Impediments Associated with Requirements in Agile Projects

Master's thesis in Software Engineering

Fredrik Holmdahl

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2018

Master's thesis March 2018

Impediments Associated with Requirements in Agile Projects

Fredrik Holmdahl

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2018

Impediments Associated with Requirements in Agile Projects
Fredrik Holmdahl

Supervisor: Eric Knauss at Department of Computer Science and Engineering, Philip Jungstedt at Avanade
Examiner: Robert Feldt, Department of Computer Science and Engineering

Master's Thesis 2018
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Agile related words relevant to the thesis, created with WordClouds.

Typeset in LaTeX
Gothenburg, Sweden 2018

Impediments Associated with Requirements in Agile Projects
A Case Study at Avanade
Fredrik Holmdahl
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

The agile principles does not lean on heavy documentation, and support changes by allowing changing requirements and even welcoming brand new during the development. Agile practitioners specify requirements on different abstraction levels, depending on the framework and personal preferences. Minimal documentation, no upfront requirements, and rapid working software are some of the luring traits of agile development. Terms like "just barely good enough" and "just in time" conflict with agile practitioners wanting more defined user stories and acceptance criteria. Organizations without existing agile knowledge can opt to outsource the development over applying internal education. Customers juvenile in agile development, a conflict of documentation levels in agile by different practitioners can cause challenges, such as customer availability, requirements lacking in detail, inappropriate architecture and contract conflicts.
This study set out to identify and present these challenges and provide solutions related to agile requirements engineering in an outsourced environment.
The data was gathered through a case study at a large consultant company. Interviews were used as the main source of collecting information, with the addition of forms. Related research was also included to relate the findings too, and collect any missing information not obtained from the case.
The results are presented as impediments that hinder the project, accompanied by solutions on how to address them. The themes which include the impediments are *lacking customer availability and engagement*, *insufficient architecture*, *requirements lacking in detail* and *customers lacking agile knowledge*. Solutions are *dedicated product owner*, *improved initial architecture*, *greater requirement detail* and *customer agile education*.

Keywords: Agile, Scrum, Requirements Engineering, Impediments, Challenges, Solutions.

Acknowledgements

This research was performed as my master's thesis at the Department of Computer Science and Engineering at Chalmers University of Technology in Gothenburg, Sweden.
First and foremost I would like to thank my academical supervisor Eric Knauss for providing great support and guidance during the whole process.
I would also like to thank Avanade for making the research possible by letting me conduct the study with their personnel. Moreover, I would like to thank my industrial supervisor Philip Jungstedt, and all the employees that participated.
Finally, I would like to thank my examiner Robert Feldt for providing great feedback at the thesis' milestones.

Fredrik Holmdahl, Gothenburg, March 2018

Contents

1 Introduction **1**

2 Background and related work **5**
2.1 Agile development 5
2.1.1 Agile methodologies 6
2.1.2 Agile Manifesto 6
2.1.3 Scrum 6
2.1.4 Agile requirements engineering 7
2.2 Outsourced development 8
2.3 Avanade Stockholm, Sweden 8
2.4 Related work 9

3 Methodology **11**
3.1 Data collection 11
3.1.1 Interviews 12
3.1.2 Structure of interviews 12
3.1.3 Sample 12
3.1.4 Validation from participants 14
3.2 Data analysis 14
3.2.1 Inductive approach 14
3.2.2 Transcribing interviews 14
3.2.3 Coding interviews 15
3.2.4 Analyzing data 15
3.2.5 Keeping track with memo 15
3.3 Presenting the findings 16

4 Results **17**
4.1 RQ1: What impediments are associated with agile requirements engineering? 17
4.1.1 Impediments found in this study 17
4.1.1.1 Lacking customer availability and engagement 17
4.1.1.2 Insufficient architecture 19
4.1.1.3 Requirements lacking in detail 20
4.1.1.4 Customer lacking agile knowledge 22
4.1.2 Impediments from related research 23

4.2 RQ2: How can the impediments caused by agile requirements engineering be addressed? . . . 24
4.2.1 Improvements found in this study . . . 24
4.2.1.1 Dedicated product owner . . . 24
4.2.1.2 Improved initial architecture . . . 25
4.2.1.3 Greater requirement detail . . . 26
4.2.1.4 Customer agile education . . . 27
4.2.2 Improvements from related research . . . 28

5 Discussion **31**
5.1 Contribution to research . . . 31
5.2 Implications for practice . . . 33
5.3 Threats to validity . . . 34
5.4 Significance of the study . . . 35
5.5 Future research . . . 35

6 Conclusion **37**

Bibliography **39**

A Appendix A **I**

B Appendix B **III**

1
Introduction

Traditional software development using the waterfall model follows a well planned and executed process. The linearity of it makes it ideal for projects that require being planned into great detail. Agile on the other hand was created as an alternative to the heavy documentation driven process by making it more lightweight and flexible [1]. Agile thus aims to counter many of the challenges present in a waterfall process, caused by e.g. following a strict plan. In particular, Requirements engineering (RE) in agile development differ from traditional RE, as it allows for requirements to change during the process. A systematic literature review (SLR) done by Inayat et al mapped the traditional RE challenges and presented them to be about e.g. communication issues, requirements validation, requirement documentation, as well as rare customer involvement [2]. All of these challenges were solved by different agile RE practices such as frequent face-to-face meetings, on-site customers, one continuous scope flow and user stories.
However, agile is not a silver bullet and its failure rate is according to Mersino (measured between 2011 and 2015) at 9 percent and challenged at 52 percent, leaving a success rate at 39 percent [3]. The categories are are created by The Standish Group and contains three constraints (schedule, cost and scope) which have to be met in order for success, while two of three has to be met for challenged. Failed projects are those that are cancelled or completed but ends up not used. The percentage of which agile projects fail and are challenges are therefore at a majority with 61 percent, indicating that there is much room for improvements. A debated area highlighted by Arthaud is the use of RE in agile projects, and how user stories compare to traditional requirements [4]. The starting misconception lies with the word "over", used in each of the principles in the agile manifesto, see Figure 1, and that there is only value on the left side in contrast to the right [5]. This misconception creates a mindset that only the agile methods hold value in agile projects, and anything from the traditional processes is thus neglected. The conclusion drawn by Arthaud was that there is still room for traditional requirements in agile projects, especially for large teams, complex systems, and systems which are expected to last long. Another debated area is that detailed requirements are not as emphasized in agile projects as they are prone to be changed. The term "good enough" is often used to describe how agile requirements should be documented, although without offering a concrete definition. The closely related term "just barely good enough" (JBGE) is explained by Ambler to be the most efficient way of documenting, as it yields the best value for the effort [6]. The graph used to illustrate JBGE is drawn with one axis representing value and the other effort, where JBGE is marked as the point before the value starts decreasing in a bell curve. However, the point of JBGE

is something that changes between cases, and to determine when the optimal documentation can be hard, but according to Ambler, it comes sooner than expected. Documentation is by this definition thus very minimal and is something that Meyer includes on the list of the ugly parts of agile [7]. Meyer also includes on this list e.g. no upfront requirements, abstract requirements replaced by user stories, and using tests instead of a specification.
The SLR on agile RE practices and challenges by Inayat et al show findings of minimal documentation, customer availability and inappropriate architecture that can result in traceability issues, increase in rework and increase in cost [2]. RE in agile could thus stand for many of the challenges in agile projects. It can, therefore, be crucial that there is a good definition, template or guidelines to go on when specifying the requirements in regards to them being "good enough" or JBGE. The requirements in agile are usually specified on the form of user stories, which translates into tasks in the backlog.
A research done by Lunder set out to identify how thorough the requirements should be specified in an agile RE process [23]. The results showed that it varied between different stages in the process. In another study done by Zhu, who investigated how to conduct RE in an Agile Scrum environment, showed findings that problems were e.g. that requirements were often not specified enough, and not on any format [24]. The key problem is that minimal documentation is too loosely defined and that using different interpretations of it can result in many different outcomes. Outcomes caused by incomplete or insufficient documentation can be very problematic and could be related to the low 39 percent success rate among agile projects. The purpose of this study is to identify these problems in a company that uses the agile framework Scrum. It does so by investigating how different employees in different departments in the company is affected by the use of agile RE. It also provides suggestions on how to address these problems to decrease the failure and challenge rates for future agile projects.
The definition of problems used in this research is impediments. This is because impediments represent hinders that are not full stop obstacles, but rather problems that can cause for a project to be challenging or cause it to fail if not handled properly.

Research question 1 (RQ1):
What impediments are associated with agile requirements engineering?

Research question 2 (RQ2):
How can the impediments caused by agile requirements engineering be addressed?

These questions were answered based on an exploratory qualitative case study using semi-structured interviews at a consultant company. The method was chosen as it encapsulates how the agile process is impacted in a client-customer environment which contains many of the agile specific challenges but also e.g. customer availability, customer involvement and contract negotiation. Four main themes were identified, and four resulting suggestions on how to address them, which were vali-

dated through feedback gathered from the participants in a follow-up form.
The scope of this research was limited to a case study at a consultant company for business and IT solutions. They use the agile framework Scrum in their work, and develop and provide different systems and solutions for their customers. More about the company can be found in Chapter 2, Section 2.3.

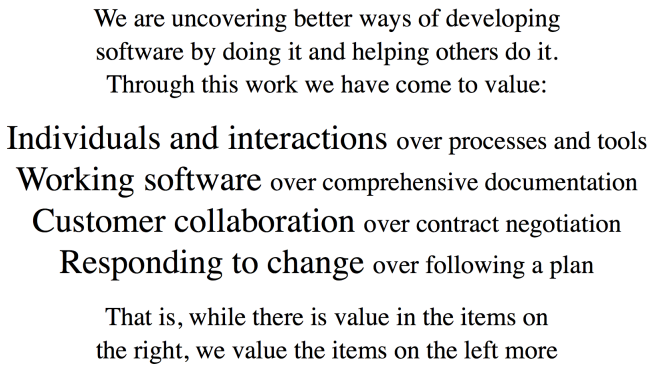

Figure 1. The Agile Manifesto.

2

Background and related work

Agile software development was officially created with the agile manifesto back in 2001 [1], yet it has traced all the way back to 1957 according to an article by C. Larman and V. R. Basili from 2003 [11]. According to Gerald M. Weinberg, who was quoted in that article, they used incremental development at IBM already at that time, which was very close to the techniques used in Extreme programming (XP). However, there are so many different agile techniques apart from XP, such as Scrum, Kanban or Feature-driven development (FDD). There are also different ways of using these in practice, such as using in-house development, off-shore developers or outsourcing. With so many different parameters and so many possible implementations, to fully understand the benefits and drawbacks we need more extensive research in this. Some systems require a more detailed approach used in traditional software engineering, but many software solutions today requires a more agile approach to be able to adapt to the market. With an increased use of agile principles in the industry, it becomes more and more important to understand the benefits and drawbacks, Jeremiah [12]. The market for technology changes faster today than ever and agile inherently fixes a lot of problems that traditional software development tackles with, as presented by Inayat et al [2]. This chapter will first cover agile development in Section 2.1, then briefly describe outsourced development in Section 2.2. The company which participated in this research is presented in Section 2.3, and related research in Section 2.4.

