the companies could not risk their core business with going into agile, BCG suggested in the article that one way to adapt agile is to use both agile and waterfall. However, in the same article they claim that the time for using both agile and waterfall seems to be over. Companies can no longer use agile for some parts of the company and waterfall for others. They should start moving into agile across the whole company to stay competitive (Ketterer et al., 2016). This form of project management is easier to understand with software engineering as software is much easier to change and evolve than for example a bridge (Coram & Bohner, 2005). This is true both during the project and after the project has been completed.

This method of only applying agile methods to new or less important areas of the company is not the only way that businesses have tried to incorporate agile while still using the traditional methods that they have grown accustomed to.

Later in this chapter we will discuss challenges with requirements engineering, what causes them and how agile might be able to handle some of those challenges. Inayat et al. (Inayat et al., 2015) identified instances where shortcomings in traditional requirements engineering were patched by using agile methods, however, others have identified issues that agile requirements engineering might introduce (Cao & Ramesh, 2008).

4.2.1. Agile Methods for Requirements Engineering

There are many agile methods used for requirements engineering in projects. Which method is best is related to the type of project. These methods can be very different from each other, however, the thing that they have in common is how they respond to change. As the name implies, agile projects strive to respond to change quickly and effectively. However, not all of these methods will be pure agile and choosing the right type of method depends on the type of project you are working on. In theory, these methods will help project managers to manage risk and keep the project cost down (Coram & Bohner, 2005). Each method can have different tools and processes that fall under the same type of method. Many of these tools and processes will be mentioned and explained, but there are many tools out there and understanding and explaining them all in great detail would require much effort. Moreover, explaining tools and processes in great detail is not the purpose of this thesis. If there is any tool or process that is of interest, then most of these tools and processes have plenty of material out there in the form of educational material or research papers.

Discovering the requirements of a project is an enormous and important task. Failing at discovering requirements accurately and effectively has been speculated to be the reason why many projects fail (Lethbridge et al., 2003). It is also one of the hardest jobs in projects (Abdullah et al., 2011). In traditional project models, the requirements need to be gathered so that the complete work-based schedule can be created. The requirements have to be identified so well that at the end of the project all the specifications have been fulfilled (Wysocki, 2010). If the requirements have not been identified correctly, the result could be that the project falls behind schedule or it has to be cancelled altogether. The approach to requirements engineering in agile projects is more flexible. However, no two agile projects are alike. Projects that are agile can be either iterative or adaptive. What type of agile project fits best depends on what type of product should be delivered and what the Project Manager knows about said product (Wysocki, 2010). The same goes for what methods should be used for gathering the requirements, modelling and managing the requirements as well as how to adapt to changes in requirements.

The research into agile requirements engineering lacks maturity and would benefit a great deal from more empirical research with a focus on practices and industrial cases (Inayat

et al., 2015; Racheva, Daneva, Herrmann, et al., 2010) , however, there is still a great foundation of research that focuses on requirements engineering. This thesis will continue by summarizing the findings from the literature review articles that could be found relating to agile requirements engineering and then also do the same thing for case studies and other materials that focus on agile requirements engineering.

The literature reviews that are referenced in this thesis are *A systematic literature review on agile requirements engineering practices and challenges (Inayat et al., 2015), Investigating the Link between User Stories and Documentation Debt on Software Projects (Soares et al., 2015)* and *Agile Requirements Engineering: A systematic literature review (Schön et al., 2017).*

At the time of writing there could not be found a newer literature review relating to requirements engineering or collecting requirements. However, these are not all of the literature reviews that can be found on the subject. The other reviews were excluded because they were deemed not to add anything substantial to the thesis that the other literature reviews did not include.

In *'A systematic literature review on agile requirements engineering practices and challenges'* Inayat et al., (2015) identify 17 different agile methods for the requirements engineering phase of a project, they also found five challenges that they linked to traditional requirements engineering that agile methods were able to solve. They could also identify eight challenges that faced agile requirements engineering. The conclusion they had from their literature review was that more research needs to be done in this area. They went as far as calling the field *"[...]immature and needs further empirical evaluation of practices in industrial cases"* (Inayat et al., 2015). This literature review was published in 2015 and at the time of writing, this was five year ago. In that time, many more empirical studies have been published relating to agile requirements engineering. However, even more recent studies claim that requirement engineering lacks in-depth research compared to other areas of agile project management (Shim & Lee, 2019).

In the rest of this chapter, the methods used in requirements engineering will be examined. The area of requirements engineering is vast. Therefore, the methods have been divided

into two categories. The first one is how requirements are modelled and managed, also what type of methods, tools and processes there are to help project managers with this task. The second category is about how these requirements are gathered and prioritised. This category has plenty of tools and processes to write about, however, not all tools and processes are equal. Some are more used than others and others are just additions to other tools and processes.

The categories are interlinked, however, the decision to write about them separately was to make it easier to structure the methods in this thesis. As these methods are agile, they all are tied to the client. Therefore, there is a separate section dedicated to the client's role in agile methods. This section will go further into the role of the client and how the client is represented in agile projects. Finally, the challenges of requirements engineering in agile will be explained as well as what methods and progress there is to manage these challenges.

4.2.2. Modelling and Managing Requirements

The Agile Manifesto (Fowler et al., 2001) valued functioning software over documentation, however, that should not be interpreted as no documentation. Requirements can be hard to explain and they can change throughout the life of a project (Lopez Lorca et al., 2018). The methods used for modelling and managing requirements should therefore be able to clearly describe the requirement, be able to handle changes to those requirements and also not be too heavy with the documentation so that the speed of the project is affected.

The agile methods that were identified in the literature review by Inayat et al. (2015) are based on techniques either identified or developed in other studies. Some of these techniques try to do the same thing but they tackle the problem from different angles. For example requirements modelling is a technique used for stakeholders to help them with decision making. Requirements modelling is also used in projects following traditional requirements engineering methods. As the traditional requirements modelling does not adhere to agile methodology, there have been techniques developed to adapt requirements modelling into projects that are agile. Inayat et al. (2015) identified two different techniques for this task. There is goal sketching, which is a technique that has been

developed with techniques relating to goal oriented requirements engineering or GORE (Boness et al., 2008). The other technique is paraconsistent reasoning. This technique aims to be as flexible as possible. Ernst et al. (2012) argues that conflicts and obstacles cannot always be resolved and therefore they propose the paraconsistent solution to work around the issue that this presents to the requirements model. You can either isolate and ignore the issue until it can be resolved or proceed with one of the conflicting requirements and later in the project correct the model to suit the choice that has been made (Ernst et al., 2012).

Requirements modelling has been used in traditional projects, however, the methods used to form these models have been altered (Inayat et al., 2015). The methods used in traditional projects have been re-imagined to fit the agile methodology. Requirements modelling is a technique that can be valuable to stakeholders. These models can help them make important decisions relating to the project. Something that agile might raise concerns, especially when the project relates to emergent systems. These concerns affect decisions relating to investment and responsibility (Boness et al., 2008). The goal-sketching technique was developed with this problem in mind. This technique starts with forming a goal graph. This graph is a rough estimate of the intention behind the product that the project will produce. The goal graph will give stakeholders insight into the final product before development has begun but after each iteration a stage graph is created. A stage graph uses the goal graph as a guide during each iteration. An iteration is called a stage in goal sketching. Hence the name stage graph. In each iteration, a portion of the goal graph is examined and refined. A stage graph can be updated within an iteration. After each iteration, the goal graph is updated to reflect the work done in the past iteration. This will continue until the final goal graph has been shaped (Boness et al., 2008).

The other technique relating to requirements modelling is paraconsistent reasoning. This modelling technique tackles the problem of how to solve conflicts by being more flexible when analysing requirements and creating requirement models (Ernst et al., 2012). A conflict in requirements is when two or more requirements cannot both or all be solved. It can also mean that none of the requirements can be solved. As requirements are derived from goals that the stakeholders have placed upon the project (Project Management Institute, 2017), it is therefore logical to assume that a conflict in requirements is a conflict

between the stakeholders. Ernst et al. (2012) reason that conflict comes from when stakeholders are inconsistent over a requirement. However, the authors reasons that the conflict is between requirements and if stakeholders are in conflict then it is not over requirements but relating to some other aspect of the project and techniques used. With paraconsistent reasoning, inconsistency is tolerated and considered important. The framework RE-KOMBINE was defined for managing inconsistency problems relating to requirements that were brought by variability and evolution. As a simplification, the framework creates models based on data (refinements, conflicts, tasks and goals) and logic. RE-KOMBINE supports flexibility in requirements model in a way that allows postponement of decisions to a time when more information is available and that new decision can quickly be evaluated when requirements changes, to see how it affects the rest of the project. These changes can be caused by learning new information such as best practices. In the case of a conflict, RE-KOMBINE is able to set this conflict aside and search for a solution without that conflict until a time that new information has helped dissolve the conflict (Ernst et al., 2012).

Goal-oriented models in only one way to create requirements models. Another way is to create motivational models. This lightweight modelling approach that uses emotions to build up a model is great for systems that are designed with persons interacting with technology (Burrows et al., 2019). Motivational models use three words to elicit requirements. They are “do”, “be” and “feel”. They also use the word “who”, however, that is reserved for stakeholders. During a session to discover requirements, the facilitator of the session will elicit responses from the stakeholders by asking them “what they want to do”, “how they want that product to be” (design) and also “how they would like to feel”. After these requirements have been discovered, they are prioritised. After they have been prioritised, the results will be summarized into a result. The results will be reviewed at a later date by the participants of the session. At that point the participants can further explain what they meant and give general feedback (Lopez Lorca et al., 2018).

4.2.3. Gathering and Prioritising Requirements

As flexibility is an important aspect of agile projects, the process of discovering these requirements should also follow this philosophy. In traditional requirements engineering, collecting requirements is done before the project enters the development stage. This

requires a detailed plan to be created and approved before development starts. If problems to the approved plan are discovered during development, changes to the whole project plan has to be made. This will cause both resources and time to be wasted. (Wysocki, 2010). As traditional models do not incorporate feedback throughout the project, these changes might not be realized as soon as they could have been. Agile values interaction and collaboration with stakeholders. Changes during development is also encouraged in the agile philosophy (Fowler et al., 2001).

A great way to collect requirements is by creating user stories. It helps to open up a discussion with the customer about the requirements (Bjarnason et al., 2011a). Using this method might help capture the user perspective of using the product. User stories is a popular method used for collecting requirements (Daneva et al., 2013). This method usually focuses on the goals of the user (Haugset & Stalhane, 2012). The user stories have a simple template that they usually follow *"As a <role>, I want to <goal>"*. A benefit can also be added to the template if needed. With this addition the template is as follows *"As a <role>, I want to <goal> so that <benefit>"(Lin et al., 2014; Shim & Lee, 2019)*. This method is used to describe a desired functionality or enhancement from the product. E.g "As a customer, I want to add products to the shopping cart so that I can later buy them". Anybody that is part of the project can create these stories, however, it is usually the product owner that has the main responsibility for them (Lin et al., 2014).

The user stories go through iterations so that the impact of implementing the user story is known. After the project team is happy with the user story, it is presented to the customer. Once approved, the story is added to the backlog of the project. As it would be hard to implement the user story as such, the story is broken up into small tasks which the project team later develops (Lin et al., 2014).

The concept of user stories was further developed into delivery stories (Daneva et al., 2013). Delivery stories are user stories but with functional specification, high-level design and test scenarios added to them. In theory, this would help to deliver the user story (Daneva et al., 2013). Adding test scenarios to a user story is not only beneficial for testing the feature but it also helps the whole project team verify that they are understanding the user story in the same way. Test scenarios also give the developer a way to check when the feature can be said to have been delivered (Haugset & Stalhane, 2012). This method

of adding tests to user stories is sometimes called testing before coding. Known processes that are used for this are test-driven development, behavior-driven development and acceptance test-driven development (Moe, 2019). With test-driven development the developer creates tests for parts of the functionality of the program before the developer starts coding. Behavior-driven development focuses on the behaviour of the program. What the behaviour should be is determined with the customer. The behaviour is then verified throughout the development of the program. Acceptance test-driven development has seen an influx in popularity lately (Haugset & Stalhane, 2012; Moe, 2019). The purpose of acceptance test-driven development is to promote communication throughout the team (Moe, 2019). These tests have been set up with detailed communication with the customer, which is a difference from test-driven development where the developer writes the test. When a test is run against a feature, that feature should give the right response. If it does not give the right response, then either the developers have not understood the feature or they have done something wrong during development. This way they will know that this feature is not yet deliverable without talking or having the customer test the feature (Haugset & Stalhane, 2012).

To further assist the effectiveness of user stories, there have been studies that would suggest that when collecting, evolving and clarifying requirements within the team through communication, a deeper understanding of the product's requirements can be had. Abdullah et al. (2011) gave this process the term shared conceptualization. This is because the understanding is formed within the team through collaboration.

Another method identified for agile requirements engineering is iterative requirements (Inayat et al., 2015). It is difficult and often even impossible to know the requirements at the start of a project, therefore, agile requirements engineering suggests that the requirements will be discovered and refined throughout the life of the project through collaboration with the customer (Cao & Ramesh, 2008; Ramesh et al., 2010). To not lock down the requirements but letting them form naturally throughout the project's life cycle, has had the benefit of making the requirements more stable. When the requirement needs to be implemented in the project, they have been added to the development. This has also improved the scope of the project by making the problem of overscoping not that

prominent. Overscoping is when more features than are needed or feasible are added to the project (Bjarnason et al., 2010, 2011a).

Not all requirements are equal, and they cannot all be delivered at the same time if you are following agile development. Each requirement has a value to the product and the more value that can be achieved by fulfilling a requirement, the higher this requirement should be prioritized. How much a requirement is actually worth to the end product is not that clear and as more information is collected throughout the life cycle of the project, these priorities need to be reassessed. This is a stark contrast from traditional requirements engineering methods used in for example Feature-Driven Development model where requirements are prioritised only once and that is before the development phase starts (Wysocki, 2010).

