Master of Information Systems: Digital Business & Management and Innovation

Awareness of non-functional requirements in agile software projects

Sebastian Berg Hestvedt, 705792 & Marie Raae Stedje, 705940

A report submitted in partial fulfillment of the requirement for the degree of Master of Information Systems: Digital Business & Management and Innovation

Supervisor: Asle Fagerstrøm

Restricted: ☐ Yes ☐ No

Kristiania University College
Prinsensgate 7-9
0107 Oslo
Norway

Abstract

In this master thesis, we have researched how different companies in the software development sector in Norway handles non-functional requirements, and which requirements should be emphasized. This has been conducted by interviewing eight informants on four different projects with different experience and roles. Our findings indicate that software teams have some awareness of non-functional requirements, but they are not prioritized and handled at the same level as functional requirements. Also, the technical knowledge of both the client and customer has an impact on how non-functional requirements are emphasized.

Keywords: non-functional requirements, requirements engineering, agile software development, agile teams

Acknowledgements

This master thesis is written as a part of the master's program in Information Systems at Kristiania University College in spring 2020. The purpose of the assignment is to look more closely at how agile software teams handle non-functional requirements in software development projects and why it should be emphasized. This thesis was carried out in collaboration with several companies that are within the software development domain.

We want to thank our supervisor, Professor Asle Fagerstrøm, for useful guidance, motivation and valuable feedback. Furthermore, we would like to thank all the informants from the different companies for the interviews we have conducted. There have been some special circumstances during this period, and we appreciate that all the informants could take out some time to bed interviewed on a busy workday. Last but not least, we would like to thank friends and family for knowledgeable input on corrections, spelling and grammar.

I certify that the work presented in the thesis is my own unless referenced

Signature: Sebastian Hestvedt Marie Kaas Skadje

Date: 24.05.2020

Total number of words: 18 839

Table of Content

1. Introduction **7**
1.1 Background **7**
1.2 Purpose **8**
1.3 Structure **9**
2. Literature Review **10**
2.1 Requirements engineering **11**
2.2 Agile requirement practices **12**
2.2.1 Challenges with agile requirement engineering 15
2.3 Requirements **15**
2.3.1 Functional and non-functional requirements 15
2.3.2 Classifying of non-functional requirements 16
2.3.3 Challenges related to non-functional requirements 20
2.4 Solution proposals for non-functional requirements **21**
2.4.1 Non-functional testing 21
2.4.2 Documentation proposal 22
2.4.3 AFFINE 24
2.4.4 Modeling non-functional requirements 26
3. Method **27**
3.1 Research Method **27**
3.2 Research design **29**
3.2.1 Case study 29
3.3 Selection of cases **31**
3.3.1 Case requirements 31
3.3.2 Collection of cases 32
3.4 Data Collection **34**
3.4.1 Interview 34
3.4.2 Sampling Informants 35
3.4.3 Interview guide 35
3.4.4 The interview process 37
3.5 Data analysis process **38**
3.5.1 Transcribing 38
3.5.1 Coding 38
3.6 Validity and reliability **39**
4. Findings **41**
4.1 Research **41**
4.2 Oil **43**
4.3 IT Grocery **46**
4.4 Software Care **49**
4.5 Cross-case analysis **51**

5. Discussion **52**
5. 1 Knowledge about non-functional requirements **52**
5.2 Handling of non-functional requirement **53**
5.3 Customer involvement **54**
5.4 Our own thoughts on non-functional requirements in software development **55**
5.5. Which requirements should be emphasized? **56**
5.6 Limitation **56**
5.7 Further research **57**
5.8 Implications **57**
6. Conclusion **59**
References **61**
Appendix A: Ethical approval **64**
Appendix B: Interview guide **65**
Appendix C: NSD – Norsk senter for forskningsdata **67**

List of figures

Figure 1: Model of literature review collection 11
Figure 2: Model of agile approach and practices 14
Figure 3: Overview of non-functional requirements (Sommerville 2014) 17
Figure 4: Definitions of non-functional requirements 20
Figure 5: Non-functional variables for successful agile software project 21
Figure 6: Guidelines for documenting non-functional requirements according to 23
Figure 7: The scrum based AFFINE method (Bourimi et al. (2010) 25
Figure 8: W8 user story card (Farid 2011, 46) 26
Figure 9: Qualitative research process (Yin 2009, 66) 28
Figure 10: Designs for Case studies (Yin 2003, 40) 30
Figure 11: Overview of informants 33
Figure 12: Cross-case analysis 51

1. Introduction

This chapter will initially present the background to the study, before further explaining the purpose. As a result of the background and the purpose will the research question the study attempts to answer be presented. At the end, the structure of the rest of the report is described.

1.1 Background

High costs and failure rates of software projects using a plan-driven methodology, engaged researchers and practitioners interest in agile software development methods. Almost two decades ago, in 2001, the agile manifesto was published and brought considerable changes to the software engineering field (Dingsøyr et al. 2012). After the introduction of the agile manifesto, a large number of different approaches appeared in the landscape: Extreme programming (Beck 1999), Scrum (Schwaber 1997), lean software development and feature-driven development (Palmer and Felsing 2001; Dingsøyr et al. 2012). All of these methods share the core principles in the Agile manifesto, but their features and application domains differ (van der Heijden et al. 2018).

Many companies transformed their development processes from plan-driven (traditional) methodology to agile development methods (Papadopoulos 2015). The agile methods showed: higher satisfaction, feeling of effectiveness, increased quality and transparency, and increased autonomy and happiness, and earlier detection of defects (Laanti et al. 2011). Even though many of the challenges in traditional methods were resolved using an agile approach, they also introduced several limitations (Ramesh et al. 2010). One of the critical steps in a successful software development project is the requirement engineering. Requirement engineering's purpose is to identify user requirements, analyze, document and validate for the proposed system (Inayat et al. 2015). From a traditional requirement engineering perspective, the process consists of four sequential activities: requirement elicitation, requirement analysis and negotiation, requirement documentation, and requirement validation. Although requirement engineering is seen as one of the critical steps in the software development, the term "requirement engineering" is avoided in the agile community (Ramesh et al. 2010). Often it is considered to imply substantial documentation with significant overhead.

Also, the software development environment is still unfamiliar with the requirement engineering practices in agile methods. Many of the agile methods are relying heavily on

feedback from the customer, and some advocate moving into coding without even having a centralized requirement analysis and design phase (Ramesh et al. 2010). Even though the shift from traditional to agile software development methods changed the requirement engineering field; it is evident that agile software development support many of the traditional requirement engineering activities. Elicitation, documenting, validation, analysis and negotiation are all present in the agile software development, only with different practices compared to traditional requirement engineering.

Even though many of the challenges in traditional software development were solved by moving over to agile software development, some others were raised in the context of agile requirement engineering practices. Neglecting non-functional requirements in the software development project is seen as one of the significant risks in the agile requirement engineering field (Behutiye et al. 2017; Chung et al. 2012; Ramesh et al. 2010). Non-functional requirements have been a significant concern in the agile methods and can be the reason for massive rework and lapse. Even though there are proposed numerous approaches and frameworks to deal with this challenge, there is no formal agreement in the literature about common or best practice (Mishra and Mishra 2011). Therefore, the suggestion for further research is needed to look into current practices on how agile software development projects handle non-functional requirements.

1.2 Purpose

This master thesis aims to identify how agile teams take into account non-functional requirements in their software development. Non-functional requirements are perceived as an essential felt that deserves more attention. We are also interested to see which consequences projects have by not raising awareness and understanding of not implementing routines for non-functional requirements in the development phase. By challenges, we refer to factors that negatively impact the organization, the end-users experience and the software quality. The purpose of the study is to gain in-depth knowledge, and hopefully, to contribute new insights to the subject non-functional requirements in agile software development. Taking a deep dive into the agile world and non-functional requirements, based on 1.1 Background, we will try to answer the research question that is as follows:

How do companies handle non-functional requirements in agile projects and why should non-functional requirements be more emphasized in agile software development method?

By interviewing a selection of team members from different agile software development projects, we want to investigate and understand what team members experience and awareness on the subject, and how they cope with non-functional requirements in their projects. This is the data of the study, which we will analyze and further present. We will then discuss key findings against existing literature to form a picture of how the situation is today.

1.3 Structure

This master thesis is divided into six chapters. Chapter 1, the introduction, presents the background and purpose of the research project, followed by a presentation of the research question. Chapter 2 proved an overview of established theory and literature related to agile teams, non-functional requirements and models on how to deal with non-functional requirements. Further chapter 3 describes the methodological choices of the study. The results are presented in chapter 4, followed by a discussion of the study's central findings set against the theoretical basis in chapter 5. Chapter 5 also addresses the study's contribution, limitations to the study, suggestions for further research and practical implications. The conclusion is presented in chapter 6, where the study's research question is attempted to be answered. Reference list and appendices follows in chapter 7 and 8.

2. Literature Review

The development of systems based on pre-specified requirements pressures severe challenges in a rapidly changing environment, stakeholder's preference, software technology and time-to-market (Ramesh, Cao, and Baskerville 2010). As addressed earlier, agile methods seek to support shorter development cycles in order to respond to a fast-moving competitive marketplace. The requirements tend to evolve very quickly and become obsolete; sometimes even before project completion, software development organizations are thus forced to deal with this. This has challenged requirements engineering, and new methods have had to be developed to support requirements engineering activities (Ramesh, Cao, and Baskerville 2010). Today, in an agile world, requirements engineering, and agile approaches are considered incompatible. There is a tradeoff between requirements engineering relying on documenting to share knowledge, while agile approaches depend on face-to-face collaboration between customers and developers. (Paetsch, Eberlein, and Maurer 2003).

This chapter starts with outlining the previous literature and theories concerning traditional requirement engineering and the adopted practices in the agile field. Furthermore, we will continue with the challenges in the agile requirement engineering field and will continue to go deeper into one of the challenges; the neglect of non-functional requirements in agile software development. We will outline the theory on non-functional requirements and existing literature investigating the challenges. Finally, we will present different theories and models on dealing with the challenges with non-functional requirements.

The process of the collection of literature review

To collect relevant info on non-functional requirements, we have used different search words and combinations of those at different databases. The different databases we used are Google Scholar, Oria and Google Search. Relevant literature was filtered out by reading the title and the abstract before being passed on to the next step, where we read more carefully through the articles, reports or books.

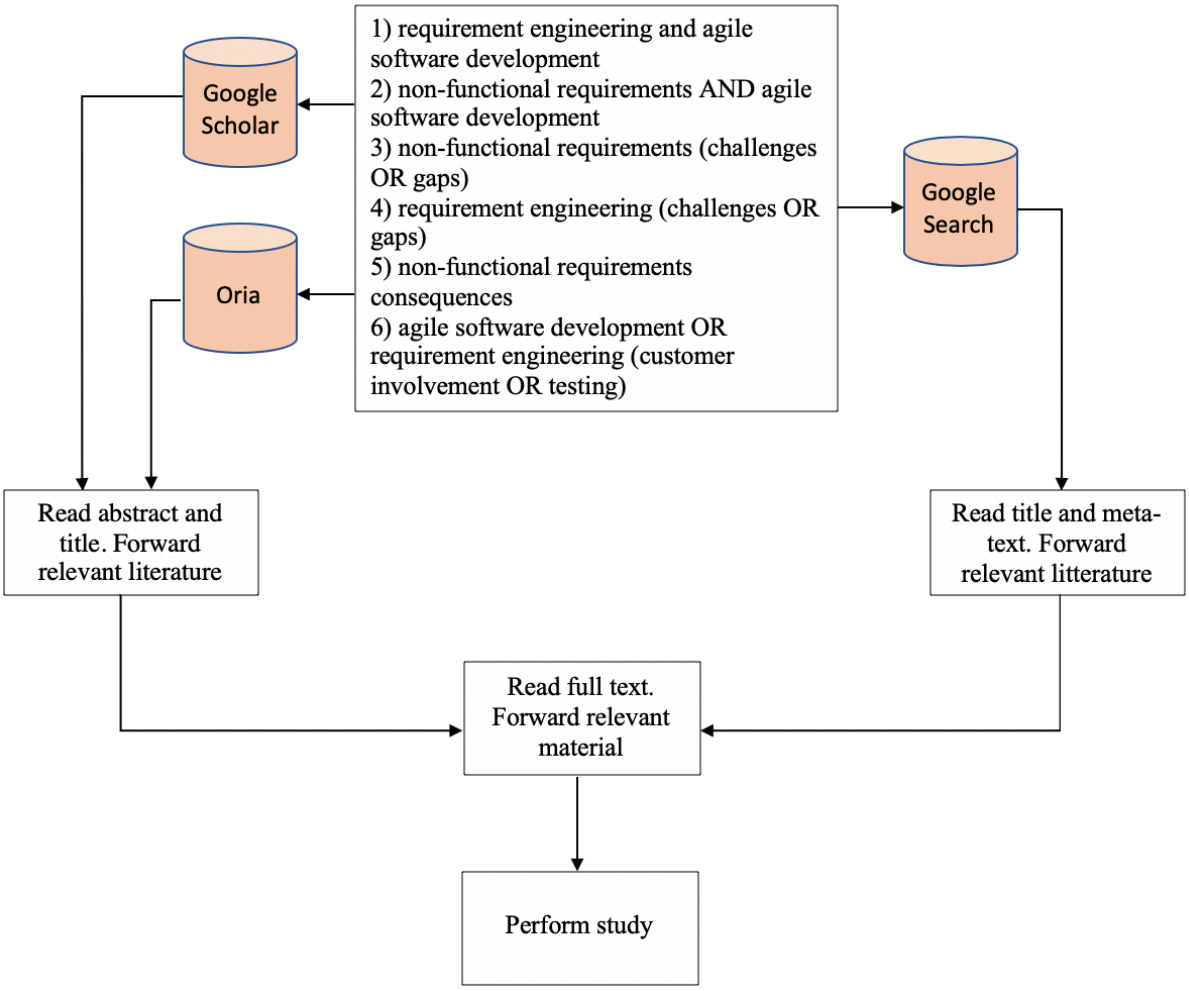

Figure 1: Model of literature review collection

2.1 Requirements engineering

Since the early days of software development, requirements engineering has been perceived as one of the key steps in a successful software development effort (Sillitti and Succi 2005). The description of what the system should do, the service that it provides, and the constraints of its operation are referred to as requirements (Sommerville 2011). The purpose of requirements is to reflect the needs of the customers for a system that serves a specific purpose, such as finding information. Finding out, analyzing, documenting and checking these constraints and services is called requirement engineering. Zave (1997) provides the most precise definition of requirement engineering:

"*Requirement engineering is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behavior, and to their evolution over time and across software families*" (Zave 1997, 315)

From a traditional requirement engineering point of view, the process consist of eliciting individual stakeholder requirements and needs, and developing them into detailed, agreed requirement document. They should be specified in such a way that they can serve as the basis for all other system development activities (Sillitti and Succi 2005) The main goal is to specify management requirements to minimize the risk of delivering a system that does not meet the stakeholders' desires and needs (Pohl 2010).

