Heidi Puttonen

REQUIREMENTS RISK MANAGEMENT IN AGILE SOFTWARE DEVELOPMENT PROJECTS

JYVÄSKYLÄN YLIOPISTO
TIETOJENKÄSITTELYTIETEIDEN LAITOS
2018

TIIVISTELMÄ

Puttonen, Heidi Annika
Järjestelmävaatimuksien riskien hallinta ketterissä järjestelmäkehitys projekteissa
Jyväskylä: Jyväskylän yliopisto, 2018, 95 s.
Tietojärjestelmätiede, pro gradu -tutkielma
Ohjaaja(t): Tuunanen, Tuure

Erilaisten ketterien järjestelmäkehitys menetelmien kasvanut suosio on vaikuttanut perinteiseen tapaan ymmärtää järjestelmävaatimusten hallintaa. Ketterissä järjestelmäkehitys projekteissa vaatimusmäärittely prosessin täytyy mukautua kehitysympäristöön, jossa keskitytään pienempiin osakokonaisuuksiin valmiiden tuotteiden asemesta, joissa muutos on jatkuvaa ja missä asiakkaan odotetaan olevan vahvasti osallisena. Tämä vaikuttaa luonnollisesti myös järjestelmävaatimuksiin liittyviin riskeihin. Tämä pro gradu tutkielma selvittää kuinka järjestelmävaatimusten riskejä hallitaan ketterissä järjestelmäkehitysprojekteissa, sekä miten ketterän järjestelmäkehitys filosofian valinta vaikuttaa projektin järjestelmävaatimusten riskikenttään.

Tutkielman teoreettinen osuus tarkastelee järjestelmäkehitys alan ajankohtaista kirjallisuutta. Yhdistämällä tietoa järjestelmäkehityksen yleisestä riskien hallinnasta, vaatimusten riskien hallinnasta ja ketterästä järjestelmäkehityksestä, kirjallisuuskatsaus muodostaa kuvan järjestelmävaatimusten riskien hallinnan keskeisitä ominaisuuksista ketterissä kehitysprojekteissa: Näissä projekteissa myös itse vaatimusten riskien hallinnan täytyy sopeutua iteroivaan toimintatapaan ja jatkuviin muutoksiin. Tämä lisää myös riskien muutosalttiutta ja samalla tarvetta muokata riskianalyysin laajuutta sopimaan kulloiseenkin kehitys iteraatioon. Samoin kirjallisuuskatsaus paljastaa kuinka järjestelmävaatimusten riskien hallinnan pitää olla läpinäkyvää ja mahdollisuuksien mukaan ottaa mukaan myös projektin muita sidosryhmiä varsinaisen kehitystiimin lisäksi.

Empiirisessä osuudessa kirjallisuuskatsauksen tuloksia rikastetaan tapaustutkimuksella. Siinä Requirements Risk Prioritization -metodia käytetään vaatimusten riskien hallintaan ketterässä järjestelmäkehitysprojektissa. Tapaustutkimus paljastaa kuinka ylätasolla myös ketterät projektit kohtaavat saman tyyppisiä haasteita kuin perinteisempiä kehitysmenetelmiä käyttävät projektit. Tapaustutkimuksesta kuitenkin selviää neljä riskiryhmää, jotka tulosten mukaan ovat keskeisessä roolissa ketterissä projekteissa. Nämä riskiryhmät liittyvät asiakkaan rooliin ja käyttäjäkokemukseen, ketterän järjestelmäkehityksen prosesseihin, järjestelmävaatimusten laajuuteen ja projektitiimiin. Tapaustutkimus myös korostaa kuinka tärkeää ketterissä projekteissa on pystyä kommunikoimaan järjestelmävaatimusten riskeistä siten, että kaikki sidosryhmät pystyvät niitä ymmärtämään – erityisesti asiakas, jonka odotetaan olevan aktiivinen osallistuja projektissa. Tässä kommunikaatiossa ketterät projektit voivat hyötyä erilaisten työkalujen käytöstä, kuten Requirements Risk Prioritization menetelmä, jota tässä tapaustutkimuksessa testattiin

Yleisesti tämän tutkimuksen tulokset korostavat kuinka järjestelmävaatimusten riskien arviointi on haastavaa ketterissä projekteissa erityisesti koska sen tekeminen vaatii laajaa tietoa projektista itsestään sekä sitä ympäröivästä organisaatiosta. Henkilön tai henkilöiden, jotka kyseistä analyysiä tekevät täytyy päästä käsiksi tähän tietoon. Samaan aikaan ketterä prosessi pakottaa analysoijan keskittymään riskeihin pienemmässä laajuudessa kerrallaan kuitenkaan unohtamatta niiden riippuvuuksia projektin kokonaisriskiympäristöön. Vaikka järjestelmävaatimusten riskit ovat erittäin tärkeitä, niitä ei kuitenkaan koskaan voida arvioida ja hallita irrallisena projektin muusta riskikentästä.

Asiasanat: järjestelmävaatimusten riskit, vaatimusmäärittely, järjestelmäkehityksen riskien hallinta

ABSTRACT

Puttonen, Heidi Annika
Requirements risk management in agile software development projects
Jyväskylä: University of Jyväskylä, 2018, 95 p.
Information Systems, Master's Thesis
Supervisor(s): Tuunanen, Tuure

The grown popularity of agile development methods has affected the traditional understanding of requirement management. In these kinds of projects, requirement engineering process needs to adapt to an agile environment where the focus on development is in smaller iterations instead of ready products, changes happen often, and a customer is expected to be highly involved in the process. This naturally effects on requirement related risks that agile projects may face. This Master's thesis investigates how the requirements risk management is done in agile projects and how selecting an agile development method affects requirement risks.

The theoretical part of this study reviews contemporary research literature from the IS field. By combining knowledge from IS development risk management, requirement risk management, and agile development it was possible to summarize some key attributes of requirement risk management in agile projects. In these projects also, requirement risks management needs to adapt to the iterative development cycles and constant changes. This increases the volatility of the requirement risks as well as the need to adjust the scope of the risks assessment to fit the size of the development scope of any given time. Similarly, the prominent role of the customer in agile IS projects emphasize how also the requirements risk management should be transparent and involve, not only the project team but also different project stakeholders.

The empirical part of this study further enriches these finding by testing the Requirements Risk Prioritization method in a case study, in an agile project where it was used as a tool to identify and prioritize requirement related risks. The case study revealed how in higher level agile projects face similar risks as projects where more structured methods are used. However, this study identified four other types of risks that seem to play an increasingly important role in the agile projects. These types related to customer's role or user experience of the system, agile development process itself, requirements scope as well as projects team. The case study also highlights the importance of communication about requirement related risks in a way that it is understandable for all the stakeholder – especially when the customer is expected to be involved. Lastly, it showcases how agile projects could benefit from having a tool to support this kind of communication.

Results of this study also show how assessing requirements risks is challenging in agile projects since the people doing that needs to have a wide range of knowledge from the project as well as the organization where it exists. This

individual or individuals should have also an access to that information. At the same time the agile process forces them to be able to do a similar assessment for risks on a smaller scale while not still forgetting to consider requirement risks as a part of project's overall risk environment: No matter how important requirements risks are, those never exist in a vacuum and should not be managed as such.

Keywords: requirement risks, agile development, requirement engineering, ISD risk management

FIGURES

TABLES

TABLE OF CONTENTS

1 INTRODUCTION

Requirements engineering is an important part of information system (IS) development. It is the phase when customers', users' and different stakeholder groups' goals and ambitions toward the system are captured. This information is combined into system requirements which describe what is needed from the system to fulfill the needs of the end-users. One could say that requirements serve as a blueprint for the IS implementation, ensuring that the final solution will successfully fulfill the purpose it was created for. Naturally, having flaws in this kind of plan can significantly hamper the quality of the end result.

It is unfortunately relatively common knowledge in the IS field that development projects tend to struggle in to achieve their goals. Many IS development projects have challenges with keeping their budgets and schedules or delivering a satisfying product to the end user. Some studies argue that almost half of the IS projects are considered to be some level of failures. One culprit for this has been found from the requirements engineering process: Both research literature and practical experiences have revealed that shortcomings in the requirement engineering process play often a notable role in IS project failure. This is why studying common pitfalls in the requirement engineering process is important. It will enable IS practitioners to understand how requirements should be managed so that we are moving towards more successful IS development projects.

Avoiding common pitfalls, in other words, risk management is a critical part of IS development project management and well-studied topic in IS research. Requirement engineering as part of the IS development presents its own set of risks that are often caused by challenges in collecting, analyzing, designing and finally implementing IS requirements appropriately. These so-called requirement risks can be a result of a wide range of reasons and they can appear at any time throughout the project's lifecycle. In addition, requirements risks are not just caused by internal issues inside the project team but those can appear also because of external, organization-wide topics. Examples of such risks are changes in organizational structures, strategic direction as well as turnover of the key project team members. Varying nature of these risks and impact they can have for

the projects means that managing requirements risks is by no means a simple task for the project team.

Current research literature represents a relatively good understanding of the IS risks in general. There are plenty of studies which offer tools and best practices for the IS risk management, for instance (Boehm, 1991; Barki, Rivard & Talbot, 1993; Keil, Cule, Lyytinen & Schmidt, 1998; Kontio, 2001; Lyytinen, Mathiassen & Ropponen, 1996). Surprisingly, the same cannot be said for the requirement risks management. Those studies which identify requirements related risks as a separate topic from other IS project risks, mainly focus only on identifying typical requirement risks. There are very few papers that share insights into how to manage requirements risks in practice or provide tools for project managers to do that in real life projects.

One exception is by Tuunanen, Vartiainen and Mehdi (2018) who defined a method that helps the project team to prioritize requirements related risks. This method is called Requirements Risk Prioritization method (RRP method). It offers lists of typical requirement risks for different requirement engineering process phases. These checklists can help project teams to identify risks as well as avoid overlooking topics they would have missed otherwise. Writers claim that by evaluating the impact of identified requirements risks, the project teams can recognize the most important ones and prioritize them accordingly. The method also provides risk resolution techniques for different risk types. Writers argue that even though the method has not been created only a specific development philosophy in mind, it could be used in all kinds of IS development projects.

After the emerge of agile methods, the traditional way of seeing software development has been challenged by short development cycles, iterative development mentality, and quick releases. The selected development method has naturally impact on how requirements are managed. Differences in requirement engineering process in agile and more traditional development methods is covered relatively well in the current research literature. These differences can affect, for example, how requirements are gathered, when those are collected and how much the end user or customer is involved in the process. All these differences affect naturally also the risks the project may or is likely to face.

As different kinds of agile adaptations have grown in popularity in IS industry so has the need to understand how such methods affect IS project risk environment and through that also requirement risks. Fundamentals of agile development, such as high customer involvement, emphasizing teamwork, less focus on documentation and embracing the change create their own twist for the already easily overlooked requirement risks. This can mean, for example, that as requirement engineering process needs to adopt iterative development cycles, requirements are not gathered fully at once and changes can appear at any point in the process. This, in turn, can increase the ambiguity of requirements or the likelihood of missing critical requirements. Similarly, as collaboration is highly valued in agile projects, the risk of not sharing the correct level of knowledge

inside the development team can significantly complicate the effective communication of requirements among stakeholder groups.

Understanding what requirement related risks in the agile development projects are and how to manage them is important for the IS projects today: Accurate identification and correct prioritization of requirement risk could positively impact project general risk management and success. However, a limited amount of information is available to support agile projects teams to accomplish this. To fill in this knowledge gap, the goal of this master's thesis is to shed light on this topic. This master's thesis aims to understand what requirement related risks are in agile IS development projects and how those should be managed. Secondly, this thesis investigates a real-life case project where RRP method is tested as a tool to assess requirement related risks hence giving more visibility how this process could be supported by the method provided by IS research. The research topic is divided into the following research questions:

- How requirement risks management is done in agile software development projects?

Furthermore, this main research question is divided into the following subquestions:

- Does the selected development method affect ISD projects risk environment?
- Does Requirements Risk Prioritization method help requirements risks management in an agile project?
- Are there typical requirement risks in agile projects?

This master's thesis first forms the theoretical background for the research topic by reviewing contemporary IS research literature on the research topic. The goal of the literature review is to form an understanding of requirement risks management in agile development projects. This is done by combining knowledge about how risks management is done in IS development projects in general, how requirement risks are considered in relation to that and how both topics can be considered from the agile development point of view.

The empirical part of this study is conducted as a single case study. There RRP method is tested in practice as a tool for requirement risk assessment. Research case gives a glance to the practicalities of a real-life requirement risks management process in the agile project environment. This further enriches and deepens the understanding of this process which combined with literature review findings can help to answer research questions.

2 REQUIREMENTS ENGINEERING IN AGILE SOFTWARE DEVELOPMENT

> "Evolution favors those that operate with maximum exposure to environmental change and have optimized for flexible adaptation to change. Evolution deselects those who have insulated themselves from environmental change and have minimized chaos and complexity in their environment" (Schwaber, 1997)

Software development is never done in a vacuum. It is a complicated process, which is affected by the organizational environment, time and available resources as well as different stakeholders. The question of how software development should be organized in order to deliver better, faster and more cost-effective solutions has been a hot topic in the ISD field for decades. The fast development of the field itself and the way information systems are now involved in almost all aspects of our lives, mean that developing such systems requires the ability to respond to this dynamic environment. The idea of coping with constant changes is a key principle in most of the agile software development methods. Quickly evolving requirements, rapid change of market environment, changing technologies and time constraints are just a few examples of the reality of software development today and at the same time the development environment where agile methods usually perform well (Highsmith & Cockburn, 2001).

Requirements engineering is one of the key processes that guide the direction of a system development project. It is naturally highly affected by the development environment and the development method in use. Some agile principles such as high customer involvement, frequent feedback from stakeholders and flexible attitude towards documentation have challenged the traditional view of requirements engineering (Cao & Ramesh, 2008; Kassab, 2014). This chapter gives an overview and background of agile software development and how requirements the engineering process has evolved as part of it.

2.1 Agile software development

In traditional, often referred as plan-based or heavy-weight view of software development, the development process is considered to be linear and predefined. This "engineering-approach" claims that problems can be fully specified and optimal solutions for them exist. It also assumes that all sources of variations are identifiable and possible to be eliminated by refining the process itself. The traditional methods commonly rely on upfront requirement definitions, detailed planning, and heavy documentation. (Dybå & Dingsøyr, 2008; Jiang & Eberlein, 2009; Nerur, Mahapatra & Mangalaraj, 2005). Software development methods, which are based on these concepts usually struggle when they face unexpected changes or outputs from any of the intermediate process phases (Schwaber, 1997).

Two prominent examples of traditional methods are the Waterfall model that is based on Winston W. Royce's (1970) article and the Spiral model which was first described in the paper by Barry Boehm (1986) (Jiang & Eberlein, 2009).

To improve software development methods to better answer to the constant changes of both development environment and system requirements, experts and experienced practitioners have come up with methods labeled as *agile* (Dybå & Dingsøyr, 2008). Agile software development includes a wide range of different methods, which typically promote similar concepts such as suppleness, nimbleness, dexterity and responding to changes (Beck et al., 2001; Cohen, Lindvall & Costa, 2003; Erickson, Lyytinen & Siau, 2005). One of the first known agile methods is the Dynamic System Development Method (DSDM), followed by Extreme programming (XP) (Larman & Basili, 2003). Other well-established agile methods are for example Crystal methodologies, Feature-driven development, Lean software development, and Scrum. (Dybå & Dingsøyr, 2008).

The term "Agile Method" is not a new one. It has been around for more than a decade and many of its key principles even longer. (Cohen et al., 2003; Dybå & Dingsøyr, 2008). In the literature roots of the term *agile* can be traced back to the 1990s manufacturing industry's lean development or lean manufacturing (Jiang & Eberlein, 2009). Although agility has a relatively long history, there is no full agreement on what it really is (Qumer & Henderson-Sellers, 2006). According to Wong and Whitman (1999), to be agile means the ability to effectively respond to unexpected and rapid changes with flexibility. Hendersson and Sellers (2005), in turn, argue that an agile entity gains knowledge and experience from its external and internal environment and improves. Qumer and Hendersson-Sellers (2006) propose in their article the following definition:

> "Agility is a persistent behavior or ability of a sensitive entity that exhibits flexibility to accommodate expected or unexpected changes rapidly, follows the shortest time span, uses economical, simple and quality instruments in a dynamic environment and applies update prior knowledge and experience to learn from the internal and external environment."

All these different definitions approach *agility* from slightly different angles. They all still seem to share similar core ideas or *values.* These agile values and principles have been summarized in the Agile Manifesto. It and its content are discussed more in detail in the following chapter.

2.1.1 Agile values and principles

The starting point for agile software development "movement" is usually said to be Manifesto for Agile software development. This declaration was a result of a summit where 17 software development gurus got together in Utah, Snowbird in 2001. They shared a common goal to address known issues that troubled traditional software development such as lengthy and stiff development projects, which ended up exceeding their budgets and didn’t satisfy the end users. (Agile

foundations: Principles, practices and frameworks, 2015). The resulted manifesto consists of four key values (figure 1) and twelve supporting principles (table 1).

FIGURE 1 Four values of Agile Software (Beck et al., 2001)

TABLE 1 Agile principles (Agile Foundation: Principles, practices and frameworks, 2015)

Principle 1	Our highest priority is to satisfy the customer through the early and continuous delivery of valuable software.
Principle 2	Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
Principle 3	Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
Principle 4	Business people and developers must work together daily throughout the project.
Principle 5	Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done.
Principle 6	The most efficient and effective method of conveying information to and within a development team is a face-to-face conversation.
Principle 7	Working software is the primary measure of progress.
Principle 8	Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
Principle 9	Continuous attention to technical excellence and good design enhances agility.
Principle 10	Simplicity--the art of maximizing the amount of work not done--is essential.
Principle 11	The best architectures, requirements, and designs emerge from self-organizing teams
Principle 12	At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

The first value, focusing more on individuals and interactions over the tools and processes, means emphasizing the relationship and communality of the people working on the software project instead of institutionalized processes and development tools. (Abrahamsson, Salo, Ronkainen & Warsta, 2002). This means that in place of standardizing people to the processes, agile methods aim to adapt processes to the people by capitalizing on the strengths of each team and individual. This is achieved by promoting interactions and communication among the project team members and having the customer in the core focus. An agile project team is typically self-organized and collaborates intensively within and across organizational boundaries. (Cockburn & Highsmith 2001). The impact of this is visible as different collaborative practices included in different methods like pair programming, co-location etc.

The second value highlights another common practice in most of the agile methods, which is the continuous delivery of working software (Abrahamsson et al., 2002; Beck et al., 2001; Jiang & Eberlein, 2009). This means producing new releases in frequent intervals. Usually, this is achieved by building a software in several iterations. Each iteration is an independent "mini-project" that includes a set of activities such as requirement analysis, design, development, and testing. After each iteration follows an iteration release where a stable system, or a part of the system, is released. Several releases may be needed to finalize new features or a finished product. In agile the working software is typically considered to be the best measurement for overall project progress (Beck et al., 2001; Larman, 2004).

Over documentation has been one of the detrimental problems in traditional software development. Extensive documentation is challenging to keep up to date and both generating as well as maintaining the documentation takes a lot of time. Some even argue that the only documentation one can trust is the code. (Paulk, 2002). The documentation burden in agile methods is kept to the minimum by making sure that code itself is simple, straightforward and technically as advanced as possible (Abrahamsson et al., 2002; Nerur et al., 2005). Overall, agile methodologies favor more tacit than explicit knowledge and the knowledge is transferred by human interactions rather than by formally documenting it (Paulk, 2002).

Customer collaboration and focus are in a central role in agile software development methods. As Highsmith and Cockburn (2001) state: "Delivering customer value (however the customer defines it) measures the success of the agile project". This means for example that co-operation between the customer and the developer is preferred over strict contract negotiation (Abrahamsson et al., 2002). In some methods such as XP, the customer (if not already internal one) is preferred on site (Manzo, 2002). When the customer is involved in the development project, they can follow up the progress more closely and have a clearer understanding of what is coming out from the pipeline and when. This mitigates the time spent on reporting and documentation of the project progress (Cockburn, 2002). The customer is also able to provide feedback, answer questions and react

faster if the project is, for some reason, starting to stray from the correct path. Paulk (2002) even uses the phrase "*agile relationship*" to describe the collaboration between a customer and a development team. He also mentions the inability to establish such a relationship as one of the significant barriers to adopt agile methodologies.

The fourth value means that all the participants in an agile development project are prepared to make changes and react to changed situations during the entire life cycle of the project. In traditional, plan-driven development any change from the original plan triggers heavy alterations in project management processes. Because of this, changes are very hard and expensive to make. In agile methods, the incremental approach to the process allows changes during the whole project. It also means an individual change can be handled more flexibly. Continuous feedback from the customer and constant re-prioritization of the development tasks ensure that any needs for change are spotted early on and the effects they have on the project resources are more manageable. (Abrahamsson et al. 2002; Nerur et al. 2005; Higsmith 2002).

Nerur, Mahapatra and Mangalaraj (2005) summarize in their article topics which usually separate traditional and agile software development. This summary can be found from table 2. In addition to the previously mentioned agile values, they also emphasize differences in organizational factors. Agile methods benefit from a management style that supports collaboration and lacks heavy control. The agile organization itself is usually very organic, less bureaucratic and inclusive. Again, an ideal agile team encourages flexible roles and changes in responsibilities compared to more traditional approaches where individual specialization is valued high.

There is naturally some criticism against agile methods. Firstly, some arguments state that agile methods don't fit for all types of development projects - especially when the developed system is in life-critical or high-reliability context. (Paulk, 2002). Similarly, as it is critically different to manage a battlefield and managing warehouse logistics, the agile methods seem to shine on projects that involve exploratory actions (volatile requirements, experimental technology etc.) whereas traditional methods are more suitable for projects involving repeatable processes and high predictability (Highsmith, 2002). Previous literature has also suggested that agile methods are more suitable for small teams than larger projects in complex organizations (Dybå & Dingsøyr, 2008; Poole, Murphy, Huisman & Higgins, 2001).