To be able to prioritise requirements it is vital that requirements are discovered and collected. Agile allows that development be started without knowing all the requirements and lets requirements be added and removed during development and even after. In a simplified view, the requirements go through two phases. They are discovered and developed. These two phases are repeated throughout the project (Shim & Lee, 2019). The delivery phase will try to deliver the most value to the customer at any iteration. There have been plenty of agile methods identified for collecting requirements. They vary but they all have incommon the principles of the Agile Manifesto that development should value interaction and collaboration with the customer (Fowler et al., 2001). These methods do not only identify requirements, but they also introduce changes to existing ones (Inayat et al., 2015). Collecting requirements in agile projects value customer collaboration. The requirement gathering methods that Inayat et al. (2015) identified all include collaboration with the customer in some form. Face-to-face communication between the project team members and the customer is valued in agile requirements engineering. Improving the communication with the customer gives the customer the ability to steer the project because they are able to understand the project better. The requirements will gradually be identified and then communicated within the team and with the customer so that they can efficiently be agreed upon (Bjarnason et al., 2011a). However, it is not only with the customer that there must be communication. In agile projects, the teams developing the product are cross-functional teams. This means that

people with different backgrounds, specialities and tasks sit together. The manager is also a part of the development team as well as a customer representative (Bjarnason et al., 2011a).

Continuous planning is when the development team doesn't have a fixed plan but planning the job is a recurring task. Therefore, the development team can avoid trying to fit changes into the plan as the changes were already integrated into the process. Also, when there is continuous planning there is always communication with the customer, this could lead to cases where major changes could be avoided altogether (Jun et al., 2010).

Feedback is an important factor of agile project management and the methods Inayat et al. (2015) identified reflect that. Feedback is included in the way the teams are formed, how the communication with the customer is handled and also how the product is tested. One of the ways to receive feedback is to use prototypes. A prototype is an early version of the product that is being developed (Paetsch et al., 2003). These prototypes are usually classified as low or high fidelity. A low-fidelity prototype can be as simple as paper sheets with static interfaces on them that the user can simulate the user experience with. High-fidelity prototypes can be something close to an end product (Rudd et al., 1996). Low-fidelity prototypes might not have any value out of collecting requirements but high-fidelity can be used for collecting requirements and also be a part of the final product. Evolutionary prototypes are functioning prototypes that the customer can work with throughout the project and can also become part of the final product (Paetsch et al., 2003)

Prototyping can therefore be used to overcome the challenges of requirements validation. The purpose for requirements validation according to Paetsch et al. (2003) is to *"[...] certify that the requirements are an acceptable description of the system to be implemented."*. Requirements validation is also a challenge with traditional requirements engineering (Inayat et al., 2015). A possible solution for this is to continuously reprioritize the requirements. After each iteration of the project, a product is produced. This product with the help of user stories are used to demonstrate to the stakeholders that intended requirements were fulfilled. This gives stakeholders the possibility to voice concerns and give other forms of feedback to the project team.

Another method of feedback and requirements validation in agile projects is the use of retrospectives or review meetings. As stated before, the agile development process is based on iterations. The final product will come together in increments. In the Scrum framework for agile development, after each iteration there is a retrospectives meeting. To understand how the retrospective meetings function in Scrum, it will help to first understand other components of the Scrum framework. The Scrum framework has all the components of a project, however, they have their own terms and definition for them. For clarification, the roles used in Scrum will be defined and then the roles of Scrum will be linked to roles of project management in general. The main three roles in Scrum are Team, Scrum Master and Product Owner. The Team is the development team. They fall into the normal characteristics of a team working on an agile project. They should be self-organised and have cross-functional skills (Carlson et al., 2012). Scrum Master is the closest to a Project Manager, however, Scrum Masters do not function as a traditional Project Manager. The Scrum Master's main objective is for the team to work efficiently by removing obstructions. The Scrum Master will also plan the meetings and see that the Scrum framework is followed (Carlson et al., 2012). The Product Owner is someone or multiple people that represents the customer or the end users of the product. The reason to have more than one person for this role is to fulfill all domains and technical knowledge needed by the team. The Product Owner can also be the customer. Their main task is to collect requirements to the project's backlog (product backlog in the Scrum framework) (Carlson et al., 2012).

The closest role to Project Manager in Scrum is the Scrum Master. It is their responsibility to organise the retrospectives meeting. This meeting should be four hours or less. The purpose of this meeting is for the team to discuss what they have developed and what goals they have fulfilled during the latest iteration of the project to the stakeholders (Carlson et al., 2012). This will inform the customer of the state of the project and it is also in this meeting that the team can demonstrate the current product to the customer. This meeting also functions to transfer knowledge to the next iteration, however, the purpose of these meetings are not to identify error, resolve issues nor is it meant for planning. At the end of each retrospectives meeting, the Product Owner, the person or people representing the customer or end user, will either accept the work of the iteration or postpone the acceptance until the product is more mature (Carlson et al., 2012). Even

though one of the objectives with the retrospectives meeting is to receive feedback and to discover issues, in reality, the value for the development team from these meetings might be minimal. In Ramesh et al., (2010) case study, only minor issues were discovered during these meetings and to the case study interviewees it seemed that these meetings mostly functioned as progress reports to the customer. The case study also implied that a better way to receive feedback for work done is to keep constant contact with the customer throughout the project.

In agile, this form of collecting requirements is prefered over granular documentation. This will give the stakeholders of projects more control to change the project.

4.2.4. The Client's Role in Agile Methods

To form an understanding of the product that the project will produce, there needs to be someone that judges what is valuable for the end user of the product. With agile, the philosophy is to prioritise interaction within the team and collaboration with the customer as stated in The Agile Manifesto (Fowler et al., 2001). What the requirements are and how they are prioritised will therefore fall upon the client. The client is a stakeholder of the project. The client is not the only stakeholder of projects but they carry the responsibility of assigning importance to requirements (Bakalova et al., 2011). The client will function as a substitute for the end user as their judgement will assign a value to each requirement (Bakalova et al., 2011). However, requirements prioritization is a complicated task and more research needs to be done to understand how it affects decision making (Bakalova et al., 2011).

How much the client is involved in practice is not that clear. According to Bakalova et al. (2011), the literature relating to agile project management states that the client's involvement has never been higher as it is when using agile methods. However, the research done by Racheva et al. (2010) discovered that the responsibility of requirements often falls on the developers. In their case study they could only find one case where the client was actively involved in the decision-making process of the project. Even then, the representative from the client side could not deliver the desired information that the developers relied upon (Racheva, Daneva, Sikkel, et al., 2010). However, in Cao's and Ramesh's (2008) case study into multiple businesses, they found that pressuring the

customer to participate in the requirements process helped the team and the customer come to an agreement over the requirements. Being involved also helped the customer to become satisfied with the project. They felt that being part of the requirements process helped the project team to deliver high value throughout the project life cycle (Cao & Ramesh, 2008). This seems to indicate that involving the customer, even if there is resistance in the beginning, has a positive effect on the product and also with the relationship between the project team and the customer. Therefore this relationship should be encouraged. This is not the only benefit that Cao and Ramesh (2008) identified, they could also see that collaboration and feedback throughout the development phases, decreased the likelihood of big changes to the product after development had ended. Agile allows for changes even after development. However, it seems that collaboration during the projects makes the possibility of changes after development less important.

Communication skills are not every person's most defined skills. Stakeholders can have a hard time communicating with other stakeholders. This is an obstacle that must be overcome for agile to be as efficient as possible. This problem has been particularly prominent when communicating with non-technical clients or end-users. There have been processes to help along with this. As an example, motivational models are designed to elicit a response from the client or end user. Burrows et al., (2019) noticed in their project that stakeholders would only need to be presented with the model to find something that they would like to talk about. They believed that it was because of the use of motivational models that made the stakeholder comfortable to talk. This would be particularly effective if the topic was embarrassing to talk about, the authors of the article speculated.

4.2.5. Challenges with Requirements Engineering

In the section about agile methods we discussed different methods for requirements engineering. We also touched upon some challenges with traditional and agile requirements engineering. In this section we aim to go further into the subject of challenges and form an understanding of when agile requirements engineering might be a good option and when it might now.

Because of the importance of requirements engineering for projects it is important to identify and solve challenges that come with requirements engineering. The more

research that is done, it stands to reason that we have a better understanding of the subject. Case studies have been important in finding these challenges and potential solutions to them. Some of the challenges that have been identified with the traditional project management methods might have solutions in agile methods. The reason why challenges in both traditional and agile project management are raised in this thesis is to investigate the importance of adapting to new methods when new information has been discovered. Agile requirements engineering has the potential to solve problems that arise when using traditional project management. It is also important to note that agile is not without its own challenges (Cao & Ramesh, 2008; Inayat et al., 2015; Mendes et al., 2016). Fernández et al. (2016) stated that agile might not solve the problems of traditional requirements engineering but it might make them more visible.

The ability to overcome these challenges will give project managers better abilities to finish projects on time and within budget (Inayat et al., 2015). It is also likely that it will increase motivation in the project team and within the whole company .

Another reason to mention both traditional and agile methods is that the industry has tried to use agile methods to fix shortcomings in traditional requirements engineering (Inayat et al., 2015). The industry has also tried to fix shortcomings in agile requirements engineering with traditional methods (Boness et al., 2008). There is no silver bullet and therefore it is good to be aware of the limitations with traditional and agile methods.

Inayat et al. (2015) identified five challenges with traditional requirements engineering that agile was able to resolve. They also identified eight practical challenges with agile requirements engineering. Many of the challenges that agile might have the ability to resolve are related to better communication, both within the team and with the customer.

Limited communication is a challenge within itself with traditional projects but it also causes other issues such as overscoping which will be discussed later in this chapter (Bjarnason et al., 2010). The Agile Manifesto (2001) lists interaction with individuals and collaboration with the customer as two of the items it values. Therefore, it can be reasoned that communication is an important part of the agile mindset. In the agile framework Scrum, there are multiple ways to accommodate communication within the project. One of them is the daily stand-up meeting. It is a quick meeting where each member of the

team answers what they have done since their last stand-up meeting, what they plan on doing today and if there are any issues that need to be resolved before they will be able to accomplish their work (Carlson et al., 2012). Another meeting is the iteration planning meeting. This meeting is held at the beginning of each iteration. In the meeting the team plans together what tasks must be finalised during this iteration. At the end of each iteration, a retrospectives meeting is held. The purpose of these meetings is to transfer knowledge to the next iteration and also share status with the customer (Carlson et al., 2012). All of these meetings are set up to inform, transfer knowledge and/or partake in feedback sessions.

The Agile Manifesto values communication because the authors believe that good communication profits the whole project. We will focus on how communication can benefit requirements engineering and how poor communication can affect requirements engineering. Bjarnason et al. (2011b) discovered through interviews with practitioners in the software-development market numerous communication gaps that affect requirements engineering. They also discovered what causes these gaps and what effect these gaps will have on the project. One of the issues that was discovered related to the complexity of the product. Because the product is complex, the amount of different roles in the project was large. Therefore the team had a hard time drawing a clear picture of the project as they did not clearly understand the responsibility of the different roles. This gap in communication affected the requirements (Bjarnason et al., 2011b). Over time this issue seems to become worse as well. As their research seems to indicate, the problems became worse over time as work and responsibility changes over time within the life cycle of projects. During hand-overs between roles, important requirements might not be transfered or they might be misinterpreted.

The effects of these communication gaps are plenty. Bjarnason et al. (2011b) identified nine effects that are related to requirements engineering. These issues affected the customer, the motivation of the team, the quality of the product and also caused unnecessary work. Insufficient communication can cause requirements to not be fully understood or might not be noticed at all. Communication with the customer seems to be specifically hard. To get the customer to agree on a requirement can take time. When they finally agree on a requirement, it might be that the requirement has become outdated.

There might also be cases where the customer has been promised a feature that the developers have not signed off on. The task might be too big to finish or it could be impossible. Therefore, the developers will not be able to implement that feature and it will cause the customers' expectation not to be met. In other cases, changes to requirements are not communicated fast enough throughout the whole project team. This would result in unnecessary work because the work that was done was related to a now outdated requirement. In other cases communication gaps would cause frustration within the project team (Bjarnason et al., 2011b).

Overscoping is also caused by communication gaps (Bjarnason et al., 2010, 2011a, 2011b). Overscoping is when too many features are added to the project. These features are either too labour intensive, not needed or they can not fit the schedule (Bjarnason et al., 2010). The challenge with overscoping is to some degree minimized by agile project management because agile methods help with overcoming communication gaps and it allows for change throughout the life cycle of the project. Bjarnason et al. (2011a) have found evidence that would indicate that agile methods such as continuous reprioritization of requirements and improved communication with stakeholders has addressed some of the problems with overscoping. The risk of overscoping is still there, however, it can be managed better as there is more visibility to the scoping process.

One of the items that the Agile Manifesto (Fowler et al., 2001) values is *"Working software over comprehensive documentation"*. This does not mean that there should be no documentation, however, the focus should be on functioning software. Inayat et al. (2015) identified multiple issues related to deficient documentation. The issues seem to become worse if the customer is not present and there are changes to the requirements.

The issues of insufficient documentation seem to spawn from misinterpreting the Agile Manifesto (Voigt et al., 2016). However, it also might be that the tools used for documentation should become more compatible with the agile methodology (Lethbridge et al., 2003). It seems that wikis have been implemented as a collaborative documentation tool for agile projects and for the purpose of documentation of requirements, it functions quite well (Voigt et al., 2016). However, agile projects might benefit from a documentation tool that is better interconnected to the tools used within projects (Voigt et al., 2016). No dedicated widespread documentation tool for agile seems to be used yet

in the industries that have adopted agile. However, there is research being done to develop one (Voigt et al., 2016).

A more detailed documentation of the requirements can be beneficial if someone outside of the development team needs to understand these requirements (Goetz, 2002). Agile does promote having a closer relationship with the customer, however, that is not always possible. One artefact in agile that benefits documentation is the use of user stories. Earlier in this thesis when discussing user stories, a few additions were mentioned that could add value to user stories. These additions to the user stories does contribute to the documentation of the project.