2.1 Agile development

Agile development is based on principles that allow for an adaptive process where the system evolves through iterations. The priority lies in satisfying the customer, which can be achieved by continuous delivery of valuable software. Many of the principles in agile origins from lean development, which focuses on eliminating waste such as excessive documentation, and instead focus on bringing value to the customer [13]. Value to the customer is not limited to working software as soon as possible, but also the use of doing it often, in iterations. It is thus providing value in flexibility, which lets the customer change or add requirements during the whole process. Having this flexibility gives customers an edge in competitive markets.

2.1.1 Agile methodologies

There are many different lightweight methods in comparison to the heavier waterfall method, in regards to the process, tools, and documentation. These different lightweight methodologies aimed to solve the problems that waterfall development had, and each method could be more or less beneficial depending on the different needs of the project [14]. The methodologies include e.g. XP, Kanban, FDD and Scrum, and all processes use an iterative or incremental approach. All of these also have similar values that lie in minimal documentation and high user or product owner involvement. Scrum is specifically in the scope of this research as the company involved uses this methodology, and is thus described in more detail below in Section 2.1.3.

2.1.2 Agile Manifesto

The agile methodologies share similar values, and in 2001, many representatives from the different methodologies gathered to find common ground [1]. The result was the Agile Manifesto which combined 12 prominent principles and created 4 core principles that made up the manifesto [15]. The Agile Manifesto can be seen in Figure 1.
The first principle is about the importance of driven individuals, using face-to-face communication instead of different processes to convey information. The second principle is about delivering valuable software with as little waste as possible by minimizing the amount of documentation. The third principle describes the importance of collaboration with the customer, and that business people and developers work together to enhance rapid development instead of negotiating over contracts. And lastly, the fourth principle highlights the need for flexibility by welcoming requirements during the whole process instead of following a plan defined at the start. The use of all principles assists in a sustainable process.

2.1.3 Scrum

Scrum is an agile methodology that sees itself as a framework which can within itself deploy several different techniques or processes [16]. Scrum uses self-organizing teams which are cross-functional to best accomplish tasks. They can manage to plan and execute tasks alone without being directed or obtaining help by personnel outside of the team. The Scrum team consists of several members. The product owner is responsible for the delivery of the product and is selected because of his/her expertise in the product or market. As an expert, the product owner is in charge of organizing the backlog to maximize the value for each iteration. The product owner is working as a bridge between the customer and developers to ensure the products best interest with all stakeholders opinions and developers suggestions in mind.
The development team is made up of developers, with cross-functional skills and there are no titles among them. The development team itself chooses how to tackle the items in the backlog, but there are no sub-teams. However, the skills may vary among the members so the self-organizing team may allow for specializations for the most efficient implementations, although everyone has the same accountability.

The Scrum Master works differently with different parties. To the development team, the Scrum Master works in a supporting way to coach and facilitate Scrum events.The Scrum Master also helps the product owner to ensure that the development team understands the goal and scope of the product. The Scrum Master also helps the product owner arrange the backlog to maximize value.
Daily Scrum is a daily event which is held by the development team and is no longer than 15 minutes. The Scrum Master makes sure that the event is happening but is not taking part in it, as the questions are related to the developers. The questions each member has to answer is:

- What did I do yesterday that helped the Development Team meet the Sprint Goal?
- What will I do today to help the Development Team meet the Sprint Goal?
- Do I see any impediment that prevents me or the Development Team from meeting the Sprint Goal?

The questions are constructed to enhance collaboration between members of the team, and they usually meet afterward to adapt or replan the rest of the sprint.
The sprint is a period no longer than one month were goals are set to be accomplished. Each sprint is planned and no changes can be made to it that would endanger the set goals. The end of a sprint is a resultant increment of the product that is developed.
The product backlog is a list of ordered items which are everything known to be needed in the end product. The backlog is never complete and it evolves during the whole development process. Each sprint has a set of items from the backlog which are to be completed by the sprints end.

2.1.4 Agile requirements engineering

Instead of eliciting all requirements in the beginning, and detailing them before the development starts as in traditional waterfall development, agile RE promotes the requirements to be introduced during the whole development. This enhances the flexibility of a product to adapt to changing customer needs or development obstacles. Creating a comprehensive documentation at the start is therefore considered to be waste. Eliciting requirements from stakeholders, specifying the requirements in grand detail and verifying the end product before the development starts could result in the development of the wrong product by the time it is finished. Another consequence could be that competitors could manage to launch their equivalent product earlier. Such consequences are avoided by agile, as the development starts earlier, and working software is presented early. This makes it possible to launch the product faster and continuously work on it after release to ensure an advantage over competitors on the market.
Traditional requirements engineering elicit a great number of requirements and specify them into a comprehensive documentation [17]. The document contains functional and quality requirements as well as thorough depictions of the dependencies

on different forms, such as different diagrams and tables. In Agile RE there are usually only user stories that depict the requirements. User stories are high-level definitions of requirements and are containing just enough information so that developers can create estimates of effort for implementing it [18]. User stories are simple and provide little detail. It can, therefore, be beneficial to let the stakeholders write them as they are easy to learn, and they can thus help formulate requirements with their own words. User stories are often focused on functional requirements, and does not take quality requirements into account. It is therefore important to keep them in mind while writing the user stories so they can be specified were possible, or they might be absent altogether. There are guidelines and frameworks to follow for specifying user stories which specify formats to use. A format could be a skeleton template consisting of the crucial information, which is the role, the means, and the end [8]. The format is often written as "As a <role>, I want to <means>, so that <end>", and neglecting any of these parts would make the user story incomplete.

2.2 Outsourced development

There are many reasons for companies outsourcing development to a third party. It can be the lack of having own developers in-house, having no developers available, a question of money, and much more. The outsourced developers are usually part of a consultant firm which can be specialized in developing software systems, but also designing them. The company who outsources the development, containing the stakeholders, is considered to be the customer to the developers, who are also usually referred to as consultants, or suppliers [13]. Developers in outsourced projects often use a person that works closely with the customer and stakeholders. This person should be an expert in the product, or market for the product, be available, and have decision power to both support and guide the development adequately. This person is in agile projects often an onsite customer as in XP [19], a product owner in Scrum [16], or a separate entrepreneur [20].

2.3 Avanade Stockholm, Sweden

Avanade is a joint venture between Accenture and Microsoft, who joined forces to create an IT consulting firm with Accenture's consulting expertise and Microsoft scalable platforms and technologies. In Sweden, they work with consulting services to provide and develop software solutions. They use both their own development team as well as Accenture's, in e.g. India, and work closely with the Microsoft platform for business solutions, such as the Office 365 platform. This research was performed at Avanade in Stockholm, Sweden, where agile development is in focus using the Scrum framework, whom they also have a partnership with [21]. Avanade is thus on many levels ideal for this research, but it also has a few disadvantages which can cause different threats to validity. These disadvantages lie in that Avanade is potentially already very potent in agile, and has eliminated many agile RE impediments already. A full breakdown of the threats can be found in Chapter 5, Section 5.3.

2.4 Related work

The terms “just in time” and “just enough” are concepts that are closely related to the agile methodology, yet they are seldom specified and serve more as guidelines [22]. As a result, Lunder set out to identify how thorough requirements should be specified [23]. However, the research found that the subjects did not agree on the requirements level of detail before the projects are started. Although they wished for the requirements to be specified as early as possible to easier find dependencies and problems. When the development was ongoing and during sprints, the subjects were in larger agreement on the specification level, concluding that it shouldn't either require them spending time interpreting, or taking away their imagination of implementing it. The paper also investigated when user stories could be enough as a requirement, which it could if the project wasn't too complex and were an in-house project. The results were obtained from a case study at a consultant company who used the agile framework Scrum.
The systematic literature review (SLR) done by Inayat et al gathered information in order to map RE practices and challenges faced by agile teams to understand how traditional RE problems are solved using agile RE [2]. In the results of the adopted practices, we can find face-to-face communication, customer involvement, user stories, iterative requirements, requirements modeling etc. Challenges found for agile RE include minimal documentation, customer availability, customer inability, and agreement. The SLR gathered their data through 21 studies that discussed agile RE.
A study conducted by Zhu investigated how to conduct requirements engineering in a Scrum environment [24]. Obtained results relate to the problems and suggestions of possible solutions to the requirements detail level or lack of. The problem was that the requirements were often not specified enough, and additionally, they were not written in any format. Other findings pointed toward no formal documentation in general, where only backlog items where found, which were far from formal. There was also no trace of any non-functional requirements. Minimal documentation and lack of detail level thus pointed towards possible traceability issues. Suggestions were, therefore, to write user stories on a specific format, and include them in a new recommended template for writing requirements. The study used a case study on two different development teams at the same company.
Cao and Ramesh conducted an empirical study about benefits and challenges of agile RE practices [29]. In regards to iterative RE, minimal documentation was one of the discovered challenges, along with neglecting non-functional requirements, and cost and schedule estimation. Minimal documentation was concluded to be the reason for many types of problems, such as scaling the software, evolving the application over time, and introducing new members to the development team. It was also shown in the same study that face-to-face communication can be the root cause of problems associated with minimal documentation. Projects which cannot achieve high-quality interaction has a high risk of requirements lacking in detail or being completely wrong. The main challenges were customer availability and trust between customer and developers. In regards to changing requirements during the development, the biggest challenges lied in redesigns, which could result in refactor-

ing code, or throwing away complete modules because of inappropriate architecture. This means that the cost of the project would potentially increase significantly. The results of this study were obtained from 16 organizations which used agile approaches.
Berry wrote in 2002 a paper, based on own experience and observations of others, about pains caused by requirements when it comes to developing software in computer-based systems [30]. He means that neglecting requirements because they will change later is not a great argument. Berry adds that the code refactoring that needs to be done because of inappropriate or absent architecture is painful and time-consuming. Rewriting or throwing away code because the architecture has changed is both wasteful and increases the development cost. In addition to that, programmers will in time learn that any code is written has a high risk of needing refactoring, and as refactoring is a painful step, it is also subject to be postponed. Each time refactoring is postponed into the next iterations, it will be harder and harder to do.
Meyer published a book in 2014 called "Agile: The Good, the Hype and the Ugly". The book contains, as the title suggests, what Meyer considers the ugly and good parts of agile. Related ugly parts consist of no upfront requirements, using user stories instead of abstract requirements and tests as replacement for specification. The related good parts consist of daily meetings, focus on working code, frequent iterations and acceptance of change.
Miller identified and provided solutions for agile problems, challenges, and failures [32]. The related results include findings of problems such as communicating; managing day-to-day operational problems; gaining buy-in from management, customers, and team members; changing culture and mindset; and gaining experience and making it work. These include approximately fifty sub-challenges, e.g. company culture, no single product owner authority, too many meetings and new to agile. The data was gathered through literature from books, internet research, and reference materials.