TABLE 2 Traditional and agile development method comparison (Nerur et al., 2005)

	Traditional	Agile
Fundamental assumptions	Systems are fully specifiable, predictable, and can be built through meticulous and extensive planning.	High-quality, adaptive software can be developed by small teams using the principles of continuous design improvement and testing based on rapid feedback and change.
Control	Process-centric	People-centric
Management Style	Command-and-control	Leadership-and-collaboration
Knowledge management	Explicit	Tacit
Role Assignment	Individual-favors specialization	Self-organizing teams -encourages role interchangeability
Communication	Formal	Informal
Customer's role	Important	Critical
Project Cycle	Guided by tasks or activities	Guided by product features
Desired Organizational Form/Structure	Mechanistic (bureaucratic with high formalization)	Organic (flexible and participative encouraging cooperative social action)

Secondly, researchers have noticed some issues with the common agile practices: In reality, customer availability for development projects, especially on site, is not always possible. Lack of written documentation and focus on tacit knowledge may lead to problems when new people join the project team. Agile methods may also lead to a situation where developers should be more generalists than specialists. These people are not easy to find when at the same time, agile methods encourage the use of new, cutting-edge technologies. (Paulk, 2002). As in everything, finding a middle ground is important. Selecting what best suits the development project and organizational culture seem to be the key to the ISD project's success. This has led to a situation where many organizations decide to adopt a development method that is a mixture of the plan based and agile (Tuunanen et al., 2018).

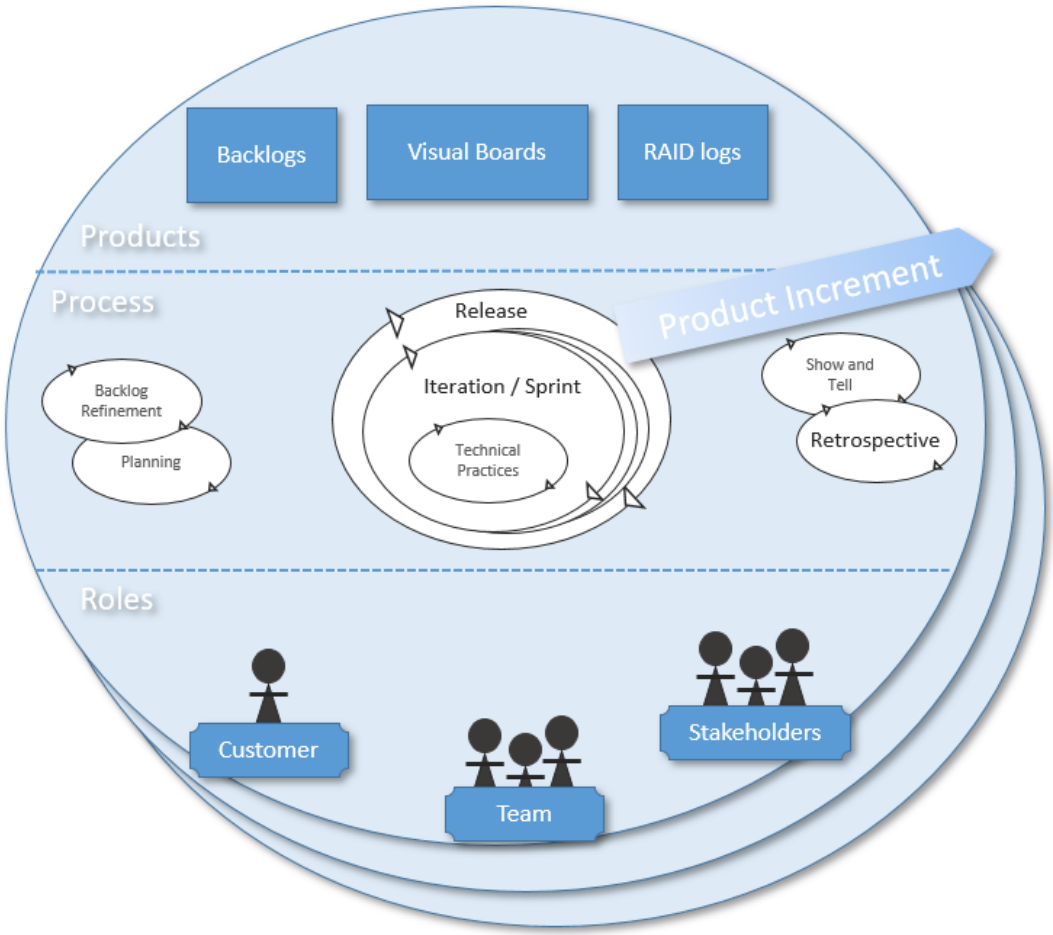
To summarize, the “agile family” includes different development methods that all share the same key values. They are characterized by a collaborative and communication driven development environment where the value is delivered in small increments fast and cost-effectively without losing time on extensive documentation and unnecessary following of predefined plans. It is important to note that all the agile values are enabling other ones. Paulk (2002) summarizes this accurately in the following way: "An agile method that ignores customer collaboration and incremental development would almost certainly fail. Agile practices are synergic, and the success of agile methodologies depend on the emergent properties of the set of practices as a whole."

2.1.2 The generic agile development model

Each of the agile development methods has their own take on how to deliver information technology products. Selecting which agile framework should be used depends a lot on the project and the environment where it is developed. Measey (2015, 38-42) offers a simplified description of a generic agile model that helps to showcase how agile processes aim to reach their goals.

Figure 2 illustrates this generic agile process. In this model, everything revolves around product backlog and development cycles. Customer, as one of the key actors in the process, continually evolves the product backlog. The backlog is a collection of requirements for the developed system, for example, new features or functionalities. Backlog management is supported by the development team which may consist of developers, ISD project manager, product owner etc. Backlog content is prioritized and organized to the development iterations (sprints). Which in practice means deciding what is developed in which development cycle. The team then delivers product increments according to the schedule. After one development iteration, the next one starts, and the cycle begins again. (Measey, 2015, 38-42).

FIGURE 2 Generic Agile process (Measey, 2015, 38)



The planning activities happen in all layers. First, backlog content is planned with the customer according to the requirements derived from their needs and goals. Secondly, each sprint/iteration content is planned by prioritizing the content of the backlog and selecting features to be delivered. Lastly, each release is planned, and a suitable amount of iterations are included in the released version of the product. Agile methods encourage frequent releases and one release can include content from one or more sprints. Sometimes if the agile process is scaled, planning actions spread across the projects, for example, when several projects share a similar release schedule. (Measey, 2015, 38-42)

Daily meetings or stand-ups are held within the sprint. These are used to keep everyone involved up to date and to guide the day-to-day activities in the project. The current work and status of the releases or iterations are visualized in different visual boards. These boards can be digital or, for example, made from post-it notes. Their purpose is to offer transparency of the current development status to everyone involved. RAID logs are used to track things which may affect later development cycles (risks, assumptions, issues, and decisions).

The agile process is fueled by feedback loops. These loops are included in the process in form of reviews. 'Show and tell' sessions are organized to review deliverables of different process phases. Examples of such reviews are release and sprint reviews. The feedback from these meetings is used to improve the solution, to recognize possible challenges and provide the basis for the next iterations. Another form of feedback is gathered from retrospectives (discussed more detail in chapter 2.3.2), which are also performed in selected times of a process. Information from retrospectives is used to improve the current way the team works. (Measey, 2015, 38-42).

Figure 2 showcases also four key actors in an agile process: customer, the team, stakeholders and agile lead. 'Customers' are the ones who "own" the product. They make the final call of what will be done and what is the priority of those deliverables. They hold and know the vision of the product and should be able to answer the question 'Why are we doing this?'. Customer role is not a simple one and in an agile process the customer ideally is willing to be involved, actively participating in the decision making, is knowledgeable enough to make correct decisions and is available for the project team. (Measey, 2015, 38-42) Customer involvement is also emphasized in the agile values and principles (Figure 1, table 1) (Fowler & Highsmith, 2001).

The team is the actor that keeps the process rolling. It has a responsibility of deciding how to work to best achieve the current goals. The team works in close collaboration with the customer to refine requirements and typically divide bigger entities into smaller tasks. The team is responsible also for estimating how much efforts are needed for each iteration and who is focusing on which tasks. An agile team is usually self-organized. This means that the team has often the freedom to choose their ways of working. The teamwork is supported by the agile lead who is responsible for enabling the team's way of working. Agile lead facilitates and enables different agile processes. A person holding this role coaches teams to continuously improve their performance and spot inefficiencies. (Measey, 2015, 38-42)

Stakeholders are those parties in the organization who have an interest in the developed solution and are actively involved in the ISD project. Their responsibility in the agile process is to ensure that the needs and interests of the group they advocate are accurately represented. Even though stakeholders do not necessarily possess the same kind of decision-making power as the customer does, they can still positively or negatively impact the ISD project. Identifying correct stakeholder groups is important for the project success (Measey, 2015, 38-42).

2.2 Requirements engineering

The success of a software system depends on how well it meets the purpose for which it was intended. In other words, how well real-life needs, or goals are supported by different functionalities of the designed system. Requirements engineering (RE) is a branch of software engineering that is focused on this aspect of

development processes. RE processes include all the activities involved in identifying stakeholders, discovering their needs, documenting and maintaining them and, at the same time, understanding the constraints of the software system and architecture. (Kotonya & Sommerville, 1998; Nuseibeh & Easterbrook, 2000; Zave, 1997) RE is a human-centered and multidisciplinary process including tools and techniques not just from the computer science, but also, for example, cognitive psychology, sociology and linguistics (Nuseibeh & Easterbrook, 2000).

In order to understand RE as a process, one needs to answer the question: What is a requirement? Requirements describe how the system should behave, what the constraints on the system's operation are or they specify the system's attributes or property. Requirements are derived from the different stakeholders of the developed system. These stakeholder groups are for example end users, business management and development team. (Wiegers & Beatty, 2013, 5-9; Kotonya & Sommerville, 1998). According to IEEE (1990) standard, a requirement in system development means:

> "(1) A condition or capability needed by a user to solve a problem or achieve an objective. (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification or other formally imposed documents. (3) A documented presentation of condition or capability as in (1) or (2)."

In short, requirements are base for understanding what the system should do hence being a guide for what should be implemented to the system.

In many occasions, the requirements are divided into functional and non-functional requirements (Glinz, 2007). Functional requirements describe actions that system or end-product must be able to perform (IEEE standard glossary of software engineering terminology 1990; Kotonya & Sommerville, 1998). It also explains inputs to the system (stimuli) and outputs from the system (responses) and how the system behaves between them (Davis, 1993).

The definitions for non-functional requirements vary in the related literature. Glinz (2007) summarizes in his article:

> "A constraint is a requirement that contains the solution space beyond what is necessary for meeting the given functional, performance and specific quality requirements. An attribute is a performance requirement or a specific quality requirement (…) A non-functional requirement is an attribute of a or a constraint on a system."

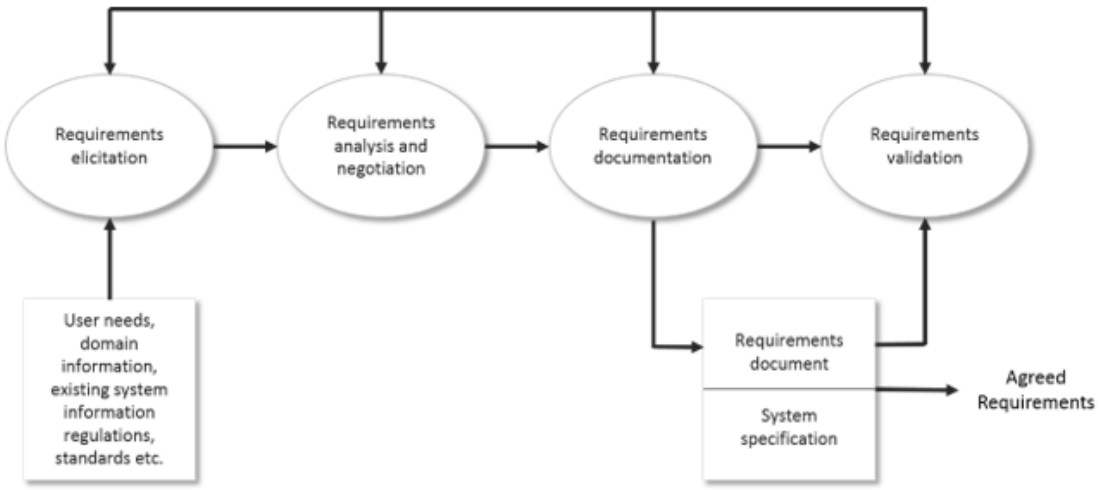
This means that these non-functional requirements don't focus on functionalities of the system but define the overall qualities or attributes of the resulting system. Non-functional requirements can be related to for example security, safety, usability and reliability of the system. (Kotonya & Sommerville, 1998).

2.2.1 Requirement engineering process

Requirements engineering or requirements management is traditionally regarded as a front-end activity that happens in the early stages of the development

process, but it may also play a role in the implementation phases and even in maintenance (Kotonya & Sommerville, 1998; Nuseibeh & Easterbrook, 2000). Requirement engineering as a process can be structured in different ways and different organizations tackle it in radically varying ways. Kotonya and Sommerville (1998) divide the process into the following activities: requirement elicitation, requirement analysis, and negotiation and requirement validation. In figure 3 these activities are displayed in coarse-grain activity model in relation to another. Writers have chosen the coarse-grain activity model because it shows the major activities in a particular process and their approximate sequencing. In reality requirement activities rarely follow a simple sequential pattern but are more likely to overlap each other in different phases of the software development.

FIGURE 3 Coarse-grain activity model of the requirements engineering process (Kotonya & Sommerville, 1998)



Eliciting requirements happens in the first phase of requirement engineering process in which the needs and goals of the different stakeholders (clients, developers, users etc.) are collected or captured. Typically, this means finding the problems that need to be solved. This forms the boundaries that define the context of the developed system. (Nuseibeh & Easterbrook, 2000; Paetsch, Eberlein & Maurer, 2003). Although other strategies are possible, requirements elicitation is often done either with a product-centric or usage-centric approach. The product-centric strategy aims at defining features that focus on reaching the business success. The usage centric approach derives necessary system functionalities from user goals. (Wiegers & Beatty, 2013, 48-49)

Different methods can be used for requirement elicitation like interviews, use cases /scenarios, observations, focus groups and prototyping. The goal of the different techniques and tools is to improve the clarity and reduce the ambiguity of the requirements. (Chua, Bernardo & Verner, 2010). Eliciting requirements re-

quires knowledge of the organization, the application domain and business processes - an overall understanding of the environment where the system will be used. Other terms used to describe this phase are requirement discovery and requirement acquisition. (Kotonya & Sommerville, 1998).

When requirements are collected from different stakeholder groups it is almost inevitable that some conflicts will arise. Requirements analysis and negotiation are activities during which elicited requirements are elaborated further to establish agreed set of requirements that are consistent and complete. During this phase, requirements are checked for their necessity, completeness, feasibility, and consistency. This means that requirements are actually presenting necessary functionality, they don't contradict each other, no constraints are missing, and requirements are feasible to be implemented when considering the budget and schedule of the development project. The goal of this phase is to reach a rich and precise understanding of each requirement. (Wiegers & Beatty, 20131 121; Paetsch et al., 2003).

Requirements analysis and negotiation phase is very closely related to requirements elicitation. It is also the phase when possible problems are discovered. This means trying to recognize missing, overlapping, conflicting, ambiguous and unrealistic requirements. In this phase problems in the initial requirements are discussed and compromises are formed to achieve a solution that will cater to all project stakeholders. (Kotonya & Sommerville, 1998; Ahmad, 2008). During negotiations, additional requirements may arise. This may result in updating previous requirements and re-entering the requirement elicitation phase. (Wiegers & Beatty, 2013, 139; Ratchev, Urwin, Muller, Pawar & Moulek, 2003).

The purpose of requirements documentation is to provide a way to communicate system requirements between different stakeholder groups (customer, management, developers etc.). In this phase, the goal is to store the collected requirements knowledge in well-organized fashion. Usually, requirements are documented in a formal way, but there is no standard name or format for the document and practices may vary from one organization to another. Some commonly used terms for the document are requirements document, functional specification and requirements specification. (Paetsch et al., 2003).

Requirements themselves are usually written in a natural language accompanied by diagrams, equations, and models. (Kotonya & Sommerville, 1998; Zhang et al., 2010) According to Paetsch et al. (2003), qualities of a good requirements document are: unambiguous, complete, understandable, consistent, correct and feasible. The level of detail in which the requirements are documented varies. High-level requirements are used for communication with the customer and they focus more on describing user's actual need. Later these descriptions can be refined and adjusted into more detailed and technical specification. (Zhang et al., 2010). The most important quality of the selected documentation format is that it is suitable for the use, review, and comprehension of the intended audience. (Wiegers & Beatty, 2013, 181-186).

The requirements document can be structured in many ways. The structure varies depending on the organizational practices, the type of system which is being developed and the level of detail needed. According to Kotonya and Sommerville (1998) the requirements document usually describes at least the following:

- The service and functions which the system should provide
- The constraints under which the system must operate
- Overall properties of the system
- Definitions of other systems which the system must integrate with
- Information about the application domain of the system
- Constraints on the process used to develop the system

Some large organizations like IEEE and US Department of Defense have created standards for requirements document (Kotonya & Sommerville, 1998).

The last activity in the requirements engineering process, requirements validation, is a phase when requirements are certified. This means answering the question: Do requirements really represent an acceptable and valid description of the developed system? The activity of requirements analysis has a lot in common with the validation phase. Both involve analyzing requirements, judging their representation of the real stakeholder need and recognizing possible problems and conflicts. The key difference between the two phases is the completeness of the requirements that are evaluated. When in analyzing phase, requirements are usually incomplete and unstructured as the goal of the analysis is to answer to the question: Do we have the right requirements? Requirements validation is focused on fine-tuning the requirements document from where all the known inconsistencies and incompleteness have already been removed. The focus question of the requirements validation is: Do we have the requirements right? The output of the requirements validation process is a list of system requirements that are then used to implement the desired solution. Ideally, requirements validation process produces also a list of problems affecting the current requirements setup and agreed on actions how to deal with them. (Kotonya & Sommerville, 1998; Paetsch et al., 2003)

2.2.2 Requirements development techniques

Collecting requirements from the correct stakeholder groups and transforming them eventually to system features is anything but a simple task. There are several examples in the literature about the common pitfalls in the requirements engineering process. For instance, Kauppinen, Savolainen, Lehtola, Tohonen, and Davis (2009) identified in their article challenges that plague IS development which could be addressed with the requirements engineering process. Pitfalls they mention are generalizing users to too few separate user groups, not supporting customer's processes well enough, releasing deficient features due to the

schedule pressure, adding too many features or fine-tuning individual feature too much or general lack of overall picture in the development.

To avoid these pitfalls several different requirement development techniques have been generated. Requirements development technique means those methods which can be used in all the phases of the RE process. Those are used to achieve a goal of that specific RE process phase. Mathiassen, Saarinen, Tuunanen and Rossi (2007) identified from the previous literature in their article four different trends in requirements development techniques: Requirements discovery, requirements prioritization, requirements experimentation and requirements specification. These techniques are listed in table 3. Writers note that not all the techniques can be identified to belong to only one technique family but can utilize principles from several of them.

TABLE 3 Requirements development techniques (Mathiassen, Saarinen, Tuunanen & Rossi, 2007)

Technique	Definition
Requirements discovery	Customer-centric and based on identification or prediction of customer needs
Requirements prioritization	Resource-centric and based on analysis of and the choice between identified requirements
Requirements experimentation	Software-centric and based on iterative processes involving end users
Requirements specification	Documentation-centric and based on abstraction and textual or graphic representation

With discovery techniques, the focus is on user or would-be users. These techniques facilitate identifying and explication of requirements without having a rigid approach to documentation. In these techniques communication is usually less about software artifacts but instead, these techniques rely on having a personal contact between user and developers. Discovery techniques emphasize understanding and predicting the needs and beliefs of different user groups. These techniques borrow ideas from approaches used in marketing (Delphi, laddering etc.) and take advance of group dynamics like focus group interviews, where requirements are discovered through stakeholder group interactions. (Mathiassen et al., 2007).

Prioritization techniques are based on the fact that software development has limited resources like time, costs, technologies etc. These techniques apply various types of decisions, support, and analysis to help to focus the requirements development towards the right goal. One example is the critical factor technique that guides the project to focus on the most important requirements from the top management's point of view. (Mathiassen et al., 2007).

On the opposite for discovery techniques, experimentation techniques use design and software artifacts as a key tool to communicate with users. These software-centric approaches differ in the level of user involvement, but in order to be used, there has to be some representation of the software available. This may mean using prototypes or preliminary versions of the software components and then using user feedback to guide further requirements development. Example of experimental technique can be workshops where developers and users jointly discuss and debate about requirements and solve problems. This kind of practices has been widely used in the software development world. (Mathiassen et al., 2007).

The last technique family in Mathiassen at al. (2007) categorization is specification techniques. With them, the focus is on documentation. These techniques are distinguished from others by focusing on providing an agreed upon and an explicit basis for the future development. Some of these techniques rely on decidedly formal concepts and precise notation schemes. One example is Box structures that offer a representation of requirements with execution semantics. Not all specification techniques are formal tough, and those that are not, usually utilize a natural language that can be still augmented with more formalized notations and concepts. With specification techniques, requirements are acquired from the users, by studying the existing software or developing a graphical presentation of requirements. Entity-relationship modelling is one example of the specification techniques. (Mathiassen et al., 2007).

2.3 Requirements engineering in agile software development

There has been a debate about what kind of role RE plays in agile software development. Ramesh, Cao and Baskerville (2010) even state that the term 'requirements engineering' is avoided in the agile community because it is often taken to imply heavy documentation with significant overhead. They also mention that many agile methods advocate moving into coding without centralized requirements engineering in the design phases of the software project. Instead, agile methods such as Extreme programming support RE processes throughout the development lifecycle in small stages (Cao & Ramesh, 2008; Paetsch et al., 2003).