Limited documentation raises the likelihood of the issue of documentation debt which is not as prominent in traditional as it is in agile projects (Mendes et al., 2016). Documentation debt is a form of technical debt. The debt is a metaphor meaning that the developers of the system have neglected parts of the development process to speed up the development. The project manager or developer could have been aware of the technical debt they took on and weighed the risks of that debt. I.e. intentional debt. However, some debt is unintentional. This debt could accumulate because of not following procedure, lack of attention or just not doing enough tests (Seaman & Guo, 2011). This debt might at some point have to be repaid, usually by having to do more work later (Seaman & Guo, 2011). Documentation debt is when documentation has been neglected in favor of development to such a degree that it causes problems with development further in the project or when the product is already in production (Seaman & Guo, 2011). In Mendes et al. (2016) case study into the subject of documentation debt and what kinds of issues this would cause, they identified 65 out of 132 maintenance tasks of a recently finished software project to be related to documentation debt. The list of tasks that were identified related to this software were observed within a 18 month timeframe. Out of the tasks that were related to documentation debt they further categorized the tasks into three categories. They were lack of information, volatility of requirements and lack of non-functional requirements identification. They further reasoned that the lack of non-functional requirements identification is also a form of lack of information. Out of the 65 tasks that were identified, 42 tasks were categorized to be related to lack of information, volatility of requirements counted for 18 tasks and the remaining five were related to lack

of non-functional requirements identification. As lack of non-functional requirements identification was also categorized as a lack of information, 47 out of 65 of the tasks were related to lack of information. This was estimated to take up 455 hours to resolve these tasks. Which is a significant amount of work over the 18-month timeframe. It is hard to say to which degree these tasks could have been prevented with better documentation and would that task be worth it. Let us say that the total maintenance hours related to documentation debt over the lifetime of the software is 1000 hours and to not have said issues would require more than 1000 hours of documentation, then it seems to be of more value to just not do the documentation. However, if documenting for 200 hours would cut the maintenance task in half, then it would seem that the documentation would have more value. This is a very shallow way to look at documentation debt. There are many more variables to take into account to determine if it is profitable to take the debt or pay it off. If the "interest" of the debt has to be paid during development to such a degree that the software suffers, then debt has to be managed better (Seaman & Guo, 2011).

This was only a single software project, however, it seems to indicate that documentation debt can be associated with higher maintenance effort and cost (Mendes et al., 2016).

5. EMPIRICAL RESEARCH

When it comes to software development, it seems that agile methodologies have taken over. In the Greater Helsinki area there are plenty of companies offering training in agile methods used for software development. These course topics include Scrum, Kanban and agile leadership. The use of agile methodologies are quite widespread in software development (Hess et al., 2017). They have also found use outside of software development projects (Carlson et al., 2012).

This chapter will go through the empirical research that was performed to collect data about the methods used for requirements engineering and what issues project workers have identified. The aim for this research was to investigate if agile methodologies can be found in the sample and what impact agile has had. The data collected from the sample was compared to the literature about requirements engineering.

5.1. Requirements Engineering in the Greater Helsinki Area

The environment for this research was the Greater Helsinki area in Finland. The sample was compiled from people that are working with projects from that area. The research was a mixed-method study. The first data-gathering method was a questionnaire. People in the Greater Helsinki area with project related tasks in their daily work were asked to fill in a questionnaire with questions related to requirements engineering methods and challenges.

It seems that companies, start-ups, researchers and private persons are always collecting data through questionnaires through different media. Acquiring respondents to a questionnaire can be labouring and will probably also be time consuming. As there was no clear direct incentive for people to fill in this questionnaire, the questionnaire would have to be as easy and as fast as possible to answer. Furthermore, convincing people to help spread the questionnaire would probably help to receive more responses.

To form a deeper understanding of agile methods of requirements engineering in projects, interviews were conducted with people that are either Project Managers or they are involved in training project management in the Greater Helsinki area. As the title Project Manager is not always used when working in agile projects, Scrum Masters and other roles with similar roles as Project Manager were combined into Project Manager.

The questions this thesis tries to answer are:

> *RQ1: Has agile requirements engineering impacted project management?*
>
> *RQ2: Does the product benefit from using agile requirements engineering?*
>
> *RQ3: What relation does the method of requirements engineering have on the success of a project?*

The plan was to collect data from a small sample of employees that work with projects in the Greater Helsinki area. When using a questionnaire to gather data for research purposes, it is important that the questions in the questionnaire reflect research objectives (Denzin, 2017). In this case, the research objectives are the research questions listed above.

Data collected from the demography, would then be compared with the literature to examine how well the sample collected would reflect the findings of past studies about the subject of requirements engineering. To collect data, the questionnaire was distributed to people working on projects in the Greater Helsinki area. Based on the data from this questionnaire, the interview questions were formed. The questionnaire was only in English even though Finnish is the majority language in the Helsinki area. This choice was made to minimize confusion because of potential translation errors. The terms in Finnish might also be less known than the English counterpart and that people would use the English terms more than they use the Finnish ones. There is also the possibility that some terms have not been translated yet to Finnish and mixing Finnish and English terms can be messy and therefore add confusion.

The purpose of the questionnaire was never to form a deep understanding of the project environment in the Greater Helsinki area. The purpose was to examine the state of agile project management in a sample collected from the Greater Helsinki area. It was also the purpose to collect opinions about agile project management. The sample size was too small to form any base line of the demography, however, it can be studied in relation to literature to see if there are any indications to the methods used and what are the problems that they are experiencing.

The questionnaire had twelve questions. Eight of the questions had predefined answers for the responder to choose from. The other four questions the responder could write in their own response. These four questions were optional so that the responder could choose to answer them or just leave them blank. The purpose of these questions was to catch answers and opinions that were not included in the questionnaire and also to collect subjects to talk about in the interviews. The research is limited to requirements engineering, so all of the questions revolved around that theme. To validate the questionnaire and to catch mistakes, the questionnaire was sent to the supervisor of this thesis before it was distributed to the public. The questionnaire was electronic and spread through the social media LinkedIn via direct message and also through posts on LinkedIn. The questionnaire was also distributed through other media. This type of sampling is called convenience sampling because the sample was chosen based on easy accessibility to the researcher and the participants chose to be part of the sample (Etikan et al., 2016). The only criteria was that it should mainly reach people that live in the Greater Helsinki area. If the questionnaire did reach someone outside of the Greater Helsinki area, the second question in the questionnaire filtered them away as it asked the responder what area they were from. If the responder chose another option than the Greater Helsinki area, they were excluded from the data set.

The purpose of the interviews was to form a view of project management and requirements engineering in the Greater Helsinki area. The main interest was to observe what tools are used and also to listen to their opinions of the tools and how well these tools are used. The interviews were semi-structured because the purpose was to have the interviewees talk freely about working in projects in the Greater Helsinki area. The participants for the interviews were chosen via convenience sampling. The aim was to

have people from different industries and also from different types of projects. Another focus point was to have people with different levels of experience as that might have added to the diversification of the data that was collected. People fitting these criteria were contacted and asked if they would be willing to participate.

5.2. Result

This portion of the thesis is dedicated to go through the finding from the research. The data for the research is collected from a small sample of project workers in the Greater Helsinki area. The purpose of the research is to see if similarities can be found in the sample as the findings that are presented in the literature regarding requirements engineering. The focus is on the methods for requirements engineering and problems that have been discovered regarding these methods. This data will then be used to analyse this thesis' three research questions. The first topic will be the questionnaire and to see if there are trends that are visible in the questionnaire responses. The questionnaire also included open-ended questions that will also be analysed.

To understand the responses from the questionnaire and also to collect more data from the demography, three interviews were also included in this study. In an interview the interviewee is able to express themselves in a different way than in a questionnaire. The interview also allows the interviewer to ask follow-up questions if the answer does not satisfy or if it is an interesting discussion that might give valuable data to the research.

5.2.1. The Questionnaire

The questionnaire received 39 responses. Out of the 39 responses, 32 were from the Greater Helsinki area. This is a significant amount of responses and trends should be visible from the data. The data from this questionnaire were used as a base for questions for the interviews. So this data did not stand on its own, as the data was corroborated by the findings from the interviews.

The following data will represent the 32 responses from the Greater Helsinki area if not stated otherwise.

For a better view of the responses, the first question in the questionnaire was about the responders role in projects. This was to see how well each role is represented in the questionnaire responses but also to get a better understanding of the answers received. Working in projects might look differently based on the role of the respondent. The data from this question has been analysed with the answers from other questions in the questionnaire. If any findings deemed interesting or important has been found in relation to this question, that finding has been explained later in this chapter when findings from the other questions have been explained.

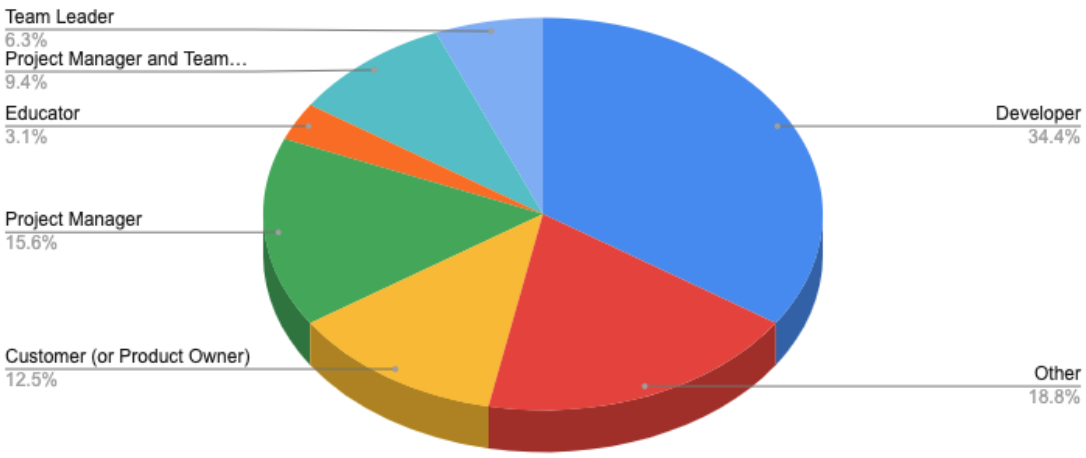


Figure 2: Project Role of Questionnaire Responder (n=32)

The role that was represented the most by the respondents was *Developer* (34.4 %). This is not a surprise as that is probably the role that is most prominent in many projects. The biggest portion of respondents after *Developer* was the group *Other* (18.8 %). This was a bit of a surprise but is probably quite logical. To speculate, this group probably consisted of experts, consultants or roles from projects that are set up differently. In hindsight, an option for consultant/expert should have been added. The differing terminology used by agile frameworks probably contributed to the *Other* group being so highly represented. For example, Scrum, which is a popular agile framework does not have Project Manager or Team Leader mentioned as a role in the framework. The closest to those roles is the role of Scrum Master (Carlson et al., 2012).

Even though Scrum Master was not mentioned as an option, a big portion of the responses came from respondents that identified as having a leadership role in projects. The proportion of respondents that identified as *Project Managers were* 15.6 % and the combined percentage of *Project Manager*, *Team Leader* and *Project Manager and Team Leader* was 31.3 %.

The proportion of respondents that identified as *Customer (or Product Owner)* was 12.5 %. Product Owner was combined with customer because a Product Owner is someone who either represents the customer or end user, or they in fact are the customer. Their main responsibility is to validate and prioritise requirements. Therefore, the methods they use should be similar.

There was only one respondent that identified as an *Educator*. This means that this perspective is underrepresented. This is unfortunate as the role of Educator probably has a unique view into project management. They might have a more objective view as they themselves do not work with projects directly. They teach others to work with projects. They might also have had experience with different types of projects and tools used to manage the projects. Their view of projects in the Greater Helsinki area might also be more of an overview. This is a role that could be more represented in a future study.

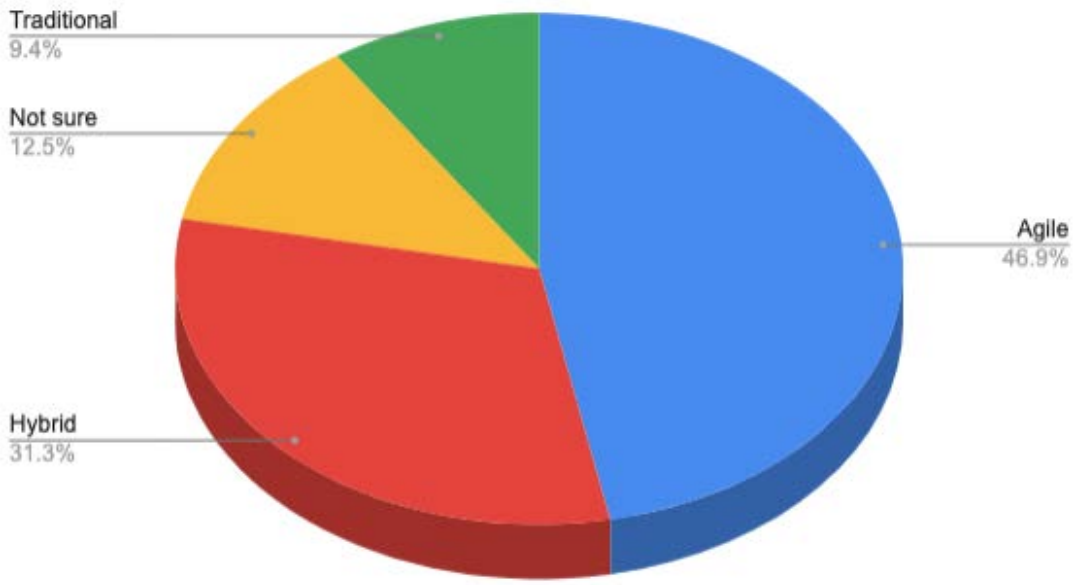


Figure 3: Type of Projects in Greater Helsinki Area (n=32)

To understand the types of projects the respondents are usually involved in, a question was included on this topic. It was also a possibility to see how well the agile methodology has been adapted in the Greater Helsinki area. As the sample size of this questionnaire was small, to form a more realistic view into what methodologies are used, further research needs to be done. The main purpose of this question was, however, to observe how the project model type relates to the requirements engineering methods and also the encountered issues.

The percentage of respondents who identified as working in *Agile* projects was 46.9 %. This follows the trend that the popularity of agile projects is increasing (Cao & Ramesh, 2008; Carlson et al., 2012; Daneva et al., 2013; Ramesh et al., 2010). There are still 31.3 % who use *Hybrid* models, and 9.4 % who use *Traditional* ones. Out of the responses, 12.5 % answered *Not sure*. This means that four people in the Greater Helsinki area who answered the questionnaire are not sure what type of projects they participate in. Two of the participants identified themselves as *Developers*, one as a *Project Manager* and the

last one as *Project Manager and Team Leader*. The reason why these individuals are unsure about the type of projects that they participate in is interesting. This could easily be that they work on many different projects who use differing management styles, and therefore, they are uncomfortable answering which type of model they usually work with. It could also mean that they have not thought about it or that they lack the education of identifying which type.