3
Methodology

This research is fundamentally based upon understanding the problems that are caused by the requirements in agile projects. It sets out to ask the question what, but also why and how, to discover problems, causes, and solutions. The questions are formulated for answers ideal to obtain from the use a qualitative approach to research design. The data was collected from a consultant company with broad experience in the agile field, with Scrum in particular, and as such it characterizes itself as purposeful, meaning that it is considered information-rich in the selected domain [25]. There are many benefits of performing a case study on a real company and the biggest is the rich and in-depth data of real-life uses, which is harder to obtain in experimental research methods [26].

3.1 Data collection

The company used for collecting data was selected because of its vast experience in agile principles and is thus related to both the purpose of this research and the related work by e.g Lunder [23]. Using a case study where the company itself functions as the subject allowed for the data gathered to be looked at as one entity and therefore analyzed as such, and not individually.
As a case study, there are several ways of collecting the relevant data, including interviews, looking at old documentation and performing observations. Documentation over requirements from previous projects was however not obtainable as it was protected by a company policy to protect the customer. Observations were also excluded as they fell under the same policy. Interviews thus became the main source of data collection, with forms added as a secondary method. The interviews were thus modified to include questions about the documentation in order to fill that possible loss of data from observations and old documents. Nevertheless, it is also unclear how much documentation that exists in regards to requirements, and interviews presumably supply the richest data. In-depth interviews allowed for detailed data which was focused to answer the particular research questions in this study. On top of this, feedback was gathered from the participants after the result had been analyzed to validate the findings. The interview template and validation form can be found in Appendix A and B and is described Chapter 3, Section 3.1.2 and Section 3.1.4 respectively.

3.1.1 Interviews

Using interviews for data gathering offers a high flexibility, as it allows for either probing questions, adapting existing questing or even adding new questions depending on the situation. Because of the nature of a consultant company, many employees in the organization will have different views on the research questions as their contact with requirements may differ. This is good for variety in the sample and such different opinions could possibly not been captured in any other way than by interviews. The interviews were thus kept semi-structured, with both defined questions and room for probing and diverging questions, to adapt to the different participants and their viewpoints. The flexibility which semi-structured interviews provide was therefore important to obtain rich data.
Another benefit of using interviews, or specifically face-to-face interviews, is that you can read and understand body language, voice tone, and notice where the participant's focus is. This makes it easier to understand their situation and adapt and steer the interview by it. The questions in the interviews were built around the subjects own experience, opinions, and feelings regarding obstacles, problems, possible solutions, the process of specifying requirements and what works good right now.

3.1.2 Structure of interviews

All questions were created to fulfill certain goals. First was questions about the participant, which related to their title and tasks. These questions were introduced to identify how the participants came in contact with the requirements, which helped steer the interview later, as well as provided context for each subject when analyzing.
Then came questions related to Avanade itself, how things were done and what differed from theoretical knowledge previously learned. This gave more practical knowledge into the context of the case that is Avanade, and thus helped with adapting the interview from learned theoretical knowledge to more relevant case questions. These questions were also present to make up for the loss of obtaining physical documentation and conducting observations.
After the contextual questions, more specific about the research at hand were introduced. These questions, like the ones before, was exploratory and focused on the requirements and the RE process. The questions were about opinions about how requirements were handled, what problems that could occur and what the participants personally would fix to solve them. The final question related to the participant's optimal way of specifying requirements. The interview template can be found in Appendix A, and an overview can be seen in Table 1.

3.1.3 Sample

The research focused on getting real-life data by doing a case study at a global IT consultant company. The sample is therefore only the company and interview subjects are chosen by convenience [27]. The convenience sampling is based on their knowledge in the area as well as their availability. The company does not use

traditional titles, however, in the sample, we could find titles that are similar or equivalents to project leader, business analyst, Scrum master and system architect to name a few, see Table 2 for all participants. The company uses both in-house and offshore development, which differs from previous research by Lunder [23]. There was, however, no developers interviewed in the sample of this study because the purpose of this study is focused more towards the specification of the requirements, and not the implementation. The relevance of the subjects was therefore judged based on their contact with the requirements in regards to eliciting requirements from the customer, acting link between developer and customer, or the writing of the requirements themselves. This allowed for a varied sample of subjects having a deeper understanding of different areas. Typically, industrial case studies need to balance sample size between availability of interviewees and ability to draw valid conclusions. 6 interviewees were selected for this case study, which is on the lower end of subjects. However, a saturation point could already be seen by the fifth interview with less novel data being found.

Soft categories	Types of questions
Intro	Presenting myself, the structure and goal of the interview
Questions about the participant	Including their position/title, workflow, duties etc.
Questions related to Avanade and their process	Including stakeholder identification, requirement elicitation methods, requirement documentation, detail level, developers contact with requirements etc.
Research specific questions	Stakeholder changing requirements, developers changing requirements, handling changing implemented and not implemented requirements, updated and newly added requirement detail level, using more processes and tools, elicitation process etc.
Final open ended questions/discussion	Problems caused by requirements in an agile process, ideal requirements in an agile process, additional comments from the participants.

Table 1. Table with an overview of the questions asked during the interview.

Participant number	Titles	Responsibilities
1	Manager	Project leader, requirement elicitation
2	Analyst in software engineering	Business analyst, Scrum master
3	Project leader	System archtitect, technical project leader
4	Analyst	Customer and developer management
5	Analyst	Change management / change lead
6	Senior analyst	Infrastructure consulting

Table 2. Table over the participants with their titles and responsibilites.

3.1.4 Validation from participants

Feedback was obtained from the participants after the analysis was completed and a theory for the results could be presented to them. The participants were asked to fill in a form where statements of the results were written, and the participants could share their opinion on how correct the statements were. The form was created using Google Forms and the participants were able to fill it in optionally at their own time. The reason for this was to obtain feedback that was sincere and not rushed through. The participants answering the form would then be genuinely interested in making sure the research got their opinions right. Half of the participants answered the form, and each result of the form can be seen in each section of the result.
This method was preferred at this stage of the research, as it was considered more important than e.g. reaching out to more people for additional interviews. This was both because confirming findings seemed more relevant and that an additional form of data gathering was preferred over extending the single existing one. Time limitation attached to this research gave time for only one, and the preferred method was the one which would add the most evident value.

3.2 Data analysis

An inductive approach was used to group data and look for relationships [28]. Transcribing the recorded interviews was, therefore, the starting point. The transcripts were systematically reviewed in an iterative cycle containing coding, analyzing and writing a memo. Using a similar formal language for all codes helped with identifying interesting themes and connections.

3.2.1 Inductive approach

The inductive approach is a ground-up approach focused on building theories based on patterns and relationships from the collected data [28]. It does not set out to prove a theory by collecting data, but rather collecting data to create a theory, which suited the research question well. The interview questions were not written to answer a defined hypothesis, but rather enabling the research to be exploratory. The inductive analysis is, therefore, suitable as it builds on learning from experience [28]. The experience from the subjects is, as mentioned earlier in this chapter, very relevant. The process of an inductive analysis includes coding the transcripts, identifying trends, form relationships between them and generate a solution theory based on emerging themes and patterns. A memo was written during the analysis phase to record thoughts and findings to keep track during the analysis cycle.

3.2.2 Transcribing interviews

All interviews were recorded and transcribed into different documents, or transcripts. No dedicated transcribing software was used. Instead, the audio file was played back and the content written down systematically. There was thus no additional functionality added to this process, such as automated speech to text.

Everything said during the interviews was written down, both from the interviewee and the interviewer to make the context clear for the analysis. Even sections during the interview with less or no apparent relevance were written down for thoroughness.

3.2.3 Coding interviews

The coding of interviews was an iterative process. Codes were first written on the first read through based on initial thoughts and feelings after reading text segments. Most codes were unique, even for similar excerpts. By reading the transcripts over and over, and by using a memo, the codes could be refined and defined more properly. The codes could thus help shape patterns already at this stage, making connections between current approaches, problems and solutions clearer each time they were refined. Codes could first be written in general about a topic, such as "initial customer contract negotiation", and be refined to be more specific underlying problem e.g. "customer agile knowledge problem", thus making the code clearer to what was described.

3.2.4 Analyzing data

An Excel sheet was created in order to get a higher level overview of the codes. Each code represented a row, and analysis sections were added into columns to grow horizontally. Each code had, first of all, a description and a trend column. The trend column was obtained by calculating the frequency of each code. The codes were then rearranged based on their description and general type. This allowed for a relationship column which explained how the particular code related to other codes. Each code did not relate to any other code within the same type, which created cross-relationship between different types. This relationship was described in a new relationship-describing column to explain how they related, e.g. causation or similar. These columns together then gave an overview for each codes importance in the whole picture.
When a theory was created, it resulted in four categories of impediments, and four categories of solutions. Each category was written as a statement and put into a document, where each statement became a section. Each section was then filled with all quotes coming from the related codes. The document then contained every quote which validated each finding, i.e. statement. Tables were also created over how the statements related to each other, as well as to which research question.

3.2.5 Keeping track with memo

Transcribing interviews, assigning codes and analyzing them is not always done sequentially but rather in a cycle. The memo helps to keep track of anything notable. It contains first of all thoughts, ideas and early trends found when transcribing the interviews. It also keeps track of codes used when coding the transcripts, helping to see all codes from all interviews compiled. The overview of codes allowed for easier code reusing where relevant as well as observing the first patterns. Thoughts were also captured during analysis when trying to find themes, as not all themes

worked out, keeping the old theories intact allowed for a more structured analysis experience.

3.3 Presenting the findings

In order to answer the research questions, we must first define how to present the findings. As mentioned earlier, all results gathered are put together into one entity. This is fundamental because of the nature of this research, where the company lies in focus in this case study, not the subjects that were interviewed. The results gathered are therefore a generalization and are not the thoughts and experiences of every participant. The participants had all varying roles and titles which made a good representation of the company. A larger sample size would have been preferable but even at this size, answers already had similarities or more exact matches.

4
Results

This chapter covers all impediments found at Avanade in this study. It also provides impediments (or challenges) found in related studies which were not found in this study, along with possible reasons to why. On a similar form, this chapter also provides suggested solutions on how to address the impediments, and lastly solutions found in related research. It also reveals where Avanade has already solved known impediments, and how. The two research questions are presented separately in two main sections, with two subsections that first provide the findings from this research, followed by the findings from related research.
Both impediments and solutions are presented below as part of underlying themes. A full overview over the results from RQ1 and RQ2 can be found in Table 4 and a heat map over the participants answers in the form can be found in Table 3, both at the end of this chapter.

4.1 RQ1: What impediments are associated with agile requirements engineering?

This section presents the findings of research question 1. It includes found impediments, how they relate to previous research and known impediments from related studies not discovered in this study.