2.3.1 Differences in requirement engineering in agile and traditional development projects

One of the biggest differences that agile methods bring to traditional RE is the iterative nature of the development process itself. Even though the traditional RE also includes the idea of not following the sequential process (Kotonya & Sommerville, 1998) and going back to previous phases, this is even more prominent when considering RE with agile methods. In agile RE all the traditional RE phases blend together and clear distinction between them is hard to make. RE activities

may follow each other in the same order as in traditional approaches, but those activities are repeated in each iteration: Requirements are elicited, analyzed and negotiated, documented and validated continuously. This means that efforts put to the requirement engineering are frequent, but less extensive at the same time. (Wiegers & Beatty, 2013, 383; Paetsch et al., 2003).

Cao and Ramesh (2008) summarize the differences between the RE process phases in traditional and agile system development (Table 4). In the following table 4 it is concluded how the requirement elicitation, analysis, and negotiation, documentation and validation are affected by agile principles like incremental development philosophy, high customer involvement and less focus on documentation.

TABLE 4 Traditional and agile approach to requirements engineering (Cao & Ramesh 2008; Dybå & Dingsøyr 2008)

Phase	Agile requirements engineering	Traditional requirements engineering
Elicitation	Requirements evolve during the project lifecycle and new requirements are discovered throughout the development process	Discovering requirements is usually done in the early stages of the project. All requirements are discovered upfront
Analysis and negotiation	Focus on refining, changing and prioritizing requirements iteratively	Focus on resolving conflicts
Documentation	No formal documentation	Formal documentation that contains detailed requirements
Validation	Face-to-face communication	The consistency and completeness of requirements document

Typically, in an agile project, detailed requirements are rarely pre-defined. As requirement tend to emerge during the development process, RE processes are forced to be flexible and adaptive. High level of elicitation activities does usually take place at the project's beginning similarly to traditional methods, but those only provide a general level of information about the critical software features. (Cao & Ramesh, 2008; Zhang at al., 2010) Frequent interaction among stakeholders is used to ensure that requirements can evolve with a lesser investment of time. The traditional RE, in turn, aims at recognizing all the requirements before the development starts. (Elghariani & Kama, 2016)

Heavy customer involvement in agile project affects all the RE phases. From the Cao's and Ramesh's (2008) summary (table 4), it is visible how this agile value has its effects in every step of RE process: Face-to-face communication with the

customer is preferred over extensive documentation and constant planning and extreme prioritization is done together with the customer. (Cao & Ramesh, 2008; Zhang et al., 2010)

A customer plays a significant role also in negotiation and analysis of requirements; particularly as new requirements emerge. In traditional RE requirements are typically prioritized once as the customer involvement is high only in the early phases of the project (Paetsch et al., 2003). With agile methods, prioritization happens constantly. The goal is to focus on those features first which bring the most value to the customer. In traditional development methods, completely new requirement, that appears in the later stages of the development projects, must usually go through change-management processes and then through all the traditional phases of the RE (Overhage, Schlauderer, Birkmeier & Miller, 2011). In agile methods requirements prioritization is done in planning meetings, usually before each new development cycle or iteration (Cao & Ramesh, 2008).

As mentioned earlier the documentation of requirements in agile methods is usually kept to the bare minimum. In many agile methods creating complete and extensive documentation is not seen as cost-effective or feasible. In many cases, the possible long-term problems caused by the lack of documentation is compensated with focusing on to the clean and compact code. Most of the agile methods have some kind of documentation recommendations, including the use of a requirements document. (Paetsch et al., 2003) Instead of a formal requirements document, in agile RE requirements are usually documented informally to lists of features or stories. The need for formal documentation is replaced with intense collaboration and communication between the customer and the development team. (Cao & Ramesh, 2008). The formal documentation that exists has a scope that is limited and focuses on the core aspects of the project. This also reduces the time spent on keeping the formal documentation up to date. (Paetsch et al., 2003).

The requirement validation in agile RE is usually done during different kinds of review meetings after the latest development cycle. For example, in Scrum process, after each sprint, developers and business stakeholders come together to sprint review where developers demonstrate new features and other participants can provide feedback and ask questions. In agile RE validation activities focus confirming if requirements reflect current user needs. The completeness and consistency are not in focus since formal requirements document is seldom available. (Cao & Ramesh, 2008; Paetsch et al., 2003). During review meetings, customers are usually able to use or test the software, experience how functionalities support their needs and what parts are already implemented. These moments are also an opportunity to discuss the implementation with developers and ask changes in design. (Paetsch et al., 2003).

One of the key values in agile development is to build a software in small working increments. The real working software can serve as an actual representation of the requirements for the customers. The project team and the customer should then together evaluate and refine the solution and plan the next develop-

ment iterations. The current solution can be regarded as a starting point for requirement elicitation rather than the end of the RE process. This combined with frequent releases makes it easier for the customer to understand the overall stage of the project and provide timely feedback to the development team. (Zhang et al., 2010).

Agility is not without its challenges when considered from the RE point of view. Project budget and time estimations are hard to make due to unstable, ever-changing requirements and constant planning and design actions. (Elghariani & Kama, 2016). This is the reason why instead of fixed price/fixed scope contracts, agile software development projects often use time and expense as a base for the contract (Paetsch et al., 2003).

Another common hardship in agile RE process is the customer's role. In reality, the customer may not always be available to give feedback and comment on the ever-changing requirements. (Elghariani & Kama, 2016). This is especially the case when two or more customer groups are involved, the developed system is very complex or customers are involved in different areas of the system. (Cao & Ramesh, 2008) Customers role is not an easy one and the challenge is that agile methods often assume that the customer representative is competent enough to answer to all the developers' questions and can make right and binding decisions. The image of the stakeholders is less idealized in traditional RE processes. Different elicitation techniques are used which aims to get all the necessary information from the stakeholders which in turn, enables resolving all the inconsistencies. In addition, traditional RE uses reviews and externalization to make sure that all the requirements are known, and all the possible conflicts are found. (Paetsch et al., 2003).

Traditional RE focuses on spending much time to define how to do things right. Finding out and knowing what the right in the early stages of the project is often very challenging. Agile RE focuses on postponing the efforts spent on detail defining of requirements to the last possible moment. As the requirements are likely to change often before actual implementation, this approach offers a chance to save time and resources in changing the already collected requirements. This, of course, puts a higher emphasis on skilled developers and efficient communication with the customer to develop what is needed. (Paetsch et al., 2003).

2.3.2 Common agile techniques which support requirements engineering process

Agile development offers some common techniques which support the requirements engineering process. These same practices can be used in traditional development methods. Many agile methods (such as Scrum and XP) introduce these as part of the development process. These are commonly associated as part of the agile development toolbox. The following section describes examples of such practices.

The first example of such a practice is a user story. User stories, or just stories, are used to communicate system requirements. They describe what is expected or needed from the system - and do it from the perspective of a user. A User, in this context, is usually an end-user for a system, but sometimes this user persona can be also another system. User stories are usually short and simple, written in the natural language and answer to the questions: whose needs are described, what is the expected feature and why it is needed. Typically, user stories are not meant to describe a system in detail but serve more as a way to communicate a general understanding of what the system should be able to do. User stories can also be refined as the development continues. User stories are collected to the backlog and then prioritized for the development iterations. Because user stories are written in a way that it is easy to understand, those are a good basis for communication with different stakeholders. User stories can support collaboration when system requirements are analyzed, negotiated and discussed with stakeholder: The discussion with different stakeholders are easier when a complex system specification is left to another forum. (Chopade & Dhavase, 2017; Measey, 2015, 54-56; Bik, Lucassen & Brinkkemper, 2017).

The second example of an agile technique which relates to requirements is retrospectives. Those are meetings where learnings from previous phases are discussed in order to improve the current processes and ways of working. In retrospectives, typical questions are: What went well? What didn't go well? What we are going to do differently next time? The time for retrospectives is usually at the end of a development cycle, but one should be organized whenever it is needed. There can be several different types of retrospectives - depending on which topics the team wants to focus on. Common examples are sprint retrospective - where the previous sprint is analyzed or bug retrospectives - where some critical system error is revisited and learnings from it are used to prevent similar cases in the future. Retrospectives provide plenty of information which can be used in the requirements management process: Those can point out different problems in the defined requirements set up and reveal new requirement related risks. New requirements can be elicited based on the discussion, ambiguities can be clarified, and new requirement dependencies revealed especially if other stakeholder groups are present. (Measey, 2015, 77-81).

Prototyping is commonly used in the elicitation phase of the requirements to help the process of gathering all the customer needs. A prototype is an early version of the developed product which is launched for the customer. With a prototype, the customer can test the existing version and better recognize additional requirements for the further iterations. It helps to recognize different requirements related challenges such as missing features or misunderstood requirements. Prototypes become extremely useful when defining non-functional requirements which relate to the user experience and usability. (Rehman, Khan & Riaz, 2013).

The downside of prototyping is usually the time needed to prepare sufficient prototype. Even though the prototype can be done also in non-digital format, nevertheless creating one usually requires a lot of time. Also, more complex

the developed feature, more complex the prototype. A poorly made prototype can also lead the discussion to inessential topics: For example, when the focus should be on functional requirements, unpleasant visualization can distract the discussion to focus on the non-functional requirements. (Rehman et al., 2013).

Heavy prioritization is important when using an agile process to ensure that the focus is always on the most important topics. Prioritizing can be done outside the development cycle to arrange upcoming requirements (usually written as user stories) to future iterations, inside the development time-box (for example sprint) or when refining the content of the backlog. Several different techniques are introduced in agile methods such as 'MSCV' or 'YAGNI' (Measey, 2015, 58-60)

'MSCV' refers to Must have, Should have, Could have and Won't have this time. The technique is aimed to help prioritization task inside an individual time frame (sprint/iteration). Must have is often called also a minimum viable product (MVP). This priority is given to the stories that must be completed during the corresponding iteration. Should haves too, are important and will cause significant issues if not included in the intended development cycle. 'Could have' has less effect on the customer's satisfaction, but is still important and. 'Won't have this time' is a priority that is used when something is agreed to be left out from the iteration in question. 'YAGNI' on the other hand, is an acronym for 'You Ain't Gonna Need It' and it is used as a reminder when the content of the backlog is considered. It can be applied when it is decided if the user story should be included in the backlog or not. For example, when the developed solution's life cycle is relatively short, the technical quality does not need to be remarkably robust. (Measey, 2015, 58-60).

3 REQUIREMENT RISK MANAGEMENT IN AGILE PROJECT

Risk management in information systems development has been a widely discussed topic in the literature. Especially because ISD projects have a reputation to fail more often than succeed to be finished in a fully satisfactory manner. (Cerpa & Verner, 2009; Glass, 2006). In fact, software projects that exceed their budgets, the end product which does not fully serve the purpose it was created or IS projects that are cancelled, are well-known struggles in the software industry (Schmidt, Lyytinen & Keil, 2001). Many post-mortems of such projects have indicated that problems they faced could have been avoided or greatly reduced with early identification and timely resolution of the projects' high-risk elements. On the other hand, when project managers of successful ISD projects have been investigated, they seem to have one thing in common: their skills in risk management. (Boehm, 1991).

Requirement engineering, as an important part of the development process, plays a significant role in forming the project's risk environment. It has been even stated that deficient requirements are the single biggest reason for an ISD project failure. (Hofmann & Lehner, 2001). For example, requirements, however, expressed, can fail to accurately represent user's need and consequently lead the development completely to a wrong direction. (Lawrence, Wiegers & Ebert, 2001). This chapter gives an overview of a risk management in ISD projects. It investigates what types of risks are associated with system requirements and how these two topics are considered in the agile software development context.

3.1 The concept of risk in ISD

There are several definitions of the concept of risk in information systems. According to Rowe (Rowe, 1975), the risk is "the potential for realization of unwanted, negative consequence of an event". In U.S National Institute of Standards and Technology's Risk Management Guide for Information Technology systems (NIST, Goguen & Fringa, 2002), the concept of risk is defined as "The net negative impact of the exercise of a vulnerability, considering both the probability and the impact of occurrence". To summarizes, a risk consists of probability or possibility and consequences of something negative.

Few other related key concepts can be found in IS risk management literature. These are a *risk factor* or *risk item*, *risk event*, *risk outcome*, *risk exposure,* and *risk impact*. The factor that causes the risk situation is called a *risk factor* or *risk item*. For instance, poor communication inside the project team can be a risk item. *Risk event*, in turn, is a situation where the risk happens. For example, when a design flaw causes the system to break down in the critical moment, we are talk-

ing about risk event. In ISD world risk items can be both project-specific or generic. General risk items are common to all kinds of projects, for example, poor communication or insufficient planning. Project-specific risk items reflect certain aspects of a given project. These can be for instance unrealistic schedules or personnel shortfalls. *Risk outcome* is a result after a risk event occurs. (Boehm & Ross, 1989; Kontio, 2001).

Boehm (1991) approach to the risk management relies on the quantification of a risk. He introduces *risk exposure* as one of the key concepts of risk management. He states: "For each source of problems causing a potential loss to the project, the exposure is identified as a product of the probability of the potential loss, probability (loss) multiplied by the size of the loss":

$$\text{Risk exposure} = \text{Probability(Loss)} \times \text{Size(Loss)}$$

Lastly, *risk impact* describes the utility loss of project stakeholders when a risk event and the resulted outcome (usually negative) has occurred. (Kontio, 2001).

It is important to understand the difference between a risk and other similar concepts such as problem and constraint. Risks are something that cannot be fully predicted, but usually after recognizing it, there is a possibility to affect it - in other words, change the project's risk exposure. Typically, the level of control over risk items varies. Risk item exists as long as the risk event doesn't occur. When it does, risk turns into a problem. Where a problem is a risk outcome and is harmful to a project, a constraint is something in the project's environment that affects (usually limits) projects success. Constraints are usually known, but not controllable. Problems and constraints can form new risk items when the situation in these elements change. (Mursu, 2002).

3.2 ISD risk management process

Risk management in ISD field is an activity that focuses on identifying, addressing and controlling risks (Boehm 1989). It is not a one-time activity. Instead, it is an ongoing process that continues throughout the project's lifecycle. Risk management allows the balancing of economic and operational costs of protective measures as well as protecting information systems and data which are supporting the organization's mission. (NIST et al., 2002). Risks of an ISD project can affect it from different levels. Lyytinen, Ropponen and Mathiassen (1996) have separated these levels as three different environments: 1) system environment which is the one where the software system operates 2) development environment where the development process takes place and 3) management environment which holds all the software management activities.

The risk management process has inputs and outputs. Inputs consist of project's knowledge, context, project goals, and planning. The outputs are understanding of risks and actions to control them. (Kontio, 2001). Between inputs and

outputs lays the risk management process. It is supported by different risk management tools and methods. The process itself is no-way unique to IT context and the same kind of decision-making is present in all areas of our daily lives. The risk management is typically divided into the two primary steps: *Risk assessment* and *Risk Handling* or *Risk Control*. Boehm & Ross, 1989; Boehm, 1991; NIST et al., 2002). The ISD risk management process, its inputs and outputs, and all its steps are summarized in Figure 4. From the figure, it is visible how the risk management process takes inputs such as project status information, moves through risk assessment to risk handling or monitoring. Risk handling aims at limiting projects risk exposure where monitoring actions ensure the preservation of the projects status quo. Finally, the output of the process is understanding of risks and well as an action plan for future risk management actions. Both of the risk management process phases are discussed more in the following chapters.

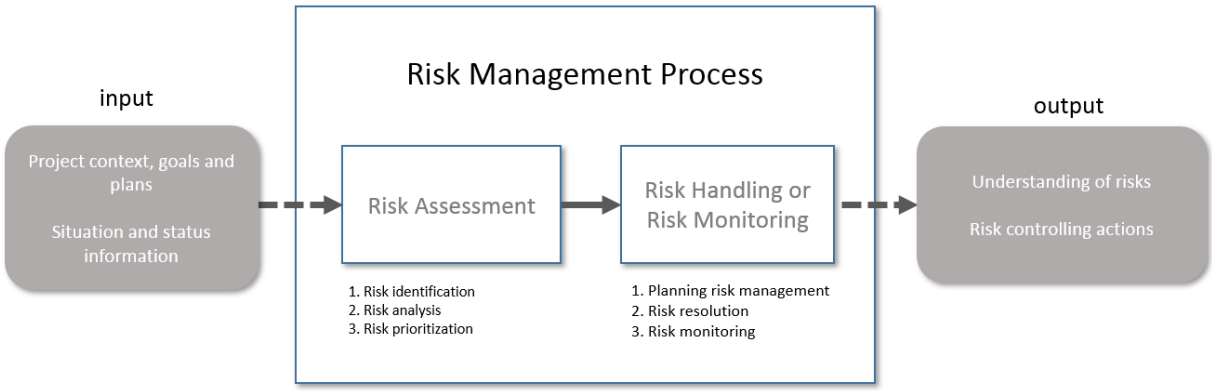


FIGURE 4 Risk management process and its inputs and outputs (Boehm & Ross, 1989; Boehm, 1991; Kontio, 2001; NIST et al., 2002).

3.2.1 ISD risk assessment

The goal of risk assessment is to determine the extent of the potential threats and to unravel risks related to ISD project. Boehm and Ross (1989) divide the risk assessment into three subsidiary steps: risk identification, risk analysis, and risk prioritization. Risk assessment is the key element for the risk management process. When it is done poorly, the whole risk management process is not effective. (Schmidt et al., 2001). There are several approaches to how the risk assessment should be done. Keil, Cule, Lyytinen & Schmidt (1998) offer one approach for the risk assessment. Their framework views risk items through two dimensions: perceived level of control and perceived level of importance. The first dimension focuses on how much project managers perceive that their actions can prevent the risk event from occurring. The second dimension evaluates the relative importance of a specific risk in relation to other risk items. The Importance in this context is a combination of risk items frequency and impact or outcome. (Keil, Cule, Lyytinen & Schmidt, 1998; NIST et al., 2002).

To start any type of risk assessment, one must first try to understand what the possible risks could be. Identifying risk means recognizing those risk items that may compromise the ISD project's success. Different checklists, studying the past or analogous situations, decomposition and examination of drivers for decisions are all typical risk identification techniques. (Boehm & Ross, 1989; Boehm, 1991; Schmidt et al., 2001). Checklists, as one of them, help ISD project managers not to overlook some risk items but also to identify sources for them and categorize them. Several of these checklists exist in literature and these checklists use different categorizations for risk items. (Boehm, 1991; Schmidt et al., 2001). One example of such categorization is done by Keil et al. (1998) in their article. In his categorization, risk items are divided into four categories: Customer mandate, Scope, and requirements, Environment, Execution.

Analyzing known risk items is the second step in risk assessment. During this step loss-probability and loss magnitude, associated with identified risk items, are evaluated. To accomplish this, it is important to understand what processes the system is performing, what is the system's value and importance to the organization and how sensitive data the system is using. Losses assessed in this step can be tangible or intangible. Typical techniques for risk analysis are network analysis, cost models and performance models. (Boehm & Ross, 1989; NIST et al., 2002).

Eliminating all the risk items from the software project is usually impractical or close to impossible. Risk prioritization step's goal is to direct the focus towards the important risk items which in turn, reduces project's overall risk exposure to the acceptable level (NIST et al., 2002). The output from risk prioritization produces a prioritized ordering of known risk items. This step helps project managers to determine which risk items are the most important to address. Typical techniques involve cost-benefit analysis, risk-exposure analysis, and group consensus or Delphi techniques. (Boehm & Ross, 1989; Boehm, 1991).

3.2.2 ISD risk handling

The second primary step in the risk management process is risk handling or risk control. In this phase, the focus is on planning the risk management, risk resolution, and risk monitoring. In other words, dealing with the risk items which were recognized in the previous step, preventing them from turning to risk events and monitoring possible changes in the risk environment.

Risk planning, as the first step, produces a plan for addressing known risk items. This means coordinating the effects that this plan has on the overall project plan in terms of time and resources. One of the common tools for this step is risk resolution checklist. In Table 5 there is an example of such a list. In the table is listed the top 10 common sources of software risks based on Boehm (1991) and resolution technique for resolving them. Generally, risk management planning should focus on answering questions: why, what, when, where, who, how and how much.

TABLE 5 Example of risk items and resolution techniques (Boehm 1991)

Risk Item	Resolution technique
Personnel shortfall	Staffing with top talent, job matching, team building, key personnel agreements, cross training, prescheduling key people
Unrealistic budgets	Detailed multisource cost & schedule estimation, design to cost, incremental development, software reuse, requirements scrubbing
Developing wrong software functions	Organization analysis, mission analysis, ops-concept formulation, user surveys, prototyping, early users' manuals
Developing a wrong user interface	Prototyping, scenarios, task analysis, user characterization
Gold plating	Requirements scrubbing, prototyping, cost-benefit analysis, design to cost
Continuing stream of requirements change	High change threshold, information hiding
Shortfalls in externally furnished components	Benchmarking, inspections, reference checking, compatibility analysis
Shortfalls in externally performed tasks	Reference checking, pre-award audits, award-free contracts, competitive design or prototyping, team building
Real-time performance shortfalls	Simulation, benchmarking, modelling, prototyping, instrumentation, tuning
Straining computer science capabilities	Technical analysis, cost-benefit analysis, prototyping, reference checking

After risk management planning follows the resolution step, where actions for solving different risk items are put to the action. Planning or resolving risks is rarely a straightforward process. The level of control for the risk items may vary depending on the circumstances. For example, in their research Keil et al. (1998)

interviewed 40 software project managers about risk management and their results show how the most important and impactful risks were the ones that appear outside the direct control of the project manager.

Lastly, different monitoring actions are needed to complete the risk management process. In this final step, the progress towards resolving risk items is tracked and corrective action should be done when needed. In this step changes in the overall risk exposure are analyzed based on the findings from the previous process steps. (Boehm & Ross 1989, Boehm 1991).