Hybrid projects still have a big representation in the responses to this questionnaire. It seems that there still might be some use for this type of project in the Greater Helsinki area even though articles have advocated that companies should concentrate on agile when starting new projects (Ketterer et al., 2016). Projects following traditional models have a very small percentage in the responses, however, the model is still present in the sample. What type of project model to follow might have a strong correlation with what type of product the project is set to produce. The reason could also have to do with the organisational structure or the management of the organisation. The reason for using hybrid and traditional models will be explored more in the interviews as well as the reason why some responses do not know what type of project model they are following.

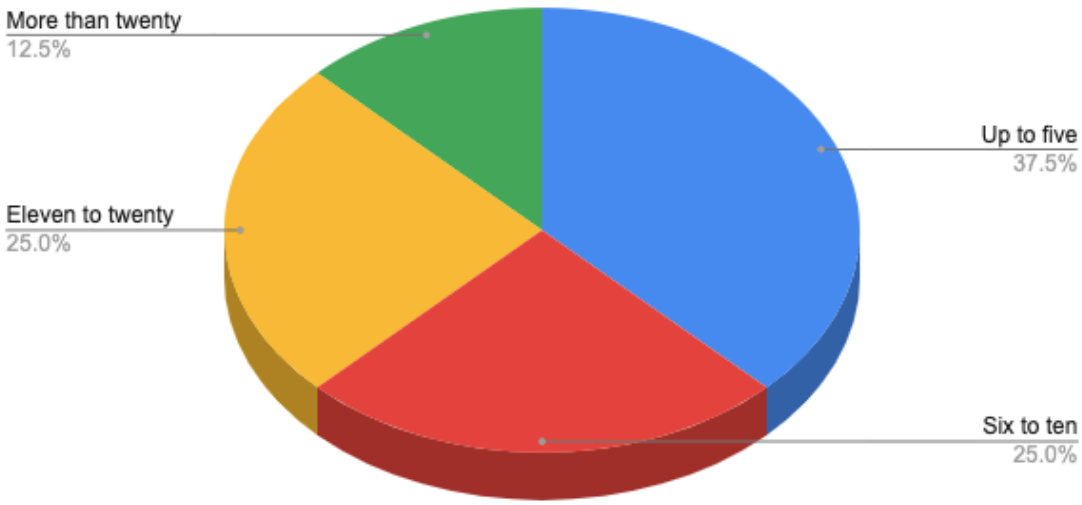


Figure 5: Participants of a Typical Project in Greater Helsinki Area (n=32)

From what vantage point requirements engineering is viewed and in what type of project it is observed, is an important aspect to understanding the data collected. However, how complex a project is will probably also affect the type of methods used to manage a project. The amount of participants of a project does not determine if a project is complex or not. However, more participants in a project does add to the complexity of a managing said project. It is logical to think that managing a project and communicating effectively is more difficult if more people are involved. There was also an optional question related to the complexity of projects that will be explored later.

The responses do represent differing sizes of projects which will help to give a better picture of methods used in various-sized projects and also increase the probability of having responses that have identified different issues with the projects that they have worked on. The amount of participants of a typical project according to the respondents varies. The most common one, with 37.5 %, seems to be *Up to five* participants. Projects with *Six to ten* participants accounted for 25 % of the respondents. This follows the guidelines that one of the more popular agile frameworks, Scrum, states. In Scrum, there are three roles. The role with the most participants is the team which the guidelines states that should consist of five to nine people (Carlson et al., 2012). Adding a Scrum Master and Product Owner on top of that, makes the whole team around seven to eleven participants. Which could indicate that the projects in the Greater Helsinki area do follow the guidelines of popular agile frameworks that push for small and intimate project teams. An interesting aspect of project work in the Greater Helsinki area is to examine what frameworks and processes are used. This will be investigated further in the interview portion of the research.

Eleven out of twenty (25 %) of the respondents answered that *Eleven to twenty* participants was the norm for them and the rest, 12.5 %, answered *More than Twenty.*

As the agile methods mature, as it has rapidly done since the publication of the Agile Manifesto (Carlson et al., 2012), agile will be able to handle more complex projects (Inayat et al., 2015). It seems, however, that 75 % of the respondents that answered that they typically work on projects that have more than twenty participants, that the projects

use a hybrid model. Only one response stated that they use agile even in large-scale projects. This could be an indication that the old ways are hard to give up on or that the projects are more suitable to a hybrid approach. (Ketterer et al., 2016) did state that businesses should move to agile methods throughout the organisation and staying with traditional or a combination with both traditional and agile would in the long run not be the best solution. This is something that will be elaborate more on in the interview portion of the research.

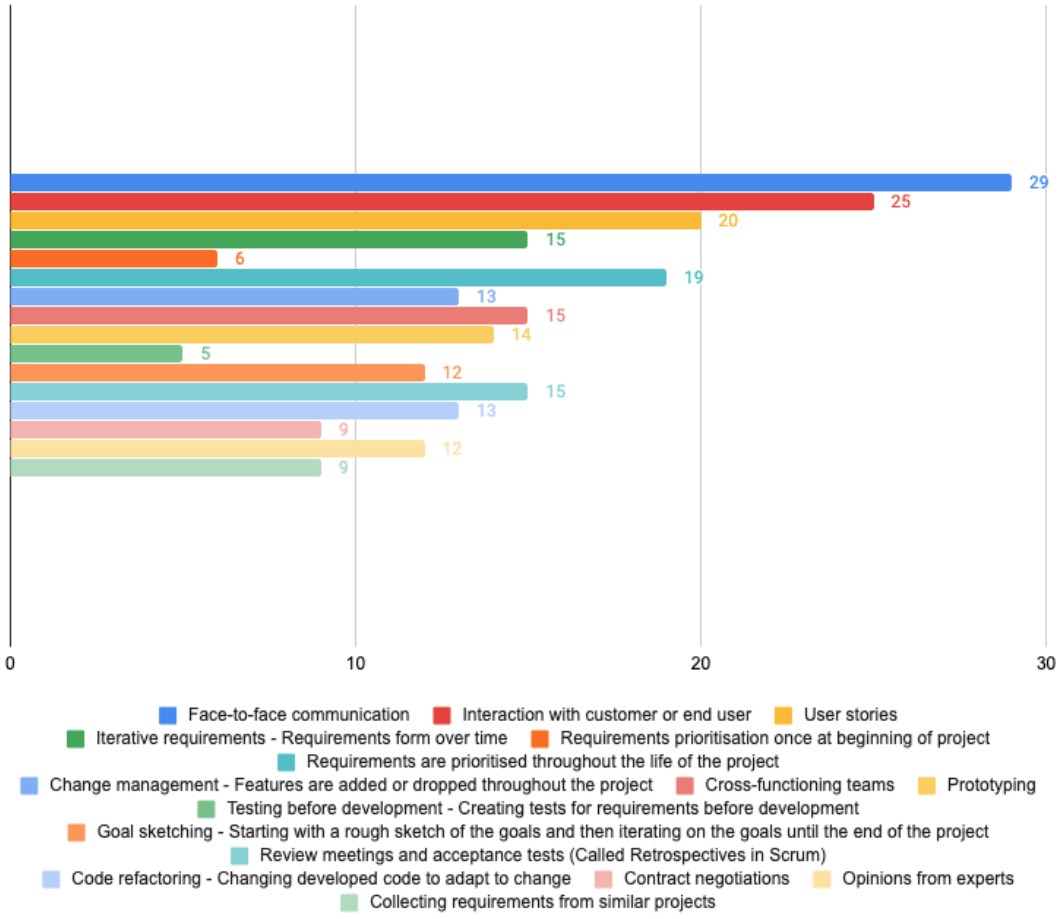


Figure 6: Methods Used to Manage Requirements in Projects in Greater Helsinki Area (n=32)

The question was *What methods do you use to manage requirements in a project?*. The purpose was to collect answers that would show trends in what methods are popular in the demography. This would then be compared to the type of project model used.

Based on the methods used for requirements engineering, agile methods are very well represented in the sample. 90.6 % of the respondents answered that they use face-to-face communication to manage requirements. 78,1 % of the respondents mentioned that they interact with the customers or end users to manage requirements. This seems to indicate that there is an emphasis on collaboration. Which also fits in with the utilization of cross-functioning teams which was reported by 46.9 % of the respondents. To understand the use of methods better, the results will be analysed based on methodology used for project management.

Agile projects was the most common type of model reported by the respondents of the questionnaire. Most of the respondents reported that they use from eight to eleven of the methods that were predefined in the questionnaire. Eleven reported that they use user stories which seems to be a popular way of discovering requirements in the sample. Very few responses mentioned the use of contract negotiation or opinions from experts as a way to manage requirements. These methods might not seem as very agile as they do not seem to incorporate change, flexibility or feedback throughout the project. The Agile Manifesto, however, only states that it values feedback and change (Fowler et al., 2001), and not that methods should not be used that do not make use of these traits. It is hard to incorporate contracts into the agile methodology and it might also lead to the contracts changing the requirements engineering methods (Daneva et al., 2013).

The amount and types of methods used by the respondents that reported that they work on projects that follow hybrid models, seem to suggest that their use of methods reflects that model. This seems to indicate that the hybrid model is also represented in the methods used for requirements engineering as they use both many of the agile methods but also make use of methods that are not as agile such as contract negotiations.

The three respondents that answered that they work on projects that use traditional models, reported that they use fewer methods for managing requirements. All of them reported that they use face-to-face communication, however, other methods that are not equated to using the agile methodology, such as opinions from experts, requirements being prioritised only at the beginning and collecting requirements from similar projects were reported. It also seems that the respondents are adapting agile methods into their traditional models because two of the responses mentioned code refactoring. Code

refactoring is when you change developed code so that it can accommodate changes in the project (Inayat et al., 2015). Being flexible to change is considered to be part of the agile methodology (Fowler et al., 2001).

The responses reporting that they are not sure what model they use, seem to all be using a hybrid set of methods for requirements engineering. They all seem to be implementing either feedback, collaboration or tools that minimize documentation such as user stories, or a combination of all. These are all values that are derived from the Agile Manifesto (Fowler et al., 2001). They also use methods that are not usually equated to agile such as contract negotiations or collecting requirements from similar projects.

The issues that have been observed and reported by the respondents will give further insight into project methodology, methods used and how they might be related. The interview portion of this research will also contribute to what methods are used and why.

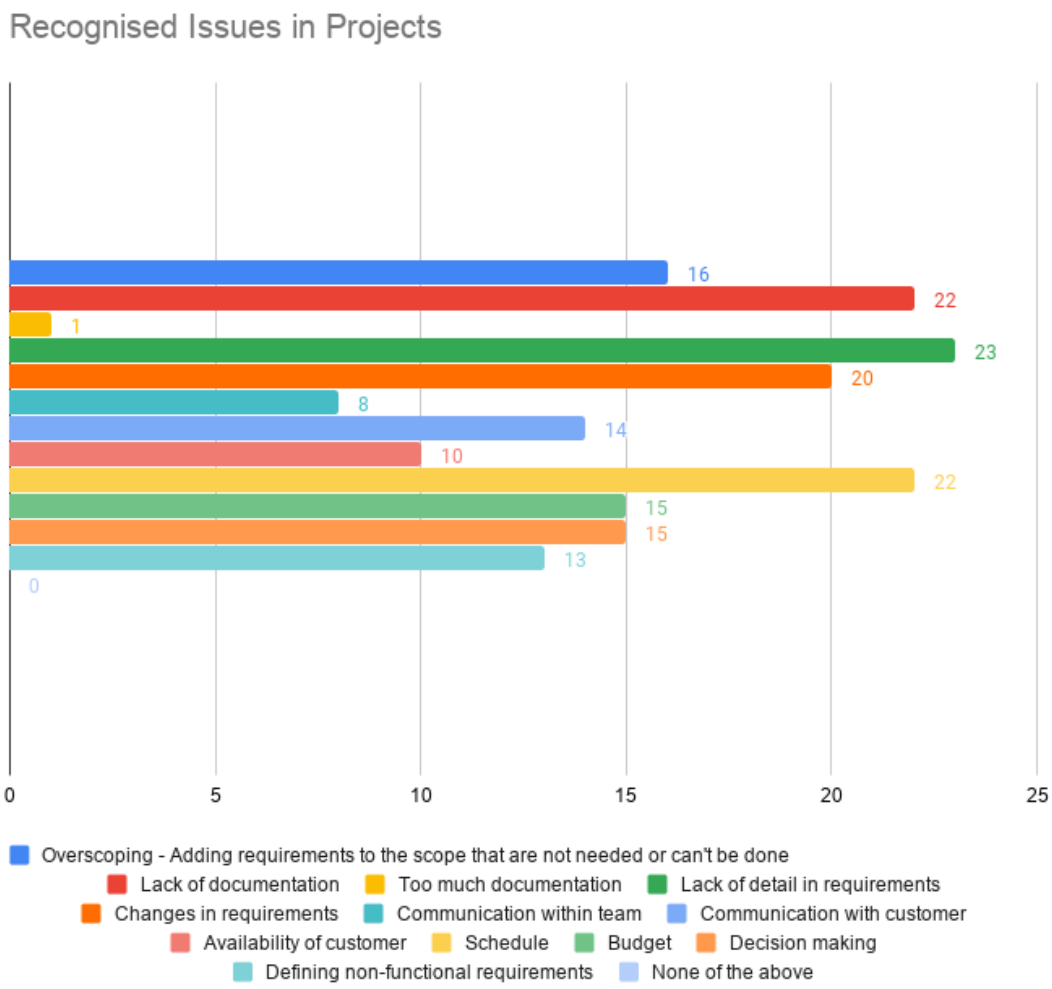


Figure 7: Recognised Issues in Projects

The interest behind asking about issues in projects was to discover what issues are found in projects in the Greater Helsinki area and to see how that reflects with other research that has been done in this area. The purpose was also to see what type of issues can be observed in which type of projects and also if there are issues that are observed in projects using a specific type of method for requirements engineering.

It seems that all respondents have observed at least a few issues when working on projects. The most prominent issue found was *Lack of detail in requirements* but the issues *Lack of documentation, Schedule* and *Changes in requirements* were almost as frequently reported. The lack of documentation is an issue that has been observed frequently according to previous research into agile requirements engineering (Inayat et

al., 2015). This could be related to the frequent use of user stories for discovering requirements. 62.5 % of the respondents of the questionnaire reported that they use user stories for managing requirements. Out of the respondents that reported that they use user stories, 60 % or twelve people reported problems with lack of documentation. However, 22 of the responses overall stated that they have issues with lack of documentation. Which makes up 68.8 % of responses. Which would indicate that the problem of documentation is related to other factors than to the use of user stories or at least the use of user stories is not the only factor.