4.1.1 Impediments found in this study

The sections below are problem areas, or themes, which consists of the impediments. The themes were created as many of the impediments closely relates in regards to the main underlying problem.

4.1.1.1 Lacking customer availability and engagement

As mentioned in the Chapter 2, Section 2.1.2, agile development depends heavily on customer collaboration to be successful. It is one of the core principles of the Agile Manifesto. Different agile practices uses different methods to achieve this crucial collaboration, which in the case of Scrum, is the use of a product owner. There was two main impediments found associated with the product owner and customer collaboration.
First, the *product owner not staffed properly*. There was many reasons found why this is considered an impediment to the project. First of all, it was revealed that

it is the single most important role on the customers side in regards to driving the whole project and the requirements. The product owner has a huge responsibility to the planning process, with both backlog management and sprint verification. Other responsibilities includes making the right architectural decisions and supplying apparent goals for the developers. The product owner must thus be staffed properly to be able to handle these demanding and crucial tasks for project success.
It was also revealed from the participants that when it is not working smoothly, it can impact the project in a negative way on several levels, including development and management. In addition, the product owner is also the link for the development team to the customers, and an additional responsibility thus lies in making it work to enhance collaboration. And as mentioned earlier, participants also agreed that the amount of collaboration with a customer can make or break a project. For these reasons, the product owner should be a dedicated position for a single person as a full time job, and not half staffed because it is required by the Scrum process.
As mentioned in Chapter 2, Section 2.4, Cao and Ramesh revealed two of the main challenges in agile RE to be customer availability and trust between customer and developers [29]. This directly ties into the findings in this study, with participants acknowledging that the product owner should be a full time job, and being able to make decisions and provide goals for the developers.
Second, the *Lacking customer vision*. The reasons for this impediment is due to several causes. The first cause being not providing enough detail to the development team. The cause for this was found to be of two reasons, either the customer did not entirely know their end vision, or they failed to provide it. In the latter case, an example was given for a project where the customers had a clear vision of a new intranet, but took for granted that the developers knew about their old specifications, leaving some of them out. Other similar examples was provided during the interviews, however those contained scenarios where the customer was specifying requirements, but too vaguely. This leads to the next cause, which is misinterpretations.
Misinterpretations can be caused by either being too vaguely defined or too short. Both leaves out information which causes the misinterpretation. A reason for this is not thinking of the others point of view, which hinders from giving out a proper explanation. One participant recognized this issue as it was personally experienced the other way around when communicating with developers in India. The last cause for this impediment ties in with the previous about customer availability. There are cases when customers have a lacking presence in the specification process. Those cases were found to be rarer, but with a potential higher negative impact to the project.
The findings here is also validated by Cao and Ramesh, who reveals that face-to-face communication requires high quality interactions to avoid the high risk of requirements lacking in detail, or being completely wrong [29].
The participants from Avanade validated this result with 1 selecting strongly agree and 2 selecting agree, see Diagram 1.

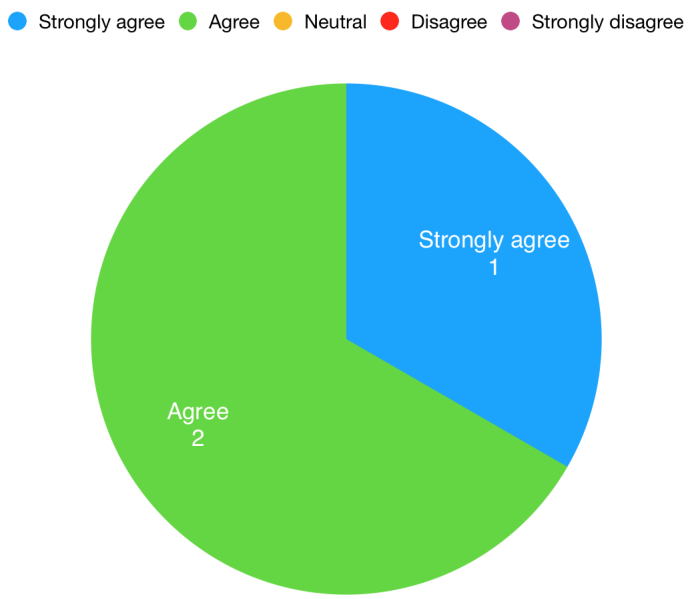

Diagram 1. Participant validation over lacking customer engagement.

4.1.1.2 Insufficient architecture

Working software over comprehensive documentation is one of the principles in the Agile Manifesto [1]. This commonly results in minimal documentation with user stories as requirements. However, participants in this study showed a tendency to it being insufficient for understandability of the system.
The data behind this lies in one major impediment found, which is *lack of system understandability*. A great focus is on user stories and acceptance criteria which validate the implementation of user stories. There are also implementation tasks, which are used for developers. However, when specifying the project, the lack of visual representations can cause understandability issues. The data shows that the majority did not desire a comprehensive documentation with e.g. component diagrams, but preferred a visualization in line with a mock-up or sketch (usually wireframes). It also showed acknowledgment of the short lifespan and high cost of such visualizations, in comparison to user stories and acceptance criteria. Despite that, it was requested to have it at the start of the project, but without the need to update it during development. Some participants provided examples where the use of wireframes and HTML mock-ups helped the execution of projects tremendously. This impediment relates to the customer's vision not getting through, as a clear vision would allow for less architectural changes, giving sketches and mock-ups longer lifespans. It also relates to what the customer is concerned with, which is mostly user stories and acceptance criteria because that is what is needed to validate the product against the contract. Adding time, and cost, to other areas not valuable to the customers, is considered waste. It was thus identified to be a very fine line dependent on the different customers.
Creating too much documentation will cause for redesigns when requirements change, which is the biggest reason for avoiding it. However, as Berry described it, this is not a solid argument [30]. Code refactoring caused by an inappropriate or absent

architecture is painful for the developers, but also time-consuming. This, in turn, adds cost to the project. In addition, rewriting or throwing away code because of architectural changes is wasteful on its own.
Cao and Ramesh also found that the biggest challenge in changing requirements lies in redesigns and refactoring, also concluding a resulting increased cost. It is thus easy to understand that the single most requested improvement was, with a clear customer vision, visualization of the system from the start. This enhances both understandability as well as decreasing potential architectural changes during the development.
From the validation, it was shown that 2 answered that they strongly agree with this, while 1 selected agree, see Diagram 2.

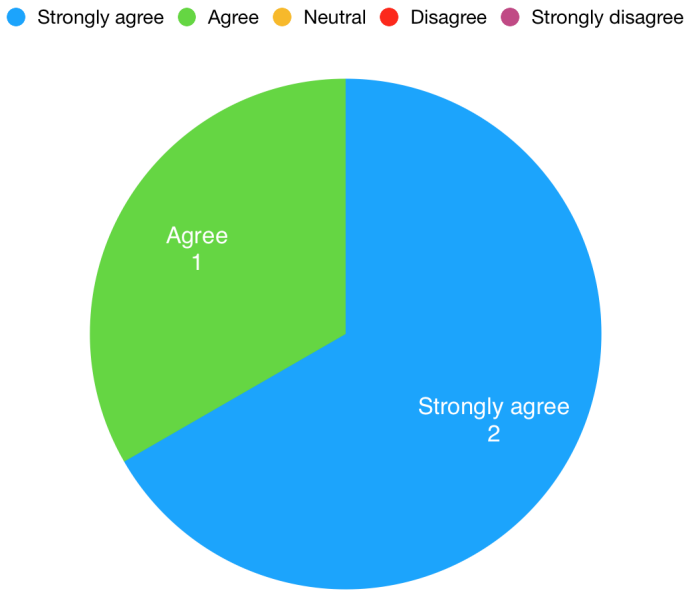

Diagram 2. Participant validation over visualizations.

4.1.1.3 Requirements lacking in detail

User stories are the primary form for requirements in different agile frameworks, e.g. Scrum. The purpose of user stories is to provide developers with just enough information to estimate the time to implement it [18]. The data showed tow impediments that occurs due to lacking detail in user stories.
First, *requirements impede development*. Data showed that there were different reasons for requirements to impede the development process. As shown in the sample in Chapter 3, Section 3.1.3, no developers were interviewed. However, the participants explained that more structured written requirements would not only help the developers but all parties who come in contact with them. Problems included misunderstandings between parties, but also about the requirement itself. There is no format used for specifying requirements at Avanade, which results in detail levels to be different in different projects, depending on different stakeholders. For example, the relationship with the customer or the maturity in the customer's organization can determine how detailed information Avanade manages to obtain. A participant

states that perfect user stories can thus be hard to specify in reality.
The findings show that requirements written more specific could benefit the process greatly. However, not all participants were in agreement on using a dedicated format, as it could be hard to adapt requirements to use the same parameters between different projects. Guidelines were instead favorable for specifying e.g. user stories, to decrease possible ambiguity, and remove the potential time for interpretation.
It was also found that language barriers could be a cause for misunderstandings. Resources in different countries can have different interpretations of the same requirement. Findings show that misunderstandings, from either the lack detail or language barriers, can cause collisions down the road on different levels. The worst effect can be developers implementing the feature wrong, or the wrong feature. This impediment can cause the need for refactoring and adds time and cost to the project.
Second, *requirements causing sign-off conflicts*. It was found that not only requirements in the form of user stories was requested to be specified in greater detail, but also the acceptance criteria. While acceptance criteria was found to be extremely beneficial when working as intended, it was not without its problems if it was not. Smallest impediments caused for misunderstanding, and sometimes even better solutions implemented than the validation required. However the bigger misunderstandings due to ambiguity could lead to a contract twist. Ambiguity in the documentation leads to traceability issues, and results in a word against word conflict.
Zhu uncovered similar results, reporting findings of requirements that were often not specified enough, or written on any format [24]. The minimal documentation and lack of detail level thus also pointed towards possible traceability issues.
Lunder revealed that the detail level of requirements during the project should not require time for interpreting them[23]. The same study also uncovered that user stories could be enough as requirements if the project wasn't too complex or an in-house project. Meyer goes even further and puts user stories as a replacement for abstract requirements as one of the ugly parts of agile development [7]. Even Cao and Ramesh reveals minimal documentation to be a great challenge, being the the reason of many problems such as challenges scaling the software, evolving the application over time, and introducing new members to the development team [29].
Diagram 3 reveals how 2 of the participants selecting strongly agree and 1 selecting agree in the form, as seen in Diagram 3.