3.3 Requirements risks and their handling

Requirements engineering process involves a wide range of actions from identifying correct stakeholders to understanding the constraints of the developed system. This means that also risks associated with it, range widely across the whole spectrum of the system development. The current literature does not offer a clear distinction between requirements related risks and other ISD development risks. Furthermore, requirement risks are rarely discussed and analyzed from the risk mitigation point of view. (Tuunanen et al., 2018). True, in some instances such a differentiation might feel even artificial. Yet, because of the important role in software development, the requirement management process should include a conscious risk management actions. For this research purposes, the requirements risks are considered be the following: 1) risk items that can directly or indirectly affect the success of a requirement management process and 2) risk items which can be controlled by requirement management process.

3.3.1 Requirement risks

Existing literature has mostly focused on answering, what are the most common requirements related risks and how those can be resolved. Several different, sometimes overlapping, set of risks have been presented in the previous literature: Already in the 80's Davis (1982) had in his article Strategies for information requirements determination relatively one-dimensional view of a risk - uncertainty. He proposes that the uncertainty level is evaluated for the collected system requirements and defines different resolving techniques to be used depending on the level of this uncertainty. In his article Davis (1982) identified three different uncertainties in respect of requirements: uncertainty of requirements stability and existence, the uncertainty of users being able to specify the requirements and uncertainty in their ability to elicit and evaluate requirements. In more recent years DeMarco and Lister (2003) list five core risks in ISD projects of which

three are related to requirement engineering: intrinsic schedule flaws, specification breakdown, and scope creep. Lawrence, Wiegers and Ebert (2001) describe in their article the following top risks:

- Overlooking crucial requirements
- Inadequate customer representation
- Modeling only functional requirements
- Not inspecting requirements
- Attempting perfect requirements before beginning construction
- Representing requirements in the form of design

It is clear that every project will face some risks which are utterly unique, but many of the writers mentioned above, have noticed similar themes that are typical for requirements risks.

To help the risk assessment for requirement related risks, Mathiassen, Saarinen, Tuunanen and Rossi (2007) have summarized and categorized different risks recognized in the previous literature into different risk types. In their article, they present three risk types, which are: r*equirements identity risks*, *requirements volatility risks* and *requirements complexity risks*. Later Tuunanen et al. (2018) expanded the categorization with one more risk type: Requirement integrity risks. All the of these risk types are listed in the table below (Table 6). This research uses this categorization as presented in the work of Tuunanen et al. (2007) for depicting requirement related risks a project can face.

TABLE 6 Requirement risks (Mathiassen et al., 2007; Tuunanen et al., 2018)

Risk	Definition
Requirements identity risk	The availability of requirements, high identity risk indicates requirements are unknown or indistinguishable
Requirements volatility risk	The stability of requirements, high volatility risk indicates requirements easily change as a result of environmental dynamics or individual learning
Requirement complexity risk	The understandability of requirements, high complexity risk indicates requirements are difficult to understand, specify and communicate.

Requirements integrity risk	The completeness and accuracy of requirements elicited from the end users, high integrity risk means indicates requirements were only partly captured and their origin is not traceable.

Requirement identity risk type includes all those risks that are related to the availability of the requirements. This means that there is a communication gap between the developers and end users due to a conceptual, cultural or physical distance. This can happen for example when the system is developed to mass markets, or when the project stakeholders have challenges agreeing on the requirements. (Mathiassen et al., 2007). In addition, the category includes those risks where all requirements are not recognized. This means overlooking a certain user class or missing critical quality or performance attribute. One other type of example is when the team is focusing only on functional requirements. In this scenario system quality attributes such as performance, security, reliability, and usability are easily neglected. (Lawrence, Wiegers & Ebert, 2001; Mathiassen et al., 2007; Wiegers & Beatty., 2013, 542 - 546). Lawrence et al. (2001) named overlooking crucial requirements as one of the most critical risks in requirement engineering. This is also supported by Tuunanen et al., (2018):

> "Having clearly defined requirements would not only lay a solid foundation for subsequent development efforts but would also ensure that the proposed system matched clients' expectations, thus a higher level of customer satisfaction could be achieved".

Risks related to requirements volatility mean challenges with requirements stability. Requirements are not stable when changes happen in the development project's external or internal conditions or environment. These changes can appear, for instance, because of market impacts such as emerge of competitive products or company impacts such as business strategy and direction changes. An additional source of volatility is a situation when stakeholders learn more about the system as the development progresses. Also, especially in agile projects, the software itself evolves continuously and therefore requirements will evolve with it. (Curtis, Kellner & Over, 1992; Mathiassen et al., 2007; Mills, 1999). Lawrence et al. (2001) argue that trying to avoid volatility type of risks by perfecting requirements before starting the implementation is a risk on its own:

> "Today we live in the emergent world - some information simply isn't available early in our project - and only emerges later. We can't possibly know everything we'd like to know before we start development. It's safer to assume that our requirements are going to change than that they won't."

Requirements complexity refers to how hard or easy it is to understand requirements. As software is inherently complex, also requirements tend to be so. Requirements need to be communicated effectively to project stakeholders. Stakeholders, on the other hand, are likely to have different, varying, and contradicting view of the developed system. This means that challenges arise in understanding, describing, communicating and specifying system requirements. (Mathiassen at al., 2007). DeMarko and Lister (2003) also pointed out that systems are hardly ever developed with only one user group's requirements in mind. Today's IS systems affects several stakeholder groups. On the other hand, sometimes even the different user groups don't share a common agreement on the desired outcome. Requirement complexity risks arise when different stakeholder groups fail to concur on project goals resulting in struggles in the later parts of the developments. (Wiegers & Beatty, 2013, 543)

The fourth risk categorization, requirement integrity risks, according to Tuunanen et al., (2018), refers to the level of accuracy and completeness of requirements elicited from the end users or stakeholders. Issues related to these risks emerge because of the lack of requirements review and inability to trace the origin of the requirement. Furthermore, writers argue that setting a fixed budget and deadlines for the overall project might limit the time spent on eliciting the requirements. This may result in a situation where requirements are not being fully captured. This is a very typical situation when tight development schedule forces a project team to limit time spent on analysis and negotiation phases. This, in turn, may lead to development being pushed forward even though open issues still exist. (Wiegers & Beatty, 2013, 545).

3.3.2 Requirement risk management

Requirement risk management should not be done in isolation from other ISD project risk management activities. Because requirement related risks play such a significant role in projects overall success, managing them consciously is important. For instance, if requirements are initially defined poorly, clarifying them later often increases the efforts needed to complete the project. This, on the other hand, requires more resources (time and money) and easily escalates to other problems. This situation is commonly known as a *scope creep*. In short, it is a phenomenon when project content is extended with new requirements or demands, but the project resources (time, money, people) are not increased accordingly. Identity and volatility types of risks are a common culprit for this situation, for example when all the stakeholder groups do not share the same understanding what the end-product is supposed to be. (Wiegers & Beatty, 2013, 472-473).

Risk environments defined by Lyytinen, Ropponen and Mathiassen (1996), which were mentioned in chapter 3.2 apply naturally to requirements risks. Due to their business implications, requirements risks are also affected by another level: business environment. This means that requirements are often affected heavily by changes in business processes, business direction and end customers

using it. The relationship between risk environments and requirements risk management is illustrated in Figure 5. It shows an extended version from Lyytinen, Ropponen and Mathiassen (1996) approach to environments which affect risk ISD risk management and in this case, requirement risk management. From the figure, it is visible how requirement risk management is affected by four different environments.

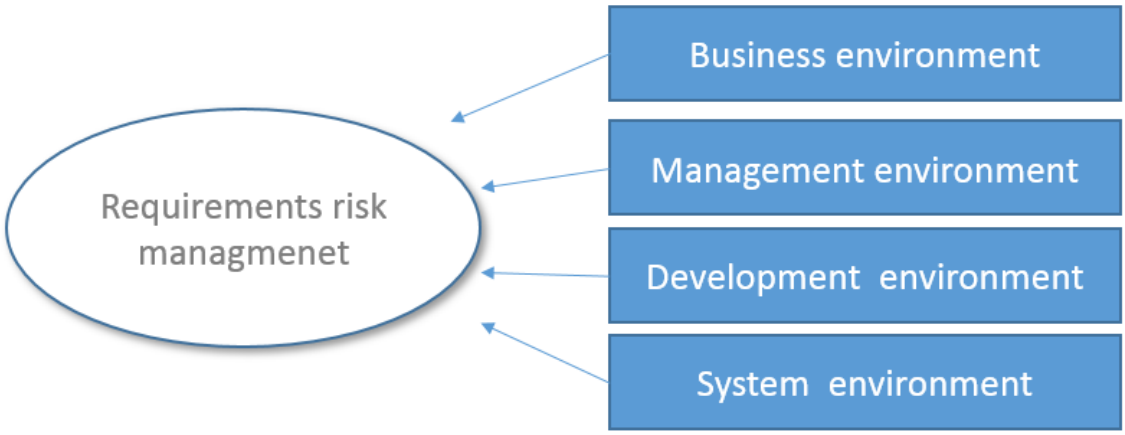


FIGURE 5 Risks environment and requirement risk management process (extended version). (Lyytinen, Mathiassen & Ropponen, 1996)

Controlling requirement related risks should be done with close collaboration between the project stakeholders and the development team or at least doing that seems to increase the project success rate. (Wiegers & Beatty, 2013, 546-547). For example, there is evidence that having a collaborative requirement elicitation with extensive user involvement, can reduce the requirements creep by half (Jones, 1996).

The process of managing requirements risks can be described with the same process as general ISD risk management one, discussed in chapter 3.2. Current literature offers some options for risk handling process for requirement related risks. Matthiassen et al. (2007) have in their article a set of different requirements development techniques which were discussed in chapter 2.2.2. In the same article, writers summarize how those techniques can be applied as risk resolution techniques supporting risk handling process. Writers combined the earlier risk categorization (shown in table 6) with the recommended technique to be used for resolving them. The risk types combined with their recommended resolution technique can be found in the previous table 7.

TABLE 7 Requirements risk categorizations and resolving techniques (Mathiassen at al., 2007)

Requirements risk type	Resolution technique
Identity	discovery techniques
Integrity	prioritization techniques
volatility	prioritization and experimentation techniques
complexity	prioritization and specification techniques

Matthiassen et al. (2007) claim that based on the literature they covered, techniques applied for requirement identity risks should focus on identifying and connecting users and possibly involving them in the development efforts - using discovery techniques. This approach would reduce the communication gap between the users and developers. It could also enable defining requirements using the real voice of the user and stakeholders. Wiegers and Beatty (2013, 543) also emphasize the importance of recognizing correct stakeholder groups early in the project and determining which of them serves as a literal voice of a user. These stakeholder groups should review the requirements as well as define their own acceptance criteria for them.

For requirements volatility risks writers suggest using a combination of prioritization and experimentation techniques. This helps in to stabilize the elicited requirements and make it easier to put the focus to correct topics. New requirements' priority should always be evaluated against the work remaining to be done (backlog). This helps to identify those tradeoffs that addition of these requirements will cause, hence limiting the likelihood of scope creep to appear. Tuunanen et al., (2018) propose the use of prioritization techniques also to the requirements integrity risks.

When requirements stability risks are probable, Matthiassen et al. (2007) suggest increasing the focus on detailing and specifying. This means that requirements complexity risks would benefit from prioritization and specification techniques. The specification can be done in different kinds of reviews where the correctness and quality of the requirements are refined with the stakeholders. (Wiegers & Beatty, 2013, 51).

3.3.3 Requirements risk management in Agile development and Requirements Risk Prioritization method

Requirements related risks mentioned in previous chapters are not only substantial when using traditional development methods. Characteristics of agile development methods bring their own twist to the ISD project's risk environment. This does not mean that the risks themselves are completely different, but some risks might be more common depending on which development approach is used. For

example, requirements integrity and complexity risks may appear when new requirements emerge during short development cycles. Requirement identity risks that are caused by the communication gap have an important role when development depends on human communication without extensive documentation. Close customer collaboration may also generate its own layer of requirements identity risks when people from business and development environment try to understand each other. The fundamental principle of agile development: the ability to adapt to constant changes could increase the likelihood of volatility risks.

Continuous risk management is part of both traditional and agile development methods. When agile methods are used, it seems to be compulsory. (Wiegers & Beatty, 2013; Lyytinen, Mathiassen & Ropponen, 1996). Risks that have not been correctly identified and handled in a timely manner can have serious consequences for the project in the future. Tuunanen et al. (2018) argue: "if risk item was not addressed at a particular phase, any other phase of the project could be affected".

Current literature provides a relatively small amount of methods for requirements risk management that focuses on requirements engineering process in agile software development. There exist some general-purpose requirements risks risk management methods like earlier mentioned A contingency model for requirement development (Mathiassen et al., 2007) and A Unified Model of Requirements Elicitation (Hickey & Davis, 2004). However, those methods are not specifically developed for agile requirement engineering process. Tuunanen et al. (2018) have developed a Requirements Risk Prioritization method (RRP method) which they argued to be suitable for continuous requirement risk management and hence would be suitable to be used with all kinds of development methods - including agile.

RRP method (Figure 6) was developed with experienced practitioners from the ISD development industry who manage their projects in cooperation with the New Zealand Project Management Institute. The method uses Mathiassen et al. (2007) model and Keil et al. (1998) A Framework for Identifying Software Project Risks as theoretical background and inspiration. The research behind the method was a qualitative research that used focus group interviews and a Delphi survey.

First Tuunanen et al. (2018) used both research methods to identify set of requirements related risks that focus group identified and categorized them using the categorization from Mathiassen et al. (2007) extending it with one new type dimension: requirements integrity risks (discussed in chapter 3.3.1). After that, they asked the focus group to indicate the level of risk each risk item possessed and the development phase that it was likely to affect the most.

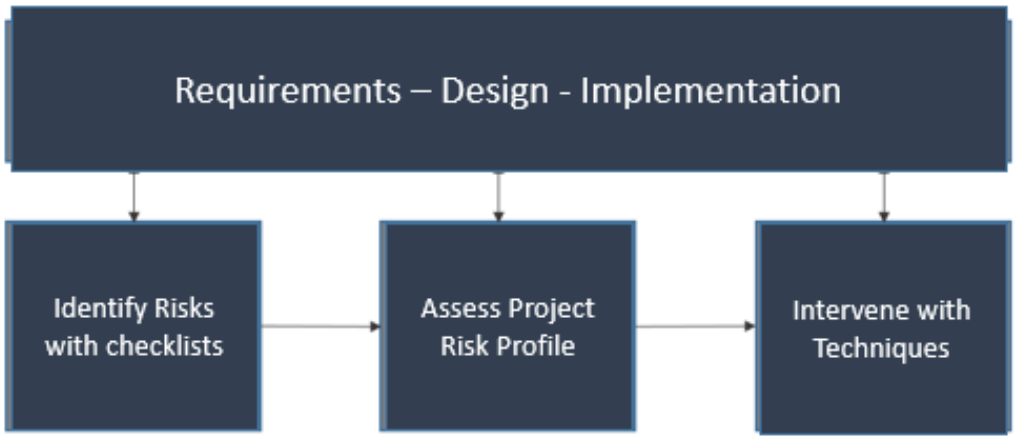


FIGURE 6 Requirements Risk Prioritization method (Tuunanen et al., 2018)

In their research, the ISD Development process was divided into three phases: requirement phase, design phase, and implementation phase. Based on their finding writers composed requirements checklists for all the previously mentioned development phases. The method doesn't assume that ISD includes only previously mentioned phases, instead, the development process may include many cycles that individually include those three phases. In the method the requirement risk management process includes three steps as well: Identifying risks, assessing the risk profile and intervening with techniques. As figure 6 show, all these three steps are targeted at each development phase.

Below, Figure 7 illustrates how the RRP method relates to ISD risk management process presented in chapter 3.2. As it is visible from the Figure 7 Tuunanen et al. (2018) approach vary slightly by dividing risk assessment into two separate steps: From these two the first one focuses on identifying risks. Risk analysis and prioritization of risk in traditional ISD risk management process are part of the Assess Project risk profile step in the RRP method. The last step, intervene with techniques, lands more on the risk handling phase of the traditional ISD risk management process. Another risk handling topic of the traditional risk management process, risk management planning and risk monitoring are not as clearly part of the RRP method. Figure 7 showcases this relation between the two risk management processes.

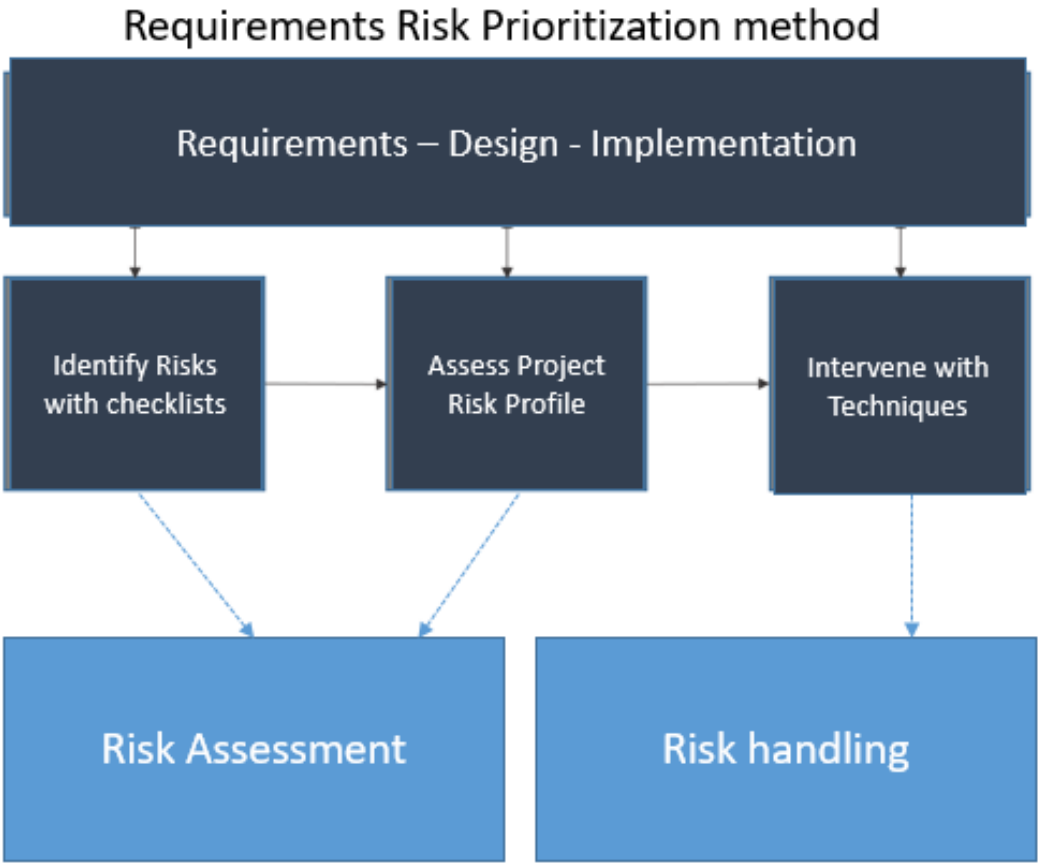


FIGURE 7 ISD risk management and the RRP method

The process in the RRP method starts by 1) utilizing the checklists to identify correct risk items. Checklists offer a set of potential risks commonly associated with the corresponding requirement management phase. A similar approach is used for example by Persson et al. (2009) in their article. After risks are identified, 2) the project risk profile is assessed by recognizing individual requirement risks. The profile is finalized with impact evaluation (high, medium, low) for each risk item. This information is used to prioritize the requirements risks correctly. In the last phase intervention is done by 3) utilizing risk resolution techniques (presented earlier in Table 7) in the following order:

- If identity risks are high, put a high emphasis on discovery techniques.
- If integrity risks are high, put a high emphasis on prioritization techniques.
- If volatility risks are high, put a high emphasis on experimentation techniques.
- If complexity risks are high, put a high emphasis on specification techniques.
- If three or more risk items are high, follow the above sequence of applying techniques (from 1 to 4).

4 RESEARCH METHODOLOGY

To further understand the specifics of requirement risk management in an agile development environment empirical part of this study focuses on using the RRP method in a real agile project environment. This was achieved with a case study where the method is used for requirements risk assessment. As each ISD project possesses its own characteristics despite the development method used, a case study approach was selected because it allows studying the phenomenon flexibly inside its own boundaries. A case study is a widely used research method - it offers good opportunities to investigate the phenomenon within its real-life context and aims at understanding the dynamics of a single setting (Eisenhardt, 1989; Yin, 2013). The research data was collected through semi-structured interviews and thematic analysis was used to analyze the research data. This chapter defines the research strategy and explains how data collection and analysis were done. It also introduces the research case as well as describes how the empirical part of this research was executed.

4.1 Case study as a research strategy and case description

A case study is a commonly used method in qualitative IS research. As the goal of this study is to investigate a phenomenon in its real-life context, the case study offered the best tools to understand the dynamics of its 'natural environment'. Qualitative research, in general, relies primarily on human understanding and perception. It is descriptive, and common sources of qualitative data are observations, interviews, documents, and the researcher's own perception. Contrary to quantitative research, qualitative research aims to understand people and the cultural context surrounding them. (Klein & Myers, 1999; Myers, 1997; Stake, 2014).

Qualitative research can be positivist, interpretive or critical. This study follows an interpretive stance. The interpretive approach assumes that reality can be accessed only through social constructs such as shared meanings and language. When compared to positivist research, where reality is assumed to be describable by measurable properties which exist independently from the observer, interpretive studies aim to understand the phenomena through the meanings people give them. It does not consider independent or dependent variables but explores the full complexity of human reasoning. (Klein & Myers, 1999; Myers, 1997)

According to Yin (2013) case study approach can be used when 1) the study aims at answering questions "how" and "why" 2) the phenomenon and context don't have clear boundaries 3) it is not possible to manipulate the behavior of people involved in a study; and or 4) covering contextual conditions is relevant for the phenomenon that is studied. The design for a case study can consist of a

single case or multiple cases. A case should be something that represents the topic of the study empirically and is interesting and relevant to the studied phenomenon. The unit of analysis is the source of data itself; for example, the interviewed individual. (Yin, 2013).