Research has identified issues related to user stories when it comes to documentation. A user story is a simple description of a functionality of a system told from a person interactive with the system(Lin et al., 2014). User stories can be written in differing detail. Who writes the user stories might also contribute to what level of documentation they contain. Researchers have raised the issue of loose-structured user story models and that they might not be able to catch the real goals that the stakeholders intend. This could be because the stakeholders do not always know themselves what the goal in reality is (Lin et al., 2014).

To be able to understand the cause of this issue, further research needs to be performed. It is difficult to understand what causes the lack of documentation issues in projects if we do not know what the documentation looks like right now. It would also be interesting to understand if the lack of documentation is in any way related to the use of user stories or if there are other factors that have a stronger correlation with the issue. However, even a strong correlation does not mean that it would be the cause of the issue.

22 of the respondents reported having schedule issues. This issue was reported in all types of projects with roughly the same proportions. Out of the 22 responses, 45.5 % were agile, 27.3 % hybrid, 9.1 % traditional and 18.2 % of the responses that reported that they were not sure which type of project that they work in. The methods used also did not point to anything significant. This could indicate that schedules are still a hard part of the project to manage. Whatever the cause of problems related to schedule are, no conclusion to what that cause could possibly be could be found from this questionnaire.

Communication within team was only reported as an issue by eight (25 %) respondents. Most of the respondents (83 %) reporting this issue reported using agile models. Communication is an important part of agile methodology (Fowler et al., 2001). Even if most of the respondents that reported that communication with team members has been an issue, the number is quite low in regards to other issues reported.

All of the respondents have stumbled across a few of these issues at least. Which could be seen as an indication that there is value in researching the topic of requirements engineering in the Greater Helsinki area. Issues with documentation and scheduling could probably improve the motivation in projects and the quality of the product that the project produces.

Figure 8: Percentage of Projects That Finish on Time

The schedule of a project is important. With agile, the product does not need to be finished for it to be in use. However, if a project is not finished, then it will take up resources that could be used for other projects or tasks in the organisation. It could also cause delays in other projects or tasks. Agile is seen as a possible solution to better manage project schedules by being efficient and effective (Goetz, 2002). Other studies have found that agile methods could lead to more difficulties estimating a schedule as the requirements

are discovered and re-evaluated over time (Ramesh et al., 2010). This is reflected in the sample as the problem has not been solved with the use of agile. It seems that the issue of scheduling is quite equally reported in all project-management methodologies.

6.3 % of the respondents to the questionnaire answered that 100 % of their projects finish on time. This could be because the industry that they operate in does not allow for delays or the competition is so strict. The complexity of the project and the strictness of the schedule is of course a factor and should be explored more in the interviews.

44.4 % of respondents reported that 75 % of their projects finish on time and about the same at 39.7 % said that only 50 % of their projects finish on time. 6.3 % have only 25 % of their projects finish on time and 3.2 %, or one respondent, answered that none of their projects finish on time. The topic of schedules and deadlines in projects seem to be an area that needs further research. The interviews will give more insight into the reality of the situation, however, a more thorough study should be had into the effect of agile methods in projects.

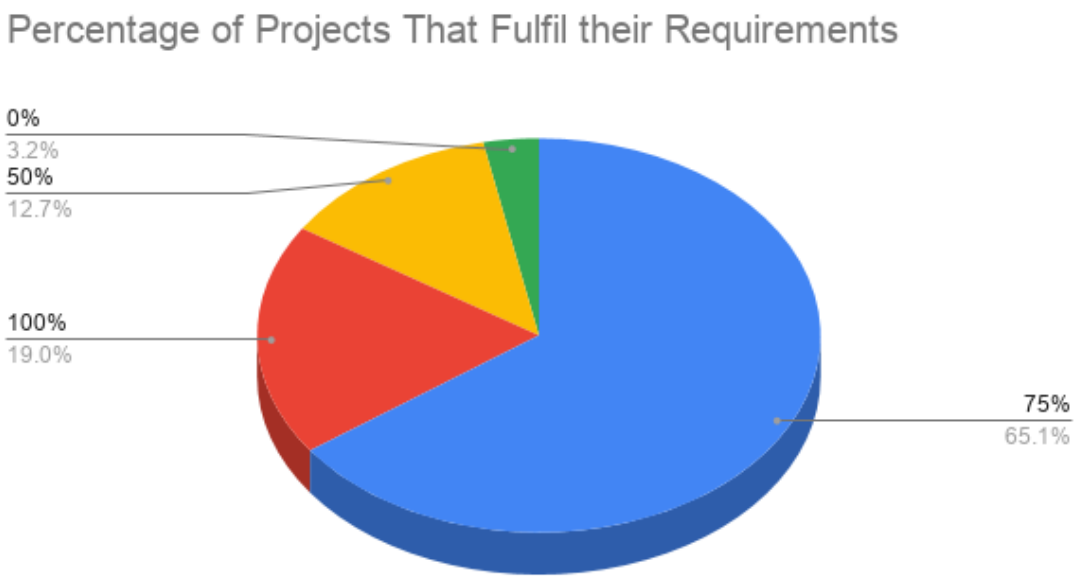


Figure 9: Percentage of Projects That Fulfil Their Requirements

In agile projects, the requirements change throughout the lifecycle of the projects. The requirements that bring the customer the most value are the ones that should be

implemented next (Bakalova et al., 2011) . If requirements are deemed not important or that they do not bring enough value, then they should be removed from the project's backlog. Therefore it is interesting to see that only 19 % of the respondents stated that 100 % of the requirements of a typical project are fulfilled. This could be related to scheduling problems or that requirements were deemed not worth the effort yet still something that was desired. This could also simply be related to how the respondent interpreted the question.

Most of the answers reported that 75 % of the projects fulfil the requirements set for the project that they are involved in. Only 12,7 % reported that 50 % of projects and one person said that none of the projects fulfil all of the requirements.

Out of the 32 respondents 25 responded that they either fulfil 75 % or 100 % of the requirements. Out of these responses, only two reported that their projects finish on time 100 % of the time. Twelve reported that 75 % of the time, ten reported that 50 % of their project finished on time and the remaining one reported that only 25 % finished on time.

5.2.2. Analysis of Optional Questionnaire Responses

The open-ended questions in the questionnaire, were meant as a way for respondents to voice their opinions more freely about a subject. The questions were a continuation to the required question that the responder answered before the optional one, with the exception of the last one. The last question was a question about the effectiveness of project management in general. These questions were all made optional in the questionnaire as answering open-ended questions can take time and more effort than answering questions with prefilled answers to choose from. Requiring the responder to answer these open-ended questions might have lowered the amount of submitted responses to the questionnaire.

The first open-ended question in the questionnaire was: *If one of your methods for managing requirements were not listed above, please write about it below.* This question received the least amount of answers out of all the questions in the questionnaire. The most obvious reason for this could simply be that most of the methods used are the common ones that the educational material and the researchers discuss. Another reason could be that the responder just could not remember more methods or just did not bother

answering. From the few answers to this question, a few mentioned specific tools used and also Minimum Viable Product which is a high-level prototype. This is, however, a common prototyping method within Lean methodology (Münch et al., 2013). In hindsight, it would have probably been beneficial to write "Prototyping (MVP included)" in the questionnaire. Another respondent wrote *"documentation somewhere in a common drive, focusing on a MVP solution"* which could suggest collaborated documentation as a way to manage requirements but also the use of working software. The Agile Manifesto (Fowler et al., 2001) did state that agile values *"working software over comprehensive documentation"*. This response seems to imply that this specific responder does use lightweight documentation, however, the respondents focus is on the working software. This goes to the core of what the Agile Manifesto wanted to convey. There is value in both but the focus is on the working software. However, it is hard to know without asking for a clarification from the responder.

The second question was *Describe a typical project that you are involved in. (Team size, project size and duration).* This was one of the questions that received the most interesting and eye-opening answers. This could be because no two projects are alike. The projects might have similarities, however, they vary in size, duration, components and structure. Most of the responses seem to be about software development but there were a significant amount of software deployment projects as well. Some of the projects the answers described were only three to five people and lasted only a month or two. Another project had 40 people involved and others had a life cycle of around three years. It also seems that the timeframe can be flexible as the answers contain words such as *up to* and *approximately*. One response even used the word *indefinite* which means that the project has no clear end or no end at all.

Many of the responses did include what roles there were in a typical project for them. Some of the roles mentioned were *developer*, *system administrator*, *customer, project manager*, *customer service manager*, *designer* and *product owner*.

Almost all responses abstained from including further detail than team size, project size and duration of the project. There were a few exceptions that delved into what type of roles can be found in a project and a few responses also stated what a typical project was

about. Only one response described the whole project process from their point of view from beginning to end in great detail.

A developer that was not sure what type of project model he/she works with, gave a long response in describing how a project usually looks like for them. *"I think it's impossible to describe our exact process, since I see projects needs to be handled differently, based on stakeholders, requirements, developers ("devs"), etc. But I think I can more or less split up our process in the following phases:"*

Even though the respondent is unsure about the project type, the model he/she is explaining does follow agile methodology. The first phase they describe is selecting people. *"[...]I have found that the best way to do software projects is to include people from the right domain already in the beginning/planning stage. E.g. if I need to touch a system X, I probably want to invite someone who knows that system and can do those change effectively, and also understand our needs and requirements. If we later figure out that they are not needed after all, they can always leave and work on something else."* The respondent seems to be explaining cross-functional teams, which is a method recognised to be an agile requirements engineering method (Inayat et al., 2015; Schön et al., 2017). In the questionnaire, the respondent was also asked which methods they use for managing requirements. The respondent did not select cross-functioning teams from the list. This could mean that the person is unaware of what cross-functioning teams entails or that the respondent does not see cross-functioning teams as a way to manage requirements.

The respondent also described their requirements process in detail. The process invites open documentation and feedback. The set of requirements that they agree upon has to be approved by the development team and other shareholders before development can begin. Once the requirements are approved, the requirements can only be changed through an approval process and the changes have to be documented. This follows the values that the Agile Manifesto described (Fowler et al., 2001). The documentation seems to be lightweight, there is collaboration between the stakeholders and there is a process for handling changes.

Testing also follows agile methods. They test throughout development. The respondent phrased it so: *"We test along the way, but usually it's nice to show the stakeholders and get their hands-on approval, once we think the project is close to done[...]"*. They also adapt their testing to fit the need. *"If it is a large project or we think there is something we are unsure of, we might have extra testing phases in the middle of the build-stage."* The respondent had minimal to say about what caused the extra testing. From the response it could indicate that it is related to unclear requirements that need more validation. Inayat et al. (2015) mentioned unclear requirements as a challenge with traditional requirements engineering that could be mitigated with agile requirements engineering. The respondent's answer seems to indicate that they have followed a similar approach. As they are unsure of a requirement, they have set up extra testing opportunities in the middle of development to receive validation for their requirements.

The third optional answer question was *What other factors have you experienced that have caused friction in projects?* These answers mentioned issues that other researchers have also identified . One issue is the lack of commitment from management and also inside the team. The reasons mentioned by the responders to the problem of lack of commitment inside the team, were that members of the team were not fully allocated to the project and could have many projects that they were working on at the same time. A response that was memorable mentioned that individuals could work on over ten projects at a time. Another factor that caused concern according to the responses was with communication. The issue was both within the team and with management/decision makers. The problem with the communication with management/decision makers that one response mentioned, was that they would decide important decisions without the developers. Then they would not communicate the decision to the project team in an effective enough way. The problem with communication also stretches to third-party providers according to responders. One response mentioned problems with third-party providers not effectively listening and adjusting based on feedback given. There were also mentions about collaboration issues within teams that caused decision-making problems and made members of the team feel unmotivated and not appreciated. According to the findings of Inayat et al. (2015), the issue of communication gaps is a challenge for traditional requirements engineering. Bjarnason et al. (2011a) research into requirements communication indicated that agile requirements engineering methods such

as cross-functional teams, iterative requirements and acceptance tests could improve the motivation inside the team and also to keep the requirements up to date. The research in this thesis did not investigate the reason for the communication gaps and why some project members are feeling unmotivated, however, it might be that better implementation of agile methodologies could be the solution for these issues.

The final optional question and also the final question of the questionnaire was *What are the reasons projects are managed effectively or not effectively at your place of work?*. This question was formed so that the responders could either answer what makes project managers effective at their place of work or how the projects are managed ineffectively at their place of work. They could also choose to answer both angles as some responders did. These answers varied vastly. The answers evolved around the same topics from both the positive and negative side and they all seem to be about management, communication, team and requirements in differing ways.

According to the responses, management was handled well when the project team had autonomy and that the management did not try to micromanage the project. The managers also needed to understand their role in the project so that it can be managed well. This way of handling a project is reflective of the Scrum framework. In Scrum, the Product Owner will focus on stakeholders and requirements, the Scrum Master will see to it that the development team is not distracted away from development and the development team should focus on development (Carlson et al., 2012). A response focusing on when projects are not effective voiced concern about lack of effective management and not taking ownership of the project. Contracts also seem to cause problems as one answer cited unclear contracts and contract negotiations as a presumed reason for ineffectiveness. Requirements were also a popular topic. One such response was *"When project requirements are written from technical perspective and not pay attention to user experience, projects fail."* As mentioned in the chapter about agile requirements engineering, a typical complaint for agile is that there is minimal emphasis on non-functional requirements or that non-functional requirements are only prioritized later in the project. Inayat et al. (2015) identified multiple researches that have come across the problem of identifying non-functional requirements in agile projects, but also researchers that suggested solutions to modelling for non-functional requirements.

Other problems that were listed were that people took on too many projects and also that management started too many projects in regards to the amount of resources the company had. Responses also discussed concerns about staying agile and not reversing back to a hybrid model. Drifting from agile methodology was a direct answer from one of the responders but others did state key agile principles as what was making them effective such as communication, close collaboration and frequent planning.

These answers were considered when the questions for the interviews were planned. Therefore, these answers will also be mentioned and discussed in the section about the interviews. There they will also be analysed against previous research.