The traditional requirement engineering process generally comprises four main activities: Elicitation, documenting, analysis and negotiation, and management (Paetsch, Eberlein, and Maurer 2003). During elicitation, the goal is to discover requirements and to identify system boundaries by consulting stakeholders. The initial information regarding the requirements are gathered and the context (Pohl 2010). The most prevelant techniques are interviews, use case, observation and social analysis, focus groups and prototyping. In the next phase documentation, the elicited requirements are described adequately. Validation and negotiation include checking requirements for necessity, completeness and feasibility. The goal is to guarantee that the predefined quality criteria are met. Therefore, documented requirements must be validated and negotiated early on. The most used techniques for the analysis are joint application development sessions, prioritization and modeling. The requirement document is the baseline for the evaluation of subsequent products and processes. The last activity is requirement management, which has the purpose capturing, store, disseminate and manage the requirements. Thus, the requirements can maintain consistency after changes, to be used by different roles and to ensure their implementation.

2.2 Agile requirement practices

According to Orr (2004) , some agile developers advocated that requirements engineering is not necessary for an agile environment. Some agile approaches attempt to give users what they say they want as quickly as possible rather than understanding the user´s business and information needs. There seems to be an impression that you can code yourself into success in the new agile world, and indeed the code is a requirement. Nevertheless, people need to understand what to build, thus, proper requirements are necessary. In addition, without proper

requirements, the project risks increase dramatically. This is why it is essential to do requirements engineering to ensure successful system development (Orr 2004). According to Ramesh et al. (2010), the literature that addresses explicitly requirements engineering in agile development is still nascent. The term "requirements engineering" is avoided in the agile community because it implies substantial documentation with significant overhead.

Ramesh et al. (2010) studies show that agile requirement practices do not follow traditional requirements engineering principles or guidelines defined in the requirements engineering literature. On the contrary, agile development is often done in an environment where it is impossible or inappropriate to develop unambiguous and complete requirement specifications. Requirements engineering is a traditional software engineering process, but agile methods have adopted many of the basic ideas to a new environment. According to Paetsch et al. (2003), elicitation, documenting, analysis, negotiation, and management are part of in all agile processes but implemented by different practices. Figure 2 provides an overview of some agile requirements practices:

Traditional requirements engineering processes	Agile approach and practices	Sources
Requirement elicitation	Iterative requirements engineering and face-to-face communication. For agile development, requirements are not pre-defined, they emerge during the development cycle. In agile requirement engineering the customer can freely communicate with the team throughout the software development life cycle. This means that agile methods accommodate changing requirements, even late in the development cycle, and require that requirements evolve over time.	Batool et al. (2013); Ramesh et al. (2010)
Requirement documentation	No formal documentation of requirements. Informally as lists of features or stories. On the contrary with traditional requirement engineering the formal document is replaced with intensive communication with the customer and the development team.	Ramesh et al. (2010)
Requirement validation	Agile approaches usually require frequent review meetings for requirement validation. At the end of every development cycle, a meeting that involves developers, customer, quality assurance, management and other stakeholder is scheduled. The purpose of these review meetings is to increase customer trust, and confidence in the team, to highlight problems early, to show that the project is on target, and to check if the requirements reflect user's needs.	Paetsch et al. (2003)
Requirement analysis and negotiation	Facilitated by iterative requirement engineering. Face-to-face communication, extreme prioritizing and constant planning. The purpose of the requirement analysis and negotiation is to help redefine, prioritize and change requirements.	Sillitti and Succi (2005)

Figure 2: Model of agile approach and practices

2.2.1 Challenges with agile requirement engineering

The literature indicates that many of the challenges concerned in traditional requirements engineering were resolved with agile requirement engineering practices (Inayat et al. 2015). Some of these are communication issues, over scoping, requirements validation, requirements documentation and rare involvement. These challenges are solved by practices such as frequent face-to-face meetings, cross-functional teams, requirements prioritization, user stories and requirements prioritization by the customers. Although that agile requirement engineering practices resolved many of the challenges experienced in traditional requirement engineering, they also introduced several limitations (Inayat et al. 2015; Ramesh et al. 2010). See figure 2 for an overview of some of the challenges.

2.3 Requirements

One of the main challenges with agile requirements engineering is the neglect of non-functional requirements. Here we will present the literature covering requirements and non-functional requirements. Farid and Mitropoulos (2012) argue that agile methods have not adequately identified modeled and linked non-functional requirements and their potential solution (operationalization) with functional requirements during early requirements analysis phases. In this section, we will present the literature concerning Agile.

2.3.1 Functional and non-functional requirements

According to Glinz (2005), there is a broad consensus of the definition of “functional requirements”. According to existing literature, the definition follows two threads that correspond to a large extent. The first thread emphasizes the function. Sommerville (2011) defines functional requirements as “statements of services the system should provide, how the system should react to particular inputs, and the system should behave in a particular situation. In some cases, the functional requirements may also explicitly state what the system should not do”(Sommerville 2011). The second thread on functional requirements emphasizes the behavior aspect of a system (Glinz 2007). According to Glinz (2007), there is only one semantic difference that may arise between the different definitions; the timing requirement. They may be viewed as behavioral, but most publications consider them to be performance requirements, which is classified as non-functional requirements. On the contrary, there is no clear definition of what a non-function requirement really is.

Many organizations operate with non-functional or quality requirements. Such requirements are often defined in general terms and are not handled in the same way as functional tasks. The result is that the requirements rarely take the form of specific tasks that can be developed, verified or prioritized in line with other functionalities. Requirements that are neither concrete nor measurable can neither be met nor verified and are thus often overlooked (Glinz 2007).
A challenge with theory about non-functional requirements is that there are many different definitions of what to include in non-functional requirements. There is a large variety of types of non-functional requirements, something that can create difficulties. In addition, different authors use different names for what is the same term.

Where Davis (1993) talks about 'non-behavioral requirements' and Skagestein (2005) about 'quality requirements', one can also use the term non-functional requirements. However, one agrees that a common denominator for these types of requirements is that they help to define the functions and give the positive or negative limits to which operating personnel should keep track. On the other hand, Cleland-Huang et al. (2007) argue that non-functional requirements describe important constraints upon the development and behavior of a software system. Disagreement within the theory shows that there are infinitely many ways one can classify and group non-functional requirements. We find this definition of non-functional requirements most suited for our thesis:

"Describe the nonbehavioral aspects of a system, capturing the properties and constraints under which a system must operate." – (Glinz 2007, pp. 22)

2.3.2 Classifying of non-functional requirements

Due to the ambiguous and imprecise definitions, for the purpose of this thesis, we have chosen to create a model. This will be achieved on the basis of non-functional requirements derived from both system development and operation. The sections below will first prevent various other models of non-functional requirements and theory about the operation where one also looks at the non-functional requirements.

Chung et al. (2012) present a list of violations that are included in non-functional requirements. The book presents a way to represent and analyze non-functional requirements using a framework that will make it easier for developers to relate to non-functional requirements. The

list of non-functional requirements has been compiled to give the reader a sense of the breadth of areas that fall under non-functional requirements. This list is not complete, but nevertheless provides a good starting point, and covers important non-functional requirements such as security, user-functionality and accessibility.
Sommerville (2004) provides a model of system properties. This divides the non-functional requirements into three main groups; product requirements, organizational requirements and external requirements, which again are turned into smaller subgroups.

Product requirements are the requirements that specify the way the product is behaving. An example of this is performance requirements, which means how quickly the systems manage tasks, a requirement for how many errors that can be accepted and user-friendliness requirements. Organizational requirements come from the policies and procedures of the client and developer organization. An example of this is the standards that will be used, limitations on the implementation of the system in relation to which programming language and system design, and requirements for when the system and documentation will be delivered. External requirements include all the requirements that come from factors outside the system and the development process. These are requirements for the system to cooperate with systems in other organizations, statutory requirements that must be followed or ethical requirements. Figure 3 represents an overview of the different specifications of non-functional requirements:

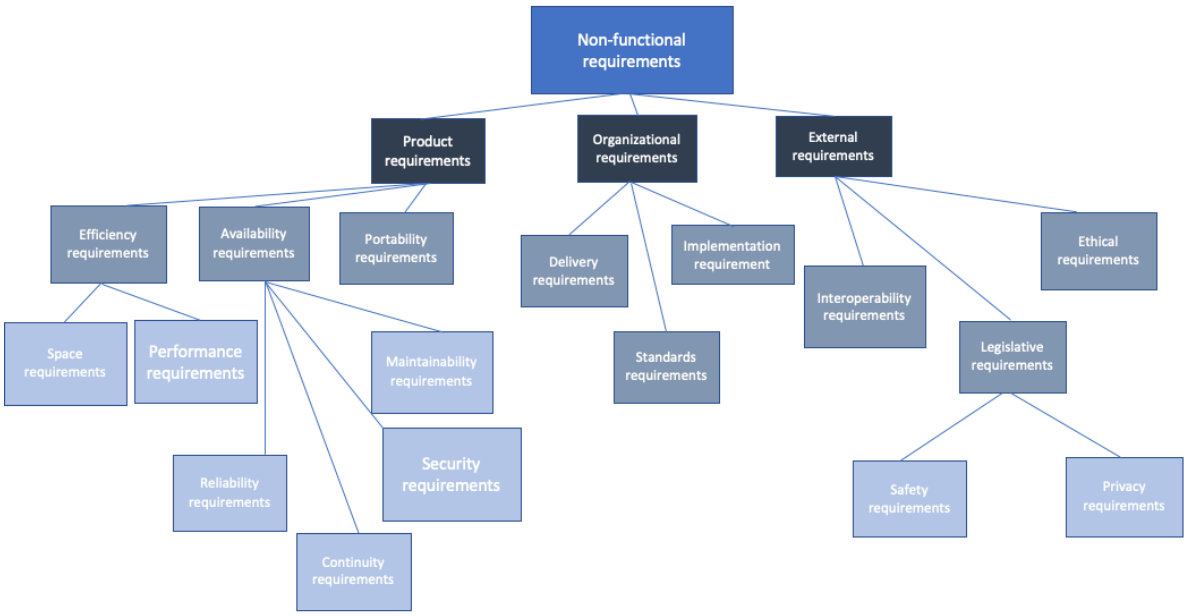

Figure 3: Overview of non-functional requirements (Sommerville 2014)

Definitions of the concepts in the model

To gain a better understanding of the model, the various concepts in the model are presented in figure 4 below. All of these concepts fall under the term 'non-functional requirements' and help lay down guidelines on how a system should perform its functions. These guides can give positive or negative limits that the operating team must keep track of. Non-functional requirements have an impact on the system as a whole and do not affect the specific functions that the system will perform. In this thesis, the main focus will be on product requirements, which are defined as:

Name	Description	Sources
Product requirements	Product requirements are requirements that specify the way the product behaves. It thus encompasses all the various non-functional requirements that apply to the system as a product to be used in an enterprise or organization.	(Sommerville 2011)
Availability requirements	Availability requirements are requirements for the system's accessibility. This is an IT service's ability to perform its required functions within an agreed time or time period. Accessibility is often described as a percentage, that is the percentage of time the is actually available to users within the agreed service time. Availability is a collective term that encompasses many different non-functional requirements.	(Evans and Macfarlane 2001)
Maintainability requirements	Maintainability requirements are the requirements for maintenance and maintenance of the system. This is a component or IT service's ability to maintain to a state where it can perform its required functions. It also includes maintenance requirements, and the procedures that one has to carry out.	(Evans and Macfarlane 2001)
Security requirements	Security requirements are security requirements, and more specifically the data that the information systems contain, processes and delivers. This is the process of ensuring that services are used properly and by the right people. You should be able to trust that the data you enter, or retrieve from the systems are available, has integrity and is confidential.	(Evans and Macfarlane 2001)
Efficiency requirements	Efficiency requirements are requirements for the system's efficiency. This is the goal of deciding whether a system or component performs its assigned functions with the least possible use of various resources.	(Wiegers and Beatty 2013)
Performance requirements	Performance requirements include requirements for how the system will perform its tasks. This involves the functions of the system, and whether it is running fast enough and whether the customer / user can perform the operations needed.	(Wiegers and Beatty 2013)

Space requirements	Space requirements address requirements for the system's capacity. It is critical to have enough storage space, and not at least that the components are capable of handling the amount of data that goes through the system.	(Wiegers and Beatty 2013)
Privacy requirements	Privacy requirements are privacy laws, both for users, developers and others involved in the system. It is critical that no private data in the system is released to others who should not have access to it.	(Wiegers and Beatty 2013)

Figure 4: Definitions of non-functional requirements

2.3.3 Challenges related to non-functional requirements

To ensure a project's success, there are several variables related to non-functional requirements to consider. The next section will present these shortly. The variables will construct a basis for the hypothesis for our research.