The case study was a natural selection as a research method for this research. The performance of RRP method in an agile project cannot be considered without the context of the agile project itself. Since each agile project is different in terms of practices used and even the "level of agility", it would be impossible to have a true picture of the person's perceived usefulness of the method without considering the context in which it was used. Also, the ISD project where the requirements risk management actions were done, cannot be considered without its organizational and social settings.

Another benefit of using a case study is that it enables capturing multiple different perspectives (Yin, 2013). This was found especially beneficial for this study since the experts chosen for the case represented different business functions as well as varied in their experience in agile software development. Taking this diversity into account made it possible to draw a more thorough picture of the research case and finally about the research topic in general.

4.1.1 Case description

This research was conducted as a single case study. The case of interest is an agile software project in a company that operates in B2B conference business. The case company is based in Finland but brings together IT solution providers and investors from all around the world. The organization is focusing mainly on markets in Europe and South-East Asia. Although not a software development company, the case organization has a long history of developing software in-house from design to implementation to support its unique business requirements and support their customers more flexibly.

One project from the IS development portfolio of the company was selected for the focus of this study. At the time of writing this study, the case project is still ongoing in the active development stage. The project's goal is to develop a product which provides customers of the company an interface to manage their conference participation both before and at the event. The end-product offers customers a self-service channel to provide necessary info about their participation which enables better customer service. On the other side, the platform provides customers with all the information they need to get the best return for their investments at the conference itself. The product is also used for managing customers' schedules during the event, and for providing feedback about the conference.

The project started two years prior to this research. The product first launched at pilot conferences and at the time of the study, it is already being launched to the whole case organization. After the initial launch, the project has continued focusing on further improving the current implementation and adding new features based on customer feedback.

As the project follows agile development, it is developed iteratively. To ensure representativeness of the research results and to focus research's scope, one feature development was selected to the scope of the study. This made it possible to evaluate all the phases of the development with the RRP method without one being already completed.

The focus feature is a possibility for solution provider customers to approach the investors in the service and express their interest in having a face-to-face meeting during the conference. This consists of a personalized message that the user can send to another party inside the platform. This feature was not a part of the first iteration of the product. After customer feedback, it was decided to be added to the next development round. At the time of the study, this feature was in its early design stage.

4.1.2 Agile development in the case project

The case organization follows the agile approach in their software development. The development process uses practices mainly from Scrum. The base of the Scrum process is a sprint, a time frame in which the selected features are developed. One sprint usually lasts 2-4 weeks. Requirements are collected to a backlog from which they are selected to the development sprints to be implemented. At the beginning of each sprint, there is a sprint planning meeting where selected requirements are introduced to the development team and discussed. The sprint ends at the sprint review where the completed features are introduced and reviewed.

Other typical Scrum "ceremonies" are different retrospectives and daily scrum meetings. Retrospectives aim at evaluating the work done so far and improve the current process further. One example of retrospectives is a bug retrospective. A daily meeting takes around 15 minutes where each team member focuses on the following three questions: What I accomplished yesterday, what I shall do today and what prevents me from doing that.

In the case organization, most of the Scrum ceremonies are in use. One sprint is 2 weeks long and sprint planning is done before each development sprint starts. Requirements are collected in backlog from which each sprint's content is selected based on the current business priorities. The requirements are collected and defined by a product owner who is also participating in the daily management of the development team with the development team lead. Product owners of the different software projects also gather each morning for their own daily meeting, to align with the current situation in each project.

Product owners are responsible for collecting business requirements for their own products in close collaboration with different business functions and other project stakeholders. They also "own" the development priorities and manage the order of tasks - or user stories - in the backlog. Their prioritization decisions, in turn, are done after the high-level direction is decided by the top management and heads of different involved functions. The development priorities

are monitored and evaluated on a monthly basis by top management and other key stakeholders.

In case organization retrospectives are used but those are not scheduled for every sprint. They are organized when an extra focus for a certain situation is needed. For example, bug retrospectives are used when technical challenges appear, to avoid similar situations in the future. Sprint retrospectives are used to evaluate the success of the previous sprint. Project teams organize also feature retrospectives for those feature developments that are important to share across teams.

Risk management for the development is mainly a responsibility of both management and product owners. Product owners manage risks that affect their projects on a daily basis, but there is no agreed risk management process in place and the risk management is mostly done as a part of other activities such as planning and prioritizing. Before this research, the organization has not made exercises which focus precisely on requirement level risks.

4.2 Data collection

A case study can accommodate a wide range of data sources such as archival data, interviews, observations, and survey data. For this study, an interview was selected as the main method of data collection since it is an efficient way to collect rich empirical data and has a focus on the individual perception of the research topic. As the research questions focus on people's perception on the requirement risk management and the use of the RRP method, it is important to be able to see connections between a person's position in the case project and organization, as well as their opinion on the exercise done within the case study. Because a person's position might affect a lot how he or she sees the software project itself, it is also important to be able to read between the lines and find possible contradicting views.

There are several types of interviews, which are commonly used in qualitative research. These types are an open interview, a structured interview, and a semi-structured interview. An open interview is a conversational situation, where participants discuss preset research topic. An open interview has open questions and the response choices are not limited. This way the open interview is the most informal of the three types of interview. On the opposite side, in a structured interview, all the participants answer the same questions. Answer options are also often defined before the interview starts. (Galletta, 2012)

The type of interview used in this research is called a semi-structured interview, which is a combination of the two mentioned above. It can include both open and closed questions. This means it is structured enough to address the research topic, while it still leaves space for flexibility. According to Galletta (2012): "A key benefit of semi-structured interviews is its attention to lived experience while also addressing theoretically driven variables of interest." Most of the participants in this study are representing different business functions and working

with the project from different angles. In this situation, the interviewer needs to have room to probe a participant's answer for clarification, critical reflection and meaning-making. This flexibility is the main reason for choosing semi-structured interview for this study.

4.2.1 Conducting a semi-structured interview

The case company was contacted about the interview in January 2017. And the interviews took place May 2017 - August 2017. All the interview participants were confirmed that all the interview answers are treated sensitively and case company or the participants cannot be identified within the study. Study results were shared with the organization's' management, but none of the individual answers were distinguishable.

Selection of interviewees was done based on the following three criteria: 1) person must be closely working with the case company's software development projects either as part of the development team or as a business requirements owner 2) person's position in the company must be specialist or mid to top management. 3) A person must be familiar with the agile software development process in the case company.

Since the case company is not a software company, all the employees are not familiar or involved in the software development projects the case company has. The first criteria ensure that only those people who have sufficient involvement in the software development projects were selected for the interview. The sufficient involvement in this context means that the person is either working directly in a software project as an architect, developer or product owner or the person is a direct stakeholder from the business side. The second and the third criteria aim at focusing on having only interviewees who have enough knowledge and such a position in the organization, that they can in their work directly affect how development projects are run. This way it was possible to make sure that interviewees can evaluate requirement related risks at a proper level and the results are comparable, despite the different business functions they represented. These criteria meant, for example, that only lead developers were selected to this study and more emphasis was put on the product owner and architect level personnel.

The following list shows the selected interviewees and their function title:

2 Software architects
3 Product owners
1 Lead software developer
1 Business manager
1 Customer experience manager

Because the project team is relatively small and heterogeneous, it would be very easy to identify individual interviewees from more detailed background information (age, gender, education etc.) This could have affected interview answers and lowered the reliability of the interview results. To protect the anonymity of interview participants, more detailed demographics were not published.

The interview was divided into three phases. The first phase focused on using the RRP method in case context. This meant that participants were asked to use the RRP method to generate a risk profile for the focus feature. In phase two the participants were asked to evaluate data they had produced with the help of the method. The evaluation was done by commenting on the risk profile they had created for the focus feature. Lastly, interviewees were asked to assess the method's usage in their own projects. This included evaluating the usefulness of the method and how they see that the method could or could not support their requirement risk management work.

Time reserved for each interview was 60 minutes. The interview time was extended when needed, to make sure that all the same elements were covered with all the participants. All the interviews were done individually, and participants answered the same predefined interview questions. The interviewer asked additional questions in case some clarification was needed. All the interviews were recorded, and answers were littered later for the data analysis. The full structure of the interview and interview questions are presented in APPENDIX 1 and APPENDIX 2.

All the interviewees received a glossary of terms before the actual interview. That helped to reduce the confusion that might have appeared because of the IT jargon that was not familiar to all the participants. This glossary of terms can be found from APPENDIX 3. Extra attention was paid for making a clear difference between terms *customer* and *user* in research case's context. In the case company customer means an internal business owner and user is mainly the end customer of the case company. Half of the interviewees spoke Finnish and the rest other languages. For Finnish speakers, the interview was conducted in Finnish and for the rest in English. All the materials were provided in English to avoid different interpretations of the terms used.

All the participants had different levels of knowledge about the focus feature that was selected for this study. The interview started with a focus feature introduction, to make sure all the participants shared the same level of understanding about it. After the feature was introduced, general introduction for the RRP method followed.

From all three requirement risk management steps, the method includes, the third, using recommended techniques to handle identified requirements risks, was excluded from the scope of this study. Introducing risk solving techniques to the research would have required teaching them to the interviewees since none of the interviewees were familiar with these specific techniques the method utilizes. Understanding those techniques well enough was not possible within the time resources allocated for this study. This means that any results related to the

risk solving techniques would not have been representative enough for the purpose of this study.

After introductions, the first interview phase started. It included the creation of the requirement risk profile for the focus feature. Each requirement management step was analyzed separately (requirements step, design step, and implementation step). Analysis started from a requirements step. At first, the interviewee was asked to write down risks that he/she felt might be related to each step to a separate paper. This was done without seeing the risk checklists of the RRP method. After this, the method's checklist for the corresponding requirements management phase was presented. Then a participant was asked to read through the checklist and mark those risks 1) which correspond to the ones the interviewee had written down before seeing the list 2) which the interviewee didn't consider before, but which are still relevant for the project and focus feature 3) which are not relevant at all for the project and the focus feature.

Those risks that participant wrote down before seeing a checklist, but which did not correspond to any risk item in the method's checklist were underlined and included in the final list of relevant risks for each phase. An identical process was repeated for all the design and implementation steps. In case the interviewee didn't understand an item on the checklist, the interviewer provided an explanation.

After analyzing relevant requirement risks, interviewees were asked to assess the risk impact for each risk item they considered valid for the project and focus feature. The impact evaluation was done for risk items by marking "+" for high impact risk and "-" for low impact risk. A risk with normal impact was left without a symbol. For risks which interviewees considered irrelevant for the case, interviewees did not have to give an impact evaluation.

The result from the first interview phase was a risk profile for the focus feature development. This included a list of valid risks for each requirements management step (either from checklists or identified by the interviewee) with risk impact evaluation. Interviewees could make changes to their risk profile before moving to the second interview phase. The risk profile for the focus feature development was visible for the participant for the rest of the interview.

In the second interview phase, the participants were asked questions that evaluated how well the risk profile they have created actually represent focus feature developments requirement risk situation. Interviewees were able to comment if they think the checklists really included enough relevant risks for them to be able to evaluate reliably focus features requirements risks. They were also asked to elaborate risks that they felt were missing from the original checklists, and if the risk profile they created would help them prioritize risks for the focus feature.

In the last interview phase questions focused on how likely the participants would be to use the RRP method in their current or future agile projects. Participants were asked to assess what benefits and limitations this method would have in their own project work. Interviewees were asked if using the method makes

requirement risk management for their own projects easier compared to their current way of managing requirement related risks.

4.3 Data analysis

Data analysis started after all the interviews were held. First, all the data from the risk profiles the interviewees created was combined into one table. Then a thematic analysis was conducted for the rest of the interview data. Combined data from the interview participants' risk profiles are summarized in Table 8. Results of the thematic analysis of the interview questions are showcased in APPENDIX 5.

4.3.1 Analysis of risk profile summary

For each risk item in the RRP method's checklists it was calculated how many of the interviewees thought:

1. The risk corresponds to the one the interviewee had written down before seeing the checklist
2. The risk interviewee didn't consider before, but it is still relevant for the case
3. The risk is not valid for the case

The risk impact evaluations were also added to the table for each risk item. This means the number of participants who evaluated the risk item to have a low, high or medium impact was added to the table.

TABLE 8 Risk profile summary

		Risks validity			Risk Impact		
Requirements phase	**Risk type**	**Valid** a participant identified as a risk without a checklist	**Valid** a participant identified as a risk from a checklist	**Not valid**	**High**	**Low**	**Neutral**
Absence of project Sponsor	Identity	2	5	1	5		2
Access to Clients (Proximity to Source)	Complexity	2	4	2	2	1	3
Ambiguous Requirements	Identity	6	2		7		1
Change in Business Strategy and Direction	Volatility	4	3	1	3	1	3
Change in External Regulations	Volatility		3	5	1		2
Client Commitment	Identity	3	4	1	6		1
Constrained User's Knowledge	Complexity	3	4	1	2		5
Fixed Budget and Timelines	Integrity	3	2	3	3		2
Incorrect Stakeholders	Identity	4	3	1	3	1	3
Misunderstood Business Needs	Identity	5	2	1	5		2

Underestimation of Change Magnitude	Volatility	4	4		3		5
Unrated Requirements	Volatility	4	3	1	3		4
Design phase	**Risk type**						
Ambiguous Requirements	Identity	5	3		3	1	4
Change in External Regulations	Volatility		4	4	2	1	1
Client Commitment	Identity	3	5		4		4
Compliance with External Regulations	Identity		4	4	2		2
Conflicting requirements	Integrity	2	6		3	2	3
Missing Requirements	Identity	5	3		3		5
Delivering What the Client Requires	Identity	6	2		4		4
Emerging Requirements Dependency	Volatility	3	4	1	3	1	3
Fixed Budget and Timelines	Integrity	4	3	1	5		2
Knowledge Gap between Coworkers	Complexity	4	3	1	2		5
Lack of Collaboration	Complexity	7	1		5		3
Technology Changes	Volatility	1	6	1	1	1	5
Underestimation of Change Magnitude	Volatility	2	6		3	1	4
Unrated Requirements	Volatility	3	4	1	2		5
Implementation phase	**Risk type**						
Ambiguous Requirements	Identity	6	2		6		2
Change in External Regulations	Volatility	1	2	5	3		0
Client Commitment	Identity	4	1	3	4		1
Fixed Budget and Timelines	Integrity	3	3	2	4		2
Hostile Users	Identity		5	3	2	2	1
Project Team Member Turnover	Volatility	1	7		4	2	2
Unrated requirements	Volatility	5	2	1	2		5
Underestimation of change magnitude	Volatility	5	3		4	1	3

Risks profile summary was used to evaluate how well risks items in the RRP method's checklists were matching participant's perception of the research case's risk situation and if checklists were able to highlight risks the participant didn't think of without the checklist. It was also possible to analyze what kind of a risk impact the participants were assigned for both types of risks.

To evaluate the agreement between the participants, the following scale was applied: If the risk item had six or more participants answering the same way, that was considered to be a significant agreement between participants. When the number of participants varied from 3 to 5 that was considered to be a moderate agreement and if less than 3 participants answered the same way, participants were not agreeing about that risk item.

Risks that the interviewees recognized but were not included in any of the original checklists of the RRP method, were collected into their own table. These risks can be found below (Table 9) as well as the development phase an interviewee considered it be likely to affect the focus feature. All the interviewees wrote down at least one of this type of a risk. The table 9 has also the number of interview participants who mentioned a risk with the same content.

In total there were 9 additional risks for the requirements phase, 10 for the design phase and 6 for the implementation phase. Interviewees were not asked

to define a risk type nor an impact level for these risks. During the analysis, common themes for these risks were found. These common themes are *project team related*, *customer & user experience related*, *requirement scope related*, and *agile development related*. These themes are also visible in table 9 but are described more in detail in the next chapter.

TABLE 9 Risk missing from the original checklists

Requirements phase	**Number of participants mentioned the risk**	**Theme**
Undefined / Unclear responsibilities in the project team	1	Project team related
Challenges in adding the new feature as a part of the existing system	1	Agile development related
Difficulties in defining a scope for a feature	2	Requirement Scope related
Not considering all / correct user/customer groups	2	Customer & User experience related / Requirements scope related
Overview and ownership from the client side	3	
Lack of iterating/missing proof of concept	2	Agile development related
Requirements are rated incorrectly in relation to other developments	1	Agile development related
Challenges in evaluating and measuring the benefit new feature would bring	1	
Rapid changed in business needs	1	
Design phase	**Number of participants mentioned the risk**	
Lack of clear UI design processes	1	Customer & User experience related
Challenges with UI design	4	Customer & User experience related
Not considering user experience enough	1	Customer & User experience related
Communication between projects stakeholders	1	Project team related
Splitting/structuring requirements to small enough pieces for agile development	1	Agile development related
Collaboration with external / multicultural teams	1	Project team related
Lack of human resources	2	Project team related
Lack of technical understanding of the project team	1	Project team related
Challenges in adding the new feature as a part of the existing system	2	Agile development related
Unclear work estimations	1	
Implementation phase	**Number of participants mentioned the risk**	
Understanding the business need	2	
Requirements dependencies	1	Requirements scope related

Communication between project stakeholders	3	
Limitations in technical skills inside the project team	1	Project team related
Collaboration external / multicultural teams	2	Project team related
Challenges in adding the new feature as a part of the existing system	3	Agile development related

4.3.2 Thematic analysis of the interview questions

Data analysis continued with a thematic analysis of the answers from the second and third phases of the interview. It is a commonly used method for analyzing qualitative data. It emphasizes examining patterns or themes across the data set. A theme captures something important about the collected data in relation to the research question. A thematic analysis can be associated with two modalities: inductive and deductive. In this study, an inductive approach was selected because it is generally considered the best solution for the phenomenon when there are no earlier studies. (Braun & Clarke, 2006; Hsieh & Shannon, 2005; Vaismoradi, Turunen & Bondas, 2013).

The thematic analysis process started with getting familiar with the data. The data transcripts were read and reread several times. At first, ideas were written down as notes and those were later used as a base for data coding. From these notes, the finalized code set was established and the whole data set was coded. Once this was done for all the material, potential themes were selected from the coded data. After all the initial themes were set, their relation to the coded extracts and the entire data set were reviewed. This process produced a thematic map. Lastly definitions and names for each theme where selected. This was an iterative process, which included continuous refining the specifics of each theme. From this analyzed data, research conclusions were drawn. The thematic map is produced from analysis of interview questions (phase 2 and 3) is visible in APPENDIX 5. It shows all the themes and their definitions, which were produced by this thematic analysis.

5 STUDY FINDINGS

In this chapter, study findings are presented in detail. At first, results from the analysis of the risk profile are summarized accompanied by any related comments or questions, which appeared during the later interview phases. After that, results from the second and third interview phases are combined. Findings are presented with authentic citations, which support the propositions made from these topics and increase the trustworthiness of this research.

5.1 Evaluation of the risks in the checklists

Several findings could be derived from the risk profile summary (Table 8). First, it was used to evaluate, which of the risks interviewees considered to be relevant to the study case. Next, it was used to estimate if using the checklists helped the participant find risks he or she would not have considered without using them. Thirdly, it was possible to estimate how important all the valid risks were for the research case based on the impact level the participant chose for them. This chapter also introduces more in detail common themes of the risks, which participants felt were not covered by the original checklists.

Few general observations could be made from the first interview phase. While creating their risk profiles, most of the interviewees struggled to understand what specific risk item means and what kinds of issues it covers. The most questions arose regarding risks: *unrated requirements* and *hostile users*. The latter required clarification in each of the interview sessions. Additionally, the differentiation between the customer and the end user was not always clear to the interviewees. Even though the differentiation was clarified before the interview, people with different roles in the project had slightly different understandings of these concepts and their roles in relation to the requirements management. Even though specific for this research case, this setup is not a unique one and hence worth mentioning.

5.1.1 Risks validity for the research case

The relevance of the risks for the research case was estimated based on how many participants marked the risk as *valid* for the research case. In risk profile summary (table 8) the number in the "Not valid" column shows the number of participants who considered the risk to be invalid. From this column, out of the total 34 risks, none of them were marked invalid for the research case by a significant number of participants. Around one-fifth of all the risks (7 = 20,5%) were marked as invalid so that the number of similar answers reached the moderate level of agreement between the interviewees. In total 27 out of all the risks (=79%) received less than 3 answers to "not valid" column. Yet, more than half of the risks, 22 (= 65%)

received at least one answer where it was not considered valid for the study case. Risks *Change in external regulations* and *Compliance with external regulations* received most "not valid" answers despite the development phase they were related to. This is natural since the focus feature did not have any known relation for any external regulation. These results indicate at least a moderate level of agreement among the interviewees, that the risks presented in the checklists are valid for the research case. In other words, checklists do not highlight risks which would be unnecessary or completely irrelevant.

Risks that received zero answers for "not valid" column were confirmed to be valid by all the interviewees for the research case. These risks and the development phase they relate to are listed below in table 10. From there it is visible that the number of these 100% valid risks differs slightly between different development phases. The design phase had most of these risks (7 out of 14 risk items = 50%). The requirement phase had the least of these type of risks (2 out of 12 risk items = ~17%) and for the implementation phase, the result was 3 out of 8 risk items (~38%). *Ambiguous requirements* and *Underestimation of change magnitude* were risks that are present in the checklist for each development phase. Interestingly, the result in table 10 shows that those risks were also considered as valid risks in all the development phases by all the interviewees. The same table shows that most of the risks with 100% validity for the research case are either identity or volatility type with only two exceptions: *Conflicting requirements* (Integrity) and *Lack of collaboration* (Complexity).