The issues that the responses mentioned have been identified by researchers. Inayat et al. (2015) had most of these in their summary of challenges. Some of these issues also have suggested solutions that this thesis will further go into in the discussion part.

5.2.3. The Interviews

The intention behind the interviews is to get a better understanding of requirements engineering and project management in the Greater Helsinki area. The interviews are the second method in a two method process. The first research method, the questionnaire, gave a glimpse into what requirements engineering methods are used and what issues are identified with project management. The answers from the questionnaire have also been used to formulate the questions that are used in the interviews for this study. The first set of questions in the interview are about the interviewees role and experience in regards to working in projects. After these questions, the interview questions will focus mainly on requirements engineering. These questions are about what tools the interviewees use, how they gather requirements and what issues they are experiencing. The intention with the interviews was also to validate the data that were collected with the questionnaire.

The interviewees were chosen to represent different industries. Two of the interviewees develop software and the third interviewee is involved in implementing software solutions for internal users. The interviews were held in Swedish. The quotes from the interviews used in this thesis are translated from Swedish to English. Care has been taken when translating so that the original meaning is kept the same. The name of the interviewees and the organisations they work for have been kept anonymous. This was

done to keep the interviewees from being concerned that the information that they provided would not affect them or the organisation that they work for.

Role and Experience

The first part of the interviews was designed to form an understanding of the interviewees and what kind of work environment they work in. These questions were about their role in projects, what kind of projects they work in and also how much experience they have with working in projects.

Two of the interviewees had worked full time with projects for three years. One of the two did have experience with working on projects before that, however, that was less structured project work. The third interviewee reported to have worked with projects for the extent of his/her fourteen year long career. They work in quite different types of projects. One reported to work with data warehouse and business intelligence projects. Another reported to work on projects both for internal users and for customers. The third worked with software implementation projects and his/her customers were mostly internal users. The roles that they occupy in the projects vary between projects except for one of the interviewees. He/she reported to only work as a developer. The two others reported to have worked both as project managers as well as part of the core project team.

Project Models, Frameworks and Methods

After understanding their role and experience with projects, the interview questions shifted topic to project models, frameworks and methods for gathering requirements.

Two of the interviewees used agile project models and the third one used mostly the traditional waterfall model. The Kanban framework was reported to be used by all interviewees to some degree. Scrum was used by both of the interviewees that developed software. The interviewee that worked with implementing software reported using the ITIL framework. The ITIL framework is a framework specific to implementing software and services (Potgieter et al., 2005).

The methods used for gathering requirements by the interviewees differed plenty. One interviewee reported that all requirements have to be created in user stories and then added to the backlog. There were exceptions where meetings could be used to determine

requirements as well, however, the practise was that even those requirements should be transferred into user stories. The interviewee that worked with data warehouse and business intelligence reported using the vast amount of methods for gathering requirements. These methods were workshops, meetings and direct communications for discussing and brainstorming requirements. He/She also used testing and proof of concepts to test and show the requirements that will be needed for the project. On top of these methods the projects he/she was involved in used the expertise of both analysts and the project team to come up with requirements. Both of the interviewees that worked with development projects gathered requirements throughout the projects. This is different from the interviewee that worked with implementing software. The projects that he/she worked on followed the traditional project model and therefore the requirements were determined at the start of the project. These requirements were gathered in meetings and interviews and were based on the structure of the organisation. The projects must follow certain laws and regulations that apply to their industry but they also have to be so loosely defined that the requirements do not favor vendors in the public procurement process.

Challenges

One question in the questionnaire was "What issues do you come across when working on projects?". All the respondents reported that they had experienced issues or challenges when working on projects. This was not confined to one type of project model. Many of them reported multiple issues related to requirements.

Technical debt is a metaphor for a debt that needs to be paid back later. Documentation debt, which is included in the literature-review portion of this thesis, is a form of technical debt. This is when a technical solution is done quickly without the end product in mind. This will result in extra work later in development or during support of the product (Tom et al., 2013). The use of the term debt relating to poorly written code seems to have been introduced by Cunningham (Cunningham, 1992). The technical debt that was mentioned in the interview might have minimal cause related to requirements engineering but there could be a solution to this problem from requirements engineering tools. Adding testing criteria to the requirement could lead to developers understanding the requirement and therefore being able to develop a correct and functioning solution. As the cause of

technical debt is unknown in this particular case, more info would be needed to examine if agile methods could have an impact on this problem.

The second issue the interviewees mentioned was related to undefined requirements. These requirements were related to the architecture of the system and were only realized late in the project. This caused production problems because fulfilling the requirement was not something that was possible with the product that had been developed until now. This is a well-documented issue with agile models. This type of requirements is called non-functional requirements and they relate to security, capacity and architecture. It is common that these requirements are realized in the late stages of the project and therefore they are hard to implement (Inayat et al., 2015).

Documentation

One of the typical issues that came up in the questionnaire was related to documentation of requirements. To get a more insight into these issues, the topic of documentation tools and issues were raised during the interview.

How the interviewees documented the requirements was related to the project model they used. The interviewees that followed agile project models also followed the documentation tools used often in agile models. Both of the interviewees used wikis for collaborative documentation. One of them used the tools in the frameworks for documentation but also documented in the developed code. The interviewee that followed the traditional method had a more strict documentation process. The requirements had to be documented in the public procurement documentation. They also used spreadsheets to keep track of requirements. This tool was also used by the interviewee that worked with developing data warehouse and business intelligence solutions.

One interviewee explained that the responsibility of documentation falls on the developer. It was the project manager's responsibility that everyone in the project follows these instructions. Despite this, the interviewee still reported problems with the documentation. The documentation related to the logic of data transformation was raised as a specific issue, however, more general issues were also mentioned. These issues were unstructured documentation, documentation that is hard to find, bloated documentation, low-quality documentation and also documentation that was outdated. There was also documentation

that was missing altogether. The second interviewee had the opinion that their organisation had become obsessed with documentation. They have designated people that should document. It was the interviewee's opinion that these people should be the ones documenting and minimal documentation should be done by the others.

Changes in Requirements

One thing that makes agile models different from the traditional models is that agile models are designed to accommodate changes. What the re-evaluation of requirements and change process looks like in reality is of interest for this study. Therefore, the interviewees were asked how they re-evaluate requirements and how they handle changes.

Re-evaluating requirements and handling changes were done differently by all of the interviewees. The interviewee following the traditional model had a ridgid process. There was no documented process for re-evaluating requirements. However, if someone noticed that a requirement needs to be changed, they should report it to the Change Advisory Board. This board discusses and votes on new plans if the old one is no longer advisable or possible. The other interviewees did not have a board that all changes had to go through. One of them re-evaluated iteratively throughout the project when needed and the other tried to keep the backlog of tasks so low that the requirements were always fresh and did not require re-evaluation.

Testing

Testing is an important part of software development and implementation. To test that a software works the way it is intended to work requires some form of processes. In requirements engineering, testing is done to see how well the software fulfills requirements. Both types of testing are important for the product. Therefore, this was a topic raised in the interviews to incite a few examples of how testing can be handled.

None of the interviewees reported any official way of testing that the requirements are fulfilled. The developers both mentioned unit testing. This is a method used for testing developed code that it functions the way the developer wants the code to function. The test is on a module or function of the code and not a feature or the whole system (Runeson,

2006). However, the term unit testing can have different meanings depending on the person that is asked (Runeson, 2006). The other methods mentioned were ad-hoc testing, testing during development and user-acceptance testing.

Customer's role

The customer's role in projects is something that should be valued according to the Agile Manifesto (Fowler et al., 2001). However, the part of the customer for the different interviewees was quite different. One of the interviewees only had internal customers. They took part in the projects by making requests and prioritizing these requests. The developer working on data warehouse and business intelligence projects worked closely with the customer. It was the customers task to define and test requirements. The customer was also frequently part of the development team. The third interviewee was not sure what the customer's part in projects are. In some of the projects that the interviewee was a part of, the customer was an internal customer. In these cases, the customers requested features and gave feedback. However, if the customer was external the interaction was minimal. The Product Owner functioned as a proxy for the end user and tried to request features that the customer would like. The only way a customer could give feedback to the developer was through the feedback feature built into the website.

Schedule

The questionnaire showed that most of the respondents had issues with finishing projects on time. To investigate what could cause this the topic of schedule management was raised in the interviews.

Schedules in projects were handled differently in their respective companies according to what the interviewees reported. The interviewee working on implementing software with the traditional project model still applied tools from the Kanban framework to handle the schedule. They had a backlog and held two week long sprints. The other interviewees had a different experience. One of them reported that the schedule was handled by the Project Manager by following up on progress throughout the project. The third interviewee only had soft deadlines. If the deadline was not reached, the deadline was moved. This company had no Project Managers in their projects. This could sometimes require the

management to step in and assign someone as project managers if they deemed it necessary.

Feedback

Communication and collaboration is an important part of agile (Fowler et al., 2001). To discuss a project after it has finished is part of some of the agile frameworks. It is a great way to summarize the project and reflect on what went well and what could have been improved. The process after a project is finished was a subject of the interview. This was to investigate if the interviewees have any experience with this kind of feedback and also to see if this is in any way related to requirements engineering.

Discussing a project after it was over was something that the interviewees had little experience with. One of them reported that they had started doing retrospectives after finishing projects about six months ago. He/She was positive to the change and thought that value could be found in these sessions. The other interviewees reported they do not have any formal discussion after the project has finished. This was something that they realized that needed improvement. The only communication that they had has been informal discussions with other people from the project. One of the interviewees also reported that mostly people are not that positive with the outcome of the project and indicated that this could be a reason for why people might not be eager to talk about a past project. However, if the results of a project underwhelms, then it might be the perfect reason to discuss why this was the case and how this could be prevented in the future.

Experience and Opinions About Agile

The interviewees were positive to the agile project model and the models' effect on the product. The interviewee with fourteen years of experience with working in project had a few examples of the impact agile can have on a project. In the company that he/she worked in, one team worked on a project using the traditional project model for one year. When it came time to take that product into use, there was no longer a need for that product. Another example was of a new website that he/she had personally worked on. The project followed the traditional model. The interviewee explained: *"These big-bang projects when you release all work after a long time they are… That has never worked"*. The interviewee explained that too many requirements were added to the project. In the

end, they had to rush the deployment which resulted in missing features that were present in the website that it replaced. In another project that was a cloud-integration project, they were also going to follow something resembling a traditional model. The interviewee said: *"We worked on it for half a year and then we gave up. We realised that big bang is never going to work. We have to do it iteratively"*. They were going to release all the changes at once. A while into the project they understood that this would be very risky. They instead shifted to applying changes incrementally. Which the interviewee said might take more time but the risks are much smaller.

The other interviewees did not have any examples of why agile is better. However, their opinion was that agile models are better. The second interviewee said *"[...] If you are working on the normal waterfall method, then there is no opportunity to change in the middle of the project."*. He/She did think that there are places where the traditional model can be used. For example small projects that are not that complicated. Wysocki (2010) argued that if there is no clear solution defined for a project, then the product would be beneficial if an agile model was used in the project. The first interviewee explained that the choice of project model would change the schedule, cost and feature of the product. He/She argued that agile would produce a better product because the agile model is quicker to respond to changes and feedback.

5.2.4. Research Summary

To collect data from the demography, two methods were used. The use of two or more methods to gather data is called triangulation (Torrance, 2012). Triangulation is used in research as two methods might not give the same results even if the same demography is investigated. The belief is that using different methods might grant researchers distinct views on the research subject which on the other hand will give more comprehensive data on the subject (Torrance, 2012).

The summary of this research will compare the different methods and summarise the data collected. In the next chapter, the research is compared to the literature with focus on the research questions of this thesis.

The research collected data about requirements engineering and experienced issues in projects. The questionnaire had 39 responses and 32 of those were from the demography. In addition to the questionnaire, three people were interviewed.

Both research methods came to similar findings. The requirements engineering methods that were discussed in the interviews was a subset of the ones reported by the questionnaire. However, the interviewees did report the use of many agile methods.

Testing was a topic that could have been explored better in both the questionnaire and the interviews. The use of prototypes and minimal viable products was identified, however, other testing methods were minimal. Unit testing was mentioned in the interviews, however, that is a tool used for development (Runeson, 2006).

The issues that have been identified in past studies (Inayat et al., 2015; Schön et al., 2017), could also be identified in the questionnaire. In the interviews the interviewees reported issues with undefined requirements, issues with documentation and overscoping. The first two are issues related to agile, however, overscoping is an issue related to traditional requirements engineering that could possibly have a solution with agile methods (Bjarnason et al., 2011a).

The customer plays a big role in agile (Fowler et al., 2001). In the questionnaire, 10 out of 32 (31.3 %) respondents reported that the availability of the customer is an issue. However, in the interviews no issues with the customers were raised. The customers did play different roles according to each interviewee. One interviewee interacted minimally with customers in most projects, another used customers for feedback and requests and one interviewee reported that the customers are usually part of the development team.

Feedback throughout the project seems to be popular in the sample. Review meetings and acceptance tests were used by 15 out of 32 (46.9 %) of the questionnaire respondents. However, only one of the interviewees reported that they have feedback meetings after a project is finished. The two other interviewees could recognise the value of having these feedback opportunities.

Overall, the questionnaire and the interviews both seem to indicate that people are optimistic about agile project management. The examples given in the interviews also gave glimpses of when agile models could or had saved projects.

6. CONCLUSION

This thesis tries to answer three questions. The questions revolve around methods used for requirements engineering and problems that have been experienced using these methods. The questions are:

> *RQ1: Has agile requirements engineering impacted project management?*
>
> *RQ2: Does the product benefit from using agile requirements engineering?*
>
> *RQ3: What relation does the method of requirements engineering have on the success of a project?*

Two methods were used for these research. The purpose for using two different methods was to observe the sample from two different perspectives. The questionnaire's purpose was to collect statistical data and also to examine opinions. The interviews were held to increase validity to the data collected with the questionnaire but also to collect more opinions about working in projects.

The questionnaire received 32 valid responses. This was enough to observe patterns in the type of methods used and also what type of problems the respondents are experiencing with requirements engineering. The questionnaire was also able to collect some opinions about working in projects.

The research would have benefitted from receiving more responses. This is even more resonating when discussing the optional questions. One of the optional questions only received six responses. The low number of answers to the optional questions was by design. The optional questions were added because if a respondent had additional material that they wanted to add to the answer, the optional questions were designed to fill that purpose.

