Jarmo Kuusinen

MANAGING AND PRIORITIZING REQUIREMENTS RISKS IN INFORMATION SYSTEMS DEVELOPMENT

UNIVERSITY OF JYVÄSKYLÄ
DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION SYSTEMS
2018

ABSTRACT

Kuusinen, Jarmo
Managing and prioritizing requirements risks in information systems development
University of Jyväskylä, 2018, 112 p.
Information Systems, Master's Thesis
Supervisor: Tuunanen, Tuure

Keywords: Requirement risks, requirements management, requirements engineering, agile development

In design and implementation of information systems (IS), the purpose and goals of the IS under development are commonly expressed as IS requirements. The requirements acquisition, development and management processes in IS development (ISD) aim at acquiring the IS requirements from relevant stakeholders, analyzing them properly, and expressing the requirements to the IS developers in appropriate format to allow implementation of an IS that fits the purpose. Concurrently, various agile approaches in IS development have grown in importance. One can justifiably argue various agile adaptations constituting the de-facto IS development approach of today. Over the years, a wide variety of methods and techniques have been developed both by practitioners and researchers to systematically acquire, analyze and manage stakeholder requirements with concurrent ISD approaches.

However, both research literature and practical experience tell that remarkable share of unsuccessful IS development projects fail because of shortcomings in requirements engineering and management. Improperly executed requirements analysis and requirements management therefore poses risks to success of IS projects. These so-called requirement risks typically result from IS requirements that are incorrectly identified, unclearly described, technically complicated, inadequately specified, ambiguously expressed, or highly volatile. If the requirements and their risks are not properly handled throughout the IS development process, the risk of project failure increases.

The risks in IS development have been considered in numerous studies, but it still appears that the contemporary information science does not provide many methods for prioritizing requirement risks. In this thesis study, we study how the contemporary information system design methodologies consider requirement risks and their prioritization.

In the literature review, we discuss how requirements acquisition and management is approached in agile and continuous delivery, and how requirement risk management and prioritization realize in these methods. This thesis also provides the reader with a categorization of characteristics of high quality requirements, techniques to improve requirement quality, discussion on quality criteria for requirements artefacts, and measures to help in implementing requirements development practices.

In this thesis an analysis of a generic agile model's requirement risk mitigating characteristics is presented. The concurrent ISD approaches, such as agile frameworks, have many characteristics which help IS developers in requirements risk mitigation.

The review on requirements engineering and requirements management literature propose that requirements risks in IS development are managed by trying to produce high quality requirements by applying best practices in requirements elicitation, requirements development and requirements management throughout the life-cycle of the IS. The concurrent ISD approaches emphasize user-centric thinking, stakeholder collaboration and the importance of understanding the customer value creation. The assumption is that by applying user-centric requirements acquisition and development approaches, and by understanding IS customer value creation, correctness of the IS requirements can be improved and therefore confidence in building a right IS improved.

It is remarkable that even though the literature states fairly unanimously states that unsuccessful requirements engineering processes are remarkable contributor in failed IS projects, the requirements risks are seldom approached from the risk management perspective, i.e. by identifying risks the risks associated with requirements, prioritizing them and deciding on the means to mitigate them.

Tuunanen and Vartiainen (2016) represent a requirements risk prioritization method presented, which considers requirement risks, their prioritization and mitigation. In this thesis work we also study the method in a case study, to evaluate its applicability and usefulness in a case project. The empirical part of the study was conducted as an interpretive case study. The data is collected from project specification workshop observations and five semi-structured interviews for project team members of an IS development industry project applying agile development approach.

The case study results suggest that the requirement risk analysis approach was found novel by the case study participants, the method gives was considered applicable in the case project and provided valuable information about the requirements risks. However, usefulness of certain parts of the method were challenged, and improvement suggestions made. Context-specific adaptation method, and remarks on applying the method in ISD projects are presented. The results of the case project interviews suggest that the method would provide most value for the customer-side project management, because it provides them with a new tool for estimating project risk elements associated with requirements. The thesis also represents several items for further research.

TIIVISTELMÄ

Tietojärjestelmien suunnittelussa ja toteutuksessa järjestelmän tavoite ja ominaisuudet kuvataan tietojärjestelmän vaatimuksina. Tietojärjestelmävaatimusten keräämisen, kehittämisen ja hallinnan prosessit tietojärjestelmien kehityksessä tähtäävät siihen, että järjestelmän vaatimukset saadaan koottua sidosryhmiltä, ne analysoidaan huolellisesti ja ilmaistaan vaatimukset järjestelmän kehittäjille asianmukaisessa muodossa. Tämä menettely mahdollistaa käyttötarkoitusta vastaava järjestelmän toteuttamisen.