Knauss et al. (2017) highlight the importance of awareness and knowledge of non-functional requirements for teams. It is not always possible to correct or implement things while projects are in progress, especially when it comes to safety and security. Ullah et al. (2011)bring up the point that in order for software teams to meet the expectations of delivering projects with a focus on usability, safety and security it is critical that non-functional requirements are elicited as much as functional requirements. This is crucial for the project to be a success. Having the previous two points in place, it also essential to make it a priority from the start, as modifications can be difficult or impossible to implement late in a project (Dabbagh et al. 2016). Understanding how, why and what the customer is going to use the system or software for is essential. It is thus important to have a good and close dialogue with the customer, although Poort et al. (2012) see that some business conditions may have their limitations when it comes to what some parties want. Finally, it is important to define non-functional requirements. One must define how the system should operate and behave according to several scenarios. This is to be able to take into account "what-if scenarios", and this will hopefully avoid pitfalls and significant negative consequences for the project in the future (Chung and do Prado Leite 2009).

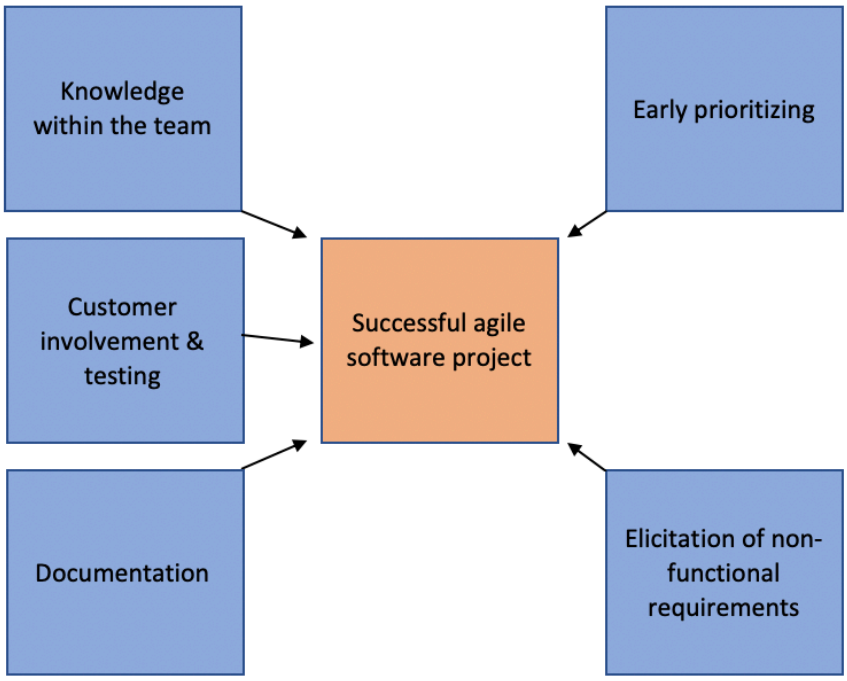

Figure 5: Non-functional variables for successful agile software project

2.4 Solution proposals for non-functional requirements

As outlined in the previous chapter, many factors that are the reason for the challenges with non-functional requirements in agile software development. As stated, before agile software development methodologies were not sufficiently identified, modeled and liked non-functional requirements with functional requirements early requirements phases. According to Farid and Mitropoulos (2012), researchers have agreed that non-functional requirements have been generally ignored, ill-defined, dealt with as an afterthought process and not treated as first-class artifacts in requirement engineering, but especially ignored in agile methodologies. Researchers have proposed different models and frameworks for integrating, planning, and managing non-functional requirements in agile software development (Behutiye et al. 2017). These various frameworks that is proposed is appeared to be too complicated and hard to implement in the development of the agile community (Umar and Khan 2011). In the next section, we will outline the different frameworks.

2.4.1 Non-functional testing

Camacho et al. (2016) argue that non-functional testing is a challenge in agile software development. Due to a business's lack of clarity of their needs in most parts of projects and its cross-functional aspects. In their case study, they found that factors such as cost, time,

awareness, culture and experience were influencing the priority of testing non-functional requirements. IT professionals consider experience as one of the most crucial factors (Camacho et al. 2016). The most experienced team members provide more attention to aspects conserving non-functional requirements and defend the testing. Younger and less experienced team members mainly focus on what needs to be delivered in functionality and give less attention to the overall system or cross-functional features. Further, non-functionality testing is a complicated task that requires specific skills and expertise. The reason is that experience influences the capacity of better identification of non-functional testing needs. A senior member also influences the team culture concerning to non-functional testing. The awareness and importance of non-functional testing were highly cited by senior members and also influenced the team perception.

The analysis over non-functional aspects can solve priority, culture and awareness of non-functional requirements during the project meetings. Project inception is the meeting where the analysis of the business characteristics should be reviewed. This analysis helps identify that non-functional testing needs support to define priority. Also, during sprint planning, the same exercise is done. The whole team participates when the senior team member reinforces the need for a more in-depth review under non-functional aspects related to that user story. This helps the dissemination of the awareness and culture of the non-functional.

Furthermore, having a "concerned" mindset is a good approach to work with. Having the risk mindset and explaining how impacted business can be and the non-functional requirements are better understood, other factors can be minimized, such as cost and time. The importance that different roles in the team participate in these meetings is essential. Also, they reported that these conversations concerning non-functional requirements should continue throughout the development process, in for example stand-up-meetings.

2.4.2 Documentation proposal

As previously mentioned, non-functional requirements are characterized as hard to define and are usually not documented. This is often because working software is prioritized over comprehensive documentation and existing requirement engineering practices fail shortly regarding the documentation of non-functional requirements. User stories in agile software development have limitations regarding specification and documentation of non-functional requirements. Traceability is difficult when non-functional are not documented and a consequence is the neglect of non-functional requirements during the development. Weak user acceptance may also be an aftereffect.

Behutiye et al. (2017) propose guidelines for documenting non-functional requirements in agile software development. They categorize non-functional requirements into three types of scope; system-wide for those that apply to the entire system, group-wide for those that apply to a set of user stories and local for those that apply to a single user story. Besides, they argue that the level of detail in which a non-functional requirement specified may vary. Therefore, they distinguish between general non-functional requirements specified as a high level of abstraction and detailed non-functional requirements specified as concrete features or toes to a concrete solution.

Due to the variability of non-functional requirements both in scope and detail, there is not a single representation artifact that is adequate to cope with all of them. Behutiye et al. (2017), therefore, argue that a proposal for documentation non-functional in agile software development should provide different artifacts for representing them and a set of guidelines to select the adequate representation depending on the features of each specific requirement. Proposed Guidelines in figure 6 show the different artefacts to represent non-functional requirements.

Scope	**Detail**	**Representation artefact**	**Observation**
Local	Generic	User story (NFR user story)	With a link to the functional user story to which it applies
	Detailed	Acceptance criteria	Appearing in the functional user story to which it applies
Group wide	Generic	Epic	The description of the epic must clarify to which group of functionalities it applies (e.g. "critical functions of the system")
	Detailed	(1) User story or (2) Acceptance criteria	(1) The description of the user story must clarify to which group of functionalities it applies or include links to the user stories it applies (2) Appearing in the functional user stories to which it applies
System wide	Generic	Epic	The description of the epic must clarify it is system-wide (e.g. by referring to "the system")
	Detailed	User story	The description of the epic must clarify to which group of functionalities it applies (e.g. "critical functions of the system")

Figure 6: Guidelines for documenting non-functional requirements according to their scope and detail (Behutiye et al. (2017)

Figure 6 illustrates that scope and detail of non-functional requirements should be able to select the adequate representation artifact for a non-functional requirement. For example, for local non-functional requirements they propose that they should be documented as user stories that should be linked to the functional user story to which they apply. For system-wide non-functional requirements they propose to use epics. For group-wide non-functional requirements, they propose a similar approach as system-wide. On the contrary, if non-functional requirements are detailed and the group of functionalities affected by the non-functional requirements is small. They propose to document them as acceptance criteria of the user stories to which they apply.

2.4.3 AFFINE framework

Bourimi et al. (2010) state when applying agile methods for building sophisticated sociotechnical systems, non-functional requirements are often considered too late in the development process. One consequence is the tension between developers and users may arise and there is a lack of guidance and support for efficiently fulfilling non-functional requirements in terms of software architecture in general. Bourimi et al. (2010) proposed the AFFINE framework that aims to simultaneously address three needs when developing sociotechnical systems by following agile methodologies. The needs were based on relevant human-computer interaction (HCI)- and computer-supported cooperative work (CSCW) literature gathering experience and a long-running project CURE. The three needs and main targets consist of simultaneously addressing previously cited shortfalls by;

> N1: Conceptually enforcing the consideration of all relevant non-functional requirements and possible trade-offs early in the development process.
>
> N2: Explicitly balancing end-users´ with developers´ needs when following agile development methods.
>
> N3: The development method must be supported at the architectural and construction level to assure meeting N1 and N2 at minimal cost. A kind of reference architecture providing support for non-functional requirements is needed.

To reduce the complexity of the involvement of the method in various phases of the following development process, they propose scrum as an integral component. They argue that scrum can

be seen as a process for empirical control of software development, which helps in handling changing requirements more efficiently. This is managed by considering human factors in the development process of both customers- and project stakeholders in general.

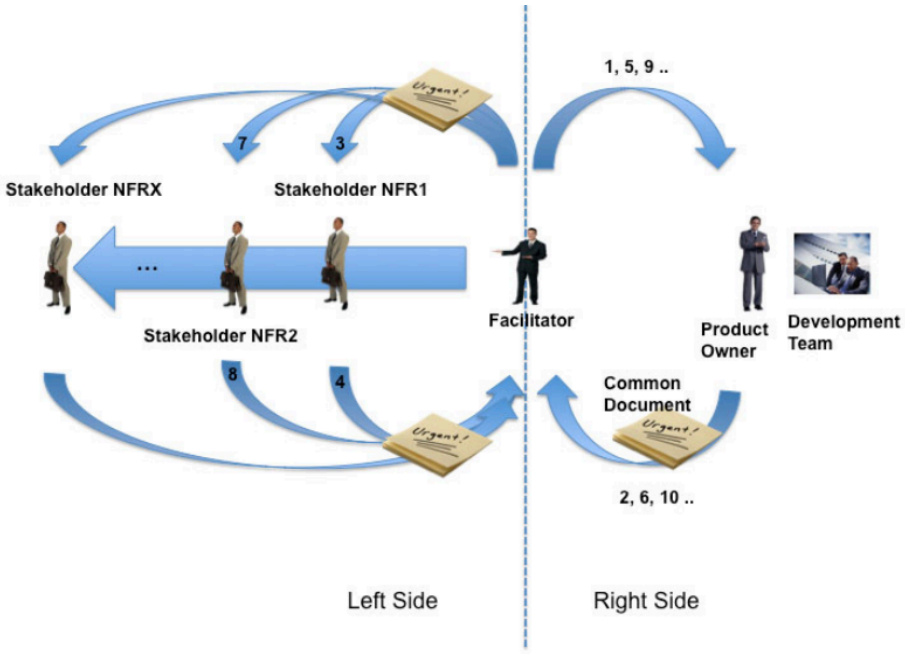

Figure 7: The scrum based AFFINE method (Bourimi et al. (2010)

Figure 7 illustrates that the Scrum development procedure is broadly represented on the right side. The loop starts and ends with a facilitator (in this case the scrum master) and in a regular scrum, a sprint backlog might be updated in such a loop. On the left side, non-functional stakeholders (experts) are introduced to the process. They should be concerned with the fulfillment and consideration of a respective non-functional. The first stakeholder gets the document from the facilitator that has to moderate the considerations of the common document. The first stakeholder has to update the document (e.g. adding warning, requirements or changing them). After the document is updated, it returns to the facilitator and loops again over to the left side (Bourimi et al. 2010).

Bourimi et al. (2010) suggest six informal steps for N1 and N2. This includes the involvement of all stakeholders of the projects and introducing the role of the facilitator. There have to be goals or use cases identification of the intended processes. Non-functional requirements must be aligned to prioritize according to the project goals or use cases. One responsible for each goal or use case have to be chosen. The single document has to circulate and contain the set of

goals or use cases and their specification and modeling. When reaching the goals i.e., implementing the use case and testing them, the circulation ends.

2.4.4 Modeling non-functional requirements

Farid (2012) proposed a method named non-functional requirement modeling for the agile process (NORMAP). The purpose of the methodology is to link non-functional requirements with functional requirements. The framework is specifically tailored for agile software development processes, such as SCRUM. The framework can be used manually (NORMANUEL) or automatic (NORMATIC). NORMAP proposes three objectives.