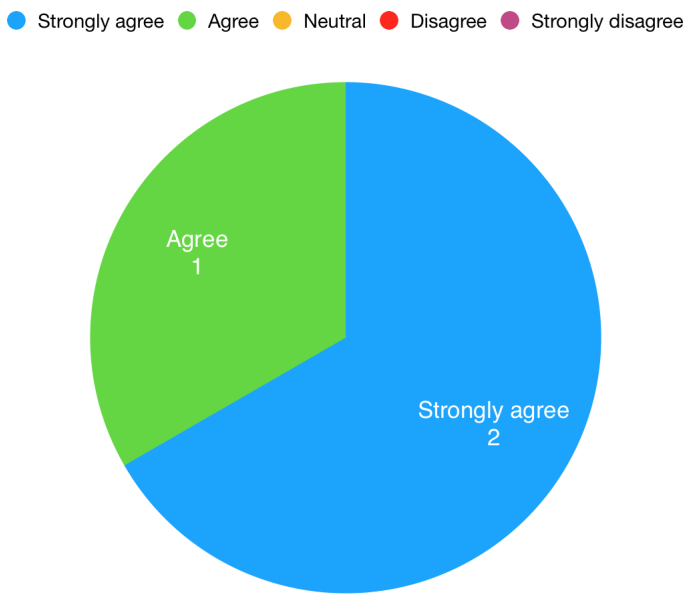

Diagram 3. Participant validation over the detail level of requirements.

4.1.1.4 Customer lacking agile knowledge

Agile has become increasingly popular and more and more are opting for an agile development process. The problem: few know how to work agile, according to the participants in this study. The main impediment, *company culture*, is backed by several findings. This impediment can spread causes to different impediments previously mentioned in this chapter but big enough to be listed on its own. Findings show that customers who wish to work agile contact Avanade because of their expertise as well as their ability to coach agile. Impediments occur as the customer don't partake in the agile process as much as needed. This means that the little or no knowledge the customer had from the start does not evolve, and can cripple the project. Occurring problems include not participating in backlog meetings, not keeping up with the required speed, or velocity, which results in falling behind the schedule. The problems originate in the culture of the organization, who can have a hard time adapting to a new methodology. Participants explain how customers are not used to the iterative process, and that the initial RE work of a waterfall method is somewhat instead spread out during the whole development. This requires hard work during the whole process, which according to participants, inexperienced customers do not realize. Data shows that project success can be dependent on the maturity of the customer's organization in agile. It also shows that it is dependent on the complexity of the project. Customers juvenile to agile has a bigger risk of disregarding documentation completely, which can lead to awkward conversations over the project outcome.
Miller provides several results of agile challenges and problems, including related challenges such as new to agile, company culture, and too many meetings [32]. Miller also provides suggested solutions on how to educate and improve, which is something Avanade reveals to be doing on different levels already, as reported in the next section of this chapter (Chapter 4, Section 4.2).

Validation through the feedback showed 2 selecting strongly agree and 1 selecting agree, as seen in Diagram 4.

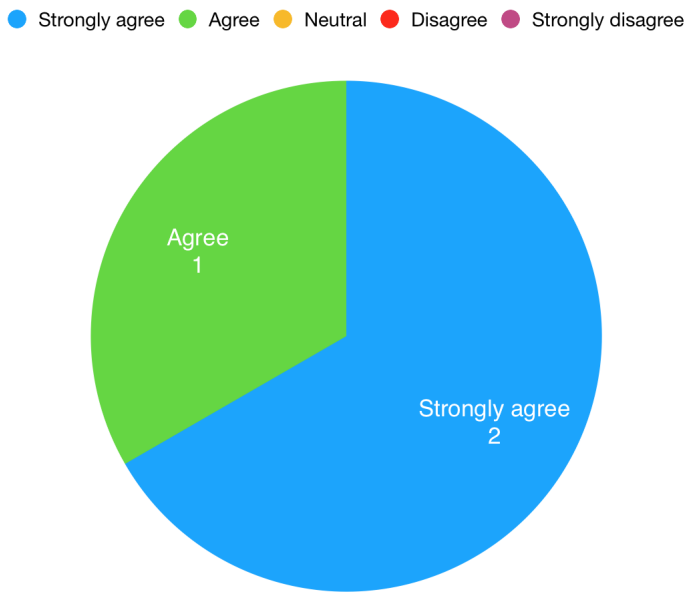

Diagram 4. Participant validation over customers agile knowledge.

4.1.2 Impediments from related research

The related work, described in Chapter 2, Section 2.4, reveals findings related to those of this study. However, the related studies also report additional findings not revealed, or investigated further, in this study. This section provides these findings, and Section 5.1 brings up the discussion on how they relate.

One common challenge to agile requirements engineering, reported by Zhu and Cao and Ramesh, is the *absence of non-functional requirements* [24][29]. Cao and Ramesh report that customers often puts focus on core functionality and neglects non-functional requirements such as scalability, maintainability, portability, safety, and performance. Zhu also reveals that developers and product owners almost never handle non-functional requirements unless there was a relevant defect reported, usually tied to the performance of a functionality. During the interviews of this study, however, there were similar indications. One participant revealed non-functional requirements do not fit the normal agile structure, which could result in potential problems later as non-functional requirements are not addressed in the agile models. This was only briefly discussed by one participant and is thus not part of the main results over impediments provided above, as it was not enough data to verify a common problem tied to it.

Another reported challenge by Cao and Ramesh is tied to *cost and schedule estimation* [29]. The initial estimation is usually based on known user stories, but as new evolve and current change, cost and estimation can vary greatly and must be adjusted. This challenge was not found in this study, except the reported impediments related to *customer lacking agile knowledge*, resulting in e.g. absence during planning meetings.

An additional reported challenge by Cao and Ramesh is *contractual limitations* as summarized by the SLR done by Inayat et al [29][2]. This challenge is tied to written contracts and changing requirements. Changing requirements can e.g. lead to new contracts needed, change of the cost of implementation, and change of implementation time. In this study, however, it was only revealed that contracts need to be specified so it can be broken down into tangible tasks. It was also revealed that the customer does not always partake in this, but is presented with the tasks later to verify it to the contract. No findings of cost and schedule estimations, only the effect of contract conflicts that can occur due to e.g. misunderstandings and traceability issues.
One challenge reported by Berry and Cao and Ramesh, that is an agile method, is *refactoring* [29][30]. This method is a solution to changing requirements but is a challenge as it causes rewriting or throwing away code completely. This subject was not discussed in depth in this study, except defining the architecture earlier in the process to increase understandability and traceability.

4.2 RQ2: How can the impediments caused by agile requirements engineering be addressed?

This section presents the findings of research question 2. It reveals suggested, requested and already implemented solutions by Avanade. It also provides known solutions found in related research not discovered in this study.

4.2.1 Improvements found in this study

The sections below are suggested improvements to address the impediments revealed above. The improvements are presented to address the themes consisting the impediments but are not limited to a single theme.

4.2.1.1 Dedicated product owner

Avanade describes the perfect customer to be the ones serious about the agile approach. There is no requirement of knowing everything from the start, but preferably engaged and motivated to improve and take part in the process. The most requested way to achieve this was revealed to use a *dedicated product owner*. The product owner position could either be staffed full time temporarily for the duration of the project, or long-term if the organization is serious about continuing to work agile. The latter can be preferable in the long run, as the product owner would be more experienced and efficient to drive the project. Participants requested that the customer should take part in more activities than showing up to weekly grooming meetings, which has happened in earlier projects. A product owner which acts as a link between the consultants and the customer can engage both parties to take part in all necessary activities. Additional activities could include being involved in writing acceptance criteria, and preferably have what is known ready before the project start. It was also revealed that Avanade has had discussions on making that

a reality, however, the fine line is to not press the customer with a rigorous process right from the start. Avanade is already to engaging the customer by giving out preparatory work, which is not always finished in time but still considered beneficial.
Lacking presence and engagement from the customer are revealed by Cao and Ramesh to be solved by a surrogate customer [29]. However, they also found that is was seldom a full-time job, and only allowed for part-time access. Similar is reported in this study, and the solution thus emphasizes on full-time, or *dedicated*.
Validation from the participants in this study showed that 2 selected strongly agree and 1 selected neutral, as seen in Diagram 5. The neutral selection can be interpreted differently. It can be seen as that participant is not affected by this solution, and are thus indifferent, or that they would be affected but did not agree that this would provide a sufficient solution.

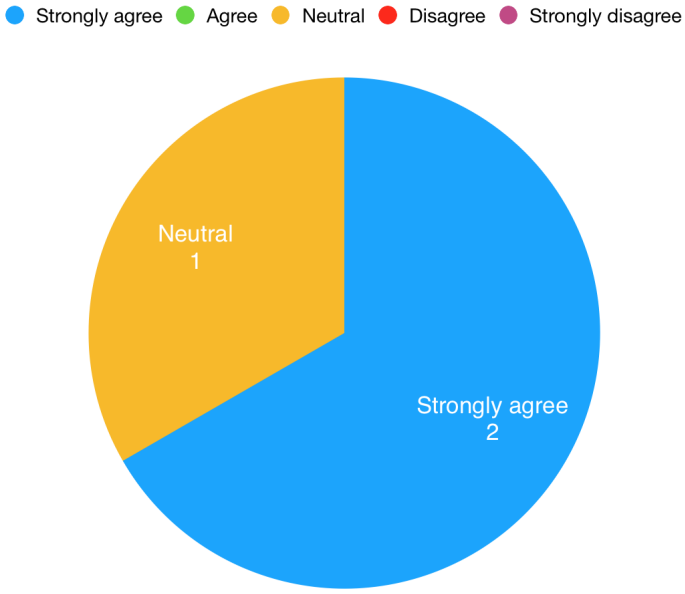

Diagram 5. Participant validation over higher customer engagement.

4.2.1.2 Improved initial architecture

Understanding a requirement is one thing, understanding the system another. *System visualizations* such as HTML mock-ups and wireframe sketches can according to the participants both help with understandability and verifying the customer's vision. User stories and acceptance criteria are fast, easy and have a low cost in contrast to mock-ups and sketches. Especially so with changing requirements, or even a change in vision, which can happen for longer developments in a changing and evolving market. However, system and contract validation for the customer are more efficient with acceptance criteria, as long as it works. Some participants revealed that sketches would be desired on everything, however acknowledging that such a case is not realistic. What some participants ultimately agreed on, however, was the use of initial sketches and mock-ups. As no developers were interviewed, which can be seen in Table 2, there was no data on refactoring code.

In related work, as reported by Berry, refactoring code is the solution to inappropriate architecture [30]. Berry although reports refactoring code as a painful and wasteful step. The conclusion to this solution of using more initial visualization is, therefore, to address the problem before it becomes a painful task. This would, in theory, increase understandability, easier initial contract sign-offs and decrease painful solutions later.
Most of the participants, 2, agreed to this solution, while 1 disagreed, as seen in Diagram 6. The participants who disagreed could have selected so due to this creating more documentation, which goes against their agile mindset. It could also be that it would not help in their work, or a combination of the two.