Tämän päivän tietojärjestelmäkehityksessä erilaisten ketterien kehitysmallien merkitys on kasvanut, ja voidaankin perustellusti väittää, että erilaiset ketterien menetelmien sovellutukset ovat tällä hetkellä tietojärjestelmäkehityksen de facto -menetelmä. Vuosien mittaan sekä tutkijat että käytännön tietojärjestelmäkehittäjät ovat kehittäneet laajan valikoiman erilaisia menetelmiä tietojärjestelmien vaatimuksia kokoamiseen, analysointiin ja hallintaan nykyaikaisten tietojärjestelmien kehitysmallien yhteydessä.

Tästä huolimatta sekä tutkimuskirjallisuus että käytännön projektikokemukset kertovat, että merkittävä osa epäonnistuneista tietojärjestelmähankkeista epäonnistuu juuri vaatimushallinnan puutteiden takia. Kelvottomasti toteutettu vaatimusten kokoaminen, analysointi ja hallinta aiheuttavat riskejä tietojärjestelmäprojekteille. Nämä niin sanotut vaatimusriskit ovat tyypillisesti seurausta vaatimuksista, jota ovat väärin tunnistettuja, epäselvästi kuvattuja, teknisesti monimutkaisia, riittämättömästi määriteltyjä tai helposti muuttuvia. Jos vaatimuksiin liittyviä riskejä ei hallita tietojärjestelmän kehitysprosessin aikana, kasvaa riski sille, että hanke epäonnistuu saavuttamaan tavoitteitaan.

Tietojärjestelmäkehityksen riskejä on käsitelty lukuisissa tutkimuksissa, mutta näyttää siltä, että tämän päivän tietojärjestelmätieteen tutkimus ei tarjoa paljoa vaihtoehtoja metodeiksi, joiden avulla on mahdollista analysoida ja priorisoida vaatimusriskejä. Tässä pro gradu -työssä tutkitaan, kuinka nykyiset tietojärjestelmien kehitysmenetelmät lähestyvät vaatimusriskejä ja niiden priorisointia.

Kirjallisuuskatsauksessa keskustellaan siitä, miten vaatimusten hankinta, kehittäminen ja hallinta nähdään ketterissä kehitysmenetelmissä ja kuinka vaatimusriskien priorisointi ja hallinta toteutuvat menetelmissä. Työ kuvaa myös vaatimusten sekä vaatimuksiin liittyvien artefaktien laatukriteerien kategorisoinnin, vaatimusten laadun parantamiseen tähtäävien tekniikoiden katsauksen sekä toimenpiteitä, joilla vaatimusten laadun kehittämistä tietojärjestelmäkehityksessä voi parantaa.

Työssä tarkastellaan myös vaatimusriskien hallintaa yleisen ketterän kehitysmallin kehyksessä ja kuvataan, miten vaatimuksiin liittyviä riskejä käsitellään ketterissä menetelmissä. Nykyaikaisissa tietojärjestelmien kehitysmenetelmissä, kuten ketterissä menetelmissä, on useita sisäänrakennettuja piirteitä, jotka auttavat vaatimusriskien hallinnassa.

Kirjallisuuskatsaus vaatimushallinnan ja –analyysin menetelmiin osoittaa, että vaatimusriskejä hallitaan pääsääntöisesti siten, että pyritään tuottamaan mahdollisimman korkealaatuisia tietojärjestelmävaatimuksia. Tämä toteutuu siten, että kehittäjät soveltavat parhaita käytäntöjä vaatimusten hankinnassa, kehittämisessä ja hallinnassa koko järjestelmän elinkaaren ajan. Tämän päivän kehitysmenetelmät korostavat käyttäjäkeskeistä ajattelua, sidosryhmäyhteistyön merkitystä sekä asiakkaan arvontuotannon ymmärtämistä. Taustaoletus on, että käyttäjäkeskeisten kehitysmenetelmien avulla sekä asiakkaan arvontuotantoa ymmärtämällä saavutetaan parempi vaatimusten oikeellisuuden taso sekä siten parempi luottamus siihen, että kehitetään oikeanlaista järjestelmää.

On huomionarvoista, että vaikka alan kirjallisuudessa melko yksimielisesti todetaan vaatimusten hallinnan epäkohtien olevan merkittävä tietojärjestelmäprojektien epäonnistumiseen vaikuttava tekijä, niin silti vaatimuksiin liittyviä riskejä lähestytään harvoin riskienhallinnan näkökulmasta. Riskienhallinnallisessa lähestymistavassa vaatimuksiin liittyvät riskit ensin tunnistetaan, sitten niiden sekä valitaan toimenpiteet riskien hallitsemiseksi.

Tuunanen & Vartiainen (2016) esittelevät tutkimuksessaan vaatimusriskien analysoinnin ja priorisoinnin menetelmän. Tässä työssä tutkitaan sitä, kyseinen menetelmä (Tuunanen & Vartiainen, 2016) käsittelee vaatimusriskejä sekä niiden hallintaa ja pienentämistä. Työssä tutkitaan erityisesti menetelmän sovellettavuutta ja hyödyllisyyttä tapaustutkimuksen keinoin esimerkkiprojektissa. Tutkimuksessa data kerättiin käytännön sovelluskehitysprojektin yhteydessä viidellä puolistrukturoidulla asiantuntijahaastattelulla sekä projektin määrittelytyöpajassa pidetyllä observoinnilla. Esimerkkiprojekti oli asiakas, jossa kehitettiin jo käytössä olevaan ohjelmistoon uusia ominaisuuksia ketterin menetelmin ohjatussa projektissa.