The first is the W story card model. To identify, link and model non-functional requirements with functional requirements in an agile process, the W8 user story card and agile requirements (AR) taxonomy are proposed. The W8 User Card is an enhancement for the agile index card technique used to gather requirements in Scrum. The story card is a physical 3x5 index card used to capture 2-5 sentences of high-level user requirements from the stakeholder point of view (figure 2). The model includes eight "W", and each "W" represents a different word that captures the most essential and necessary information of an agile user story (Farid 2011).

ID	*Element*	*Description*
1	Who	Captures the actor (e.g., "As a system administrator…")
2	What	Captures the desired core functionality that the user wants (e.g., "I want to…")
3	Why	Captures the reason or business justification for the functionality (e.g., "so that I can…")
4	Without ignoring	Captures linguistically significant terms that might capture essential system qualities (e.g., "without ignoring security, scalability…") (required NFRs)
5	While it's nice to have	Captures optional quality attributes that are of lesser importance (e.g., "while it's nice to have ease of use and trainability...")
6	Within	Captures the time frame within which the desired functionality is needed (e.g., "within 12 weeks…")—just an initial time estimate according to business and market conditions
7	With a priority of	Captures the requirement's initial priority from a business value standpoint (priority of 1 means top priority and 99 means last). *Note: this research introduced two additional priority schemes, Riskiest-Requirements-First and Riskiest-Requirements-Last*
8	Which may impact	Captures a list of requirements dependencies impacted by this user story (e.g., "which may impact requirements # 101, 105, 107…")

Figure 8: W8 user story card (Farid 2011, 46)

The second objective in NORMAP is the agile requirement classification. There functional requirements in agile use case (AUC) and non-functional requirements in agile choose case

(ACC) are classified. The third objective I s “PointCut” operators. Keywords such as “before”, “after, “override” and “wrap” are used for combining functional requirements and non-functional requirements. To link ACC to AUC, these points are used.

3. Method

In this chapter, we will explain and describe the selected research method we applied in this study. We will start with advocating and elaborate our research selection regarding research approach, research design, method for data collection, selection of cases and informants. We will conclude with an elaboration of the validity and reliability of this study.

3.1 Research Method

To answer the research question, the first choices researches have to make is whether to use a quantitative or qualitative research approach. Depending on the nature of the research and research question, both approaches have their perks and advantage. Quantitative research can be outlined as a distinctive research strategy. Resulting in numeric data to show the relationship between theory and research to be deductive and linking towards a natural science approach and as having an objectivist conception of social reality (Bryman 2016). If the research question contains words such as “what” and “who” it is appropriate to use a quantitative research approach, such as survey (Meyer 2001). On the other hand, a qualitative research is the preferred strategy for answering research questions which includes phrases such as “how” and “why”. Furthermore, qualitative research emphasizes inductive approach to the relationship between theory and research and include all non-numeric data that is generated by case studies, action research and ethnography.

If we look at our research question: “*How do companies handle non-functional requirements in agile projects and why should non-functional requirements be more emphasized in agile software development method?”.* It is clear our research question contains both the word “how” and “why”, and it is therefore appropriate for qualitative research. The appropriateness of qualitative research methods is when the study is new and there less prior theory on the subject. On the hand quantitative research method is more applicable when there is established theoretical foundation and the researcher wants to test the established theories (Bryman 2016). Extensive research has been conducted on the subject of agile project management and it is highly theorized. Likewise, requirement engineering in traditional software development. On

the contrary, research shows the need for more empirical studies that work on agile requirement engineering and its real-world impact and applications (Inayat et al. 2015). For instance, different agile methodologies, non-functional requirements, scaling or distance of project members (Inayat et al. 2015; Schön et al. 2017).

In our theoretical review, we established that the literature on agile requirement engineering is still nascent and avoided in the agile community (Ramesh et al. 2010). Non-functional requirements are ill-defined, overlooked and not properly handled in agile projects. Non-functional requirement in agile is very complex phenomenon and numerous different framework and method is proposed to deal with the different challenges conserving non-functional requirement. In addition, the term non-functional requirement is not yet clear in the theory. Therefore, we believe choosing a more flexible way in collection the data and choosing a qualitative research method will give us this flexibility.

Figure 8 represent the process of qualitative research project Yin (2009). We will in this chapter go through all the steps from plan, design, prepare, collect, analyse and share. The final step share is this written report of the master thesis. This is an iterative and a repetitive process not a linear qualitive research project, as the arrow indicate in the figure. It is important to note that share is this master thesis in written report.

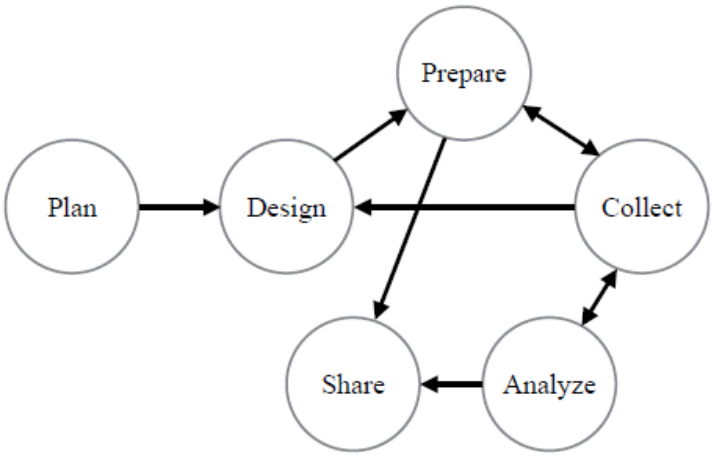

Figure 9: Qualitative research process (Yin 2009, 66*)*

3.2 Research design

In order to obtain knowledge of individuals, group, organizational, social, political and related phenomena, case study have been used in many situations as research strategy (Yin 2003). Within many different fields such as political science, social work and psychology, case study has been a common research strategy. Over the years in-depth case studies which focus on human actions and interpretation surrounding the use of computer-based information system and development have increased. This have led to some IS researcher to adopt empirical approaches which focus on human interpretations and meaning as the importance of social issues related to computer-based information systems has been recognized (Walsham 1995).

3.2.1 Case study

Yin argue that case study particularly suited when "a how and why question is being asked about contemporary set of events over which the investigator has little or no control" (Yin 2009, 13). Moreover, case study is more suited when the research question requires an in-depth description of a phenomenon and the phenomena is not supported by a strong theorical base(Benbasat et al. 1987; Yin 2003).The phenomenon non-functional requirement is highly complex and could be handled with multiple different theories, framework and methods. We therefore argue that case study is the most suited research strategy for this research project. In addition, the analysis of non-functional requirement depends largely on human actions and can only be understood through examining it in its settings. Therefore, a case study is appropriate to investigate the phenomenon on non-functional requirements in its real-life context and to considerate human interpretation and meanings.

In figure 9 below, Yin (2003) differentiate between four types of design for case studies. They are divided in four different designs. Single and multiple case design is a primary distinction in designing case studies. It is essential to determine whether to use single or multiple cases to address the research question prior to continuing with data collection(Yin 2003).

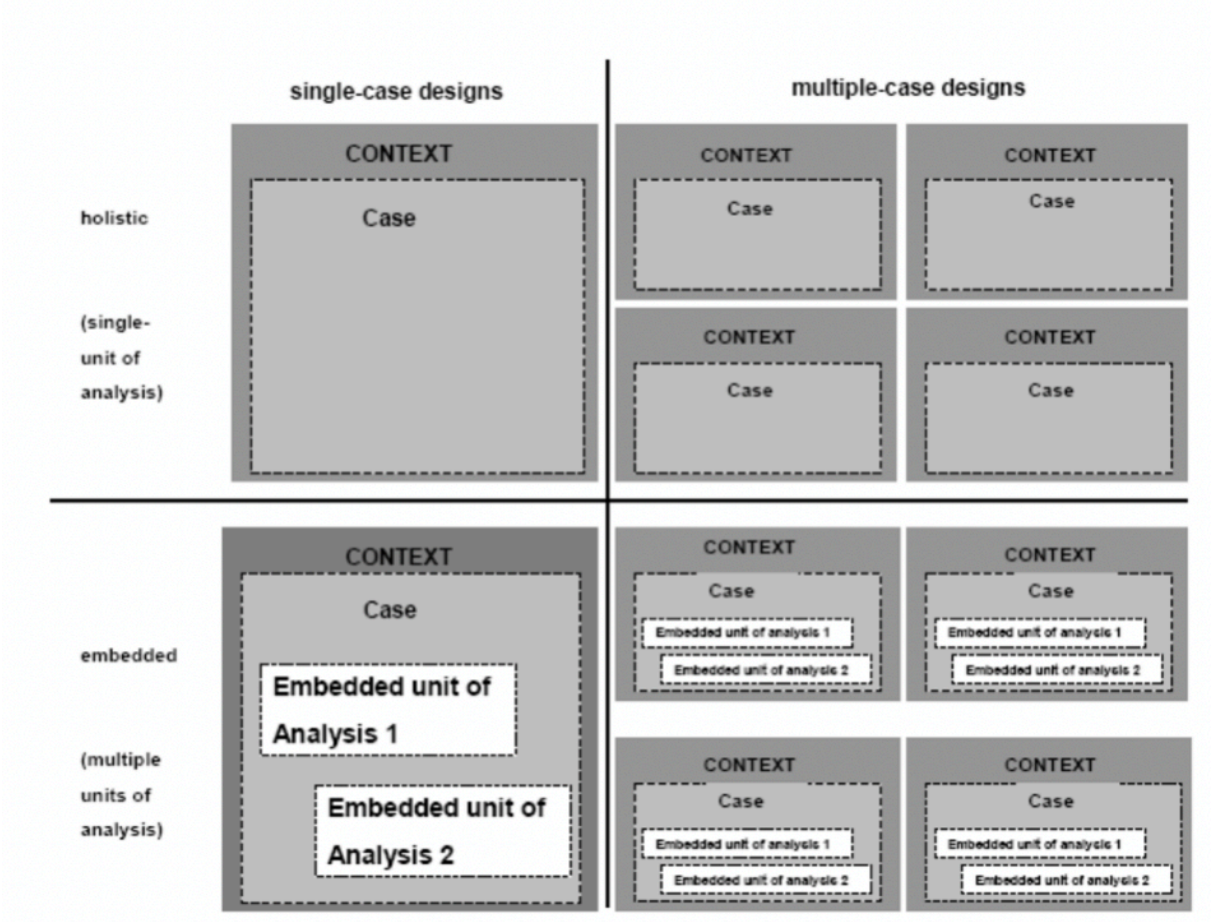

Figure 10: Designs for Case studies (Yin 2003, 40)

For this research we will use a multiple-case design. Compared to single-case designs, multiple-case design have desinct advantages and disadvantages. The overall study is regarded as being more robust, due to evidence from multiple cases is often considered more compelling(Yin 2003). Multipple-case design also allows researchers to do cross-case analysis and comparison (Darke et al. 1998). Oates (2005) argues that the most common approach is to examine one case only but the readers often find similar evidence and outcomes from several case studies more compelling then conclusion drawn from just one case study. On the contrary, a single-case study is more appropriate where it is an extreme or unique case or to represents a citical case in testing a well-formulated theory (Darke et al. 1998; Yin 2003). Another advantage is that single-case deisgn allows the researcher to investigate a phenomena in depth and therefore be able to provide rich desciptions and understanding. Since one of the main purpose of conducting case studies is the possiblity to go in-depth of a phenomenon, some researchers argue that greater number of cases means more breadth but less depth. Trade-off will result in negative effects due to increasing the number of cases are countered by certain disadvantages (Dubois and Gadde 2002). Even though we agree in some extent, we argue that using an

multiple-case study gives us the posibility for comparison and will make the resarch more compelling for the reader.

Furthermore, we have to decide wether to use a holisitic or embedded case unit of analysis. Wether to use an embedded or holistic variants depends on the type of phenomenon that is being studied and on our research question (Yin 2003). An embedded case study design is when the researcher investigates different units within the same case. Each of the units may be explored individually and further together to paint a overall picture. A challenge with embedded design is in achieving a holistic perspective from the analysis of the sub-units. A holistic case study investigates the entire case as one units. This apporach gives an helicopter view of the case, but on the other hand it is enables the change to miss changes in the unit of analysis that could impact on the appropriateness of the original research design. Due to our choice to investigate non-functional requirement on a project level and our research question, we argue that holistic multiple case design is the most appropriate research design for our study. Chosing an embedded design apporach would be too difficult to conduct due to the time dimension and the extent of the master thesis.

3.3 Selection of cases

Coyne (1997) states that qualitative research sample selection has a profound effect on the ultimate quality of the research. Neglecting to describe the sampling strategies in sufficient detail, can make interpretation of findings difficult and can affect replication quality of the study. For quantitative research, sampling concerns itself with representativeness. For qualitative research, sampling seeks information richness and selects cases purposefully rather than randomly. In case studies, the logic involves theoretical sampling. Choose cases that are likely to replicate or extent the emergent theory is the goal, to fill the theoretical categories and provide examples for polar types (Meyer 2001). We have used theoretical sampling strategy for this research.

3.3.1 Case requirements

In line with theoretical sampling, we want to investigate projects that systematical prioritize non-function requirements in agile software development, and those that does not have awareness on non-functional requirement in agile software development. The first requirement we seek was that the company had to be in the market of development of software solutions.