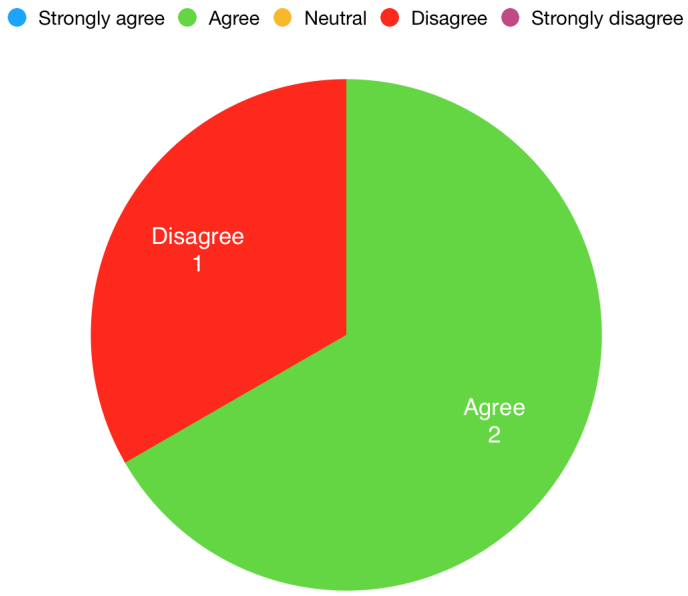

Diagram 6. Participant validation over more architectural artifacts.

4.2.1.3 Greater requirement detail

Data in this study shows that "just in time" (JIT) and "just barely good enough" (JBGE) is not sufficient for most projects. To avoid misunderstandings, leading to possible wrong implementations and contract conflicts, *greater requirement detail* was desired by most participants. Specifically desired was that acceptance criteria should be specified in greater detail, which would also require the customer to take part in this make it correct, and facilitate sign-offs. Data shows that participants found acceptance criteria to be extremely beneficial, especially when working as intended, and a larger focus should be spent on specifying them correct. User stories were also requested to be specified in greater detail, to avoid ambiguity. However, the participant was not in agreement on how to achieve it, as some explicitly stated they did not want to lock either customer or scrum master in a fixed template.
Related studies, e.g. by Lucassen, shows how a template can help write clearer requirements [8]. Writing consistent user stories in a template such as the one described by Lunder will make user stories appear on the same form across all projects, potentially increase learnability, understandability, readability, and maintainability.

A middle ground between JIT or JBGE and a fixed template is to define according to guidelines. Most participants revealed that the amount of detail would preferable be higher and that guidelines could be the answer. Avanade helps customers from the drawing board to a finished solution, but the means depends on the scrum master and customer.
This is validated by the participants, with 2 selecting strongly agree and 1 selecting agree, as seen in Diagram 7.

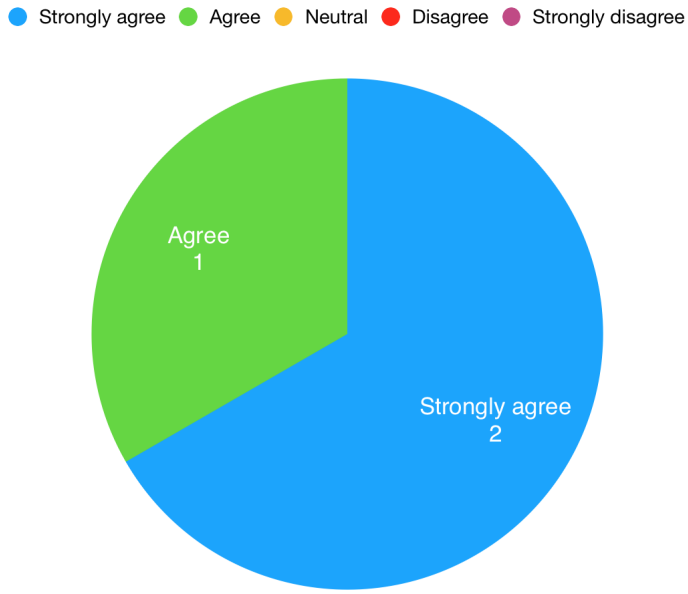

Diagram 7. Participant validation over higher requirement detail level.

4.2.1.4 Customer agile education

Participants explained that a customer intending to work agile is not required to know everything about agile, but preferably have fundamental knowledge about the agile principles and what it means in practice. In addition, participants also explained that a project increases its success rate with customers eager to improve their agile knowledge. Data shows that either *internal or external education* is recommended. Internal is often beneficial for big organizations with many projects. Having their own agile coach, or even as mentioned earlier in this chapter, dedicated product owners can potentially improve the success of their projects. Another reason for internal education is for an organization to guard their own interests. Greater internal agile knowledge can minimize mistakes that can be made for a customer juvenile in agile.
A smaller organization, or a larger who is just getting into agile development, can start with external education. The use of an external coach can, however, be more expensive in the long run. Avanade is already solving the customer agile education impediment by providing the option to use agile coaches. However, not all customers take advantage of this. It is also worth noting that even with an agile coach, it requires the customer to engage and commit to the process. Data, however, shows that coaches have been effective to increase the number of requirements and the

formality of them. It also helps with customers that are not used to the speed of the agile projects. To conclude, data showed that customers comfortable with working agile was executed better, and had increased success rate.
It can be seen in Diagram 8 that participants strongly agree, agree and are neutral to this with 1 selection respectively. Thus showing that the majority finds it important that the customer is better prepared or educated for agile development. The neutral selection is not interpreted as a disagreement, but rather that this solution would not affect their daily work as much as for the ones selecting agree or strongly agree.

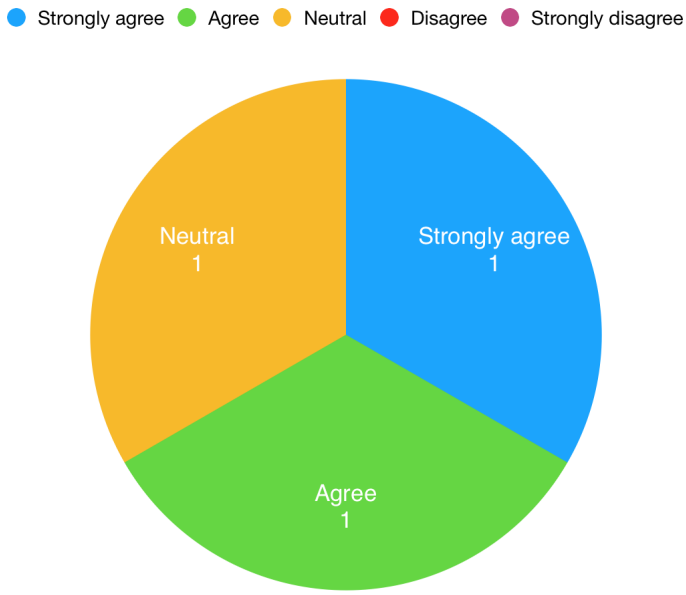

Diagram 8. Participant validation over customer education.

4.2.2 Improvements from related research

Chapter 2, Section 2.4, reveals findings from related work. A lot of the improvements, or solutions, reported in those studies relate to what was found in this one. Some solutions, however, did not appear in this study, such as *code refactoring* and following a *user story format*. Similar to Section 4.1.2, this section provides the solutions found in related research, and Section 5.1 brings up the discussion on how they relate.
Cao and Ramesh, as well as Berry, reports how code refactoring is a solution to inappropriate architecture and changing requirements [29][30]. However, Cao and Ramesh report that this solution depends on several factors, including the developers' expertise and schedule pressure. Findings also showed that refactoring did not completely address the problem of inadequate or inappropriate architecture. It could e.g. lead to throwing away code and rewriting entire modules. Berry reports similar experiences, with refactoring being both painful and wasteful.
The next solution from related work, as reported by Lucassen, involved using a format for specifying user stories [8]. While this study showed a tendency to this

solution, committing to always use a strict format was not desired as a solution.

		Strongly disagree	Disagree	Neutral	Agree	Strongly agree
RQ1	Lacking customer availability and engagement	0	0	0	0,67	0,33
	Insufficient architecture	0	0	0	0,33	0,67
	Requirements lacking in detail	0	0	0	0,33	0,67
	Customer lacking agile knowledge	0	0	0	0,33	0,67
RQ2	Dedicated product owner	0	0	0,33	0	0,67
	Improved inital architecture	0	0,33		0,67	0
	Greater requirement detail	0		0	0,33	0,67
	Customer agile education	0	0	0,33	0,33	0,33

Table 3. Overview over validation from participants.

	Impediment theme	Impediments	Presented in section	Description of impediments	Participant validation	Solutions	Presented in section	Description of solution	Participant validation
Found in this research	Lacking customer availability and engagement	*Product owner not staffed properly*, *Lacking customer vision*	4.1.1.1	Customers are not always staffing the product owner position sufficiently, lack of engagement to provide the correct project vision.	1 strongly agree, 2 agree	Dedicated product owner	4.2.1.1	*A dedicated product owner* can solve availability and engagement impediments from the customer.	2 strongly agree, 1 neutral
	Insufficient architecture	*Lack of system understandability*	4.1.1.2	Lack of initial visualisations can cause for misunderstandings and rework.	2 strongly agree, 1 agree	Improved initial architecture	4.2.1.2	Using *system visualisations* such as wireframes and mock-ups can increase the system understanding.	2 agree, 1 disagree
	Requirements lacking in detail	*Requirements impede development*, *Requirements causing sing-off conflicts*	4.1.1.3	Requirements lacking in detail can cause for misunderstandings and remove extra time for interpretations. Lacking detail level can also cause for sign-off conflicts during contract verification.	2 strongly agree, 1 agree	Greater requirement detail	4.2.1.3	*Greater requirement detail* improves the user stories and acceptance criterias, helping with e.g. contract sign-offs.	2 strongly agree, 1 agree
	Customer lacking agile knowledge	*Company culture*	4.1.1.4	Customers juvenile to agile and lack of interest to improve does not partake in the process.	2 strongly agree, 1 agree	Customer agile education	4.2.1.4	Customers can use *internal or external education* to acquire agile knowledge to enhance the development as well as protect their own interests.	1 strongly agree, 1 agree, 1 neutral
Found in related research	-	*No non-functional requirements*, *Cost and schedule estimation*, *Contractual limitations*, *Refactoring*	4.1.2	Non-functional requirements are seldom covered unless a relevant defect is reported. Cost and schedule estimations are challenging as new are revealed and old are updated. Contractual limitations often occur when changing requirements require contracts to be updated. Redactoring is reported to be an impediment when ongoing throwing away code blocks and entire modules.	-	*Refactoring*, *User story format*	4.2.2	*Refactoring* is an agile method to continuously improve and change code to adapt to the evolving system through iterations. Following a *user story format* can provide useful to structure the requirements and make them consistent through projects.	-

Table 4. Summary over the results.

5
Discussion

The impediments and suggested solutions are not all agile specific. There are thus many contexts these results can be interpreted and adapted to, both in practice and in research. There is, therefore, a section about the contribution to research and the implications to practice here, as well as discussions about significance, validity and future work.