TABLE 10 Risks with 100% validity for the research case

Risk item	Phase	Risk type
Ambiguous requirements	Requirements phase	Identity
Underestimation of change magnitude	Requirements phase	Volatility
Ambiguous requirements	Design phase	Identity
Client Commitment	Design phase	Identity
Conflicting requirements	Design phase	Integrity
Missing requirements	Design phase	Identity
Delivering what the client needs	Design phase	Identity
Lack of collaboration	Design phase	Complexity
Underestimation of change magnitude	Design phase	Volatility
Ambiguous requirements	Implementation phase	Identity
Project member turnover	Implementation phase	Volatility
Underestimation of change magnitude	Implementation phase	Volatility

5.1.2 Identifying risks with or without the checklists

To evaluate if the risk profile produced new information for the interviewees, the analysis focused on understanding whether participants considered risk item as a valid risk before seeing the checklists or only after seeing them. This information is also shown in Risk profile summary, table 8, in first two sub-columns of the *Risk validity* column. From all the risks in the checklists, only ones related

to *external regulations* and the one about *hostile users* were not recognized by anyone before seeing the checklists. Worth pointing out is that *Hostile users* was also a risk which 7 out of 8 interviewees asked clarification about its meaning - which might have affected why any of the interviewees did not consider it before seeing the checklists.

Risk profile summary (table 8) reveals four risk items which most of the participants (6 or more) marked as covered by the idea of a risk that they had written down before seeing the any of the method's checklist. These risks are: *Ambiguous requirements* (requirements phase), *Delivering what the client needs* (Design phase), *Lack of collaboration* (design phase) and *Ambiguous requirements* (implementation phase). On the opposite side, the risk profile summary (table 8) showcases a same number of risks that most of the participant (6 or more) didn't think on their own but considered as valid risk for the study case. These risks are *Conflicting requirements* (design phase), *Underestimation of Change magnitude* (design phase), *Technology changes* (design phase) and *Project Team Member Turnover* (implementation phase).

When purely comparing a number of cases when more participants recognized the risk before seeing the checklists to the ones where the risk was recognized with the checklist, the results were equal (16 risks items). In two cases the number of answers was equal for both options. These results suggest that the method indeed would help its user to focus on topics that might have otherwise been ignored or forgotten. On the other hand, results indicate that the content of the checklists highlight topics that interviewees seem to focus intuitively.

5.1.3 Risk impact evaluations

For evaluations of risk impact, the distribution of answers was quite high. In general, it appears that participants were less likely to rate the risk impact level to be low compared to high or neutral. Only 12 of a total of 34 risk items had low impact rating from at least 1 interviewee. In addition, two participants specifically mentioned during the interview that they had a hard time to evaluate any of the risks to have a low impact. From the risks profile summary, it can be seen that none of the risks which have a low impact evaluation have that with a significant or even moderate agreement between the participants. The maximum number of interviewees agreeing on a low-risk impact was two.

For high impact risks, participants pointed out few risks that they seemed to consider important. Results show 3 risks where high impact level evaluation was selected by most of the participants (6 or more similar answers). These risks were: *Ambiguous requirements* (requirements phase), *Client commitment* (requirements phase) and *Ambiguous requirements* (implementation phase). Worth noting is that Ambiguous requirements was also considered to be a valid risk by all the interviewees for all the development phases. Risks that got the least high impact level ratings were: *Change in external regulations* (requirements phase) and *Technology changes* (design phase). In both cases, only one participant rated them with high impact level. *Change in external regulations* (requirements phase) received

also the most "not valid" answer in the risk profiles. *Technology changes* (design phase) on the other hand was mostly considered to be a valid risk for the research case.

Another comparison can be seen in Table 11 below. It lists all the risks which were rated as high more often than any other impact level. This resulted in a total of 14 risks. When comparing different development phases, implementation phase had the highest number of these risks (6 out of total 8 risk items). It is important to point out that only one of these cases had the significant number of answers with high impact rating. The sheer number of high impact ratings in relations to the number of risk items in the checklist for each development phase didn't have noticeable difference. Both of these results suggest that participants didn't consider risks in the certain development phase remarkably important than in the other one.

TABLE 11 Risk items with more high impact ratings than other ratings

Risk item	Phase	Risk type
Absence of project sponsor	Requirement phase	Identity
Ambiguous requirements	Requirements phase	Identity
Client commitment	Requirements phase	Identity
Fixed budget and timeline	Requirement phase	Integrity
Misunderstood business need	Requirement phase	Identity
Change in external Regulations	Design phase	Volatility
Fixed budget and timeline	Design phase	Integrity
Lack of collaboration	Design phase	Complexity
Ambiguous requirements	Implementation phase	Identity
Change in external regulations	Implementation phase	Volatility
Client commitment	Implementation phase	Identity
Fixed budget and timeline	Implementation phase	Integrity
Hostile user	Implementation phase	Identity
Underestimation of change magnitude	Implementation phase	Volatility

If interviewee had considered the risk before seeing the checklists does not seem to affect significantly the impact level user is likely to assign for the risk item. Out of all the risks where most of the participants identified the risk item without the checklist (16 out of 34 risks) for 9 risks more participants evaluated the risk impact to be high instead of neutral or low. On the other hand, it seems that the risk which was recognized only after seeing the checklist was also more likely to receive a neutral or low impact rating. Only three of these risks received more high ratings. These risks are *Absence of project sponsor* (requirements phase), *Client commitment* (requirements phase) and *Project team member turnover* (implementation phase).

5.1.4 Risks missing from the checklists

During the interview, all the interviewees wrote down some risks that they felt were not covered by any of the checklists. These are presented in table 9. During

analysis, most of these risks were combined under similar themes. From the Table 9 it visible that the most common of these themes are related to the project team. This includes for example risks of not having clear responsibilities inside the project teams, having challenges with the communication and project team not having enough or sufficient skills to be successful. Several interviewees mentioned also risks related to working in the multicultural environment and with external teams. These topics were also put under project team category.

The second and the third common themes were related to user and customer experience and not having a correct scope for requirements. The user and customer experience in this context cover everything from not considering all/correct user or customer groups to possible challenges in designing the visual look and feel for the solution. The requirements scope means difficulties with identifying what features or parts of a feature the requirements should be describing and challenges in defining the border between other features. Even though several of the existing risks in the checklists touch the topic of requirements scope, some interviewees felt that having incorrect scope for the requirements can be a wider topic that should be addressed as a separate risk item.

The last common risk theme is agile development related. This includes issues that may arise when using an iterative development approach. Examples mentioned in the interviews were challenges including the new feature or system as part of the existing solution or architecture and the challenge of focusing on small enough increments in the development cycle so that agile approach is efficient.

5.2 Evaluation of the data the method produced

The second part of the interview focused on evaluating the data the RRP method produced when it was used for requirement risk assessment. Interviewees were asked to review the risk profile they had created and comment if it accurately represented the requirement risk environment of the research case. In general, the interviewees agreed that the method produced valid information for the research case. This is also supported by the results from the first interview phase where the number of risks marked as not valid was relatively small. None of the interviewees pointed out anything that would indicate that risk profile they created would have been incorrect or completely irrelevant. True, some of the risks in the checklists were not valid for the case in question, but the participants shared a general agreement that those might be relevant in some other cases or different projects.

> "I think it considers relatively well all the relevant topics for this case…"

> "Hits the main points for sure"

When it came to the question if the method helps to recognize risks, participants were not as unanimous. When some interviewees stated that method helped them to spot relevant risks and even to notice issues they would not have thought without it, others didn't feel the method extended their current knowledge of the project's risk situation. This dichotomy was similarly visible in the risk profile summary analysis. Interestingly, those interviewees who were not directly working with requirement management process (they are not working as a product owner or architect) were more positive about the method's ability to recognize relevant risks. For the people who have their background in the requirements engineering, the method seemed to focus on risks in slightly too general level. They felt that the risk profile they produced didn't bring that much new and they described the content to be rather obvious.

> "I had a hard time to find a correct mindset for this (identifying risks). It was hard to decide if a risk should be marked as important or not because all the risks are quite generic. I find it hard to think of any project where these risks would not have any impact. If I would think of any other project, the result would be almost the same."

> "I have not spent a day in a school about these things and I thought all of these (risks) by myself just by imagining doing a project from A to B".

The general nature of the risks was not considered only as a negative thing. For others not diving into detailed topics ensures that the tool they use doesn't bring up contrived risks and helps to avoid unnecessary problem-seeking mindset. For these participants, The RRP method performs as a guide to correct direction - not as a strict plan of action. These interviewees saw the method more as a starting point for a risk analysis than a strict guideline to follow to get direct answers. Using it still requires brainwork from the one who does the analysis.

> "…These topics are high level enough so that they don't force me in a certain direction and on the other hand it doesn't bring up made-up risks…"

> "I'm not sure if I can assume that this method alone helps me to recognize all the risks, but it might push me to the correct direction and point out things I would have not thought otherwise. I don't fully trust the idea that just by going through all the checklists would be enough"

> "I feel that these are very general level topics from which the one who does the risk analysis has to recognize the ones that are relevant for this project. If you are not able to do that or you miss something, this method doesn't provide any answers. It guides you to the correct direction, but at the end, the work is left for the risks analyzer."

Other comments about the nature of the risks in the checklists related to their interconnectivity. The *client commitment* and *communication gap between co-workers* were given as an example of this. Clients are considered as an important project stakeholder in the case project and also the agile project's in general. The interviewees commented that it is a risk itself to focus on things separately when they actually are just an implication on the same problem - in this case, the lack of

sufficient communication. This kind of causal relation was found between different risks. Other examples mentioned in the interviews were: *misunderstood business needs*, *missing requirements* and *delivering what client needs.*

The fact that risks in the checklists focus more on the business-related issues rather than requirements' relation to the technical implementation, received praise. This focus was generally perceived positively as it supports the focus on the importance of the requirement management process and enforces stakeholder collaboration:

> "I feel this (method) covers our project risks pretty well. It doesn't consider too much the implementation though and all the related issues from the technical side. I'm not even sure if it should be in a scope of this method…I actually think it is better that they are not included"

> "The method showcases well the importance of the business-related challenges. As long as you have a clear picture of what you need (from the system) the technical side will be doable one way or another."

All the interviewees agreed that some of the risks have been recognized already in the earlier stages of the case project - some even stated that almost all of them. The fact that risks have appeared in the project's history further support the method's risk selection's representativeness of real-life development environment.

During the interview, several interviewees struggled with evaluating the impact of the risks. As also apparent from the risk profile summary (Table 8), it seemed to be more natural to give a high or medium impact to a risk than a low one. One interviewee asked if she could mark all the risks with high priority. Without a big difference in impacts, the interviewees didn't feel that the method helped them to prioritize risks.

> "Majority of those impacts I added were either high or medium. Based on that, how should I be able to prioritize them (risks)? I don't feel that the method brings additional help for that".

When asked why low impact rating was hard to give, one common answer was that user felt that all the risks have a chance to escalate. At one moment, the risk might be low, but then when the next iteration comes, the situation might change drastically. Impact evaluation, in general, didn't seem to be a natural way to create an understanding of the risks severity for the case project. Participants were perplexed when they were only asked to focus on the risk impact instead of risk probability. As one interviewee stated, the risk likelihood is as important when assessing to which risks to focus on first:

> "All risks are big (have a high impact) when building something complex. It is like in a brain surgery: Even if you cut just one wrong vein, it screws up tons of stuff…The likelihood of something happening is much more useful for evaluating the priority than the possible impact."

This relation between the risk impact and the risk probability was one of the biggest challenges when using the method for the risk prioritization.

5.3 Perceived usefulness of the method for the agile software project

The last phase of the interview focused on the RRP method's usage in the agile software project. The topic of the method's representation of requirements risks in high level was also present in the interview answers for this last interview phase. When looked from the perspective of the whole project and project work, this scope for viewing risks was considered to be a positive thing. Majority of the interviewees mentioned that the risk profile they created for the case feature, could easily be from any of the other development projects the case company is currently working on. In this regard, they felt that the method doesn't have such limitations that it would not fit for the company's current development environment.

When asked if the interviewee felt that using this model would be useful for the whole project the answers were divided. One of the most evident challenges seems to be the scope in which the RRP method should be used in the agile development project. When the end-product is developed in several iterations - some bigger and some smaller - there is no clear point when the method is best to be used. Interviewees also concluded that finding a proper time to do such an analysis might be hard. In the case company, the duration of one iteration might vary significantly and several iterations can be in development at the same time. Doing the risk analysis, the way method suggests, might become quickly overwhelming if the scope is not clearly defined.

> "In agile development, there is no clear point when this kind of analysis should be done"

> "If we think our agile development, we have a focus on features which might be small. I think this model might be too heavy".

> "For one feature development, where there is a clear start and finish doing this might be possible. Then there is a clear life cycle to follow. But if we think about our project, then it is more difficult to define when this should be done."

Some participants even questioned the need for the whole idea of incorporating additional risk management steps to the agile development process. Their argument was that agile development in principle already includes risk management for example in form of retrospectives. They felt that the method might prove to be too comprehensive for the agile needs and add an unnecessary step for the development process. Because efficiency and getting rid of everything redundant are the corners stones in the agile development, time and effort put to this kind of analysis were seen as unnecessary.

> "I would say this model is not useful for us. This iterative way of ours...of doing the development...it is continuous. We are all the time trying to spot problems in our processes and ways of working on the fly. I think we should not try to put it to any heavier model."

Time resource and scope challenges lead the majority of interviewees to the conclusion that the way they would use the method would be for predefined, bigger entities that require specific risk management actions due to their complexity or business criticality:

> "I think this could be done for bigger sets of things. Instead of one feature iteration, I would see this as a tool for bigger projects or parts of a project"

> "Doing this for every feature might be good, but would we be able to - I doubt that. I think we could do it for high-level issues, at the beginning of a project and to any other entity that we recognize is big enough that requires a deeper focus on the possible risks"

Interview answers suggested that one way to use the RRP method in agile development is to utilize it as a communication tool. The level of information the method presents could help sharing information between the project stakeholders, especially when people share a different level of understanding the software development. According to the interviewees it could be used to communicate the project's challenges (or likely challenges) to help decision makers to know to what issues should be addressed. The method could also help to emphasize the connection between different issues:

> "If I would have to explain to a decision maker what is wrong, this could be an eye-opener that which issues are related. For example, that it is not the implementation that is the problem, but actually we had challenges already in the requirements phase. This is not really recognizing risks, but more communicating them through a known model to someone who is not familiar with the whole development process."

> "At the beginning of a project, this could be used to showcase which things need to be in order for all the different parties"

> "This could be a good way to communicate about challenges in our processes. I would not do this to all of our projects, but an individual example and that way bring up those issues that we should remember to address."

In the interview, participants were asked when they would use the RRP method. The common agreement between the participants was that the method would be most beneficial at the beginning of the project or development iteration. One argument supporting this was that the majority of the risks will follow through the whole development process. If those are not addressed in a timely manner at the beginning, that will cause problems throughout the development process. Interviewees also felt that if the risk is skipped in the previous phase, its effect is multiplied in the following phases.

> "In my opinion, many of these risks affect through the development process. If we, for example, don't have ownership from the client side at the beginning, it will have repercussions even in the implementation phase"

> "Well if the requirements have been misunderstood at the beginning, the problem will reappear in each and every process phase."

> "Now that I read this list (checklist for the design phase), I would say that all the risks from the previous section would be here also and their effect is multiplied. If we would design and implement something with a poor understanding of the actual business need, it doesn't change any better later on"

Following the method's approach and doing the analysis for each development phase individually could, on the other hand, increase the likelihood of spotting issues that were skipped. Few interviewees pointed out that since all the different phases include different tasks and different people, the risks analysis should be also done again - even though the majority of the risks would have been recognized and even addressed earlier. The phases would also help the development team to track which of the issues are under control and which might need additional action.

During the interview, another interesting topic arose which is not only related to the agile development: who could be competent to use the model? While doing the interview, the differences in a person's focus and how they perceived each risk was remarkably different depending on their role in the case project. This naturally affected the way the interviewee understood each risk and how he or she evaluated their impact. In the case company, different phases of the agile development have been appointed to different people and several individuals affect the requirements during their life cycle. This means also that the knowledge and overall picture is divided among multiple people. Interviewees pointed out that using the RRP method requires a quite deep understanding of the IS projects, system development, and the company's organizational environment. The user of the method should be able to reflect different business areas and should not be limited by the method but also be able to evaluate risks outside the lists.

Lack of proper requirement risk management process seems to be a challenge also in the case company. Interviewees coincided that the some of the risks in the checklists have realized one way or another in earlier developments. As seen from the comments below one participant mentioned that even though almost all the risks in the checklists have been acknowledged, the project team hasn't made actions based on that information. Another participant felt that even though risks have been recognized, the team doesn't have a possibility to influence them enough.

> "I felt it hard to get help from this (method) that I would have not known before. Still, by using this, I think we could be able to make such an observation that we normally ignore although we shouldn't".

> "We have recognized most of these already and also tried to do something about them. Still, because of reasons we cannot affect, we have not been able to fix them. This is why I didn't feel that this method brought any value to me. I can imagine if someone who had never thought why our process is not working, he could use this and get something out of this."

If the RRP method would be a good option for the case company as a tool to manage requirements risks, participants were not sure. Having a general approach and slightly unclear scope seemed to be the main reasons why not. As the method don't propose direct answers, even though the resolution techniques would be involved, participants felt that the method lacks concrete actions to take. When compared to any other risk management method, choosing this one didn't seem to be an obvious option for anyone. In turn, the method was praised of its ability to present risks in a way that in the skilled hands the method could serve as a communication tool between different stakeholder groups and really help to address the requirement related risks.

6 DISCUSSION

By combining results from both contemporary research literature and this case study it is possible to identify topics that help draw a picture of how requirements risk management is done in agile IS development projects. This research also manages to highlight common themes which are important to focus on in agile IS development projects showcasing at the same time how selected development method indeed can affect the requirements risk environment. This chapter describes in detail these specifics of requirement risk management in the agile project by comparing findings from this study case to the current literature.

6.1 Managing requirements risks in agile IS development project

Reviewed research literature and findings from the case study highlighted one key attribute of the requirements engineering process in agile projects: its iterative nature. It is very clear that when IS development project is using agile development methodology, requirements management process simply cannot be stiff and heavily structured. This is the most evident when requirement elicitation is considered: Even in agile IS projects, a heavier focus on requirements elicitation is often put at the beginning of the project. However, in agile projects, this is just a mere starting point. Original set of collected system requirements are refined as the development process continues and the focus moves to smaller iterations. System requirements in agile projects are typically highly volatile and evolve throughout the development lifecycle.

This has a lot of implications for the requirements risk management: Risk profiles or evaluations done at any given moment might be old tomorrow. This means that requirement risks understanding needs to evolve along with the changes for the requirements set up. All kinds of ISD projects will benefit from continuous risk management actions, but current literature strongly emphasizes it in relation to agile development. (Wiegers & Beatty, 2013; Lyytinen, Mathiassen & Ropponen, 1996).

As responding to constantly changing environment is one of the four values of the agile development (Agile foundations: Principles, practices, and frameworks, 2015), agile philosophy has some inbuilt mechanisms that help continuous risk management process which, in turn, helps management of requirement related risks too. The whole agile process is based on the idea of constant feedback loop which guides future development iterations. Measey (2015, 38) showcases in his generic agile process framework this loop and how the previous development iterations provide new knowledge to guide the future development. Doing things iteratively means that for example one feature may be built in few smaller increments - one iteration improving the previous one. For bigger features, this might mean that several development iterations are needed before

it is considered to be ready. New information gathered from the previous iterations may affect the requirements of the new iteration and changes are required.

Different agile methods offer a variety of practices to gather information from the prior development iterations. In the case company, for example, this was achieved with different types of retrospectives. These serve as an instrument of self-correcting both the development process as well as the development plan. In agile projects, these moments are natural points where changes to the project's requirement risks appear. Those are then also natural checkpoints where requirements risks can be reanalyzed accordingly. In the research case, some interviewees even felt that this agile feedback loop is so effective that specific risk management actions toward the requirements risks are not even needed.

As mentioned earlier, also agile projects have a heavier focus on the requirements elicitation at the beginning of the project. At that time requirements are higher level and so are risks relating to them. From the more practical point of view, this research concludes that agile projects will benefit from having a higher-level risk management discussion at the beginning of the project or bigger development cycle. This was visible in the case study results where many interviewees were seeing communicational challenges as a high impact risk and contributor behind other topics which were mentioned in the RRP method. Having this kind of discussion would increase the information sharing and understandability of the projects risk environment and create a baseline for the future discussions. Case study interviewees were also positive that they could use a tool such as the RRP method to support this kind of discussion.

In addition, both research literature and case study results showcased that requirement risks management should involve not just the project team but other stakeholders' groups too. This creates an environment which enables transparency needed for agile projects where customer and other stakeholders are expected to be active contributors for the project. Collaboration between stakeholder groups (especially customer) in the moments of change is important because requirements are highly affected by topics which are not obviously related to the development itself. For instance, organizational decision making, external regulations or even personnel changes are topics which are often not in direct control of any development project team members but can have a significant effect on requirement related risks.

After higher level understanding about requirement risk environment is formed for the agile project, the scope of the requirement risk management needs be focused to the correct level when proceeding to a cyclic development flow. Results from this study indicate that this is the point when requirement risk management starts to become harder. At this stage, more is required from the people who do the requirement risk management. They need to understand inputs from the previous iterations and apply the new knowledge at the same time understanding how it will affect the requirements risks in the iteration level but also if those affect the big picture too.

It also more challenging to find tools which support this process. As the scope and the level of risks can vary depending on the development cycle, a simple readymade list of risk, for example, is not able to support all the variations for the risk assessment. Even though different checklists are a common way to support project risk assessment (Boehm & Ross, 1989; Boehm, 1991; Schmidt et al., 2001), in agile development these can serve only as a mere starting point and should be always adjusted to fit the current development scope. The level of risks presented in the RRP model support discussion about relatively big entities or even whole projects. In the case study, several interviewees stated that the scope selected for the research case (focus feature iteration) did not feel appropriate. Their comments about risk content in the checklists being too high level or general to be useful highlight how at the feature level requirement risks need to be already quite focused.

This was also visible in other interview results: the high-level focus of the risks in the RRP method's checklists seemed to affect how useful study participants perceived the method to be. Several of them commented on how it did not bring anything "new" for them or was too general to support their risk management work in feature level. This can also be supported by the observation, that people working directly with the feature development were less confident with the method's checklist content than the ones having more distance to the development process in the study case. This might be because the latter group does not have to deal with the scope changes of requirements risks similarly as the people working directly with the development process do and hence they are happier with the high-level focus of the risk discussion.