We had no requirement regarding to which market domain the project should be in, but we targeted IT consultancy firms to be able to get in contact with a rich mix of fields within development of software solutions. The second requirement we had was that the projects had to follow agile methodologies, preferably SCRUM but this was not a requirement. The last requirement we had was that the project had to be completed. This requirement was especially important because we had to be able to see the different outcomes of the projects that systematically prioritize non-functional requirement and those that don´t focus on them.

To find out if the projects focused on non-functional requirements, we looked at the company's home page and tried to seek information. This turned out to be impossible due to lack of information about this particular subject. Therefore, we contacted companies directly over phone or e-mail. Two of the companies we contacted claimed that they continuous and systematically focused on non-functional requirements throughout the development process. We determined that we had to accept and trust their words, since we had no way to check if it was true. On the other hand, both of the companies are well established, good reputation and with a large client base, so we placed confidence it was credible.

For the two other companies we asked control questions when we contacted them to check if they had some knowledge about non-functional requirements. We then determined that the knowledge they contained was lacking or inadequate about non-functional requirements. We therefore would be able to get information about how these companies handle the importance of non-functional requirement without prioritizing them.

3.3.2 Collection of cases

After we evaluated and decided on the case requirements, we contacted multiple software companies trough telephone. We explained the purpose of our research and if the respondents found it interesting to participate, we sent e-mail attached with a project description that described the research project. We also an attached information letter that promised anonymity both for the company and the informants. The names of the companies would be replaced with pseudonyms and every participant agreed to that. The industry that the companies represent would not be anonymize in order to maintain the reliability and transparency of this research project. The informants also agreed to this. We also registered the research project at "Norwegian centre for research data" (NSD) and the respondents were informed of this.

In total we contacted 10 companies before we decided on four projects that met our pre-defined requirements. All of the four companies were in the field of software consultancy and had completed different projects in the last year. The first project stated that they tailored SCRUM method. They also argued that they had a systematic approach to deal with on the non-functional requirements were in the field of web application for research facility. This project will be referred to as *Research*. The second project that appeared to follow a tailored SCRUM method and also said that they have different methods and routines to deal with non-functional requirements were in the field of enterprise resource planning for an oil company the second company will be hereby be referred to as *Oil Industry*.

Name of project	Respondent	Project Description	Team size	Roles	Years of experience
Research	Research 1	Web application for managing application for research	5	Developer	2
Research	Research 2			Project leader	15
Oil	Oil 1	Enterprise resource system for oil industry	10	Scrum master	16
Oil	Oil 2			Front-end developer	5
IT Grocery	IT Grocery 1	A web and mobile application to support grocery logistic	10	Back-end developer	6
IT Grocery	IT Grocery 2			Front-end developer	3
Software Care	Software Care 1	A web and mobile application to support grocery logistic	15	Scrum master	10
Software Care	Software Care 2			Project leader	15

Figure 11: Overview of informants

The first project that appeared to have lack of awareness about non-functional requirements were in the field of enterprise resource planning systems for grocery market. This project also followed a tailored SCUM method. This project will be referred to as *IT Grocery.* The second

company that appeared to have not enough awareness and routines to non-functional requirements and also following a tailored scrum method were in the health care industry. This company will be referred to as *Software Care.* Figure 4 summarize the projects and participants.

3.4 Data Collection

The chosen research design together with which data collection method to deploy should be guided by the research question (Meyer 2001). Documentation, physical artefacts, archival records, direct observation, participant-observation, interviews are the most commonly used data collection methods (Yin 2009). The anchor of qualitative research is interviews and usually the primarily method for collection data. For our research we found that interviews are the most suited method for data collection.

3.4.1 Interview

The most common method for collection of data in qualitiv method and one of the most important sources of case study information is interview (Yin 2003). It is a great way to gather information and may seem straightforward at first sight, but it also has its pitfalls. Meyer (2001) brings this up and reflects around what interviewers need to be aware of when conducting data collection and refers to methods to avoid these pitfalls. In most cases, the interview object is a stranger to the researcher, which can lead to issues if the interviewer can be trusted. There is also time pressure that can cause the informant to come up with answers that are not really honest or fully thought through due to lack of time to think through the answer. Yin (2003) brings up weaknesses with qualitative interview due to the fact that there is a chance of bias since questions can be constructed poorly, the researcher could have already made up their opinion on the subject beforehand or that the informants gives the researcher what he or she wants to hear. These are points that we as researchers must bear in mind when constructing and conducting interviews under this project. For our research it is especially important to address these questions concerning bias in the data collection process, since our research concerns around comparing projects based on awareness about non-functional requirements. We believe that choosing the interview as primary method for data collection suit our goal for this research. It is a good method to obtain much information in a short period and exploring the participants opinions and viewpoints regarding the subject. In addition, it gives us the flexibility that we seek to investigate multiple different theories and framework.

3.4.2 Sampling Informants

In order to be able to assess the validity of the findings in a survey, one must be able to explain which sample the study is based on, so that it is clear to see who the results are based on. The selection of informants is therefore important in qualitative interviews. Who is to be interviewed, how many and according to which criteria they should be selected from are central questions that arise when defining and selecting the sample. The main requirement is therefore that the participants in the study need to have knowledge about the phenomenon we are researching. When contacting software concultancy companies we were lucky to come in contact with projects leaders, scrum master, front and back-end developers . Another factor is that the number of informants should not be too large since borth performing and processing the interviews are time-consuming (Coyne 1997). We therefore decided to interview two from each project with different roles.. The following requirements the informants had to fulfil:

- The informants need to have experience and knowledge about agile methodologies
- The informants need to have been on a software development project using agile methodologies.
- The informants must have to be have different roles in the agile project team; project manager, developer, product owner or scrum master.
- The informants must have to have basic knowledge about the requirement engineering field in agile development.

To be sure that the informants could contribute to this study it was important they met all of these requirements. We did not meet any problems getting informants to particpate in the study, the only problem that appeard was getting acceptance from the company. While on the other hand, when we informed orally or in written form about the project the companies were very helpful. In compliance with NSD the informants agreed that the interviews would be recorded and that the audio files would be deleted when the research project was finised. The informants concented orally and in written from.

3.4.3 Interview guide

By choosing a qualitative method and using interview as the data collection method, we wanted to capture different people's perspectives and experiences related to the projects focus on non-functional requirements. This corresponds with Qu and Dumay (2011) who claims that

interview are particularly well suited to provide information about people's experience, views and self-understanding. Different interviews can be divided into three types: structured interviews, semi-structured and unstructured interviews (Oates 2005). The unstructured and semi-structured interviews is the type that is used most in qualitative research in information systems. Semi-structured interviews are more suited when the aim of the research is to explore personal accounts and feelings. On the other, using a structured interview method would minimize the potential risk of influencing the informant. This due to the pre-determined, standarzides, identitcal questions for evry interviewe and little conversation between the researcher and informat. In the same way, stuctured interviews are more appropriate when the researchers are not so experienced and would minimize the risk for influencing the informant when there are little conversation. We decided that semi-structured interview would be more appropriate for this research study when it is highly depending on the respondents experience and thoughts. In additionn deciding on a semi-structured interview type would give us the flexibility and openness as interviewer, but at the same type allow the respondents to speak their mind. Our aim is that the interview setting will be more of a conversation instead of a formal interview to build trust but also to make the resepondent more comfortable. It also gives us the flexibility to ask follow-up questions in an unbiased manner that also serves our needs to collect data for our research question According to Oates (2005) the purpose of semi-structured interviews is rather to “discover” rather than “checking” and that supports our research objective.

Even though choosing semi-structured interviews gives us the flexibility, we developed a interview guide to make sure we captured questions related to our research question. The interview guide covered topics and key concepts of our research question. On the same time, we were careful not to over prepare the script so that the openess, flexibility and improvisation to create a conversation were present. We also avoided leading questions to avoid bias data. The interview guide were pilot tested on two practitioner that work in the software industry and some adjustment were made. In our origial interview guide we only used the term “non-functional requirement”. We found out that the term was very confusing as we outlined in chapter 2. We therefore outlined in the introduction that it also included “quality requirements” so the participant had the right understanding of that.

The interview guide was structured into five sections which togheter they would cover the areas that should form the basis for answerign the research question. The five sections was

- Preliminary questions
- Knowledge about non-functional requirements
- Requirement elicitation and validation
- Nature of non-functional requirement
- Non-functional requirement modeling

The informants were offered to take the interview both in english or in norwegian. Some particpants chose to take it in their first language and other thought it was easier to take it in english. We decided to offer the participant both choices in order to build trust and make them comfterabl and to avoid any language barrier that could prevent bias information.

3.4.4 The interview process

As stated, before all participants were contacted through telephone and informed orally about the research project. Afterwards the project information scheme where sent out on email so that they could prepare what to expect in the interview. In all cases the first participant functioned as a gatekeeper to arrange the contact with the second participant on the project. All interviews were schedule to be performed at the project site. For one of the projects the two interviews where performed at the project site in closed rooms. Due to the COVID-19 pandemic some challenges arose. Strict measures where implemented by the Norwegian Government and one of them prohibit in-person contact (this will be outlined more in chapter x, limitation for research). To be able to make progress regardless of measures from the Government we decided to take the rest of the interviews through digital video conference systems. We used different video conferencing platform such as Skype, Zoom and Microsoft Teams. The participant chose which conferencing tool we should use for the interviews. This worked perfectly and we did not have any issues regarding doing the interviews digitally. On the other hand, we learned that to deal with adversity is a part of doing research.

In total we used two weeks to conduct all interviews. We followed the interview guide to cover all key concepts but made sure to keep the conversation flow naturally and follow up on different insight and information the participant had. We informed participants beforehand that the interviews could take up to one hour each and all participant agreed that the interviews would be recorded. In average the interviews took approximately 55 minutes. One participant was interrupted after 30 minutes, due to work. We therefore schedule to take the rest of the

interview the next day. In addition to recording, we took field notes that covered our own thought, body language and statements we thought were more important to look into.

3.5 Data analysis process

In this chapter, we will describe the data analysis process and outline the choices we made and why. According to Thorne (2000) “Unquestionably, data analysis is the most complex and mysterious of all of the phases of a qualitative project." (68). The data analysis is about addressing the initial propositions of the study and bringing order, structure and meaning to the collected data.

3.5.1 Transcribing

Transcription can often be seen as a behind-the-scene task but is a powerful act of representation (Oliver et al. 2005). Getting the interviews in a written form makes it much easier to search through and analyze the data (Oates 2005). Oliver et al. (2005) argue that transcription can powerfully affect the way participants are understood, the information they share, and the conclusion drawn.

Transcription can be practiced in multiple ways, often using naturalism or/and denaturalism. Naturalized transcription is when every utterance is transcribed in as much detail as possible. On the other hand, we have denaturalized in which grammar is corrected and stutters and pauses are removed. In addition, non-standard accents are standardized (Oliver et al. 2005). For this research project we decided to use a denaturalized transcription. We chose this method due to the time frame of the research project and agreed that naturalism approach would be too time consuming. Also, the fact that we were more interested in the participant meaning and perception created and shared during the conversation rather than involuntary vocalization or depicting accents. We spent approximately 5 hours for every hour of tape. We captured the participants words as spoken but we edited them slightly to get rid of pauses and non-verbal sounds.

3.5.1 Coding

After finishing transcribing, we moved forward to coding the qualitative data. First, the researcher starts off with identifying themes in the data. According to Oates (2005) you can use three different themes; segments that bear no relation to our overall research purpose, segments

that provide general descriptive information that you need in order to describe the research context for your readers and segments that appear to be relevant to your research question. The last segment is the one we focus on. From there we categorize each segment with sub-heading that describes the theme presented by that unit of data. These names can be found using inductive or deductive approach. Inductive approach is based on categories observed in the data, for example those used by informants. This approach is based on the idea that you keep an open mind and let the data "speak" to you. The deductive approach is based on existing theories we have found in the literature or have developed our self. For this research we chose an inductive approach. We used categories and subcategories our informants used in the interviews.

3.6 Validity and reliability

Sources of error are linked to the quality of the study. Low validity and low reliability are two different sources of error that will impair quality. To ensure high validity, it is about asking the right questions, and in terms of reliability, it is about the accuracy of the survey. Validity depends, among other things, on the units and variables we choose, and whether we actually ask about what illuminates the phenomenon. Validity is linked to the relationship between the concrete data and the phenomenon we are investigating (Bryman and Becker 2012).

One pitfall a researcher must try to avoid is that the researcher's subjectivity affects the data being collected. This is to ensure that the findings can be viewed as credible and trustworthy (Bryman and Becker 2012). In order to ensure that we remained as objective as possible, the interviews were focused on containing open questions. The interviews were also arranged to be held as a conversation rather than an interview, using semi-structured interview guide. This led to the informant speaking in depth about the topic we were on, and in some cases the informant answered questions we had intended to ask later in the interview. We also made statements from theory and other researchers and asked about the informant's thoughts on this. Unstructured interviews are an interpretivist method and has its strengths and limitations.
Interpretivist methods produce qualitative data which helps researchers to uncover hidden meanings they may not have understood (Bryman and Becker 2012). This was an element that was important for data collection in our research, but due to the face that most interview had to be done over video conference there could have been some body language we did not able to observe and pick up.