Tapaustutkimus osoittaa, että vaatimusriskien analysointimenetelmä nähtiin uutena lähestymistapana tutkimuksen osallistujien näkökulmasta. Menetelmää pidettiin tarpeeseen sopivana ja sen koettiin antavan arvokasta tietoa projektin vaatimuksiin liittyvistä riskeistä. Menetelmässä oli kuitenkin vaiheita, joita ei pidetty soveltuvana projektissa. Tähän liittyen kirjattiin menetelmän kehitysideoita ja jatkotutkimuskohteita. Erityisenä havaintona tapaustutkimuksessa nousi tarve menetelmän soveltamiseen projektin kontekstin mukaisena. Tapaustutkimuksen perusteella näyttää siltä, että kyseisessä projektissa menetelmä on erityisen arvokas projektissa sovelluskehityspalvelua ostavan asiakkaan projektijohdolle, sillä se tarjoaa uuden tavan hallita projektiin liittyviä riskejä.

ACKNOWLEDGMENTS

I would like to thank everyone who has supported me when working on this thesis. It was not always easy to combine the studies with work and family life. Therefore, I thank my family for patience while I was finalizing this thesis, and my employers for flexibility that allowed this study to be accomplished.

Also, I would like to express my gratitude to prof. Tuure Tuunanen and prof. Tero Vartiainen for giving the idea for the study and providing support throughout the research process.

FIGURES

TABLES

TABLE OF CONTENTS

1 Introduction

In the design and implementation of information systems (IS), the purpose and goals of the system under development are commonly expressed as IS requirements. The requirements engineering and management processes in IS development aim at acquiring the IS requirements from relevant stakeholders, analyzing them properly, and expressing the requirements to the IS developers in appropriate format to allow implementation of an IS that fits the purpose. However, both research literature and practical experience tell that remarkable share of unsuccessful IS development projects fail because of shortcomings in requirements engineering and management. Improperly designed and implemented requirements analysis and requirements management therefore poses risks to success of IS projects. These so-called requirement risks typically result from IS requirements that are highly volatile, incorrectly identified, unclearly described, technically complicated, inadequately specified, or ambiguously expressed. If the requirements and their risks are not properly handled throughout the IS development process, the risk of project failure increases.

The risks in IS development have been considered in numerous studies, but it still appears that the contemporary information science does not provide many methods for prioritizing requirement risks. This notice sets the academic motivation for this thesis study.

1.1 Motivation for the research – service development perspective

Currently, growing share of services is motored by digital technologies: often information systems (IS) is the machinery required to run new service business models. For instance, cloud-based business models (such as Software-as-a-Service business) or platform business models rely heavily on use of ISs. Therefore, IS development, as a critical part of implementation and delivery of IS-intensive services, has gained much attention among business managers.

Concurrently, various agile approaches in IS development have grown in importance. One can justifiably argue various agile adaptations constituting the de-facto IS development approach of today. To develop IS in a manner that supports business value creation, a link between customers' and other stakeholders' needs and IS development activities must be established and maintained throughout the life-cycle of the service. Traditionally in IS development, this link has been expressed in form of requirements. The IS requirements describe customer needs and expectations, among other aspects of the IS. Over the years, a wide variety of methods and techniques have been developed both by practitioners and researchers to systematically acquire, analyze and manage stakeholder requirements.

Commonly, customer experience is seen differently by different camps in IS intensive business. Business developers talk about concepts such as customer expectations and preferences, usage analytics, and use measures such as conversion rate, page visits, online revenue - some to mention. On the other hand, technical staff focus mostly on system health and quality issues: they talk about (system) performance, scalability, quality of service, and use measures like downtime, mean time to repair, load time, or latency, exemplarily. Often these two sets of measures do not match, and the connection between system-related measures and their business impacts is not apparent. Understanding better the chain of activities from customer value creation to IS implementation is of great interest to business management today.

It also appears that this link between customer value creation and IS development activities is getting increasingly difficult to maintain, as the speed of development keeps growing, and both customer expectations and IS architectures are getting more complex. This raises questions for the designers of IS-intensive businesses: how to develop deep understanding of user needs for the IS and maintain it throughout the IS lifecycle? And moreover, what are the consequences, if this chain of actions from the user need to the delivered information service is broken?

1.2 Research objective and research questions

Abundant number of studies have considered risks in IS development, for instance Wallace et al. 2004, Persson et al. 2009, Mathiassen et al. 2007, Keil et al. 1998, Chen et al. 2015, Barki et al. 1993, and Tuunanen and Vartiainen 2016. However, Tuunanen and Vartiainen (2016) argue in their study that contemporary information science does not provide many methods for prioritizing requirements risks, and that further emphasis should be put on studying how requirements risk prioritization is done in a way that it considers of contemporary ISD principles, such as agile and continuous delivery, while supporting more structured ISD approaches. Therefore, it appears that that not very much research has been done on the field of requirement risk prioritization and management in continuous ISD context.