6.2 Requirements risks environment in the agile project

Requirement management revolves strongly around the needs of a customer and transferring those at the end to a working software. Current literature has several lists of common requirements related risks and most of them seem to focus on the nature or quality of the requirements themselves, for example, their volatility, completeness, and ability to represent clients need accurately. (DeMarco & Lister, 2003; Lawrence et al., 2001) For example, Mathiassen et al. (2007) have characterized requirements related risks based on these characteristics. However, requirements are affected by topics in the environment where they exist. Agile methodologies promote a certain kind of development environment which naturally influences what kind of risks requirement engineering process faces.

To investigate what are typical requirement related risks, risk checklists from the RRP method were used as a starting point. The method was not developed especially with some development philosophy in mind (Tuunanen et al., 2018) and so the risks in the method's checklists offered a good reference point to start the investigation from. When these checklists were analyzed during the case study, there was a strong agreement with the participants that the majority of the risks were relevant to the study case. This has two major implications for this

study: firstly, risk checklists of the RRP method seem to represent true requirement risks which are also valid in agile development. Secondly, selecting an agile development method does not save a project from some type of risks. Overall, these research results suggest that requirements management process is plagued by similar risks when agile development methods are used compared to more structured methods. Using agile development methods does not seem to make any of the topics mentioned in the checklists completely irrelevant. The difference lays more in emphasis of risk types and in which scope those should be analyzed.

When checklists from the RRP method were used as a reference, this study revealed some themes which seem to be especially important in the agile project. Several interviewees also felt that these themes were not addressed enough by the RRP method's checklists. This does not mean that these topics are only agile development specific, but based on this case study, those might have a more prominent role in agile development. These themes were: *Project team*, *Customer and user experience*, *Requirements scope* and *Agile development*.

The first of these themes, *Project team*, has its roots in people and communication. In the core of agile thinking is the focus on collaboration. This is especially true in requirement management where even the whole process could be considered as a flow of communication - communication of the client's needs. Likewise, as Wiegers and Beatty (2003, 546) state, managing requirement related risks is a collaborative action between all the stakeholder groups. While creating risk profiles for the focus feature, several interviewees felt that stakeholder communication-related topics should have been in prominent focus. It was also possible to see from the results that challenges in communication and collaboration could be a reason for several risks in the checklist. Addressing these issues could actually prevent certain risks from happening. *Misunderstood business needs* or *delivering what the client requires* are good examples where collaboration issues may have played a role. Since the communication and collaboration have so important a role in agile development, not focusing on these topics seems to be a high-risk item on its own.

Communication challenges may be a result from different issues, but one very typical one in the project such as the case company has, comes from the physical distance. It is not uncommon that the ISD project's stakeholder groups are in different locations - even different countries and time zones. The actual programming can also be done from multiple locations. The case company offered a good example of this kind of scenario with is multicultural development teams which are spread across the globe. This scenario is not, by all means, agile specific, but when combined with a high emphasis on small self-organizing teams, it is clear that agile projects demand caution in this area and requirements risks related to that may play a bigger role in agile projects.

From the people side, this theme points out the risks which can arise from not having clear responsibilities or enough skills among the team members. Even though agile methodologies promote self-organizing teams and flexible leader-

ship structures, this does not mean that agile projects perform without any structures, quite the opposite. Project environment where changes are constant and fast, the responsibilities need to be clear in order to be able to achieve project's goals. On the other hand, flexibility is born when individuals have a wide range of skills and knowledge. When the agile project team lack clarity on responsibilities and sufficient skills and knowledge, this will create risks which have a direct effect on requirements too.

The second agile specific theme which was brought up during the interviews related to customer's or user's role in the agile project. How a client is often seen as an active contributor to an agile development project is widely discussed in the current literature. (Agile foundations: Principles, practices, and frameworks, 2015; Cockburn & Highsmith, 2001; Highsmith, 2002; Paulk, 2002) This means that in requirement engineering process the customer is not expected to give finalized "order" at the beginning of a project but act more as a direction giver. In this role, the customer guides the process and ensures the business needs are met with the team - not from the outside. Being a customer of an agile project can also be demanding. The customer is expected to be available to participate in the development process and like Paetsch (2003) mentions, agile methods often assume that customers are competent enough to make good decisions.

When doing a risk assessment for the focus feature, some interviews felt that customer's role was not considered enough by the RRP method's checklists. They found several risks which they would have included in the checklist to support this customer-centric view agile project requires. These risks varied from the lack of ownership and overview from the client side to not considering the user experience enough in the end-product from the look and feel point of view. These results highlight the high expectations for the customer in the agile project and how client's relationship can generate risks both sides: customers relation to the project and the project's ability to answer to the customer's needs.

The importance of these topics can be also seen in the risk profile summary and which risks interview participants felt to have the highest impact on the research case. Risk *Ambiguous requirements* and *client commitment* were rated as high impact risk by most of the interviewees. Both of these can have heavy relation to customer's or user's relation to the development project. When combined with the case study result that *identity* types of risks received the most high impact ratings out of all the other risk types, it seems clear that this theme seems to be especially critical in agile projects. *Identity* type of risks relate to the availability of the risks such as communication challenges and overlooking a certain user group.

Based on the interview results, the RRP methods seem to have a slightly more buyer-supplier view of requirement management process: Interviewees felt that the responsibility of understanding the business need is more on the supplier side and the risks arise mainly when requirements management process fails to capture these needs correctly. Based on the literature, agile methods try to break this setup and customer's role is expected to be an active contributor to the ISD project. (Abrahamsson et al., 2002; Cao & Ramesh, 2008; Manzo, 2002) In an ideal

agile project customer relationship is not just one -way line. Instead, understanding of the developed solution is built together with the customer and development team. Assuming that the customer is able to fill this role creates its own set of risks. When those are not correctly addressed, the development project might face unexpected challenges.

The RRP method does mention risks such as *client commitment* and *constrained user knowledge*, which in a way, address this topic. However, those are assigned to the requirements phase of the development process. In agile projects, customer involvement continues ideally throughout the project lifecycle. Having focused on customer-related topics during the requirements elicitation only might cause problems in the later stages of the development. Results from this case also support the argument that focusing on customer relationship related risks is important during the whole development. Using a tool such as the RRP method, in turn, could offer required transparency so the requirement risk management can be done with the customer and so risks related to customer's own role can be minimized.

The third and fourth risk theme considered requirements scope and issues that arise from the agile development process itself: How continuing development will make it challenging to draw the line between features when developing software in iterations and how to make sure individual iterations fit as part of the bigger whole. These two topics a lot to do with each other. Agile methods underline how doing small increments and adding them to the bigger whole ensures that building working software is more manageable. (Agile foundations: Principles, practices and frameworks, 2015; Abrahamsson et al., 2002). Case study interviews revealed how this thinking can bring some challenges. In bigger projects, an individual feature may have dependencies from technical as well as requirement point of view to other requirements, features or event other projects. What is selected to be achieved within one development iteration, in turn, is not always aligned with all these dependencies. This means that understanding how these dependencies effect on requirements risks can be really challenging.

6.3 Using the Requirements Risk Prioritization method in an agile project

In this case study assessment of requirement related risks was done by using the RRP method. Developers of the method argue, that the method could be used in all kinds of development projects. In a way, it would be easy to assume that the method covers the aspect of continuous risk evaluation agile projects benefit from by assigning risks to different development phases with the checklists. When requirements change, the analysis could be repeated, and new, updated risk profile created based on the new information. However, this study highlights some of the challenges in assessing requirement related risks in agile ISD project and how using a tool such as the RRP method could be challenging. On the hand,

this study highlights how this kind of method could, in turn, help requirements risks management in agile projects.

Working in the highly collaborative environment requires people to share at least somewhat same level of an understanding of any given topic. Interviews revealed that discussing requirements risks with different stakeholder groups can be challenging due to different perspectives and level knowledge they share. Several interviewees stated that having a tool that supports the communication of different requirements related challenges with different stakeholder groups, could help the agile project to be successful. Because the scope of the RRP method is relatively general, risks that it directs project stakeholders to focus on are more likely to be understandable and relevant for all the stakeholder groups. This case study offered a good example of how a tool such as the RRP method can help people from different functions to discuss the same topics and point out common challenges. Several interviewees concluded that even though as it is, the method would not support their risk management activities in daily work, it would be a good tool to support communication with the other stakeholders about case project's requirements risks.

Agile methods promote flexibility both in roles and responsibilities as well as avoiding bureaucratic organization structures and controlling management styles. This means that agile project team members are often preferred to be generalists instead of specialists. Typically, an agile method also discourages doing heavy documentation. (Agile foundations: Principles, practices and frameworks, 2015) In this kind of environment key information and knowledge is often distributed to several individuals and is not centrally documented. Having all that in mind, one relevant question is: Who would be able to analyze requirements risk reliably in an agile project? There are some recommendations in the literature about how the person should have deep enough understanding of the actual business/client needs, organizational environment as well as the development processes. (Konya & Sommerville, 1998). This case study revealed how it is also important that a person's position in the organization allows access to the required information. One person rarely poses all these qualities. This means that, like every other aspect of agile development, also requirements risks assessment must be a collaborative effort.

During the interview, one of the biggest criticisms towards the RRP method was about using the risk impact as a measurement of risks importance to the project. Typically, ISD project risk management does not only rely on the risk's possible impact but includes also risks probability to the analysis. For example, Boehm's (1991) calculation of risk exposure includes both the probability of a loss as well as the size of a loss. The same combination is present in U.S National Institute of Standards and Technology's Risk Management Guide for Information Technology systems, where writers state that risks should be considered by both probability and the impact of occurrence (NIST, Goguen & Fringa, 2002).

The RRP method as a tool directs user's focus to only evaluate risk impact. This result rather one-dimensional visibility to a situation that in real life is more complex. When the risks prioritization is done based on only the impact, it is

possible that risk management efforts are directed to prevent something very unlikely. On the other hand, focusing only on the risk probability, limited resources could be wasted on minor issues. It is true that the model's checklists consist of risks that are proven to be common in requirement management and in that sense their probability is already high. However, as the checklists will never represent the full spectrum of the requirement risks of the project, the model should better encourage to two-sided risk evaluation to avoid incorrect risk prioritization.

Managing requirements related risks cannot be done successfully if it is isolated from the overall ISD risk management (Wiegers & Beatty, 2013, 542-546). Talking about requirements risks separately might even feel artificial - after all, the requirement management process is an integral part of the overall development. Interviewees in this study seem to have a hard time to orientate just to focus on the requirements level and talking about requirement risks felt unnatural. This is most likely due to the fact that requirements are affected by a wide range of topics surrounding the development project. Similarly, on the other side, requirement related risks rarely have only effects on requirements but can hinder the success of the whole development project. The RRP method seemed to make it easier for case study participants to understand these relationships in their own project.

6.4 Implications for the practice

This research reveals some topics which could help managing requirement related risks in the real-life agile projects. One of the research questions of this study focused on identifying typical requirements related risks in agile ISD projects. This research used a risk checklist from the RRP method as a reference point for risks evaluation in the case study. These requirement risks were considered in this research context not to be dependent on any specific development method. As stated earlier is relatively safe to say that agile projects face a lot of the same challenges as do projects where more structured methods are in use. However, while analysis risk profiles created during the case study it was possible to highlight themes which seem to be especially important for the agile projects: *user and customer experience, project team, agile development, and requirements scope*. These risk themes are discussed more in detail in chapter 6.2.

One of these themes is specifically about the agile development process and the other three have also a strong relation to the agile principles and values (Figure 1 and Table 1). This can indicate that these themes are common for all kinds of agile projects, not just the one presented in this case study. However, to confirm this, more research is needed. Even more so, although there is no possibility to conclude exactly what are typical agile project's requirements risks it is important to know for agile project practitioners that there can be some. Based on this research, selecting an agile method for the development project can affect the project's risk environment. Even more importantly: The selected development

method can affect the project risk environment whether it is an agile or more structured method or anything in between.

From the more practical point of view, people and collaboration - agile values should be extended also to the requirements risks management. One person in the agile project cannot have all the required information to evaluate requirements risks accurately. Involving the project team and stakeholders is crucial to draw a reliable picture of the requirements risks the project might face. Based on this research requirements risk management in an agile project should be transparent, cross-team and cross-functional and preferably involve customer as much as possible. Using a tool such as the RRP method can help in this collaboration. When the focus of the development moves to a more detailed level, also the scope of the requirements risks analysis should do so. At that point when the overall understanding has been formed earlier, changing the scope of the analysis to smaller entities can be easier.

Lastly, managing requirement risks should never be done in isolation from other risk management activities. This is true for all kinds of projects. Having a tool that only addresses certain types of risks can lead to a situation where other issues are overlooked. This case study showcased how project team members who did not have requirement risk management process established per se, felt that they had addressed or at least recognized most of the topics presented by the RRP methods checklists without a specific focus on the requirements risks. Yet recognizing requirements risk as a key risk type as well as understanding how different issues affect the requirements engineering process brings important insights into the overall risk management and increase projects success rate.

7 Conclusions

System requirements play a significant role in guiding IS development project from initial idea or need to the implementation of desired functionalities to a working software. Over the years, researchers have concluded that shortcomings in the requirement management process contribute heavily to the IS development projects success. After the emerge of agile methods the IS field has been introduced with development methods which focus on smaller increments, flexibility, and success through constant adjustments of the plan. The goal of this master's thesis is to uncover if agile development method affects requirements risks in projects where such methods are in use. It also aims at understanding how requirements risk management is done in agile IS development projects and showcases some topics which are more likely to affect more requirements risks in agile projects than projects where traditional development methods are in use. Lastly, this research showcases practical example of requirement risk management in an agile project with the help of RRP method.

This research consists of a review of related research literature and a case study where the RRP method is used in a real-life agile project. The literature review includes articles about requirement engineering in agile projects as well as requirement risk management as part of overall IS project risk management process. In addition, it introduces the RRP method which is later used in the empirical part of this research. Combining all these topics creates a theoretical framework for the case study to uncover answers to all research questions.

The empirical part of this research was conducted as a single case study. The study subject was an on-going agile software development project and the scope of interest was one system feature development. Case study participants were all part of the project team or close stakeholders with sufficient position to affect projects requirements risk environment. The case study was conducted as a semi-structured interview where interviewees used RRP methods to create a risk profile for the feature development. After risk profile creation, the interview continued with pre-defined questions which aimed at evaluating the data interviewees generated with the RRP method and evaluation of the usefulness of the method for the agile project work. Interview results were analyzed by combining the data of the risk profiles created by interview participants and conducting a thematic analysis for the interview question answers.

Results from this research conclude that agile IS projects also requirements management process is iterative which has a direct effect on the requirements risk management. From the simple process step point of view, requirements risk management does not seem to differ in agile and more structured development methods. In the end, it will always come down to identifying risks, analyzing them and finally resolving them in one way or another. However, in agile projects, the scope of risk analysis, as well as timing, should be aligned with the current development phase. When software is developed in smaller increments,

the requirement risks relating to them may also vary in size, criticality, and likelihood. The risk profile of any given moment can be old in the next after some changes have been done. For successful requirement risk management, this means that the analysis of risks should be continuous.

Despite these continuous efforts of managing requirement risks, agile projects can still benefit from the higher level scope of requirement risk analysis at the beginning of the project and bigger development iterations. Forming this kind of overview level understanding of the project requirements risk environment can help later more detailed risk management actions. Overall, the risk analysis should preferably involve different project stakeholder and even customer. This ensures that requirements risks, which are not only affected by issues in a development project, will be analyzed sufficiently and accurately. This also increases transparency among the project stakeholders, helps to recognize relations between different topics and reveals how changes can affect development project.

After a high-level vision for the requirement risk environment is formed this information should be used to support more detailed risk management. Agile processes support this idea with the constant feedback loop. This means that feedback from the previous iteration is used to fix any possible issues in the future ones. The knowledge of requirement risk environment of a whole agile project is created by combining both high-level understanding and feedback from smaller scope development. Depending on the agile method, this feedback is analyzed in different agile ceremonies (for example retrospectives) which are typically happening after each development iteration before the start of a next one. These moments are also good points to revisit higher level requirement risks and adjust the plan for managing them.

When considering the individual risks, agile projects seem to suffer similar risks than projects which use more structured development approaches. From the results of this case study, it is not possible to conclude if there are any specific requirements risks which project team in agile projects should always focus on. However, case study results pinpointed four themes which can be more prominent in agile projects compared to the ones not using agile methods. These themes where Project team, Customer experience, Requirements scope, and agile process. Project team theme relates to issues such as challenges in communication (internal and external), unclear responsibilities and insufficient skill set inside the project team. Customer experience, for example, means not being able to recognize all the correct user groups or not paying enough attention to the user experience in the end-product. Requirements scope included risks such as not being able to draw clear enough border between features or increments. Lastly, the agile process relates to for instance not focusing on small enough increments or problems with integrating selected feature as a part of existing architecture.

These risk themes and their close relation to the agile development, agile principles and values suggest that these can be present in different kinds of agile method. Even more so results indicate that the selected development method can affect the project requirement risk environment. This, in turn, means that while

forming the overall understanding of the requirements risks the project might face, the project team should be aware of how the selected development methods may affect them. The project team should also select such tools and processes for the risk management that best support requirements risk management in their type of development project.

Current literature does not offer many tools to manage requirements risk in agile IS projects. The RRP method by Tuunanen et al. (2018) is a tool that was used for requirement risk management within this case study. Developers of the method state that it can support requirement risks management in all kinds of projects - including agile. This research results both support and do not support that argument.

According to this study, RRP method can support agile projects at the moment when the higher-level understanding of the requirements risk environment needs to be formed. At this stage, it can help stakeholders to focus on correct things and not overlook any important topics. Based on the result from this case study, the method can help collaboration and communication about requirement risks at the level that it is understandable for all the projects stakeholders. Typically, in agile projects key information is often spread among several individuals. Having a list of risks where issues are discussed in the language that everybody can understand can increase the likelihood of managing those related risks successfully.

The RRP method includes three different checklists for different development phases: requirements, design, and implementation. With these lists the method can in theory support a cyclic development process. It would mean that these phases are repeated for each development cycle. However, in agile development, as the scope of each iteration can vary, using static checklists for the risk management is not the most effective option. Based on the interviews, already for individual feature development, a big part of the risks presented in the checklists felt too high-level and could not accurately guide the risk analysis work to the correct scope. This means that as the method works well for the high-level discussion when used to support requirements risk management work for a more detailed level, it can only be used as a starting point.

The last point about the usage of the RRP method for managing requirement risks in an agile project was about actual prioritization. Interviewees felt that the method could not help them to prioritize requirements risk mainly due to its unilateral way of evaluating risks severity. The method guides its users only to evaluate risks impact on the project and not include the likelihood to the analysis. This approach can lead methods users to direct risk management resources toward preventing very unlikely risks. On the other hand, it can lead to overlooking small impact risks which do not harm the project drastically but are still unwanted.

As an important part of IS development, requirements and risks relating to them should not be overlooked. As shortcoming in the requirement engineering process has been proven to be one of the key contributors on statistically high

number of IS project failures, limiting risks in the requirement engineering process should increase the IS projects success rate in the future. Managing requirement risks in an agile project is not too different compared to a more structured project. However, understanding the effect that the agile process does have on the requirements and their life cycle is crucial for successful requirement risk management. It starts by understanding agile values and principles which are at the core of every agile methodology. It continues continuously and consciously managing requirements risks in the scope of each development phase and utilizing correct for tools – at the same time understanding the limitations those tools can have when agile development is in question.

7.1 Limitations

This research has some limitations that should be considered. Firstly, in the theoretical part of this study, limitations relate to this study's ability to cover enough research literature and to have wide enough scope for the literature review. Current research literature offered only a few sources that would have addressed requirements risks specifically and many previous studies discuss them in a scope of overall ISD risk management instead of a separate focus point. This means that in this study the way requirements risks are considered relies heavily on those few available research papers, especially Tuunanen et al., (2018) view of requirement related risks. Including other sources, for example from ISD risks management or different key standards of requirements engineering, could have increased the representativeness of the literature review.

In addition, this study interprets and describes different agile methods at a very general level. The Measey's (2015, 38) framework for generic agile development is used as a baseline to understand key concepts of the agile development and what agile methods are. Specific techniques or ways of working in different agile projects are mainly discussed from the research case's point of view. This means that the results of this study do not cover all the techniques from different agile methods which could in practice positively or negatively affect the requirements risk management process. In addition, the case company does not provide a pure representation of any specific agile method. Therefore study results are not fully representative for any specific agile method (e.g Scrum).

Secondly how case study was conducted brings up some additional limitations. Naturally, one case is not descriptive enough to form a conclusive overview of the research topic. This case study offered a real-life example of how requirements risks are managed in the organization that uses agile development method in a very interesting collaborative environment but having more case examples to compare would increase the accuracy of the results. Comparing different cases could also help to limit project related circumstances which may have affected the results of this study and which were not identified in this study.

The selection of the case itself limited the number of interview participants who could participate in the study. This was due to both a relatively small number of project team members and other stakeholders who shared the level of knowledge about the project that was required to be able to participate. This also limited how much background information could be collected from the interview participants without compromising their anonymity. Having a bigger sampling and wider demography of interviewees would have increased the generalizability of the study results. This would have also increased comparability of the results to other similar studies. However, case company offered a good opportunity to have participants from the different business units who still work closely with the development project. Yet the representation of different business units was not big enough to make a strong conclusion about their effect on different aspects of requirement risk management. This research offers some insight into the topic and indicates that one's role in the project affects how an individual reflects requirements related risks.

Other limitations come from the case scope. At the time of the study, the case project was ongoing, and the case scope was one focus feature development of that project. In a project that has been in development few years, the understanding of the projects requirements and related risks are in a different level compared to the project that is only about to start. This means that the results of this study are not fully comparable to the studies where the project is in a significantly different development phase. Similarly, analyzing bigger development entity, for example, several development iterations, could have increased the validity of the study results.