Researcher error is any factor that may influence the researcher to have a wrong interpretation during data collection or during the analysis of the data (Bryman and Becker 2012). To reinforce erroneous interpretation, we recorded all our interview trough digital video conference, and then transcribed the audio files the same day or the day after. This reinforces the threat of research errors as we still had the interview fresh in memory. Nevertheless, there is a possibility that we have misinterpreted some answers as we do not, for example, know the respondents and their personal way of expressing themselves. We have taken care of to actively ask clarification questions during the interviews, which strengthens the research's reliability. We also made sure that we limited the number of interviews we held per day and have a day off after a day of interviews. This gave us time to transcribe, as well as having clear heads for a next interview.

The researcher´s role defines the parameters of the research situation, but it is difficult to guarantee external reliability since our social role in the interviews were unique. Since this study deals with a small sample, eight interviews of software development companies in Norway, it can be said to have a low degree of external validity. However, we got the impression that based on the questions we asked and what the informants told us; it is very likely that we would have received the same findings in a different sample.

4. Findings

In this chapter, we will look at the findings of our research, together with the data analysis. First, we will give a brief review of each company and project, next, we will analyze every case individually. We highlighted five challenges in our literature review regarding non-functional requirements in agile software project. We divided these challenges into three categories which were utilized in our case interviews. The categories will provide a framework for our findings and data analysis. Subsequently, we will present the findings in cross-case figure to show the at similarities between the cases. We will begin by analyzing the project "Research and Oil" which stated that they have good knowledge about non-functional requirements. Next, we will continue with IT Grocery and Software Care, who gave the impression that they do not emphasize non-functional requirements.

4.1 Research

The company is an information technology consultancy firm that has customers within different sectors. They focus on having cross-industry expertise to deliver value for their clients. Research is a project that develops web applications for managing applications for research for a research facility in Norway. The project team consists of 5 team members that all have different backgrounds and levels of experience. Research went over 2 years, but were not finalized due several reasons which we will clarify in chapter 5.

Knowledge about non-functional requirement

Research argued before the interview that they had good knowledge about the concept non-functional requirement. They talked about the concept on scrum meetings and on a daily basis. The informants were asked to describe what non-functional requirement means and consist of.

"Non-functional requirement is a feature in the system that is invisible for the customers. They do not affect the functionality of the project but makes them work better"

- Research 2

Furthermore, they were asked where they learned about the non-functional requirement concept. Both of the informants stated that they had knowledge about non-functional requirements before the project started. They learned about it in their studies or in previous jobs.

"I learned about it at school, but not so much in my work life. From the studies I knew that security for example when under the category non-functional requirement. On the other hand, that authentication went under functional requirement."

- Research 1

Elicitation and specification of non-functional requirement

Research states that they don't formulate nor document non-functional requirements. IT is only talked about during meetings, and it is expected that everyone on the team should have knowledge about it, and to be able to consider them during the development. They made a software requirement document in the beginning of the project. In addition, user stories to help create a simplified description of a requirement. Nevertheless, non-functional requirements were not present in the software requirements document or considered documented in another way.

"We did not formally document non-functional requirements in any way but talked about them at meetings. For me it was self-explanatory that we should consider it when we were developing the code"

- Research 2

"From school I learned how to formulate and document non-functional requirements, but on the project, we found it difficult to implement."

- Research 1

Customer involvement

For the Research project the client had a technical background on software development and were able to see the importance of non-functional requirements like security, performance and maintainability. In agile requirement, elicitation is carried out through iterative RE and face-to-face communication with the client. On the contrary, with traditional requirement elicitation all requirements are discovered before the development start. For Research they made a requirement document with all the overall features that the system should perform before the development started. This was done to be able to agree on a set of terms with the client. But overall through the whole development process the requirement evolved over time and other requirements were added throughout the development process. In the requirement document they also included points related to how many users that the web application should support (performance requirement), how many security parameters they needed (security requirements) and how much storage space (space requirement) the web application should

hold. During the development, they had continuous interaction with the client to agree on prioritizing of the requirements.

“Due to the client's technical knowledge, he saw the importance of prioritizing non-functional requirements from the start of the project. The client was therefore willing to invest the money to make the web application up to date on security, maintainability and performance.”

- Research 2

“The client had some technical knowledge and was able to see the importance of prioritizing non-functional requirements. Nonetheless, we had to argue and convince them why we should have some extra time for testing scalability and security requirements”

- Research 1

4.2 Oil

The company is an information technology consultancy firm that has customers within both public and private sectors, but their special expertise and main focus is in the public sector. Their main focus is on technological solutions and consultancy services for public sectors within fields as low, health, logistic and oil. Oil is a software project for a company in the oil industry where the main focus was to develop an enterprise resource system to support engineers. The project team consists of 10 team members, all with different backgrounds and level of experience. For this research we were lucky to interview the Scrum Master (Oil1) of the project and Front-end developer (Oil2). The project went over 3 years but are not fully closed due to continuous improvements and different features the client wants.

Knowledge about non-functional requirement

Agile software development relies on tacit knowledge within the team, and the aim is to reduce the focus on requirement specification and documentation. For the Oil project, they stated that the team were skilled on non-functional requirements, thus they had focus on it throughout the development process. Also, they stated that they have a very experienced project leader, who saw the importance of non-functional requirements. They also emphasize that they are dependent on experience and skilled people in every field to be able to deliver a high-quality product. Since the technical sector is so broad, no one is able to have knowledge about every aspect that is required. Therefore, both the architect, front-end developers and technical leader are all required to have experience and knowledge to be able to prioritize non-functional requirements. The interview object where asked where they had learned about

the concept. Oil1 stated that he learned it through work experience, Oil2 learned about it during their education. Furthermore, they stated that they had daily stand-up meetings and scrum meetings where they talked about it to be able to keep it as a priority throughout the development process.

"I had not heard about the term before, but when I heard the meaning I am very familiar with the concept. For me, non-functional requirements are how good the application perform what it is intended for"

Oil 2

"Yes, to some extent I heard about it, but it is nothing that I go around and think about. For me, non-functional requirements are not requirements that you necessarily need for the application to work or that are included from the client requirements specification. On the other hand, it is more based on specifications that you want but that are not always prioritized."

Oil 1

Elicitation and prioritizing of non-functional requirement

First, Oil states that they together with the customer created a high-level requirement document before the project started. This was done to estimate project costs and schedules, but also to get an overview of the project. Non-functional requirements were included in the document but only as statements. Oil also states that they used this document when new members were added to the team so they could get an overview of the project purpose and goal. Oil states that they elicit functional requirements through iterative requirements engineering and face-to-face communication with the customer. After every iteration they have iteration planning meeting with the customer to prioritize the requirements for the next iteration. Here they select and commit to the "stories" or requirements from the team backlog for the following iteration. The customer and the team also use the iterative planning meeting to decide to add more features or user stories to the product. Oil states that at the iteration planning meeting the client always wanted to add new features. Together with the development team they discuss the requirement in greater level of detail and the outcome is a set of defined requirements the project has to implement. From the defined requirement they make user stories that are a short description of the requirements. Oil was asked if they also define non-functional requirements after every iteration, they only discussed them, but they did not define them.

“We do not define non-functional requirements in the same way as functional requirements. They are not included in the backlog. We only talk about non-functional requirements during iteration meetings. In the requirement document we only made statements of what quality was expected of the system. Therefore, many of the non-functional requirements were not included in the development of the system from the beginning. This became a problem later with for example scalability. This is harder to do something with later in the project and the consequence is that the system was not up to today's standard.”

Oil 1

“Actually, this is a big problem in the project at the moment. We have seen that some of the non-functional requirements like scalability and routing were not prioritized in the early development. This is a major concern now that is hard to something about, but we are trying to fix it”

Oil 2

Customer involvement

Oil states that their customer emphasized the importance of security from the beginning of the project. Therefore, the team always prioritized security throughout the development process. On the other hand, they stated that their client did not have good technical knowledge and was not able to understand the technical aspect of the development process properly. Even though the team tried to argue that non-functional requirements should be included (scalability, usability and robustness) the client often prioritized the functionality instead. Oil argued that time limitations and budgets was the main reason they were not allowed to include them from the customer. As a consequence, some have to spend a lot of time and resources later on in the project to include non-functional requirements because the client sees the importance of it when the system is not to the standard it should. We are always trying to teach the client about the development and the technical aspect of the project. Also, they argue that if they spent more time testing the application towards the main users, they would be able to see the problem beforehand, and maybe the client would be able to understand the importance. They are thus trying to implement systematic testing in the project to be able to detect these problems.

“We have to implement and focus on the requirements and features the client prioritize. Since the client in this case only focuses on functional requirements, we can only argue our case to prioritize non-functional requirements, but in the end, it is the customer's call. Since the client found security important in this case, it is a high priority. On the other hand, the client did not find scalability important and therefore it is a low priority and not implemented.”

Oil1

“I would argue that if we included testing toward the end-users earlier, we would be able to see the problem with for example scalability and show it to the customer.”

Oil2

4.3 IT Grocery

IT Grocery is a project from an information technology consultancy company in Norway. The company is specialized in project management within information technology. They are a leading provider of software solutions, and work toward delivering high quality products to all sectors in Norway. They built a web and mobile application to support grocery logistics. The project consisted of 10 team members, all with different backgrounds and experience. The project was finished earlier this year and went over 3 years. They are still working on some improvements and additional features that the client wants to implement.

Knowledge about non-functional requirements

IT Grocery did not seem to have adequate knowledge about non-functional requirements when we contacted them and asked control questions. In the interview they were asked to describe what the term implied, it was obvious that they had a good understanding of the concept. On the contrary, they argued that they have good knowledge within the team on non-functional requirements but a lot of people with minimal experience (Junior developers). Many of the developers had just come out of school and had therefore not enough experience. Due to that, they came across a lot of issues during the development because the lack of experience led to inadequate attention to non-functional requirements. Many of the developers did not know how to build the application and detect problems regarding non-functional requirements.

"As I see it, non-functional requirements are not really necessary for the project to evolve and are not included in the requirement document from the customer."

IT Grocery 1

"Everyone on the project had fair knowledge about non-functional requirements, but from my perspective, experience is the key to implement it properly. Since the team consists of mainly junior developers, this has been a problem detecting bugs relating to scalability and performance."

IT Grocery 2

Elicitation and prioritizing of non-functional requirements

IT Grocery states that they do define non-functional requirements for performance and security in their project. Their customer made a requirement document before the project started that included all the functionality the systems should support, but also requirements regarding performance and security. IT Grocery was a big project that was intended for a large group of end-users. Therefore, the system had to have the capacity of a large user group. The project consisted of two weeks sprints and iteration meeting after every sprint with the customer. They prioritized requirements for the next sprint to be implemented, but also elicited new features and requirements that the customer wanted. They defined the non-functional requirement regarding capacity and security but did not include other non-functional requirements like scalability, availability and maintainability. This they explained was due to the fact that they did not see the possibility to define them. In addition, they admit that they should have thought about the problem regarding scalability beforehand. When they released the first prototype, they got a lot of feedback that the scalability did not work as it should today. They say that they actually did not think about the problem regarding this non-functional requirement. Further, they state that the example was not the only one that arose during the development. They experienced a lot of times after the release that things were not up to the standard for the end-users. This did not just have an impact on the end-users but also the IT Grocery reputations as a software developer provider. Also, the time and money that went into fixing all of the improvements regarding non-functional requirements that were not implemented, made a lot of setbacks for the project.

“I do not think that one should define the non-functional requirement the same way as functional requirement, but I believe that we should implement some kind of checklist to always keep focus on them. “

IT Grocery 1

“From my point of view, I believe that if we had a team that consisted of a lot of experienced people the issues we faced would not have happened. Unfortunately, we only have a few senior developers at the team, and no one can have the knowledge about every aspect of information technology that is needed for these applications.

IT Grocery 2

Customer involvement

IT Grocery had their office place at the customers headquarter the whole time during the development. They worked closely together with the development team for the client and person in charge. On account of that the client had a lot of experience and knowledge about information technology and software development. They therefore saw the importance of prioritizing non-functional requirements like capacity and security. On the other hand, IT Grocery argued that the client did not prioritize other non-functional requirements like scalability, and thus it was not a priority during the development. In the same way, the client mainly focused on the core functionality and getting the applications ready for the deadline and ignored issues related to non-functional requirements. *IT Grocery2* argued that prioritizing non-functional requirements mainly up to the client. If the client finds some requirement important it will be implemented like capacity and security in this case. On the other hand, the client did not find scalability important and therefore it was not a priority and the client mainly focused on release date instead.

“This is the same case at every project I have been on. If the client finds it important enough, we will get the time and money to prioritize them. On the other hand, if the client does not find them important enough, we can only argue, and try to convince them, but in the end, it is the client's choice.”

IT Grocery 2

4.4 Software Care

Software care is a software development project for the public sector. Software Care was a software development project to build a mobile and web application for patients to register medical results from home. The project went over 3 years and is still ongoing. The project is from a Norwegian based information technology consultancy company that specialize in the public sector within different segments. The project consisted of 15 team members with different responsibilities and years of experience. In the end, software Care delivered a successful application, but they came across different issues related to non-functional requirements.

Knowledge about non-functional requirements

From our control questions it seemed that Software Care believed non-functional requirements contained security requirements. Therefore, we made the impression that Software Care was not aware of the issues related to non-functional requirements and did not have sufficient knowledge of the concept. Furthermore, it seemed that Software Care's one priority was security for the applications when it contained patient data and sensitive information. In addition, they argued that most of the team members had knowledge and experience working in that respective subject domain before. The profound knowledge that the team members had enhanced from the client´s domain from previous projects made them able to advise the client better in the matter of security. On the other hand, when it came to the case of other non-functional requirements, they argued that scalability was not a priority in their development from the start and therefore they did not have a lot of knowledge regarding that matter.