The goal of the thesis is to study how the contemporary information system design methodologies consider requirement risks and their prioritization. In the literature review, we consider how requirements acquisition and management is approached in agile and continuous delivery, and how requirement risk management and prioritization realize in these methods. In addition, the literature review section also discusses how the requirements risks are handled in the contemporary ISD methods, such as agile and continuous delivery. Further on, the thesis studies how the requirements risk prioritization method presented by Tuunanen and Vartiainen (2016) considers requirement risks, their prioritization and mitigation. Specifically, the applicability and usefulness of the method is evaluated in an empirical case study.

Therefore, the primary research questions of this thesis study are set as follows:

- How the requirements risk management and prioritization is approached in the context of contemporary ISD methods, such as agile or continuous delivery?

Sub-questions, which aim at building ground for understanding the primary question, are set as follows:

- How the contemporary ISD methods, such as agile and continuous delivery, approach requirements acquisition and management?
- How requirement risks are handled in these methods?

In the theory part, a review of contemporary IS literature and research is done to chart methods and search for experiences and examples. The theory part will consist of discussion on requirements engineering and management, risk management of requirement risks and agile development methods.

In the empirical part of the thesis, the goal is to test and experiment the risk prioritization method (Tuunanen and Vartiainen, 2016) and develop it further in a case IS development project.

1.3 Research methods

In theory part of the thesis, theoretical grounds for the requirement risk handling in agile and continuous development environment are built based on a review of contemporary IS literature and research findings. The goal of the literature review is to find definition for requirement risks, identify relevant requirement risk handling methods, and to search for practical experiences and examples in the context. The theory part will discuss requirements engineering and management methods, requirement risks, management and prioritization of risks, and agile development methods. Further on, the model for risk handling and prioritization to be applied in the empirical part is described based on the literature.

In the empirical part of the thesis, the goal is to test and develop further requirement risk prioritization model described in the theory part. The model is based on the work by Tuunanen and Vartiainen (2016). The empirical study will be conducted as an interpretive case study. The data is collected by making semi-structured specialist interviews and observations in an IS development industry project applying agile development approach. The research methods are described in chapter 6.1.

2 Requirements as a representation of the vision of IS

Requirements are a representation of the purpose of the information system. In ISO's & IEEE's systems and software engineering vocabulary (ISO/IEC 24765, 2010) the term is specified as follows:

> ***Requirement* 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, system component, product, or service to satisfy an agreement, standard, specification, or other formally imposed documents* **3.** *a documented representation of a condition or capability as in (1) or (2)* **4.** *a condition or capability that must be met or possessed by a system, product, service, result, or component to satisfy a contract, standard, specification, or other formally imposed document. Requirements include the quantified and documented needs, wants, and expectations of the sponsor, customer, and other stakeholders. A Guide to the Project Management Body of Knowledge (PMBOK® Guide) – Fourth Edition*

In the next subchapters requirements categorization, requirement presentation and description methods, characteristics of good requirements and risks related to IS requirements are discussed.

2.1 IS requirement types and categories

Typically for every system there are diverse types of requirements, which derive from the needs and expectations of different stakeholders for the system. Common stakeholder groups are different user groups (such as end users, customers, customer service personnel or IS administrators), business analysts, business management and IS developers (Wiegers 2013, pp. 4-8, Patricio et al. 2009, Pohl 2010, pp. 17-23).

Typical requirements categories in requirements engineering are functional requirements, which describe what the IS should do, i.e. the functionality of the system, and non-functional requirements, which describe how the IS should be built, such as system requirements (Patricio et al. 2009). In the categorization by Pohl (2010, pp. 17-23), there are three categories: functional requirements, quality requirements and constraints. In addition to these, business requirements define the business goals of the IS. User requirements refer to requirement statements that describe the goals or tasks the users should be able to perform (Wiegers 2013, 8-10).

Often the non-functional requirements also express various quality related expectations towards the system. Quality related requirements emerge from different needs: usability, accessibility, reliability and robustness qualities for ISs are often called for. For instance, Patricio et al. (2009) discuss customer experience

requirements, which they define as "customer perceived attributes of the interaction with the service provider that contribute to a satisfying experience". The attributes related to customer experience requirements, as described by Patricio et al. (2009), are for instance usefulness, efficiency and quality of personal contact to IS.

Requirements can be approached from the perspective of cultural values, as proposed by Tuunanen & Kuo (2016). The researchers present a model for cultural value based prioritization of requirements. The typology of values could be applied in requirements engineering to separate requirements based on cultural settings or geographical locations. This approach can be applicable also in requirements acquisition phase to evaluate feasibility of functional or non-functional requirements in diverse cultural contexts.