Three individuals were interviewed for this research. The interviews were semi-structured. I.e. the topics were raised in the form of questions and then the interviewer and the interviewee could discuss the topic quite freely. The interviews were one-on-one interviews.

The research questions will be analysed and answered one at a time. Each question will include data from both data gathering methods as well as from literature about the subject.

Has agile requirements engineering impacted project management?

Agile methodologies have been around for over 20 years. With the release of the Agile Manifesto in 2001, agile methods started to be used throughout the software development industry (Carlson et al., 2012). The methods used in agile projects follow the agile philosophy. An important part of any project is requirements engineering. It has also been argued that requirements engineering is one of the more demanding tasks of project management (Abdullah et al., 2011). In traditional requirements engineering, the requirements are gathered at the start of the project and see no changes until the end of the project (Wysocki, 2010). In agile, the requirements form throughout the project. Requirements are added, removed and throughout the lifecycle of a project (Cao & Ramesh, 2008; Wysocki, 2010).

If companies have adopted agile project models, then how has that impacted projects and how they are managed? In the questionnaire, 46.9 % of the respondents answered that they are working on projects that follow agile models. Hybrid models was the second most reported answer with 31.3 % of the respondents reporting that they usually work with hybrid projects. As hybrid is a mixture between traditional and agile models, the hybrid models also use agile methods to some degree. Which means that according to the questionnaire, 78.2 % of the respondents are using agile methods in their project work. The impact of agile methods can also be seen in the methods used to manage requirements. Face-to-face communication, interaction with customer or end user, user stories, iterative requirements, requirements prioritisation, change management, cross-functioning teams, prototyping, testing before development, goal sketching, review meetings and acceptance test and code refactoring are all agile methods for requirements engineering (Inayat et al., 2015; Schön et al., 2017). All the methods listed above were identified in the sample. Many of the methods saw close to 50 % of the respondents using the method. The most popular method was face-to-face communication (90.6 %). Interaction with customer and end user (78.2 %) and user stories (62.5 %) were the second and third most reported method for managing requirements.

Inayat et al. (2015) identified a few issues related to agile requirements engineering. These issues are related to *minimal documentation, customer availability, budget and schedule estimation, inappropriate architecture* and *neglecting non-functional requirements*. These issues could also be identified in the sampel. The most reported issue was related to *Lack of details on requirements* (71.9 %). In the interviews, one interviewee explained that the responsibility of the documentation is on the developer. Only the project manager might check that the documentation is adequate. The second most reported issue was related to *Schedule* (68.8 %). Ramesh et al. (2010) argued that schedules are hard to estimate in agile projects because of changing requirements.

Overscoping is an issue that was reported by 50 % of the respondents. This is an issue related to traditional requirements engineering (Inayat et al., 2015). Insufficient communication can contribute to the issue of overscoping (Bjarnason et al., 2011a). Even though it is reported that the issue of overscoping can be handled to some degree with agile methodologies, there is no indication that overscoping would be less of a problem with agile requirements engineering in the research in this thesis. There is a possibility that the challenge of overscoping is still visible in agile requirements engineering, however, the challenge has been lessened by the increase of communication that the agile methodology value. Bjarnason et. al (2011a) findings showed that agile methods added visibility to the scoping process which alleviated overscoping. Also, because change is part of agile models, the problem caused by overscoping was less severe.

Agile requirements engineering might have also impacted people working on projects without them identifying that it has been caused by agile requirements engineering. One of the questionnaire respondents was unsure of what type of project model they were working with. However, the methods and processes used had been influenced by agile methodology. The respondent explained what could be interpreted as cross-functional teams, lightweight documentation and change management. All of which are important aspects of agile (Fowler et al., 2001).

All of the interviewees had similar opinions about agile projects. They were optimistic that agile has many benefits and is a prefered way over traditional in most projects. One of the interviewees had from past experience seen that agile can have a positive impact on a project.

The findings from the research indicate that agile requirements engineering is well adopted in the sample. The research indicates that projects use agile methods to manage uncertainty and changes.

Does the product benefit from using agile requirements engineering?

Agile and hybrid models were well represented in the sample. The use of agile methods for requirements engineering had a strong presence in the sample. The questionnaire did not directly touch upon this research question. The only metric for an improved product was the question *Approximately, what percentage of project that you are involved with fulfill the requirements set for the project?*. As the number of respondents that work using the traditional model was so low in the sample, there is no indication if an agile model helps the project team fulfil requirements better. However, the open-ended questions collected data about criteria that either benefits or hinders management of projects. According to the responses, a project is managed well when the project team has autonomy. Scrum which is a popular agile framework focuses on the autonomy of the development team. The Scrum Master responsibility is to help the development team focus on development (Carlson et al., 2012). The questionnaire also received responses stating that communication, close collaboration and frequent planning in their opinion methods that made projects successful. These factors have minimal indications that the product benefits from agile requirements engineering. It only seems to indicate that many of the respondents prefer working with agile methods.

The interviews added clarity to this research question. One interviewee described a failed project that followed the traditional model. The project took one year to complete and then was never implemented because it failed at fulfilling the project's requirements. In another project, the project team decided to change from traditional to agile to lower risk. According to the interviewee, that project would probably have failed if they did not change to an agile model.

The research seems to indicate that people prefer working with agile methods and also that some projects benefit from agile requirements engineering.

What relation does the method of requirements engineering have on the success of a project?

The usage of agile methods for requirements engineering in the sample was high. All the methods identified by Inayat et al. (2015) as being agile requirements engineering methods were identified in the sample. Issues that have been identified as challenges in agile requirements engineering could also be identified in the sample. The questionnaire only had two questions that related to a project being successful or not. These questions were *Approximately, what percentage of projects that you are involved with finish on time?* and *Approximately, what percentage of project that you are involved with fulfil the requirements set for the project?*. The answers from these two questions had minimal indications to whether agile methods increased or decreased the likelihood of a successful project.

The responses to the open-ended questions in the questionnaire indicated that communication, close collaboration and frequent planning were factors in a successful project. These factors are important parts of the agile methodology and also many of the agile requirements engineering methods. It was also reported that drifting away from agile might have a negative impact on the project.

In the interviews, the positive relation to agile requirements engineering was supported by one example where the product was never taken into use because the requirements had either changed since they started the project or the requirements had not been understood. Another example was a project that was probably saved because of the change of project model.

Overscoping is an issue that has been a challenge for traditional project management (Inayat et al., 2015). This issue was reported by 16 out of the 32 (50 %) respondents of the questionnaire. It was recognised in both traditional and agile projects. Bjarnason et al. (2011a) identified that the effects of overscoping might be able to be reduced with agile methods. The interviews gave an example of a project where overscoping and time constraints caused required features to be omitted from the final product.

Working on projects is contrasting. Each answer in the sample was unique. The interviews expanded on just how different each project's workflow is. There is still plenty to be studied when it comes to working on projects. As the industries change and as agile

methodology is adopted into new industries, this area of research will probably not be exhausted in the near future.

6.1. Further studies

Pair requirements analysis

This method is derived from pair programming. The method of pair programming is used when developing software. Pair programming is when one person is writing code while the other is sitting beside the programmer to correct the code and ask questions about the code while it is being written. The observer should also think about ways to test the code that is being written. This would streamline the testing part of development. After a while, their roles switch so that the programmer is now the observer and the observer becomes the programmer. This method can also be applied to requirements analysis. However, the amount of research that can be found on this method is limited. Yu & Sharp (2011) did report benefits of using the pair requirements analysis technique in their case study.

By researching this method, value could be added to projects. To understand and explain requirements is still an obstacle in today's projects. Increased clarity could be added to requirement description when another person is asking questions while the requirements are being discovered. A well described and understood requirement could help in delivering the requirement to the customer.

In-depth research in the Greater Helsinki area

The sample for this thesis was from the Greater Helsinki area. The research conducted in this thesis only gave an overview into project work in the sample. The data from the questionnaire and the interviews was used to see how well the sample matched previous studies. The data collected from the sample included methods used for requirements engineering and issues that have been recognized with requirements engineering. To examine if the issues mentioned in the sample can be found throughout the area, and also to determine the extent of the problems that these issues cause, an in-depth study has to be conducted. If the root causes of these issues would be discovered, there is a possibility

that these problems could be rendered less prominent as the cause of frustration within projects.

Documentation tools for agile projects

The data collected in this thesis seems to indicate wikis are a popular documentation tool used in agile projects. Many of the issues with requirements engineering discussed in this thesis were related to documentation. As the sample in this thesis was quite small, a larger sample should be questioned to examine how popular this documentation tool is. It would also be valuable to understand in which areas of project management wikis function well and where they do not. However, if a dedicated agile documentation tool was implemented, that would probably have a significant effect on the agile project landscape and would therefore be an interesting area of research.

Agile in other project types

This research mostly focused on software development and software implementation projects. During the research for this thesis, only a few examples of agile methods being used outside of software projects. A few research articles did test agile methods on other types of projects. These were often small projects such as figuring out what to do with a common area in a building. Value could be gained from doing research into other areas as well as it seems other industries are hesitant to implement agile processes into their projects.

7. SWEDISH SUMMARY

Kravhantering inom agila projekt

En jämförelse av ett stickprov med kravhanteringslitteratur

Det är populärt att dagens arbetsuppgifter baserar sig på projektarbete. Projekt varierar i storlek, längd och komplexitet. Projekt har olika stadier och dessa stadier är hanterade olika baserat på vilken modell för projekthantering som följs. Den traditionella modellen är uppbyggd så att varje skede följer den tidigare. Den traditionella modellen brukar oftast kallas för vattenfallsmodellen. Namnet kommer från att man alltid arbetar mot nästa skede och man går aldrig tillbaka till ett skede efter att ett skede har slutförts (Wysocki, 2010). Ifall nya krav upptäcks under projektets gång kan det betyda fördröjningar och att projektet går över budgeten. Nyare modeller har utvecklats för att kunna hantera det okända. Dessa modeller kallas för agila modeller och kan hantera ändringar bättre (Wysocki, 2010).

Ett viktigt skede i dessa modeller är kravhantering. Projektets krav kommer från projektets intressenter och de mål de har satt upp för projektet (Project Management Institute, 2017). I den traditionella modellen är upptäckandet av krav det första skedet inom projektet (Royce, 1970). Detta skede är inte upprepat senare inom projektets livslängd. Inom de agila modellerna är kravhanteringen anpassande och iterativ (Wysocki, 2010). Denna avhandling fokuserar på kravhantering inom agila projekt och hur metoderna som används inom agila projekt jämför sig med metoder som används inom den traditionella modellen. Avhandlingen utforskar även problem och potentiella lösningar när det gäller både traditionella och agila metoder för kravhantering.

Mycket av den utvecklingen vi sett inom agil projekthantering är p.g.a *Manifest för Agil systemutveckling* som publicerades 2001 (Carlson m.fl., 2012). *Manifest för Agil systemutveckling* (2001) föreslår lösningar till styvheten av traditionella modellen. Manifestet är inte en modell för systemutveckling utan endast en lista med värderingar.
"Individer och interaktioner framför processer och verktyg
Fungerande programvara framför omfattande dokumentation

Kundsamarbete framför kontraktsförhandling
Anpassning till förändring framför att följa en plan"
Syftet är inte att strunta i de förstnämnda punkterna utan endast att prioriteringen ska vara högre på de sistnämnda punkterna.

Agila metoder föreslår lösningar för många problem som plågar systemutveckling, men agila metoder är inte utan problem. Forskningar har t.ex. tagit upp att agila modeller inte är tillräckligt bra på att upptäcka icke-funktionella krav. Icke-funktionella krav är krav som gäller design, arkitektur och säkerhet. Det har rapporterats att dessa krav upptäcks så sent in i utvecklingen av produkten att stora ändringar har krävts för att fylla dessa krav (Inayat m.fl., 2015).

Undersökningen i denna avhandling är en empirisk forskning inom projektarbete i Helsingforsregionen. Syftet med undersökningen är att jämföra ett stickprov från demografin med litteraturen om kravhantering. Målet är att undersöka ifall samma metoder och problem kan observeras i stickprovet som de som nämns inom litteraturen. I undersökningen används två metoder. Orsaken till att två metoder användes var för att öka validiteten på data som samlades (Denzin, 2017). Första metoden var ett frågeformulär som skickades ut till undersökningens demografi genom olika medier. På basis av svaren från frågeformuläret gjordes tre intervjuer. Detta gjordes för att öka validiteten på data som samlades från den tidigare metoden. Intervjuerna var också ett tillfälle att få djupare insikt i projektarbete samt att få mera åsikter om projektarbete från individerna.

Frågeformuläret fick 39 svar varav 32 var från demografin. Detta var tillräckligt för att mönster formades och att det gick att jämföra stickprovet med litteraturen. Inom stickprovet var det populäraste modelltypen för projekthantering agil. Agila modeller bestod av 46,9 % av de som svarade på frågeformuläret. Efter agila modeller var det hybrida modeller som var mest populära med 31,3 %. Endast 9,4 % rapporterade att de arbetar med traditionella modellen. Detta passar bra in med litteraturen som säger att agila projekt blir allt vanligare(Carlson m.fl., 2012; Shim & Lee, 2019).

Det samma som gäller modelltyper gäller även vilka metoder som används för

kravhantering. Av de 32 som svarade på formuläret och var från demografin, så rapporterade 29 att de använder sig av kommunikation ansikte mot ansikte för att hantera krav i projekt. Det var också populärt att kommunicera direkt med slutanvändaren eller kunden för att hantera krav. Dessa två metoder var populära exempel inom litteraturen för kravhantering inom agila projekt. Användningen av andra metoder för kravhantering kunde även jämföras med litteraturen. De metoder som har fått mycket uppmärksamhet inom litteraturen kunde även observeras i stickprovet.

De som intervjuades för forskningen är från olika industrier. Två av dem är involverade i mjukvaruutvecklingsprojekt, medan den sist sysselsätter sig mest med systemimplementeringsprojekt. En av de intervjuade har 14 års erfarenhet med projektarbete. De två andra har jobbat ungefär tre år med projektarbete.

Båda systemutvecklarna jobbar mest med agila projektmodeller medan personen som jobbar med systemimplementering använder sig oftast av traditionella modellen. Metoder som de använder för att samla krav, varier mellan personerna. Den ena personens arbetsflöde kretsar endast kring användarberättelser (user stories), medan en annan använder sig av flera olika metoder. Personen som jobbar med systemimplementering har ett mera strikt arbetsflöde. Kraven kommer ofta från lagar och regleringar samt så måste kraven vara beskrivna på ett sätt som gör att konkurrensutsättningen inte blir drabbad.