"It is not a term we use a lot. We mainly focus on the requirement document and deliver what is expected of us. Since the client mainly focused on security this was number one priority for us."

Software care 1

Elicitation and prioritizing of non-functional requirement

According to our respondents they did not have any routines or framework to handle non-functional requirements. They only focused on the non-functional requirement security and these were also defined and elicited in the same way as functional requirement from the start. The non-functional requirements were just briefly talked about during development meeting but was not considered important for this particular software. On the other hand, after releasing the first prototype to the end-users they experience massive problems regarding performance and scalability. The scalability was not properly implemented. In addition, the platform became more popular than they expected, and the platform was not performing fast enough for the end-users. Software 1 states that they tried to fix the problem, but the only way to fix bugs is to test it. Unfortunately testing towards scalability and performance was not implemented and they did not have any routines to do so.

"I think maybe non-functional requirements need to be prioritized in the same way, you have to get a value on them, so that you can prioritize against other functional requirements, but to formulate it in the same way I have no faith in it."

Software care 2

Customer involvement

Software care has learned together with their customer the hard way of not taking into account that a service became more popular than expected, concerning the scalability and performance requirements. We cannot go into the details about this project, but it was such a critical situation that the media wrote about it. The problem was quickly solved but is a typical example of where things do not cooperate and fails since it has not been thought of before.

"Especially the of reputation is harmful, because usually you manage to close things up pretty quickly when you discover it. If it is a company's internal solutions, this is not an issue, but if it is a public solution that should reach many, then for example, for a large company it can be expensive and mean that you lose a lot of money"

Software care 1

4.5 Cross-case analysis

Figure 12 represent a visual presentation of our comparative finding from the different cases. The figure shows that there are many similarities in the different cases regarding the different categories. We will elaborate them further in chapter 5, discussion.

Companies	**Knowledge**	**Elicitation and prioritizing**	**Customer involvement**
Research	Yes	No	Yes
Oil	No	No	No
IT Grocery	No	To an extent	Yes
Software care	No	No	Yes

Figure 12: Cross-case analysis

5. Discussion

This chapter will discuss the findings presented in chapter 4 in light of the literature and theory reviewed in chapter 2. The purpose is to gain a deeper understanding of the current state of non-functional requirements in agile software development and to highlight further new insights the study has to the existing literature on non-functional requirements. We will first do a cross-case analysis of our findings to look at similarities. Next we will connect it to existing literature to present recommendations. The discussion will first focus on answering the research question:

"How do companies handle non-functional requirements in agile projects and why should non-functional requirements be more emphasized in agile software development method?"

5. 1 Knowledge about non-functional requirements

One of the main findings in this study is that the knowledge about the term and the importance of non-functional requirements are still inadequate in the software development community. From the existing literature, it was clear that the definition of non-functional requirements was not clear and the breadth of the area it falls under (Glinz 2007; Chung et al. 2012). This is also evident in our research. Even though the participants in the research were informed of the different names for the term; non-behavior requirement and quality requirement, none of them had the knowledge to which breadth it included. Furthermore, it is noticeable that some of the non-functional requirements are repeated to be more emphasized in the agile software development community, such as performance, security and scalability. These non-functional requirements are seen as the most important by our participants and are also what they think about when they hear the term non-functional requirements. For some of the participants, they are aware of the importance of non-functional requirements and have enforced some measures. This includes courses and different measurements they try to employ, such as testing towards end-users. Even though the awareness about non-functional requirements is improving, they still struggle to employ these measurements as intended, for example testing. This is since there is no formal framework or method to use and the routines are not formally implemented.

From the literature, it is also stated that experienced team members provide more attention to the importance of non-functional requirements in the development, than to team members with less experience. Experienced members can better identify non-functional needs (Camacho, Marczak, and Cruzes 2016). Our respondents emphasize this. IT Grocery 2 argued that since

their team mainly consisted of junior developers, detecting bugs relating to scalability and performance has been a problem. Camacho, Marczak, and Cruzes (2016) highlight that the senior developers can influence the team' culture if they have the right mindset regarding non-functional requirements. A solution was proposed by Camacho, Marczak, and Cruzes (2016), where they recommended that the whole team should attend project meetings, such as sprint planning and project inception. When different team members attend the meetings, awareness and the importance of non-functional requirements could be highlighted. In conclusion, the non-functional requirements will be better understood, and it will be an aligned agreement within the team on how to handle them.

5.2 Prioritizing of non-functional requirements

The theory chapter explained that some of the challenges with agile requirements engineering are the lack of requirements prioritization of non-functional requirements. They are often ill-defined and ignored during early development cycles (Ramesh, Cao and Baskerville 2010). Neglecting non-functional requirements can later be time-consuming and can lead to a massive rework. Another point previous literature address is the importance of defining non-functional requirements. This is done to take into account "what-if scenarios" and will hopefully avoid pitfalls and significant consequences for the project in the future (Chung 200). One of the informants identifies budget constraints and time as factors that cause them to neglect non-functional requirements.

None of the companies we talked to, took the time to formulate nor document non-functional requirements formally. The companies in our research experience that since there has not been much focus on non-functional requirements early in the project, this later becomes a problem – especially in the case of scalability. This requirement is more challenging to handle later in the project, as a consequence, the project did not fill the expectations of today's standard. The companies agree that non-functional requirements should be emphasized in the establishment phase of a project, and some suggested doing this via a checklist, to keep the focus on them. This can be linked up to Farid (2012) and Bourimi et al. (2010) proposals and suggestions on dealing with non-functional requirements as presented in 2.4.4.

Often, the customer is most concerned about the functions of the system and will have it delivered as soon as possible, which can damage the development and focus on non-functional

requirements. Experience is one aspect several of the companies bring up when it comes to knowledge about managing non-functional requirements. Another aspect that is important is to conduct functional tests and also performance tests. It is important to simulate a real use of the system (not always easy to estimate) because it is desirable to find bottlenecks by simulating the right number of users who will simultaneously perform the same actions or transactions. If all these aspects are taken into account, hopefully the end-user will get a good experience of both the functions and the experience of how the system behaves.

5.3 Customer involvement

A general finding was that the customer has the final word, and they are in charge of what requirements and functions that will be prioritized. *IT Grocery 2* said that they often would discuss and try to convince the customer and what non-functional requirements they should emphasize, but in the end, it is up to the customer. This is often linked up to how much money and time they would like to spend on the project. Security is a non-functional requirement that has a high priority across the teams, but *Oil 2* told us that their client did not find scalability important and was therefore not implemented – which was a problem they had later on in the development. They argue that one way of convincing the customer to emphasize more on this non-functional requirement is to include testing towards the end-user earlier. They would have then picked up the scalability-problem earlier. This tells us that the budget a company has on a project does not always match with what is needed to take into account all non-functional requirements. Ramesh et al. (2010) states, budget and schedule estimation is one of the challenges in agile teams. Since every project is not identical, it makes it hard to make upfront estimations of what should be emphasized and implemented at certain times.

So, what are the consequences for the customer of not emphasizing and prioritizing non-functional requirements? Loss of reputation is an aspect that could harm the customer severely in the long-term perspective *Software care* argues. If the requirements of security and performance are not taken into account in a project with sensitive information that should not be accessible to the public, that is more critical compared to a functional requirement not working which can be fixed within a short time. As stated by *Software care,* it is often more time consuming and more expensive to implement and fix non-functional requirements at in the late phase of a project. Consequently, it can be wise for a company to spend a bit more time and money on these “what if”-cases could occur.

In the case of *Research 2,* they were involved with a customer that had a high technical knowledge that saw the importance of prioritizing non-functional requirements from the start of the project and were willing to invest money on ensuring that the web application was up to date on security, maintainability and performance.

5.4 Our own thoughts on non-functional requirements in software development

In order for a functioning system to fit into an organization and perform the tasks it is designed for, one must identify and specify the requirements that the organization and user have for the system. This is not a simple process, and it requires efforts from both the developers and customers. Customers are not always able to communicate to the developers what requirements they expect for the system. An example of this is that customers want their new system to be more efficient. This is a vague claim that can be interpreted in many ways. Do they want more storage capacity, faster execution of tasks or cost savings? Here, the developers will have to spend a lot of time identifying what the customer actually means. If they invest in one of the interpreters, this can result in a system that does not fit into the organization and the customer's actual needs. Requirements are also usually situation dependent. As a software team, you must therefore spend time getting to know the organization and their work so that you can adapt the system to their way of working. In addition, such requirements tend to change as the development process progresses. This must also be taken into account.

If the industry wants to continue the trend toward ever-improving information systems, we must place more emphasis on those parts of the system development that have a direct impact on the users. Non-functional requirements are essential in the development of systems of how users expect them to operate. The requirements must therefore be identified, and solutions must be provided in order for the software to meet and satisfy them. Thus, in our opinion, it is important that the software development theory emphasizes these as an important part of software development and the work in order to achieve a high success rate in software projects. What our findings show is that some companies take this into account to some extent, but several of the informants agree that more and more priority should be given to it.
It may be that a lot of projects take non-functionals requirements for granted as a part of the software development. As users of different software, we expect them to be available, reliable

and secure at all times. Since we regard these functions as quite obvious should be in a software system, one will not prioritize spending time on this in the analysis and design phases of a project. If the developers are not trained to have these requirements explicitly identified and specified, one may risk that these requirements do not receive the necessary focus, and in the worst case, do not take care of at all.

All of this shows that requirement specification is a process that requires a lot of work and perhaps even more stringent guidelines. This applies between other requirements to be looked at and who in the project should be included under the requirements specification.

5.5. Which requirements should be emphasized?

Every software development project is not identical. An information system developed for one hospital may not be used directly in another hospital. The various factors that need to be taken into account will change, both from project to project and in the process of a project. There will be individual situations, and thus also various relevant non-functional requirements. This makes it tough to generalize which of the non-functional requirements that should be emphasized. What is clear is that one should include the non-functional requirements as part of the analysis and requirements specification. To help with this part of the development, you can have checklists that you go through to find the relevant requirements for a quarter project. Here, the model of non-functional requirements presented in the theory chapter may be of assistance.

5.6 Limitation

The main limitation of our study is that we based all of our data collection and analysis on semi-structured interviews. We asked our participants for permission to get access to documents such as requirements document and project plans, but this was not possible due to confidentiality towards their customers. The results are under the influence of our interpretation of the phenomena investigated is a consequence of this limitation. We interviewed different software development teams and team members with different roles and responsibilities. This made it possible to study the phenomena from different viewpoints, even though we had limited time to conduct this study. In addition, it reduced this limitation. We would like to point out that observing body language during the interviews was original in the plan of this study. Due to the COVID-19 pandemic, we had to conduct most of the interviews through video conference systems. This worked perfectly collecting data, but it prohibited us partially from observing the informants' body language.

Another possible limitation is the generalizability in a multiple case study is limited. Even though we interview four case projects with eight informants in total, additional interviews would gain additional insight into the phenomena. Due to time limitations and resourcess we were not able to do so.

5.7 Further research

In future research related to emphasizing non-functional requirements in agile teams, it would be interesting to see if the teams that neglect and prioritize more on non-functional requirements earlier in projects accomplish a high success rate or avoids pitfalls. It would also have been interesting to compare agile teams that prioritize non-functional requirements at the same level as functional requirements, as compare similarities and inequalities.

5.8 Implications

This study has both methodological and other limitations that are relevant to elucidate. In a purely methodical way, the case study deals with four companies in the IT development sector. In these companies, we have conducted in-depth interviews with seven respondents who have different roles.

This sample is not enough to draw conclusions beyond these companies. In chapter 3 Method, we discuss the research quality, and one example is the number of respondents in the study - which is limited due to the time we have been able to both plan the study and collect data, and further analyze findings against existing research in the field. If we had more than one semester available, we could have done data collection on an even wider specter of companies.

Another methodological limitation of the study is that we have not interviewed with the management team in any of the companies. This could possibly give the findings more nuances. We have nevertheless had a focus on interviewing people with a wide variety of positions. We would therefore like to say that this strengthens the study, despite the restrictions on the perception from the management.

Due to the time frame of one semester, we have made some refinements to the study. Non-functional requirement is a topic with a variety of approaches. The theme is still too broad for

us to cover everything in this study. Therefore, we have chosen to focus on elements in order to achieve a successful project and see how companies emphasize this. This is done in order for the study to be able to go into the depth of each element and aspect.

Despite the limitations above, we still believe the study has value as it provides insight into how some companies deal with non-functional requirements. The findings are interesting since there is still a lack of awareness in the industry, which has significant consequences that can lead to a failure of a project or even worse the reputation of a company which can have long-term harm on income.