Further on, Tuunanen & Govindji (2016) propose using the concept of flow to better understand information system usage experience. The flow experience refers to the notice that some system features are driven by more experiential needs, whereas others are driven by more task-oriented goals. The concept of flow describes the user experience fulfilling the experiential needs. The researchers propose that flow experience should be visualized and measured, as differences in how users see and perceive different features. The findings indicate that the users' perceived flow experience can be measured already during the early phases of information systems (IS) development projects. This enables practitioners to design IS that better facilitate flow experience for the users, which in turn will potentially lead to shortened development time and cost savings for firms.

It is noteworthy that there is no single generic model either for categorizing or for expressing IS requirements, but instead applicable requirements model, as well as RE methods and techniques, are dependent on the context, such as purpose of IS, stakeholders of IS or life-cycle stage of IS. For the sake of consistency, in this thesis we use requirements terminology derived from the reference literature. The categorization is summarized in Table 1.

Table 1. Requirement categorization.

Requirement category	**Explanation**	**Examples**
Business requirements	Expresses the business reasoning for the IS, and therefore sets vision for the IS.	"The system must decrease the time needed for processing an application."
Functional requirements	Expresses requirements for the functionality of the IS: features and capabilities of IS. Describe for instance who are the IS users and what do they want the IS to do for them.	"The applicant must be able to fill in and send the application. "
Non-functional requirements	Expresses how the IS must be built. For instance, system requirements, quality attributes, performance requirements, restrictions or external limitations.	"The system must be able to handle 100 separate applications, each by individual applicant, in hour"

The basic requirements categorization presented in Table 1 describe the essentials of IS requirements. By specifying business goal, functional and non-functional requirement, we lay ground for the design of the IS.

2.2 Characteristics of a good requirement

The elemental purpose of requirements engineering is to produce and develop as good requirements as possible, no matter which methods have been selected for the task. The requirements do not describe how to build the IS technically, but instead describe how the system should work and what other expectations it must fulfill. Therefore, ideally requirements must be, after sufficient analysis, expressed using such techniques that they can be correctly interpreted by the IS developers and other customers. This requirement expression and interpretation process is critical for project success.

In an ideal case, each requirement would have the following qualities: complete, correct, feasible, prioritized, necessary, comprehensible and unambiguous (Wiegers 2013, pp. 201-205). Table 2 explains these qualities in context of IS requirements.

Table 2. Qualities of a good IS requirement.

Quality	Explanation
Complete	Each requirement must be complete, i.e. contain all necessary information for the reader to understand it. If the requirement has known unknowns, it should be expressed also.
Correct	Correct requirement 1) describes a capability meets some stakeholder's expectation, 2) describes the functionality accurately, and 3) is not in contradiction with other requirements.
Feasible	Feasible requirement is implementable within the known limitations of the IS environment and project constraints (time, resources).
Necessary	Necessary requirement describes a feature that provides business value or is needed to conform with standard or regulation.
Prioritized	Requirements should be prioritized according to the importance in producing desired value.
Unambiguous	Unambiguous requirement is correctly interpretable by the reader, i.e. it cannot be interpreted differently.
Comprehensible	Requirement must be understandable by readers.
Verifiable	Requirement must be verifiable as correct or incorrect after implementation. Requirements that are incomplete, inconsistent, infeasible, or ambiguous are also unverifiable.

Requirements are usually represented in requirement collections, such as requirement specification document, story board or product backlog. A requirement collection should constitute a complete presentation of certain entity, such as entire IS or a module of the system. Having excellent individual requirement statements is not enough, but also the requirements collections, whatever is the

form of presentation, should be of excellent quality. Wiegers (2013, pp. 204-205) lists the following qualities for a good requirements collection: complete, consistent, modifiable and traceable. Table 3 summarizes and explains these qualities.

Table 3. Qualities of a good requirement collection.

Quality	Explanation and examples
Complete	A complete requirement collection contains all necessary requirements to specify the system.
Consistent	Consistent requirements do not conflict with other requirements or business goals.
Modifiable	Modifiability requires that requirements can be changed in a controlled fashion. This requires labeling requirements uniquely, knowing their dependencies, unambiguity, and non-redundancy of requirements.
Traceable	Requirements must be linked both backward (to identify their origin) and forward (to potential other requirements, later design phases, and source code).

The IS requirements can be expressed and documented in many different forms. In some situations, human language can be used to explain the requirement. In other cases, a more illustrative method, process-centric diagram or story-like presentation is more applicable. The most suitable method depends on the type of requirements, context of IS usage, and practical development arrangements (such as ISD approach or development team arrangements). The methods are discussed in chapter 3.

Perfect and completely unambiguous requirements or requirement collections can never be written, because some room for different interpretations will always remain. However, paying attention to the characteristics of good requirements, better requirements specifications and better software can be created.