5.1 Contribution to research

The results presented in this research is related to previous work mentioned specifically in Chapter 2, Section 2.4. This section will focus on how the impediments and solutions found relate to previous research and how it contributes to the bigger picture in this research area.
The results provide findings from both this study as well as from related research. Some of the findings are more domain specific than others, making the results contribute more to the research of this domain. Customers use Avanade for their agile expertise and domain knowledge of the Microsoft platform. The results are thus providing a lot of contextual findings. However, there are a lot of similarities between in-house and outsourced agile development, and the related research covers both.
First, there is the impediment about the customer's role during development. In the SLR by Inayat et al, one major challenge in agile RE is the customer availability [2]. The research cited there is from Ramesh et al who presents challenges in the availability of customer for requirements negotiation, clarification and feedback [29]. The impact is stated to increase in rework and that the solution to this challenge is to use surrogate customers. In practice, the surrogate customer can e.g. either be playing the role of a product owner or be an on-site developer to be a representative on the customer's side. The findings in this research however points toward more work for the consultants, time they could use on more valuable tasks, and not specifically to rework. The end cause is however the same, that extra work is needed which could be avoided by a customer taking a bigger, more present role in the development. Ramesh et al suggested in their research that the solution to the customer availability challenge is to use surrogate customers, which relates to the suggestion provided here about using a dedicated product owner [29]. The solutions are similar, but the suggestion by Ramesh is more focused on what the developers could do, and less what the customer could. From the the data gathered in this research, it seemed to be more desired that the customer took a bigger role in driving

the development, preferably with a dedicated product owner.
Most of the related research gathered did not state any challenges specific to the customers knowledge. It can be suggested that the minimal documentation or customer availability is a cause of lacking knowledge however. But this was found to be an explicit impediment in this research, which was highly validated by the participants with 66,7% answering strongly agree and 33,7% selected agree, as seen in Diagram 2. When it comes to the solution about a higher focus on customer education to solve this impediment, the participants selected strongly agree, agree and neutral with 33,3% respectively, as seen in Diagram 6.
Minimal documentation and inappropriate architecture are two challenges found in the SLR by Inayat et al, about agile RE challenges, which strongly relates to the impediment about depicting requirements thoroughly [2]. The data is provided by Zhu who shows that user stories and product backlogs are the only documents in agile methods and that it is the major cause for traceability issues [24]. The challenge lies in that there is only user stories and product backlog to lean on, which was also an impediment in this study in some cases. The difference found in this research is however that the main challenge that arise is the decreasing understandability of the scope. Traceability is however a closely related attribute, since they both lean on learnability in some way. The difference being that understandability (in this case specifically) is considered short-term while traceability is long term. Meyer is on the same page calling the fact that user stories are being a replacement for abstract requirements one of the ugly parts of agile [7].
Inappropriate architecture is also relevant to this study, and the impact found in that SLR was about increasing cost, which can now be verified again by this study. Ramesh et al explains that early depictions of the architecture becomes inadequate in later stages with new or changing requirements [29]. This was also found and validated by this research as 66,7% of the participants selected strongly agree and 33,7% selected agree about this statement, as seen in Diagram 3.
The solution for architectural changes are code refactoring, which is an ongoing agile practice. This is thus not specifically related to depicting of the architecture. The data found in this research points towards the systems visualization, which can avoid the need of adapting code later. The solution provided in this research is that mock-ups and sketches are desired mostly in the beginning of the project, keeping them up to date is thus not necessary as they are likely to change. This means that with initial sketches, less refactoring is needed, which according to Berry is a painful procedure anyway [30].
Findings in the research by Lunder states as previously mentioned that user stories could work well if the project is not too complex or is an in-house project [23]. The challenges found was e.g. that user stories became unmanageable in big projects, that they did not cover enough or that they were too user oriented. Impediments found in this study thus relate to those challenges as both studies identified that user stories are not always clear enough. This is validated through the feedback, where it shows that 66,7% strongly agreed and 33,7% agreed, as seen in Diagram 4. Meyer emphasized this even more stating that user stories are to requirements what tests are to software specification [7]. Even though there are differences here, the bottom line is that user stories are seldom as specified as they are needed to be.

Meyer stresses the importance of user stories being written the right way, using in practice the same template provided by Lucassen et al about the role, means and end [8]. This was also validated by the participants in this study, with most of them specifically desiring a clearer specified end.

5.2 Implications for practice

Not all of the solutions provided in this study are specific to agile development. They are nonetheless present and important in software development, so the implications to practice can differ in a traditional waterfall development to an agile development. Other parameters that can vary in practice are in-house development or the use of other agile practices. The company in this research used Scrum, but other agile frameworks that could be used instead are e.g. Kanban or Extreme programming. But any of these agile methodologies would benefit somewhat equally, as they all are derived from the agile principles. All suggested solutions are not to be taken as musts to fix all impediments. A single solution could impact many areas, such as putting larger focus to customer engagement can in practice lead to better collaboration, which could lead to better-written requirements, better testing, feedback and a faster more efficient development. A larger focus on customer education facilitates both the customer engagement in current projects, but also for upcoming future agile projects. Both these solutions require more time and effort from the customer, which would increase cost and manpower in the short term. However, in long-term, it would be beneficial to have employees knowledgeable in agile, especially if they intend to keep working agile. A downside is however if they put time and money to train personnel in agile, which short-term adds cost, and personnel later leaves the company and obviates the long-term benefits.
For consultants or the in-house development team for that matter, to increase the focus on the thoroughness of the requirements will also add time and effort, resulting in short-term costs. It will also add waste if the system changes a lot. However, it will also add cost if the requirements are less thoroughly written and it causes confusions, impediments or complete schedule stops. The short-term benefits are that the projects will run smoother for more or less all parties. The long-term benefits could include e.g. learnability, which is good for both junior and senior employees, to know what works and what doesn't.
Traditional waterfall projects could also benefit from the suggested solutions in practice. Customer engagement would result similarly to agile projects, while customer education in agile can be disregarded completely. Customer education, in general, can, however, give positive benefits, but that would only apply to at most junior employees. Perhaps not even them, as many are already schooled in the traditional ways from college, high school or similar. A larger focus on depicting or specifying requirements thoroughly is probably also something that is already taught and already has high focus since waterfall projects want everything in grand detail already from the start.

5.3 Threats to validity

The threats to validity in this research comes from e.g. case study as research design, sample selection and researcher's bias. The threats are not presented in any order of significance, but many relate to the threats presented by Maxwell, in regards to performing a qualitative research [31].
First off there is the threat to bias in regards to answer the research questions. Researcher's bias is both a threat and an asset in that it helps dig deeper into the analysis to prove or discover what was set out for. Noticed when analyzing the data was that the participants were generally asked about problems, and not impediments or hinders. The answers obtained were therefore mostly strong opinions and it showed that not all of these are problems, but actually impediments. Some were minor and some more significant. As it was an exploratory research, there was however no problem to pivot it slightly. Thus, the questions in this study changed from being about proving or solving a problem, to explore and provide solutions based on the findings. The bias therefore shifted during the research, which made the questions asked following one line, and the analysis following a different. This somewhat eliminated the bias to prove the research question since there was no chance later to pivot the questions which could be done during the interviews. To validate the findings there was however a feedback process where the participants in the study could fill in a form about the findings. This form acted as both a way to eliminate the researcher's bias, as well as validating the results in the study.
Another threat is that it is done at only one company. Gathering data from more companies could be another way of doing it, or to single down on e.g. project managers or Scrum masters only. The way this was handled was to only use a single company, but to interview as many different subjects as possible, that still had contact with requirements in agile processes. It is still a threat in selecting a sample this way, since they were contacted based on their involvement, and choice to participate, which could be based on their own interest or stake in this matter [27]. It could be that they are genuinely interested in helping the research, or that they have strong feeling towards the research questions, or want to impact something. A random sample would solve this, but could also include employees who are not interested and give less detailed answers, or simply does not have contact with requirements and therefore mostly supplies data that is not needed, or adding outliers. Both ways of choosing the sample has benefits and drawbacks, but it is still an important threat the the validity to keep in mind.
The company itself, Avanade, who works with the agile framework Scrum, can add a threat related to their agile knowledge. The fact that they are not juvenile makes it great for this kind of research, as this research does not focus on inexperienced agile companies. There are however both positives and negatives by collecting data from a company that has a functioning agile culture. With them having knowledge and a working processes already, some impediments could be considered minor for them but bigger for other companies. It was revealed during the interview that the way they work now works good, but the impediments could still be present and improvements could also be added. Data gathered can be seen to be normal agile concerns, and not novice agile mistakes.

Another threat is how the sample was treated as one entity. If they would have a lot off different opinions it would not be possible. But fortunately the company seemed to have all similar experiences with what worked and what didn't, and that can be due to a good culture within the company where problems are talked about and highlighted for improvement purposes. Many therefore seemed genuinely interested in participating in this research. A drawback on the same note is however that it was only the company that was interviewed, and not customers they have or are working with currently. This makes the data one-sided from the company's perspective, and though it makes it easier for comparison to other companies in similar studies, it still causes a validity threat.
In regards to the data gathering itself, the threat present there is of course the primary method of gathering data by using interviews. The other intended data to be gathered could not be performed due to reasons such as e.g. client data protection policies. Fortunately it was possible to get feedback on the result from the same participants, which makes the findings more validated at the end.

5.4 Significance of the study

The findings in this study are significant to both practitioners and researchers. It affects practitioners in one direction and one indirect way with the direction being the results taken straight from this research. The indirect way being the long-term effect of this area being more researched, which contributes so the best contextual practice can be achieved.
The definition of practitioners is anyone who is related in similar cases as the one presented in this study. That means both to the customer and the consultant's side. A customer who is making the jump to agile, or already works agile but strives for improvement should see great significance in the findings and results presented. That concerns both product owners and project leaders as well as Scrum masters and developers who can see the importance of education and knowledge in practice. It affects the consultant side similarly with it being just as important for everyone affected. The difference can lie in that the consultants at an IT-firm are perhaps already schooled in agile, so the significance can vary.
The significance to researchers also diverges. Just as this research expands on previous, so can further research expand on this, which can be done by either replicating or changing variables. It is therefore significant both for this type of case, but also among all agile research to give broader and more complete pictures of agile in practice.
The significance of this research in the context of this case is high, but not complete. Further studies are needed in different cases to factually validate the results, and thence it should hold a complete significance.

5.5 Future research

There are many possibilities for future research, most notably to replicate this case at another company. It is then possible to both compare results if they differ and

validate if they are similar or equal.
A variant of this is to use several companies, and perhaps to focus on participants of a single employment position, such as system architects or project managers. That would yield a more consistent data in term of position but a different view between companies.
Another example of future research is to conduct a similar research as this but with a focus on quality requirements instead. It could also be interesting to see that research from the customer's side, how their quality requirements are handled by the outsourced company.
There are also findings in this study about the importance of the customer's knowledge in agile. There seems to be little research about this in regards to outsourcing agile development, making this a very interesting topic for future research.