6. Conclusion

Throughout this thesis, we have presented, elaborated and discussed how agile software development teams experience regarding non-functional requirements. The study is based on four different software development projects in the Norwegian software development industry. We identified five key challenges in the literature regarding non-functional requirement agile software development projects. These are knowledge, customer involvement and testing, early prioritizing and elicitation of non-functional requirements. These challenges were also evident in our findings. This chapter will attempt to answer the research question "*How do companies handle non-functional requirements in agile projects and why should non-functional requirements be more emphasized in agile software development method?"* by concluding the insights about the challenges discussed in *Chapter 5 Discussion.*

Our findings indicate that software teams have some awareness of non-functional requirements, but they are not prioritized and handled at the same level as functional requirements. They are not defined or documented in any way in most of the cases. They are only talked briefly about during development meetings and in some cases emphasized to be necessary. On the contrary, some of our cases argued that they do not believe in handling non-functional requirements the same way as functional. This goes against the previous literature on the subject that argues that non-functional requirements should be handled the same way as functional requirements. All of the cases do not have the right approach to know how to handle them properly. Testing should be implemented narrowly during the development in order to detect defects regarding non-functional requirements. This is something our informants also point out that they are trying to improve in their development, but in all cases, this is not done. Due to the priority of resources and time from the customer.

This has led to some projects experiencing problems later in the development. Especially regarding non-functional requirements such as scalability and performance have often shown problems that occurred later on in the development. This has led to setbacks in the project that have an effect on cost and resources for the client, and in the worst case, this can hurt the company's repetition both for the customer and the software consultancy company. This is all about prioritizing and acknowledging the importance of non-functional requirements. According to our findings, the agile team can have good knowledge about non-functional requirements and know the importance, but the client often considers it to be not crucial for

the system. It is evident if the client does not have the technical knowledge, this affects the prioritizing of non-functional requirements. They often want the functionally in regard to the quality. One way to handle this is that the software development team can communicate the importance. Even though the findings in this study indicated that it is done, in the end, it is the client's decision. Therefore, we argue that it is essential that the client has technical knowledge, preferably several years of experience in the software development sector. This is also evident in the agile software team, and it needs to have a diverse selection of team members with various years of experience.

References

Batool, A., Motla, Y. H., Hamid, B., Asghar, S., Riaz, M., Mukhtar, M., and Ahmed, M. "Comparative study of traditional requirement engineering and agile requirement engineering." *Presented at 2013 15th International Conference on Advanced Communications Technology (ICACT)*.

Beck, K. (1999). "Extreme Programming Explained: Embrace Change"*Pearson Education*. City: US.

Behutiye, W., Karhapää, P., Costal, D., Oivo, M., and Franch, X. "Non-functional requirements documentation in agile software development: challenges and solution proposal." *Presented at International conference on product-focused software process improvement*.

Benbasat, I., Goldstein, D. K., and Mead, M. (1987). "The case research strategy in studies of information systems." *MIS quarterly*, 369-386.

Bourimi, M., Barth, T., Haake, J. M., Ueberschär, B., and Kesdogan, D. "AFFINE for enforcing earlier consideration of NFRs and human factors when building socio-technical systems following agile methodologies." *Presented at International Conference on Human-Centred Software Engineering*.

Bryman, A. (2016). *Social research methods*: Oxford university press.

Camacho, C. R., Marczak, S., and Cruzes, D. S. "Agile team members perceptions on non-functional testing: influencing factors from an empirical study." *Presented at 2016 11th international conference on availability, reliability and security (ARES)*.

Chung, L., Nixon, B. A., Yu, E., and Mylopoulos, J. (2012). *Non-functional requirements in software engineering*: Springer Science & Business Media.

Cleland-Huang, J., Settimi, R., Zou, X., and Solc, P. (2007). "Automated classification of non-functional requirements." *Requirements Engineering*, 12(2), 103-120.

Coyne, I. T. (1997). "Sampling in qualitative research. Purposeful and theoretical sampling; merging or clear boundaries?" *Journal of advanced nursing*, 26(3), 623-630.

Darke, P., Shanks, G., and Broadbent, M. (1998). "Successfully completing case study research: combining rigour, relevance and pragmatism." *Information systems journal*, 8(4), 273-289.

Dingsøyr, T., Nerur, S., Balijepally, V., and Moe, N. B. (2012). "A decade of agile methodologies: Towards explaining agile software development". City: Elsevier.

Dubois, A., and Gadde, L.-E. (2002). "Systematic combining: an abductive approach to case research." *Journal of business research*, 55(7), 553-560.

Farid, W. M. "The Normap methodology: Lightweight engineering of non-functional requirements for agile processes." *Presented at 2012 19th Asia-Pacific Software Engineering Conference*.

Farid, W. M., and Mitropoulos, F. J. "NORMATIC: A visual tool for modeling non-functional requirements in agile processes." *Presented at 2012 Proceedings of IEEE Southeastcon*.

Glinz, M. "Rethinking the notion of non-functional requirements." *Presented at Proc. Third World Congress for Software Quality*.

Glinz, M. "On non-functional requirements." *Presented at 15th IEEE International Requirements Engineering Conference (RE 2007)*.

Inayat, I., Salim, S. S., Marczak, S., Daneva, M., and Shamshirband, S. (2015). "A systematic literature review on agile requirements engineering practices and challenges." *Computers in human behavior*, 51, 915-929.

Laanti, M., Salo, O., and Abrahamsson, P. (2011). "Agile methods rapidly replacing traditional methods at Nokia: A survey of opinions on agile transformation." *Information and Software Technology*, 53(3), 276-290.

Meyer, C. B. (2001). "A case in case study methodology." *Field methods*, 13(4), 329-352.

Mishra, D., and Mishra, A. (2011). "Complex software project development: agile methods adoption." *Journal of Software Maintenance and Evolution: Research and Practice*, 23(8), 549-564.

Oates, B. J. (2005). *Researching information systems and computing*: Sage.

Oliver, D. G., Serovich, J. M., and Mason, T. L. (2005). "Constraints and opportunities with interview transcription: Towards reflection in qualitative research." *Social forces*, 84(2), 1273-1289.

Orr, K. (2004). "Agile requirements: opportunity or oxymoron?" *IEEE Software*, 21(3), 71-73.

Paetsch, F., Eberlein, A., and Maurer, F. "Requirements engineering and agile software development." *Presented at WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.*

Palmer, S. R., and Felsing, M. (2001). *A practical guide to feature-driven development*: Pearson Education.

Papadopoulos, G. (2015). "Moving from traditional to agile software development methodologies also on large, distributed projects." *Procedia-Social and Behavioral Sciences*, 175, 455-463.

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

Schwaber, K. (1997). "Scrum development process", *Business object design and implementation*. Springer, pp. 117-134.

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

Sillitti, A., and Succi, G. (2005). "Requirements engineering for agile methods", *Engineering and Managing Software Requirements*. Springer, pp. 309-326.

Sommerville, I. (2011). "Software engineering 9th Edition." *ISBN-10137035152*.

Thorne, S. (2000). "Data analysis in qualitative research." *Evidence-based nursing*, 3(3), 68-70.

Umar, M., and Khan, N. A. "Analyzing non-functional requirements (NFRs) for software development." *Presented at 2011 IEEE 2nd International Conference on Software Engineering and Service Science*.

van der Heijden, A., Broasca, C., and Serebrenik, A. "An empirical perspective on security challenges in large-scale agile software development." *Presented at ESEM*.

Walsham, G. (1995). "Interpretive case studies in IS research: nature and method." *European Journal of information systems*, 4(2), 74-81.

Yin, R. K. (2003). "Case study research: design and methods (ed.)." *Thousand Oaks*.

Yin, R. K. (2009). "Case study research: Design and methods. Sage publications." *Thousand oaks*.

Appendix A: Ethical approval

Kristiania University College

08th of May 2020

STATEMENT OF ETHICS APPROVAL

Proposer: Marie Stedje and Sebastian Hestevedt

The school´s research ethics committee has considered your submitted proposal. Acting under delegated authority, the committee is satisfied that there is no objection on ethical grounds to the proposed study.

Approval is given on the understanding that you will adhere to the terms agreed with participants and to inform the committee of any change of plans in relation to the information provided in the application form.

Yours sincerely,

Asle Fagerstrøm
Professor

Appendix B: Interview guide

Hi, our names are Marie Raae Stedje and Sebastian Berg Hestvedt and this interview is for our master thesis that is about agile project management. This interview will be recorded and after the research project has ended, the audio file will be deleted. Make sure the informant gives his/her consent about this. We also want to inform you that the data will be anonymized in the final report.

Preliminary questions

1.1 Are you currently or have been in a software project?

- 1.1.1 What software development method did/do you use for the development?
- 1.1.2 How many team members are on the project?
- 1.1.3 Over how long is the time frame for the project?

1.2 What role did/do you have in the project?

- 1.2.1 What responsibility did you have in the project?

Knowledge about non-functional requirements

2.1 Are you familiar with the term non-functional requirements?

- 2.1.1 Can you describe what the term non-functional requirement involve?
- 2.1.2 How is in charge to prioritize non-functional requirements in your projects?
  - 2.1.2.1 How is it prioritized in your projects?
- 2.1.3 What are your responsibility when it comes to NFR in your project?

Requirement elicitation and validation

3.1 How do you elicit and define NFRs in your project?

- 3.1.1 Do you think you have a good arrangement on how to define NFR?

3.2 How do you prioritize non-functional requirements in your project?

- 3.2.1 Is NFR often categorized in the same way as FR?
- 3.2.2 Do you prioritize the NFR throughout the development process? (or only late in the development process)

3.3 How do you evaluate the NFR in the project?

Nature of non-functional requirements

4.1 Do you believe that non-functional requirements are not prioritized in software projects ?

4.1.1 If yes / no, do you think that is also the case generally in other projects in other companies?

4.1.2 What non-functional requirements do you think are critical for a project to be successful?

4.1.3 Safety and performance are two important non-functional requirements. Will this be addressed before or during the project's time frame?

4.1.4 Do you think safety and performance are two requirements that need to be focused more on earlier in the project?

4.1.5 What consequences do you think can arise from not prioritizing NFR in a development project?

Requirement modeling

5.1 Do you have routines for managing the NFR in the project?

5.1.1 Have you implemented any measures to have more focus on NFR in the project?

5.1.2 Do you follow any models or frameworks to focus on NFR?

Summarize the interview quickly. Ask if the interview object has anything more to add or we forgot to ask. Ask if the interview object has any questions. Thank you for the interview and the interview ends.

Appendix C: NSD – Norsk senter for forskningsdata

NSD NORSK SENTER FOR FORSKNINGSDATA

NSD sin vurdering

Prosjekttittel

Non-functional requirements in agile projects management

Referansenummer

832425

Registrert

11.03.2020 av Sebastian Berg Hestvedt - hesseb18@student.kristiania.no

Behandlingsansvarlig institusjon

Høyskolen Kristiania - Ernst G. Mortensens Stiftelse / School of Economics, Innovation, and Technology / institutt for teknologi

Prosjektansvarlig (vitenskapelig ansatt/veileder eller stipendiat)

Asle Fagerstrøm, asle.fagerstrom@kristiania.no, tlf: 95075325

Type prosjekt

Studentprosjekt, masterstudium

Kontaktinformasjon, student

Sebastian Berg Hestvedt og Marie Raae Stedje, sebastian.hestvedt@gmail.com, tlf: 95496869

Prosjektperiode

19.08.2019 - 25.05.2020

Status

12.03.2020 - Vurdert

Vurdering (1)

12.03.2020 - Vurdert

Det er vår vurdering at behandlingen av personopplysninger i prosjektet vil være i samsvar med personvernlovgivningen så fremt den gjennomføres i tråd med det som er dokumentert i meldeskjemaet 12.03.2020 med vedlegg, samt i meldingsdialogen mellom innmelder og NSD. Behandlingen kan starte.

MELD VESENTLIGE ENDRINGER
Dersom det skjer vesentlige endringer i behandlingen av personopplysninger, kan det være nødvendig å melde dette til NSD ved å oppdatere meldeskjemaet. Før du melder inn en endring, oppfordrer vi deg til å lese om hvilke type endringer det er nødvendig å melde:
https://nsd.no/personvernombud/meld_prosjekt/meld_endringer.html

NSD NORSK SENTER FOR FORSKNINGSDATA

NSD sin vurdering

Prosjekttittel

Non-functional requirements in agile projects management

Referansenummer

832425

Registrert

11.03.2020 av Sebastian Berg Hestvedt - hesseb18@student.kristiania.no

Behandlingsansvarlig institusjon

Høyskolen Kristiania - Ernst G. Mortensens Stiftelse / School of Economics, Innovation, and Technology / institutt for teknologi

Prosjektansvarlig (vitenskapelig ansatt/veileder eller stipendiat)

Asle Fagerstrøm, asle.fagerstrom@kristiania.no, tlf: 95075325

Type prosjekt

Studentprosjekt, masterstudium

Kontaktinformasjon, student

Sebastian Berg Hestvedt og Marie Raae Stedje, sebastian.hestvedt@gmail.com, tlf: 95496869

Prosjektperiode

19.08.2019 - 25.05.2020

Status

12.03.2020 - Vurdert

Vurdering (1)

12.03.2020 - Vurdert

Det er vår vurdering at behandlingen av personopplysninger i prosjektet vil være i samsvar med personvernlovgivningen så fremt den gjennomføres i tråd med det som er dokumentert i meldeskjemaet 12.03.2020 med vedlegg, samt i meldingsdialogen mellom innmelder og NSD. Behandlingen kan starte.

MELD VESENTLIGE ENDRINGER
Dersom det skjer vesentlige endringer i behandlingen av personopplysninger, kan det være nødvendig å melde dette til NSD ved å oppdatere meldeskjemaet. Før du melder inn en endring, oppfordrer vi deg til å lese om hvilke type endringer det er nødvendig å melde:
https://nsd.no/personvernombud/meld_prosjekt/meld_endringer.html