Flera av de problem som litteraturen förknippar med agil kravhantering kunde även observeras hos de intervjuade. Brister inom dokumentation och svårigheter med att samla in icke-funktionella krav är något som har observerats i andra forskningar (Inayat m.fl., 2015). Dessa problem pratade även de intervjuade om.

Alla tre som intervjuades är av åsikten att agila modeller är troligtvis bättre för de flesta projekt. Personen med 14 års erfarenhet delade med sig några exemplar där projekt hade fungerat bättre om projektmodellen var agil. Hen hade även ett exempel där de var tvungna att byta till en mera iterativ modell för att kunna leverera produkten.
Avhandlingen har som syfte att svara på tre forskningsfrågor. Den första frågan är *Har agil kravhantering haft en inverkan på projekthantering?*. Nästan fyra av fem (78,2 %) som svarade på frågeformuläret jobbar med agil eller hybrid projektmodell. Alla agila

kravhanteringsmetoder som Inayat m.fl. (2015) observerat kunde observeras i stickprovet. Även alla problem som förknippas med agil kravhantering kunde observeras i stickprovet. Det finns även indikationer på att intresset för agila metoder är hög då alla tre som intervjuades för denna avhandling föredrog agila modeller över traditionella.

Den andra forskningsfrågan är *Ser produkten någon fördel från användningen av agil kravhantering?*. Frågeformuläret hade en fråga om hur bra kraven uppfylls och en annan fråga om hur bra projektets tidsramar följs. Eftersom det var så få svar från de som jobbar med traditionella modellen kunde inga trender observeras. Frågeformuläret visade dock indikationer att agila metoder var uppskattade av projektarbetare. De som intervjuades delade samma åsikt. En av de intervjuade gav även flera exempel på projekt som misslyckats p.g.a. att projektet inte var tillräckligt iterativt.

Den tredje forskningsfrågan är *Vilket förhållande har kravhanteringsmetoder med projektets framgång?*. Åsikter från frågeformuläret var att god kommunikation, bra samarbete och frekvent planering bidrog till projektets framgång. Det var även sagt att misslyckandet av att följa agila metoder kunde ha en negativ inverkan på projektet. Ett exempel från intervjuerna var ett system som aldrig togs i bruk troligtvis p.g.a. att kraven ändrats sen projektet börjat eller att kraven inte hade uppfattats till en tillräckligt bra nivå.

REFERENCES

Abdullah, N. N. B., Honiden, S., Sharp, H., Nuseibeh, B., & Notkin, D. (2011). Communication patterns of agile requirements engineering. *Proceedings of the 1st Workshop on Agile Requirements Engineering*, 1–4.

Bakalova, Z., Daneva, M., Herrmann, A., & Wieringa, R. (2011). Agile Requirements Prioritization: What Happens in Practice and What Is Described in Literature. *Requirements Engineering: Foundation for Software Quality*, 181–195.

Bjarnason, E., Wnuk, K., & Regnell, B. (2011a). A Case Study on Benefits and Side-effects of Agile Practices in Large-scale Requirements Engineering. *Proceedings of the 1st Workshop on Agile Requirements Engineering*, 3:1–3:5.

Bjarnason, E., Wnuk, K., & Regnell, B. (2010). Overscoping: Reasons and consequences — A case study on decision making in software product management. *2010 Fourth International Workshop on Software Product Management*, 30–39.

Bjarnason, E., Wnuk, K., & Regnell, B. (2011b). Requirements are slipping through the gaps—A case study on causes & effects of communication gaps in large-scale software development. *2011 IEEE 19th International Requirements Engineering Conference*, 37–46.

Boness, K., Harrison, R., & Liu, K. (2008). Goal sketching: An agile approach to clarifying requirements. *International Journal on Advances in Software, IARIA*, *1*(1). https://www.iariajournals.org/software/soft_v1_n1_2008_paged.pdf#page=8

Burrows, R., Lopez-Lorca, A., Sterling, L., Miller, T., Mendoza, A., & Pedell, S. (2019). Motivational Modelling in Software for Homelessness: Lessons from an Industrial Study. *2019 IEEE 27th International Requirements Engineering*

Conference (RE), 297–307.

Cao, L., & Ramesh, B. (2008). Agile Requirements Engineering Practices: An Empirical Study. *IEEE Software*, *25*(1), 60–67.

Carlson, R., Matuzic, P. J., & Simons, R. L. (2012). Applying scrum to stabilize systems engineering execution. *CrossTalk*, 1–6.

Coram, M., & Bohner, S. (2005). The impact of agile methods on software project management. *12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05)*, 363–370.

Cunningham, W. (1992). *The WyCash portfolio management system*. https://doi.org/10.1145/157710.157715

Daneva, M., van der Veen, E., Amrit, C., Ghaisas, S., Sikkel, K., Kumar, R., Ajmeri, N., Ramteerthkar, U., & Wieringa, R. (2013). Agile requirements prioritization in large-scale outsourced system projects: An empirical study. *The Journal of Systems and Software*, *86*(5), 1333–1353.

Denzin, N. K. (2017). *The Research Act: A Theoretical Introduction to Sociological Methods*. Transaction Publishers.

Ernst, N. A., Borgida, A., Mylopoulos, J., & Jureta, I. J. (2012). Agile Requirements Evolution via Paraconsistent Reasoning. *Advanced Information Systems Engineering*, 382–397.

Etikan, I., Musa, S. A., & Alkassim, R. S. (2016). Comparison of convenience sampling and purposive sampling. *American Journal of Theoretical and Applied Statistics*, *5*(1), 1–4.

Farid, W. M., & Mitropoulos, F. J. (2012). Novel lightweight engineering artifacts for modeling non-functional requirements in agile processes. *2012 Proceedings of IEEE Southeastcon*, 1–7.

Fernández, D. M., Wagner, S., Kalinowski, M., Schekelmann, A., Tuzcu, A., Conte, T., Spinola, R., & Prikladnicki, R. (2016). Naming the Pain in Requirements Engineering: Comparing Practices in Brazil and Germany. In *arXiv [cs.SE]*. arXiv. http://arxiv.org/abs/1611.09132

Fowler, M., Highsmith, J., & Others. (2001). The agile manifesto. *Software Development Times*, *9*(8), 28–35.

Goetz, R. (2002). How agile processes can help in time-constrained requirements engineering. *Proceedings of the International Workshop on Time Constrained Requirements Engineering*. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.19.280&rep=rep1&type=pdf

Hass, K. B. (2007). The blending of traditional and agile project management. *PM World Today*, *9*(5), 1–8.

Haugset, B., & Stalhane, T. (2012). Automated Acceptance Testing as an Agile Requirements Engineering Practice. *2012 45th Hawaii International Conference on System Sciences*, 5289–5298.

Hess, A., Diebold, P., & Seyff, N. (2017). Towards Requirements Communication and Documentation Guidelines for Agile Teams. *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, 415–418.

IEEE. (1990). *IEEE Standard Glossary of Software Engineering Terminology*. IEEE.

Inayat, I., Salim, S. S., Marczak, S., Daneva, M., & Shamshirband, S. (2015). A systematic literature review on agile requirements engineering practices and challenges. *Computers in Human Behavior*, *51*, 915–929.

ISO/IEC. (2018). *Systems and software engineering - Life cycle processes - Requirements engineering* (ISO/IEC/IEEE 29148:2018(E)). IEEE.

https://doi.org/10.3403/30230210u

Järvinen, P. (2004). On a variety of research output types. *Series Of Publications D – Net Publications*. https://trepo.tuni.fi/bitstream/handle/10024/65450/D-2004-6.pdf

Ji, F., & Sedano, T. (2011). Comparing extreme programming and Waterfall project results. *2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE T)*, 482–486.

Jun, L., Qiuzhen, W., & Lin, G. (2010). Application of Agile Requirement Engineering in Modest-Sized Information Systems Development. *2010 Second World Congress on Software Engineering*, *2*, 207–210.

Ketterer, H., Rehberg, B., Schmid, C. N., & Kleine, D. (2016). THE END OF TWO-SPEED IT. *Bcg.com*. https://www.bcg.com/publications/2016/software-agile-digital-transformation-end-of-two-speed-it.aspx

LeCompte, M. D., & Goetz, J. P. (1982). Problems of Reliability and Validity in Ethnographic Research. *Review of Educational Research*, *52*(1), 31–60.

Lethbridge, T. C., Singer, J., & Forward, A. (2003). How software engineers use documentation: the state of the practice. *IEEE Software*, *20*(6), 35–39.

Lin, J., Yu, H., Shen, Z., & Miao, C. (2014). Using goal net to model user stories in agile software development. *15th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, 1–6.

Lopez Lorca, A., Burrows, R., & Sterling, L. (2018). Teaching Motivational Models in Agile Requirements Engineering. *2018 IEEE 8th International Workshop on Requirements Engineering Education and Training (REET)*, 30–39.

McCormick, M. (2012). Waterfall vs. Agile methodology. *MPCS, N/A*. http://www.mccormickpcs.com/images/Waterfall_vs_Agile_Methodology.pdf

Mendes, T. S., de F. Farias, M. A., Mendonça, M., Soares, H. F., Kalinowski, M., & Spínola, R. O. (2016). Impacts of agile requirements documentation debt on software projects: a retrospective study. *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, 1290–1295.

Miller, T., Pedell, S., Lopez-Lorca, A. A., Mendoza, A., Sterling, L., & Keirnan, A. (2015). Emotion-led modelling for people-oriented requirements engineering: The case study of emergency systems. *The Journal of Systems and Software*, *105*, 54–71.

Moe, M. M. (2019). *Comparative Study of Test-Driven Development (TDD), Behavior-Driven Development (BDD) and Acceptance Test--Driven Development (ATDD)*. http://www.academia.edu/download/59918587/47_Comparative_Study_of_Test-Driven_Development__TDD___Behavior-Driven_Development__BDD__and_Acceptance_Test-20190702-92470-o453so.pdf

Münch, J., Fagerholm, F., Johnson, P., Pirttilahti, J., Torkkel, J., & Jäarvinen, J. (2013). Creating Minimum Viable Products in Industry-Academia Collaborations. *Lean Enterprise Software and Systems*, 137–151.

Paetsch, F., Eberlein, A., & Maurer, F. (2003). Requirements engineering and agile software development. *WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.*, 308–313.

Patanakul, P., Iewwongcharoen, B., & Milosevic, D. (2010). An Empirical Study on the use of Project Management Tools and Techniques across Project Life-Cycle and their Impact on Project Success. *Journal of General Management*, *35*(3), 41–66.

Petersen, K., Wohlin, C., & Baca, D. (2009). The Waterfall Model in Large-Scale

Development. *Product-Focused Software Process Improvement*, 386–400.

Potgieter, B. C., Botha, J. H., & Lew, C. (2005). Evidence that use of the ITIL framework is effective. *18th Annual Conference of the National Advisory Committee on Computing Qualifications, Tauranga, NZ*. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.60.6404&rep=rep1&type=pdf

Project Management Institute. (2017). *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)—Sixth Edition* (p. 592). Project Management Institute.

Racheva, Z., Daneva, M., Herrmann, A., & Wieringa, R. J. (2010). A conceptual model and process for client-driven agile requirements prioritization. *2010 Fourth International Conference on Research Challenges in Information Science (RCIS)*, 287–298.

Racheva, Z., Daneva, M., Sikkel, K., Herrmann, A., & Wieringa, R. (2010). Do We Know Enough about Requirements Prioritization in Agile Projects: Insights from a Case Study. *2010 18th IEEE International Requirements Engineering Conference*, 147–156.

Ramesh, B., Cao, L., & Baskerville, R. (2010). Agile requirements engineering practices and challenges: an empirical study. *Information Systems Journal*, *20*(5), 449–480.

Royce, W. W. (1970). *Managing the development of large software systems: concepts and techniques*. 1–9.

Rudd, J., Stern, K., & Isensee, S. (1996). Low vs. high-fidelity prototyping debate. *Interactions*, *3*(1), 76–85.

Runeson, P. (2006). A survey of unit testing practices. *IEEE Software*, *23*(4), 22–29.

Schön, E.-M., Thomaschewski, J., & Escalona, M. J. (2017). Agile Requirements Engineering: A systematic literature review. *Computer Standards & Interfaces*, *49*, 79–91.

Seaman, C., & Guo, Y. (2011). Chapter 2 - Measuring and Monitoring Technical Debt. In M. V. Zelkowitz (Ed.), *Advances in Computers* (Vol. 82, pp. 25–46). Elsevier.

Sedeño, J., Schön, E.-M., Torrecilla Salinas, C. J., Thomaschewski, J., Escalona Cuaresma, M. J., & Mejías Risoto, M. (2017). Modelling Agile Requirements using Context-based Persona Stories. *WEBIST 2017: 13th International Conference on Web Information Systems and Technologies (2017), P 196-203*. https://idus.us.es/xmlui/handle/11441/88885

Shim, W., & Lee, S.-W. (2019). An agile approach for managing requirements change to improve learning and adaptability. *Journal of Industrial Information Integration*, *14*, 16–23.

Soares, H. F., Alves, N. S. R., Mendes, T. S., Mendonça, M., & Spínola, R. O. (2015). Investigating the Link between User Stories and Documentation Debt on Software Projects. *2015 12th International Conference on Information Technology - New Generations*, 385–390.

Tom, E., Aurum, A., & Vidgen, R. (2013). An exploration of technical debt. *The Journal of Systems and Software*, *86*(6), 1498–1516.

Torrance, H. (2012). Triangulation, respondent validation, and democratic participation in mixed methods research. *Journal of Mixed Methods Research*, *6*(2), 111–123.

Voigt, S., Hüttemann, D., & Gohr, A. (2016). sprintDoc: Concept for an agile documentation tool. *2016 11th Iberian Conference on Information Systems and Technologies (CISTI)*, 1–6.

Wysocki, R. K. (2010). *Adaptive Project Framework: Managing Complexity in the*

Face of Uncertainty. Addison-Wesley Professional.

Yu, Y., & Sharp, H. (2011). Analysing requirements in a case study of pairing. *Proceedings of the 1st Workshop on Agile Requirements Engineering*, 1–6.