2.3 Requirement presentation methods

There is plenty of different methods for expressing and documenting requirements, and making an exhaustive review on them is out of scope of this thesis. The literature represents techniques such as vision documents, context diagrams, ecosystem maps, critical success factors, event list or feature tree for describing business requirements, and user stories with user personas, use cases for describing functional and user requirements (Wiegers and Beatty 2013, pp). Pohl (2010, pp. 307-323) in that requirements are documented in natural language or using some modeling techniques, such as UML, use cases etc. An exemplary listing of requirement specification, experimentation, discovery and prioritization techniques based on the paper by Tuunanen and Vartiainen (2016) is presented in Appendix 3. The IS developers should be aware of different requirements development methods and how they are applied, and select methods that are fit best to the case, and allow executing high-quality and low risk requirements engineering and management.

In agile development the requirements are often expressed in form of user stories or use cases, and they are collected epics and prioritized for implementation in product backlogs. We will have a closer look at the agile requirements analysis in the chapter 4.

2.4 Summary

When designing and implementing IS, the purpose and goals of the system, i.e. the IS requirements, must be acquired from the relevant stakeholders and expressed to the IS developers in appropriate form. The requirements acquisition and analysis activities may consist of very different methods and techniques, depending on the context of the IS. When developing a mass-market IS for product-based business model the requirement acquisition methods are very different than when developing IS for business systems for professional users (so called management information systems). The selected acquisition method also has impact on the requirement specification technique, and the most suitable methods must be chosen according to the context. However, some universal principles for good requirements exist.

Both research literature and practical experience tell that remarkable share of unsuccessful IS development projects fail because of shortcomings in requirements engineering and management. Therefore, paying attention to the quality of the requirements and the quality of the requirements engineering techniques is worthwhile.

3 Requirements development and requirements engineering

Requirements analysis and management is part of larger business analysis function. Business analysis aims at identifying and understanding business needs and finding solutions to business related challenges. Project Management Institute (PMI) defines business analysis as application of knowledge, skills, tools and techniques to determine problems and identify business needs; to identify and recommend viable solutions for meeting those needs; to elicit, document, and manage stakeholder requirements in order to meet business and project objectives; and to facilitate the project team with the successful implementation of the product, service or end result of the project or program (Smith et al., 2014). In a modern business, these solutions often have IS-based parts. Essentially, business analysis also covers other aspects, such as organizational development, process development or improvement of company policies and practices. The business analysis process begins before actual IS development and extends throughout the life-cycle of the solution.

Therefore, business analysts often play a key role in requirements management for software-based systems. From the business analysis practitioner's perspective, the importance of requirements management and engineering derives from three key outcomes of successful requirements handling: possibility to better match the IS to market opportunity, lower likelihood of IS project schedule slippage, and better control of IS development costs due to increased understanding of goals and expected outcome of IS development. (Smith et al., 2014)

As discussed in chapter 2.3, requirements can be represented and communicated in different methods and formats. Same goes with requirement development: requirement development can be approached from different perspectives and done by applying many different methods. A listing of requirement specification, experimentation, discovery and prioritization techniques based on the paper by Tuunanen and Vartiainen (2016) is presented in Appendix 3. IS developers should be aware of different requirements development methods and how they are applied, and select methods that are best applicable to the situation to produce and maintain high-quality and low-risk requirements. In the following chapters, different approaches to the topic are discussed.

3.1 Requirements engineering and management

Requirements engineering, as specified for instance by Pohl (2010, pp. 3-6) and Wiegers et al (2013, p. 15), refers to the engineering discipline that focuses on acquiring, describing and communicating user requirements for IS's. Thus, it co-

vers areas such as describing requirements artefacts, setting targets for IS, handling scenarios, documenting requirements, requirements elicitation, and requirements negotiation.

Requirements management, on the other hand, refers to processes and practices to manage and control requirement engineering activities and manage changes. For instance, Wiegers et al. (2013, 458-463) define requirements management being a part of requirements engineering, and covering version control, change management, requirements status tracking and requirements tracing. However, many practitioners see the concept of requirements management wider than that, and covering not only requirements engineering process related management practices, but also relevant cultural and people aspects in the organization. For instance, PMI's special report on requirements management (Smith et al. 2014) emphasizes importance of organizations' requirements management capabilities, which include people, processes and culture.

Project Management Institute (PMI) defines requirements management as the discipline of planning, monitoring, analyzing, communicating and controlling requirements (Smith et al. 2014). They point out that requirements management is a continuous process involving communication among project team members and stakeholders and adjustments to requirements changes throughout the course of the project.

There is a wide range of requirements engineering and management techniques, which can be applied in different contexts. Wiegers et al. (2013, pp. 45-52) describe the requirements development as an iterative process, where requirements elicitation, analysis, specification and validation phases are follow one another, and feedback is delivered to previous phases. Figure 1 illustrates the process as described by Wiegers et al. (2013, 46).