6
Conclusion

The data found in this research shows that there are challenges in agile requirements engineering, and several impediments are presented as a result. Additional results present ways of addressing these impediments. The focus of this study was to obtain information from practitioners, with the compliment of related research in this area. The impediments found was related to customer's role and agile knowledge as well as the depicting and specification of the requirements. Solutions were based on engaging and educating the customer in agile, as well as increasing the focus on thoroughly depicted and specified requirements.
The data was gathered through interviews, processed and then validated through feedback from the same participating interviewees. Related work in this area had similar findings which made for good external comparisons and validations.
The findings show that there are a lot of hinders in agile development, as well as many possible solutions. A conclusion to be made after relating findings in this thesis to related research is that a lot of problems and solutions are contextual. The domain in which this study was performed in provided impediments and solutions which can be more or less unique to the domain, in contrast to those of related studies. Using a product owner for example can be standard for an in-house agile development team, but less obvious for a customer to staff when using outsourced development. The same could be applied for the impediment with a lacking customer vision and engagement, as customers can take for granted that the consultants will solve their issues by handing them over without further taking sufficient part and responsibility for it.
The most domain specific impediment is however how requirements can cause for sign-off conflicts. In contrast to related findings which focused on changing requirements, this impediment focused on the ambiguity of the requirements in the first place. This kind of impediment is thus more improbable to occur outside of this domain's context. And lastly there is the impediment with customers lacking agile knowledge. As mentioned in the results, many interviewees stated that a customers usually wants to work agile, but lacks the knowledge in it. This can of course also occur in an in-house team, however most developers are familiar with agile through education or practice, making it an impediment more probable in outsourced environments.

Bibliography

[1] K. Beck, M. Beedle, A. van Bennekum, et al. (2001). The Agile Manifesto [Online]. Available: http://agilemanifesto.org/history.html

[2] I. Inayat, S. Salwah Salmim, S. Marczak, M. Deneva, S. Shamshirband, "A systematic literature review on agile requirements engineering practices and challenges," Computers in Human Behavior, 51:915-929, 2015.

[3] A. Mersino. (2016, Aug. 1). Agile Projects are More Successful than Traditional Projects [Online]. Available: http://vitalitychicago.com/blog/agile-projects-are-more-successful-traditional-projects

[4] R. Arthaud. (2015, March). When every new iteration can violate previously satisfied requirements [Online]. Available: https://re-magazine.ireb.org/issues/2015-3-thinking-without-limits/are-requirements-still-needed

[5] K. Beck, M. Beedle, A. van Bennekum, et al. (2001). Manifesto for Agile Software Development [Online]. Available: http://agilemanifesto.org/

[6] S. W. Ambler. Just Barely Good Enough Models and Documents: An Agile Best Practice [Online]. Available: http://agilemodeling.com/essays/barelyGoodEnough.html

[7] B. Meyer. Agile! The Good, the Hype and the Ugly, 1 ed. Switzerland, Springer International Publishing, 2014

[8] Lucassen, G., Dalpiaz, F., van der Werf, J.M.E.M. et al. Requirements Eng (2016) 21: 383. https://doi.org/10.1007/s00766-016-0250-x

[9] S. W. Ambler. Agile Requirements Best Practices [Online]. Available: http://agilemodeling.com/essays/agileRequirementsBestPractices.htm

[10] S. Snow (2015, Oct. 15) The Problem With Best Practices [Online]. Available: https://www.fastcompany.com/3052222/the-problem-with-best-practices

[11] C. Larman, V. R. Basili. (2003). Iterative and Incremental Development: A Brief History [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.14.2138&rep=rep1&type=pdf/

[12] J. Jeremiah (2015, May 25). Survey: Is agile the new norm? [Online]. Available: https://techbeacon.com/survey-agile-new-norm

[13] Sillitti A., Succi G. (2005) Requirements Engineering for Agile Methods. In: Aurum A., Wohlin C. (eds) Engineering and Managing Software Requirements. Springer, Berlin, Heidelberg

[14] M. Phil. (2015, Mar). Comparative Analysis of Different Agile Methodologies [Online]. Available: www.researchpublish.com

[15] K. Beck, M. Beedle, A. van Bennekum, et al. (2001). Principles behind the Agile Manifesto [Online]. Available: http://agilemanifesto.org/principles.html

[16] K. Schwaber, J. Sutherland (2017). The Scrum Guide [Online] Available: https://www.scrumalliance.org/why-scrum/scrum-guide

[17] Donn Le Vie, Jr. on (2010, Aug. 29). Writing Software Requirements Specifications (SRS) [Online]. Available: https://techwhirl.com/writing-software-requirements-specifications/

[18] S. W. Ambler. User Stories: An Agile Introduction [Online]. Available: http://www.agilemodeling.com/artifacts/userStory.htm

[19] D. Wells (1999). The Customer is Always Available [Online]. Available: http://www.extremeprogramming.org/rules/customer.html

[20] Selleo (2014, Jun 20). Key Players in Outsourced Software Product Development [Online]. Available: http://selleo.com/blog/software-outsourcing/key-players-in-successful-outsourced-software-product-development/

[21] Scrum.org. Scrum.org Continues to Help Avanade Build Its Worldwide Agile Capabilities [Online]. Available: https://www.scrum.org/resources/scrumorg-continues-help-avanade-build-its-worldwide-agile-capabilities-1

[22] B. Hartman. (2009, Feb 23). New to agile? Remember one thing: Just enough, just in time [Online]. Available: http://agileforall.com/new-to-agile-remember-one-thing-just-enough-just-in-time/

[23] K. Lunder, "A Case Study of Requirements Specification in an Agile Project," M.S. thesis, Dept. of Informatics, Univ. of Oslo, Oslo 2014

[24] Y. Zhu, "Requirements Engineering in an Agile Environment," M.S. thesis, Dept. of IT, Univ. of Uppsala, Uppsala 2009

[25] USC Libraries. Research Guides [Online]. Available: http://libguides.usc.edu/writingguide/qualitative

[26] Z. Zainal, "Case study as a research method," Journal of Humanity, Fac. of. Mgmt and Human Res. Dev., Univ. of Techn. Malaysia, 2007

[27] J. Dudovskiy. Convenience sampling [Online]. Available: https://research-methodology.net/sampling-in-primary-data-collection/convenience-sampling/

[28] J. Dudovskiy. Inductive Approach (Inductive Reasoning) [Online]. Available: https://research-methodology.net/research-methodology/research-approach/inductive-approach-2/

[29] Cao, L. C. L., Ramesh, B. (2008). Agile Requirements Engi- neering Practices: An Empirical Study. IEEE Software, 25(1), 60–67.

[30] Berry, D. M. (2002). The inevitable pain of software development, including of extreme programming, caused by requirements volatility University of Waterloo. In Proceedings of the international workshop on time constrained requirements engineering (pp. 1–11).

[31] Maxwell, Joseph A (1992). Understanding and Validity in Qualitative Research. Harvard Educational Review; Research Library Core pg. 279

[32] Miller, G. J. (2013). Agile problems, challenges, and failures. Paper presented at PMI® Global Congress 2013—North America, New Orleans, LA. Newtown Square, PA: Project Management Institute.

A

Appendix A

Questions:

Starting the interview:

- Present yourself
- Explain the goal of the interview
- Describe the structure of the interview:
    - First questions about you
    - Then about the process
    - Then about the requirements themselves
    - Then about your opinion

1. What is your position/title?
2. What does your work include? Tasks/duties?
3. How are stakeholders identified?
    a. Are different stakeholders opinions prioritized?
    b. Does stakeholders prioritize different functionality
4. What methods are used for requirement elicitation?
    a. Does the result of some methods weigh more than other?
5. How are the requirements documented? For example:
    a. Context diagram of the scope, stakeholder analysis, use case diagram, task descriptions, functional and non-functional requirements
    b. Are certain formats or templates used?
6. Could you describe how functional requirements are written?
    a. Following a template:
        i. What does the template look like? What does it require?
    b. Not following a template:
        i. How are they formulated?
        ii. Do they have any measurable properties?
        iii. Why are they formulated as such?
            1. Own preference / The team's opinions / Superior demands?
            2. Are they based on a JIT/Good enough approach?
7. How do the developers see the requirements?
    a. In the backlog or the whole documentation?
    b. Are the backlog tasks modified versions of the requirements?
        i. If so, how?
8. How often do stakeholder change their preferences?
    a. What are the usual causes for this?
9. How often do developers need to change requirements?
    a. What are the usual causes for this?
10. About handling changes:
    a. How are the changes handled on requirements not yet implemented?
        i. Changing existing requirements, adding notes, creating new and removing old etc?
    b. How are the changes handled on requirements that already are implemented?

        i. Changing existing requirements, adding notes, creating new and removing old etc?
11. Does changed/updated requirements have the same formulation as initial requirements?
    a. If yes:
        i. Are they less or more specified? What differs?
        ii. Why?
12. What are your thought about the current elicitation process?
    a. What would you like to change?
    b. Why?
    c. How?
13. What are your thoughts about how the requirements are formulated?
    a. If template:
        i. Would you like to change the template?
            1. How?
            2. Why?
        ii. Use no template?
            1. Why?
        iii. Is the template an obstacle?
            1. Why?
    b. If no template:
        i. Would you like to use a template? Or guidelines?
        ii. Would you like to take part in the feedback from the team / result of the research?
14. Has problems occurred due to the formulation of the requirements?
    a. From what party?
    b. What kind of problems?
    c. How are they solved?
15. How will the ideal requirements flow look like in your opinion?
16. Is there anything I forgot to ask? Anything you would like to add?

B

Appendix B

Research question 1: What impediments are associated with agile requirements, and why do they occur?

The customer does not take a big enough role in the development (no dedicated product owner etc, not agile specific)

- ○ Strongly agree
- ○ Agree
- ○ Neutral
- ○ Disagree
- ○ Strongly disagree

The customer lacks knowledge in the agile ways (specific about their agile knowledge)

- ○ Strongly agree
- ○ Agree
- ○ Neutral
- ○ Disagree
- ○ Strongly disagree

The requirements are not specified thoroughly enough (not enough sketches/mockups etc)

- ○ Strongly agree
- ○ Agree
- ○ Neutral
- ○ Disagree
- ○ Strongly disagree

The requirements themselves are not specified enough (vaguely formulated, no defined end etc)

- ○ Strongly agree
- ○ Agree
- ○ Neutral
- ○ Disagree
- ○ Strongly disagree

Research question 2: How can the impediments caused by agile requirements be solved?

Larger customer engagement, preferable dedicated product owner

- ○ Strongly agree
- ○ Agree
- ○ Neutral
- ○ Disagree
- ○ Strongly disagree

Larger focus on customer knowledge in agile (use of coach, classes etc)

- ○ Strongly agree
- ○ Agree
- ○ Neutral
- ○ Disagree
- ○ Strongly disagree

More focus on having enough documentation (a low-limit)

- ○ Strongly agree
- ○ Agree
- ○ Neutral
- ○ Disagree
- ○ Strongly disagree

More defined of how a requirement should be specified

- ○ Strongly agree
- ○ Agree
- ○ Neutral
- ○ Disagree
- ○ Strongly disagree