A semi-structured interview was a suitable data collection method for this study because it gave enough flexibility to the interview situation to form a realistic understanding of the research topic. However, there could have been interesting opportunities to enrich the study results with quantitative data, especially if the case scope would have been wider than one focus feature. Examples of such data are development cycle times, a number of generated bugs, working hours and work estimations. These pieces of data could give interesting insights into the effects of the requirement related risks and indicated which risks have caused the most challenges for the project team's productivity. In this study, these kinds of data could not be used mainly due to time restrictions.

7.2 Topics for future research

Both the study results and limitations of this study revealed some opportunities for future research which could help to increase the understanding of requirement risks management in general and in an agile project environment. Firstly, investigating different agile practices and techniques affect project's requirement related risks could offer fruitful information about how individual agile methods differ in this aspect. This could also reveal if some agile practices can potentially

help to avoid certain types of requirements relates risks or if some agile approaches can increase the likelihood of certain types of requirements risks appearing. This could offer practical tools and recommendations for the agile practitioners to improve their own processes. Similarly, comparing in a case study both agile and more structured projects could further highlight similar topics that a study such as this one may have overlooked.

Equally interesting would be to study the evolvement of requirements related risks when projects maturity increases or if there is a difference in the risk environment depending on the maturity of the project. This study focused on a project that is ongoing but still quite far in its development life cycle. Studying how requirements risk environment changes during the project lifecycle from the beginning until the maintenance, could again guide development projects to focus on correct risks at the correct time.

Requirement related risks can be born from a wide range of sources which are not always directly related to the project or project team itself. This study highlights the important role of communication about requirement risks. Studying the communicational aspects of requirement risk management would be another interesting topic for the future research, especially if it is done from a customer or business point of view. Current research seems to address requirements risks usually from the development projects side which ignores completely the fifty percent of the possibilities to affect on the requirement risk environment. In this study, this was considered by including participants to the interview people from different business units. Unfortunately, due to a small number of representatives from those units, results are not fully generalizable. However, this study shows some evidence that there can be a difference in how a person perceives requirement related risks depending on his or her background and role in a project. Having a stronger focus on the customer, other stakeholders and collaboration between different interest groups in this aspect could be an especially beneficial research topic for agile projects but could surely help all kinds of ISD projects.

From the RRP method point of view, to further investigation is needed to conclude how well the method could serve agile projects. The same kind of observations from the earlier apply also for testing the RRP method: testing the method in agile projects that use different agile methods or in a project with different levels of maturity. Also comparing its usage in agile versus a more structured method in form of case study could reveal further details that can improve the method. Lastly, RRP method's risk resolution techniques were excluded from the scope of this study. However, these techniques are an important part of how the method is intended to be used. Without including them in the process it is not possible to fully evaluate how well method supports requirement risk prioritization in agile ISD projects hence more research is needed.

LÄHTEET

Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta, J. (2002). *Agile Software Development Methods: Review and Analysis.*VTT publication 478, Espoo, Finland, 107.

Agile foundations: Principles, practices and frameworks (2015). Swindon, GB: BCS, The Chartered Institute for IT. ProQuest Ebook Central, http://site.ebrary.com/lib/jyvaskyla/docDetail.action?docID=11022409&ppg=20

Ahmad, S. (2008). *Negotiation in the requirements elicitation and analysis process*. In Software Engineering, 2008. ASWEC 2008. 19th Australian Conference on (pp. 683-689). IEEE.

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

Boehm, B. W. (1991). *Software risk management: Principles and practices*. IEEE Software, 8(1), 32-41.

Boehm, B. W. & Ross, R. (1989). *Theory-W software project management principles and examples*. IEEE Transactions on Software Engineering, 15(7), 902-916.

Braun, V. & Clarke, V. (2006). *Using thematic analysis in psychology*. Qualitative Research in Psychology, 3(2), 77-101.

Cao, L. & Ramesh, B. (2008). *Agile requirements engineering practices: An empirical study*. IEEE Software, 25(1), 60-67.

Cerpa, N. & Verner, J. M. (2009). *Why did your project fail?* Communications of the ACM, 52(12), 130-134.

Chopade, M. R. M. & Dhavase, N. S. (2017). *Agile software development: Positive and negative user stories*. In Convergence in Technology (I2CT), 2017 2nd International Conference for (pp. 297-299). IEEE.

Chua, B.B., Bernardo, D. V. & Verner, J. (2010). *Understanding the use of elicitation approaches for effective requirements gathering*. (ICSEA), 2010 Fifth International Conference on (pp. 325-330). IEEE.

Cockburn, A. (2002). *Learning from agile software development–part one*. CrossTalk Oct, , 10-14.

Cockburn, A. & Highsmith, J. (2001). *Agile software development, the people factor*. Computer, 34(11), 131-133.

Cohen, D., Lindvall, M. & Costa, P. (2003). *Agile software development.* DACS SOAR Report, 11

Curtis, B., Kellner, M. I. & Over, J. (1992). *Process modeling*. Communications of the ACM, 35(9), 75-90.

Davis, A. M. (1993). *Software requirements: Objects, functions, and states* Prentice-Hall, Inc. Upper Saddle River, NJ, USA

Davis, G. B. (1982). *Strategies for information requirements determination*. IBM Systems Journal, 21(1), 4-30.

DeMarco, T. & Lister, T. (2003). *Risk management during requirements*. IEEE Software, 20(5), 99-101.

Dybå, T. & Dingsøyr, T. (2008). *Empirical studies of agile software development: A systematic review*. Information and Software Technology, 50(9), 833-859.

Eisenhardt, K. M. (1989). *Building theories from case study research.* Academy of Management Review, 14(4), 532-550.

Elghariani & N. Kama. (2016). *Review on Agile requirements engineering challenges*. In Computer and Information Sciences (ICCOINS), 2016 3rd International Conference on (pp. 507-512) IEEE.

Erickson, J., Lyytinen, K. & Siau, K. (2005). *Agile modeling, agile software development, and extreme programming: The state of research.* Journal of Database Management, 16(4), 88.

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

Galletta, A. (2012). *Mastering the semi-structured interview and beyond : From research design to analysis and publication.* New York: NYU Press. http://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=575563&site=ehost-live

Glass, R. L. (2006). *The standish report: Does it really describe a software crisis?* Communications of the ACM, 49(8), 15-16.

Glinz, M. (2007). *On non-functional requirements*. In Requirements Engineering Conference, 2007. RE'07. 15th IEEE International (pp. 21-26). IEEE.

Henderson-Sellers, B. & Serour, M. K. (2005). *Creating a dual-agility method: The value of method engineering*. Journal of Database Management, 16(4), 1.

Hickey, A. M. & Davis, A. M. (2004). *A unified model of requirements elicitation.* Journal of Management Information Systems, 20(4), 65-84.

Highsmith, J. (2002). *What is agile software development?* Agile software development. Crosstalk the Journal of Defense Software Engineering, , 4-9.

Highsmith, J. & Cockburn, A. (2001). *Agile software development: The business of innovation.* Computer, 34(9), 120-127.

Hofmann, H. F. & Lehner, F. (2001). *Requirements engineering as a success factor in software projects.* IEEE Software, 18(4), 58-66.

Hsieh, H. & Shannon, S. E. (2005). *Three approaches to qualitative content analysis*. Qualitative Health Research, 15(9), 1277-1288.

IEEE standard glossary of software engineering terminology (1990).

Jiang, L. & Eberlein, A. (2009). *An analysis of the history of classical software development and agile development. In Systems, Man and Cybernetics*. IEEE International Conference on (pp. 3733-3738). IEEE

Jones, C. (1996). *Strategies for managing requirements creep*. Computer, 29(6), 92-94.

Kassab, M. (2014). *An empirical study on the requirements engineering practices for agile software development*. In 2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA) (pp. 254-261). IEEE.

Kauppinen, M., Savolainen, J., Lehtola L., Komssi, M., Tohonen, H. & Davis, A. (2009). *From feature development to customer value creation*. In 2009 17th IEEE International Requirements Engineering Conference (pp. 275-280). IEEE.

Keil, M., Cule, P. E., Lyytinen, K. & Schmidt, R. C. (1998). *A framework for identifying software project risks*. Communications of the ACM, 41(11), 76-83.

Klein, H. K. & Myers, M. D. (1999). *A set of principles for conducting and evaluating interpretive field studies in information systems*. MIS Quarterly, 67-93.

Kontio, J. (2001). *Software engineering risk management: A method, improvement framework, and empirical evaluation*. Helsinki University of Technology.

Kotonya, G. & Sommerville, I. (1998). *Requirements engineering: Processes and techniques.* Wiley Publishing.

Larman, C., & Basili, V. R. (2003). *Iterative and incremental developments. a brief history.* Computer, 36(6), 47-56.

Larman, C. (2004). *Agile and iterative development: A manager's guide.* Addison-Wesley Professional.

Lawrence, B., Wiegers, K. & Ebert, C. (2001). *The top risk of requirements engineering*. IEEE Software, 18(6), 62-63.

Lucassen, N. B. G. & Brinkkemper, S. (2017). *A reference method for user story requirements in agile systems development.* In 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW) (pp. 292-298). IEEE.

Lyytinen, K., Mathiassen, L. & Ropponen, J. (1996). *A framework for software risk management*. Journal of Information Technology, 11(4), 275-285.

Manzo, J. (2002). *Odyssey and other code science success stories*. CrossTalk, 19-21.

Mathiassen, L., Saarinen, T., Tuunanen, T. & Rossi, M. (2007). *A contigency model for requirements development.* Journal of the Association for Information Systems, 8(11), 569.

Measey, P. (2015). Agile Foundations: Principles, practices and frameworks. BCS Learning & Development Limited, 2015. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/jyvaskyla-ebooks/detail.action?do-cID=1759633.

Mills, H. D. (1999). *The management of software engineering, part I: Principles of software engineering*. IBM Systems Journal, 38(2.3), 289-295.

Mursu, A. (2002). *Information systems development in developing countries: Risk management and sustainability analysis in nigerian software companies* University of Jyväskylä.

Myers, M. D. (1997). *Qualitative research in information systems.* Management Information Systems Quarterly, 21(2), 241-242.

Nerur, S., Mahapatra, R. & Mangalaraj, G. (2005). *Challenges of migrating to agile methodologies*. Communications of the ACM, 48(5), 72-78.

NIST, Goguen, A. & Fringa, A. (2002). *Risk management guide for information technology systems.* Recommendations of the National Institute of Standards and Technology,

Nuseibeh, B. & Easterbrook, S. (2000). *Requirements engineering: A roadmap.* In Proceedings of the Conference on the Future of Software Engineering (pp. 35-46). ACM.

Overhage, S., Schlauderer S., Birkmeier D., & Miller, J. (2011). *What makes IT personnel adopt scrum? A framework of drivers and inhibitors to developer acceptance.* In 2011 44th Hawaii International Conference on System Sciences (pp. 1-10). IEEE.

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

Paulk, M. C. (2002). *Agile methodologies and process discipline*. Institute for Software Research. Paper 3. http://repository.cmu.edu/isr/3

Persson, J. S., Mathiassen, L., Boeg, J., Madsen, T. S. & Steinson, F. (2009). *Managing risks in distributed software projects: An integrative framework.* IEEE Transactions on Engineering Management, 56(3), 508-532.

Poole, C. J., Murphy, T., Huisman, J. W. & Higgins, A. (2001). *Extreme maintenance*. In Proceedings of the IEEE International Conference on Software Maintenance (ICSM'01) (p. 301-309). IEEE Computer Society.

Qumer, A. & Henderson-Sellers, B. (2006). *Measuring agility and adoptability of agile methods: A 4-dimensional analytical tool*. In The IADIS international conference on applied computing 2006. IADIS Press. 503-507.

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

Ratchev, S., Urwin, E., Muller, D., Pawar, K. S. & Moulek, I. (2003). *Knowledge based requirement engineering for one-of-a-kind complex systems*. Knowledge-Based Systems, 16(1), 1-5.

Rehman, T., Khan, M. N. A. & Riaz, N. (2013). *Analysis of requirement engineering processes, tools/techniques and methodologies*. International Journal of Information Technology and Computer Science (IJITCS), 5(3), 40.

Rowe, W. D. (1975). An" anatomy" of risk Environmental Protection Agency. Environmental Protection Agency, Washington D.C.

Royce, W. W. (1970). *Managing the development of large software systems*. (s. 328-338) Los Angeles.

Schmidt, R., Lyytinen, K. & Mark Keil, P. C. (2001). *Identifying software project risks: An international delphi study*. Journal of Management Information Systems, 17(4), 5-36.

Schwaber, K. (1997). *Scrum development process. Business object design and implementation.* In 10th Annual Conference on Object Oriented Programming Systems, Languages, and Applications Addendum to the Proceedings. ACM/SIGPLAN October. (s. 117-134) Springer.

Stake, R. E. (2014*). Qualitative research : Studying how things work*. New York: Guilford Publications. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/jyvaskyla-ebooks/detail.action?docID=479606

Tuunanen, T, Vartiainen, T and Ebrahim M. (2018) *Development of Requirements Risk Prioritization Method*, working paper, University of Jyväskylä.

Vaismoradi, M., Turunen, H. & Bondas, T. (2013). *Content analysis and thematic analysis: Implications for conducting a qualitative descriptive study*. Nursing & Health Sciences, 15(3), 398-405.

Wong, S. & Whitman, L. (1999). *Attaining agility at the enterprise level*. In Proceedings of The 4th Annual International Conference on Industrial Engineering Theory, Applications and Practice, San Antonio, TX.

Wiegers, K & Beatty, J. 2013. Software requirements 3 (Third edition.). US: Pearson Education M.U.A.

Yin, R. K. (2013). *Case study research: Design and methods.* Second edition. Thousand Oaks: Sage publications.

Zave, P. (1997). *Classification of research efforts in requirements engineering*. ACM Computing Surveys (CSUR), 29(4), 315-321.

Zhang, Z., Arvela, M., Berki, E., Muhonen, M., Nummenmaa, J. & Poranen, T. (2010). *Towards lightweight requirements documentation*. Journal of Software Engineering and Applications, 3(9), 882.

APPENDIX 1 INTERVIEW STRUCTURE

Introduction (3 min)

- research topic and researcher
- Interview structure

Background information (12 min)

- Introduction of the Requirements Risk Prioritization method
- Introduction of the focus feature
- Feature description and business goal
- Overview of current development status (if needed)

Interview Phase 1 (15min): Creating a risk profile

- Identifying and evaluating requirement risks of the focus feature by using the Requirements Risk Prioritization method (checklists).
  - Requirements phase
  - Design phase
  - Implementation phase
- Impact evaluation for each identified risk

Interview Phase 2 (15 min): Evaluating validity, coverage, relevance and accuracy of the risk profile.

Interview Phase 3 (10 min): Evaluating the use of the method in the agile project work

Closing words (5 min)

APPENDIX 2 INTERVIEW QUESTIONS

English:

Interview Phase 1: Creating a risk profile

- Following steps are repeated to each requirement management process steps (requirements, design, implementation):
  - Write down risks that affect requirement management process step (separate piece of paper)
  - Mark with X to the checklist those risks that match risks you identified earlier
  - Mark with O to the checklist those risks that you didn't identify, but are still valid for the focus feature
  - Strikethrough those risks from the checklist which are not valid for the focus feature

Interview Phase 2: Evaluating validity, coverage, relevance and accuracy of the risk profile.

- Evaluate how well the risk profile you created describes the risk environment of the focus feature development.
- Have some of the risks been recognized earlier?
- Did the risk profile bring up risks which has not been identified before?
- Are the risks in the risk profile meaningful for the overall success of the project?
- Has any of the risks negatively impacted the project earlier in the development?

Interview Phase 3: Evaluating the use of the method in the agile IS project work

- Does the method produce information that can help decision making regarding the project? Why?
- Would it be useful for the project to use the method? Why?
- Would you imagine yourself using the method in your project / projects in the future?
- At which stage you would be most likely using the method?

Finnish:

Haastattelu Vaihe 1: Riskiprofiilin luonti

- Seuraavat tehtävät toistetaan kaikille vaatimustenhallintaprosessin vaiheille (vaatimusten keräys, suunnittelu, toteutus)

- Kirjaa ylös erilliselle paperille ominaisuuden vaatimusmäärittelyvaiheeseen liittyviä riskejä
- Merkitse muistilistasta X:llä ne riskit, jotka vastaavat tunnistamiasi riskejä.
- Merkitse O:lla check listasta ne riskit, joita et itse kirjannut, mutta, jotka mielestäsi silti koskevat esiteltyä ominaisuutta.
- Yliviivaa ne riskit check listasta, jotka eivät mielestäsi koske lainkaan esitellyn ominaisuuden vaatimusmäärittelyvaihetta

Haastattelu Vaihe 2: Luodun riskiprofiilin oikeellisuuden, kattavuuden, merkityksellisyyden ja tarkkuuden arviointi

- Arvioi kuinka hyvin muodostamasi riskiprofiili mielestäsi kuvaa järjestelmäominaisuuden riskitilannetta.
- Onko jotkut riskeistä tunnistettu joskus aikaisemmin?
- Toiko riskiprofiili esiin riskejä, joita ei ole aikaisemmin tunnistettu?
- Ovatko tunnistetut riskit merkityksellisiä projektin kokonaisonnistumisen kannalta?
- Onko tunnistetut riskit toteutuneet joskus aikaisemmassa järjestelmän kehitysvaiheessa?

Haastattelu Vaihe 3: Menetelmän käyttö ketterässä järjestelmä kehitys projektityössä

- Tuottaako menetelmä tietoa, joka auttaa projektin päätöksentekoa? Miksi?
- Onko menetelmän käyttö hyödyllistä projektillesi? Miksi?
- Voisitko kuvitella käyttäväsi menetelmää rojektissasi/projekteissasi tulevaisuudessa?
- Missä vaiheessa projektia hyödyntäisit menetelmää kaikkein todennäköisimmin?

INTRODUCTION OF THE MODEL:

Requirements risk prioritization model

KEY CONCEPTS:

System requirement: Definition of what the software product is to do as well as what it is not expected to do. In other words, what is needed from the system/or functionality

Client: In this context client/customer for the functionality is our internal business

Risk: is the potential of gaining or losing something of value

Risk types:

Risk	Definition
Requirements identity risk	The availability of requirements, high identity risk indicates requirements are unknown or indistinguishable
Requirements volatility risk	The stability of requirements, high volatility risk indicates requirements easily change as a result of environmental dynamics or individual learning
Requirement complexity risk	The understandability of requirements, high complexity risk indicates requirements are difficult to understand, specify and communicate.
Requirements integrity risk	The completeness and accuracy of requirements elicited from the end users, high integrity risk means indicates requirements were only partly captured and their origin is not traceable.

APPENDIX 4: RISK CHECK LISTS

Requirements phase	
Risk name	Risk type
Absence of Project Sponsor	Identity
Access to Clients (Proximity to Source	Complexity
Ambiguous Requirements	Identity
Change in in Business Strategy and Direction	Volatility
Change in External Regulations	Volatility
Client Commitment	Identity
Constrained Users' Knowledge	Complexity
Fixed Budget and Timelines	Integrity
Incorrect Stakeholder	Identity
Misunderstood Business Needs	Identity
Underestimation of Change Magnitude	Volatility
Unrated Requirements	Volatility
Design phase	
Ambiguous Requirements	Identity
Change in External Regulations	Volatility
Client Commitment	Identity
Compliance with External Regulations	Identity
Conflicting Requirements	Integrity
Missing Requirements	Identity
Delivering what the Client Requires	Identity
Emerging Requirements Dependency	Volatility
Fixed Budget and Timelines	Integrity
Knowledge Gab between Coworkers	Complexity
Lack of Collaboration	Complexity
Technology Changes	Volatility
Underestimation of Change Magnitude	Volatility
Unrated Requirements	Volatility
Implementation phase	
Ambiguous Requirements	Identity
Change in External Regulations	Volatility
Client Commitment	Identity
Fixed Budget and Timelines	Integrity
Hostile Users	Identity
Project Team Member Turnover	Volatility
Unrated Requirements	Volatility
Underestimation of Change Magnitude	Volatility

APPENDIX 5: THEMATIC ANALYSIS, THEME MAP

Theme:	Description:
Understandability of the risk items in the checklists	*Interviewees had a question of a certain risk item in any of the methods checklist or had a hard time understanding a scope of the risk item.*
Using the RRP method checklist to recognize risks	*Interviewees indicated that the RRP method could help him/her to recognize new risks.* *OR* *Interview indicated that check lists didn't bring any new risks*
risk items are not artificial	*Interviewees mentioned that most of the risk items have been recognized in the project earlier or that risk have created challenges earlier in the project.* *OR* *Interviewees mentioned that risk item check lists do not include contrived risks.*
risk profile's representativeness	*Interviewee commented how well or poorly the method described project's true requirements risk situation* OR *Interviewees mentioned that method didn't bring any new information to him/her.*
risk profile's generalizability	*Interviewees mentioned that same risk profile could be from any of the company's projects*
Communication tool	*Interviewees stated that method could help communication between different stakeholders*
High level risk items	*Interviewee concluded that the RRP method present high-level risks or too high-level risks.* *OR* *Interviewees felt that risk items are too high-level for them to be able to manage them.*
risk impact evaluation	*Interviewees wanted to evaluate risk probability* *OR* *Interviewees were not able to prioritize risks by focusing only to their impact.* *OR* *Evaluating low impact for a risk was difficult.*
Who is qualified to evaluate risks accurately	*Interviewees doubted their ability to evaluate risks in the level method suggested due to their role in the project or position in the organization.*
Risk item scope challenges	*Risk items were too interconnected*
Timing of the risk analysis	*Interviewees felt that the method would most useful at the beginning of the project*
Time needed for the analysis	*Interviewees commented how much time the risk assessment with the RRP method would take.*
Agile process corrects itself	*Interviewees commented how agile process already has many points where requirement risks are managed without specific focus on them.*
Check lists as a tool	*Interviewees though that check lists are either good or bad tool to evaluate risks. Some though those limit their thinking, others thought it helps them to focus on correct things.*