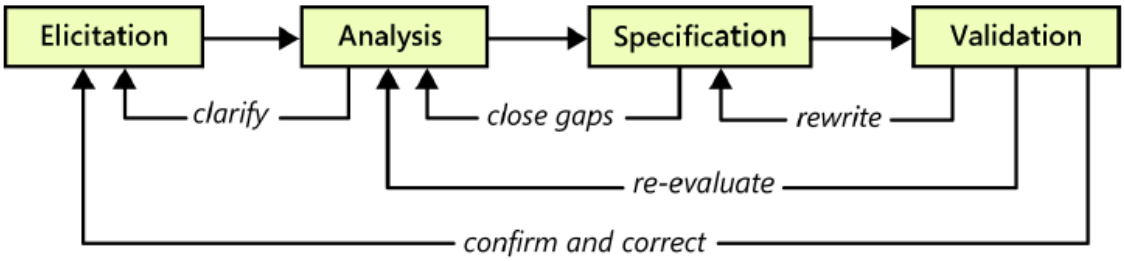

Figure 1: Requirements development process

3.2 Requirements acquisition and development approaches

The process of gathering stakeholder requirements, developing them into implementable IS features and eventually as a deployable IS, is in the core of the customer value creation of IS-intensive businesses. The importance of requirement acquisition is high: for instance, Project Management Institute's Pulse of the Profession study revealed that "inaccurate requirements gathering" was seen as a primary cause of project failure by the respondents of the survey made to about

2000 project and program management professionals globally (Smith et al. 2014). According to the survey, organizations lack resources, skills and knowledge to do requirements management properly. Furthermore, the PMI report points out that business management does not value the requirements management operations sufficiently, which very likely is one reason for shortage of resources and competence shortage in organizations.

Since IS requirements should be a representation of what is considered valuable for the IS stakeholders (such as different user groups), IS requirements acquisition, modeling and documentation are essential steps in creating customer value.

Kauppinen et al. (2009) studied the role of requirements engineering in customer value creation. They analyzed requirements specification and implementation process (i.e. requirements acquisition, documentation, change management, and eventually implementation) especially from the value creation perspective. The empirical study was conducted in Finnish companies providing software-intensive products and services. The two key findings of Kauppinen et al. (2009) are that (1) the IS developers do not understand their customers' processes deeply enough, and (2) IS developers see product features being in the core of value creation. This leads IS developers to focus the requirements engineering efforts too heavily on product features instead of customer value creation. IS developers should pay more attention to customer processes and what services or product features create value for the customer, and allocate the development resources accordingly. The researchers identify requirements engineering pitfalls hindering value creation, and propose requirements management practices that support finding customer value creation perspective in the requirements engineering of software intensive products. The identified pitfalls are related to featurism (adding too many features to the product and polishing single features too much), releasing stripped versions of features in schedule pressure, treating customers and users as one large group, failing to support the customers' processes well, and missing a big picture in development.

As a solution, researchers suggest that the IS developers should develop their customer understanding by doing proper customer segmentation, creating direct contacts between IS developers and customers, and collecting customer information actively (Kauppinen et al. 2009).

The most feasible method and techniques for requirements acquisition and analysis depend on the context and requires careful analysis of the situation. This is not a self-explanatory process. For instance, merely identifying feasible customer segments may not always be straightforward, when the systems are getting more complicated and their user base large. The ISD practitioners often have their favorite methods for requirements acquisition and analysis, which they apply in (industry) projects. Quintessential approaches today are various methods that emphasize user understanding, or user-centered systems design approaches in other words. For instance, Leikas (2009) presents a life-based design approach, which is a holistic design approach for systems design. It is based on building understanding on the user segments of the system and the use context for the

system. The life-based design process proposes to first define the design problems by analyzing the user in the use context (so called form of life), then generating possible design solutions alternatives and analyzing their feasibility, and only after the analysis phase constructing the design requirements. Life-based design process, as all user-centered design approaches, emphasize active involvement of users in the design process.

Requirements acquisition and requirements analysis have been studied in information science research widely and from different angles. For instance, contingency models have been developed to describe requirements development in specific contexts. Mathiassen et al (2007) develop and introduce a model for requirements development, which integrates different models. The major output of the study is integrative contingency model for requirements development, which also takes into account risk handling.

Mathiassen et al. (2007) serves as a base for further research on the area by Tuunanen et al (2016) to stakeholder requirements acquisition methodology. They present theory-backed and practically tested principles for methods for requirements acquisition from stakeholders by providing model for understanding the needs and preferences of various system users, for whom the requirements collection is not straightforward. They introduce a requirements acquisition methodology that builds on stakeholder theory. Stakeholders of firm consist of different parties, which have a tie with the firm. Typical stakeholders are owners, suppliers, employees, customers, or potential customers. The stakeholders can be divided in purposeful groups for defining requirements for new information systems. These groups are called stakeholder populations.

Stakeholder populations differ for instance in their roles, experience, mental models, beliefs, values and preferences (Tuunanen & Peffers 2016). Stakeholder population is a useful concept for identifying stakeholder requirements for information system. Stakeholder populations are firm-specific and also specific to the information system in question. Thus, identifying the relevant stakeholder population groups is not always trivial. Stakeholder requirements acquisition methodology (SRAM) was designed (Tuunanen & Peffers 2016) to target specific stakeholder populations to define functional requirements for new IS. The SRAM methodology builds on personal construct theory, theory of disability, diffusion of innovation, social actor theory, and media richness and information synchronicity theory. The SRAM methodology is used for selecting and designing methods to acquire the functional requirements for new applications and systems. SRAM describes five independent principles of action, which can be applied when making decisions about the requirements acquisition methods (Peffers et al. 2016):

1. Understanding stakeholder preferences, reasoning, and values can be accomplished with methods that capture, aggregate, and preserve stakeholder knowledge structure.
2. Understanding sub-population preferences can be accomplished with methods that segment stakeholder populations by identities, environments, and affiliations.

3. Using non-representative samples can be effective in acquiring requirements from populations inexperienced with new technology, risk aversion, or lack of motivation to participate.
4. Accommodating limitations to participation, from technical disabilities or population economics, through technical and procedural accommodation can overcome limitations in stakeholders' ability to participate.
5. Attending to media characteristics can enable effective acquisition of preferences, reasoning, and values in early stages and achieving concordant understanding of preferences and their value at the later stages.

In another research paper, Peffers et al. (2003) present a model that combines critical success factors (CSF) with personal construct theory to constitute critical success chains (CSC). CSC can be used to model relationships between IS attributes, CSFs and business goals. Critical success factors refer to the elements that are vitally important for an organization to keep competitive performance and eventually reach the goals. There should be only a few of CSF, but organization should put great emphasis on them to succeed. Critical Success Chain, CSC, refers to the extended framework, taking into account other factors / functions that are needed to maintain CSF performance. CSC network is a presentation of the chain.

The method provides model to allow participation on large number of people from different parts of the organization in IS planning phase. The researchers point out that the benefits of the method are larger variety of ideas, better strategic focus for the IS development portfolio, considering larger variety of options to accomplish desired objectives, and better allocation of IS resources (Peffers et al. 2003).

The CSC process consists of pre-study preparation phase, data collection phase, analysis phase, and ideation workshops. The process flows from first identifying the CSFs and people-related personal constructs with large number of people onwards to specifying the business requirements for IS with a limited number of specialists. The CSC method seems feasible for specifying for instance minimum viable product (MVP) of an IS, and showing the relationships between the functionalities of the system to participants from different parts of the organization.

In addition to this, Nemoto et al. (2015) emphasize the importance of considering value-in-context, when specifying requirements product-service systems, i.e. in the requirements analysis, specific situations in product use in different customer contexts should be considered. Nemoto et al (2015) aim at formalizing how the concept of context should be handled in publicly used IS design. When doing design, products and services are combined based on identified customer requirements. Researchers propose a framework for classifying the elements of context of customer. The framework consists of four contextual quadrants on two axes, individual vs. global and short-term vs. long term. The four context quadrants, as described in Figure 2, are customer attributes, environment attributes, environment states and customer states. Using this classification framework, Nemoto et al. (2015) present a context-based requirements analysis

process, which is described on the right-hand side of Figure 2. The process adds to the user persona-based requirements acquisition process 1) identification and description customer and environmental attribute contexts and 2) applying the context states to user persona handling. The presented context-oriented method could be applicable addition for system designers for analyzing how different usage contexts (such as user's emotional state) may affect the requirements.

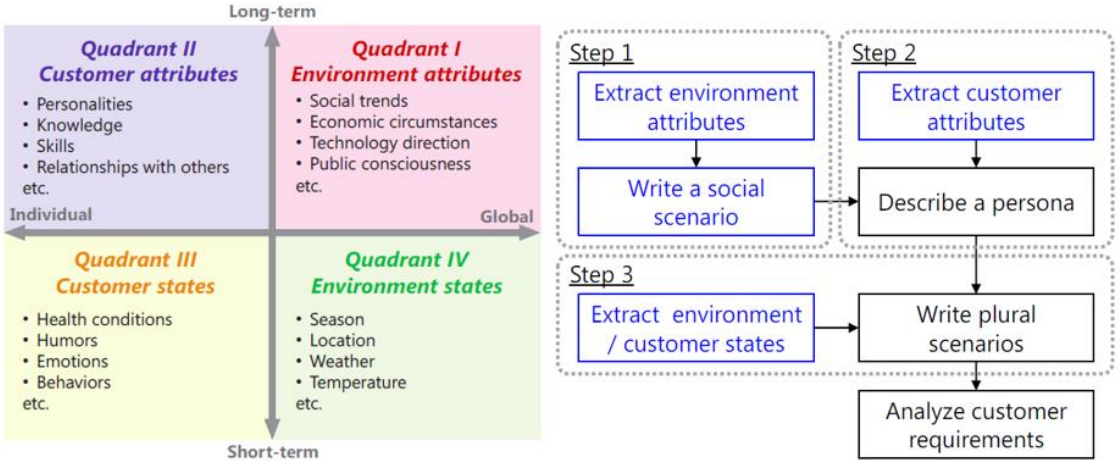

Figure 2. Contextual elements and utilization process.

Further on, service blueprinting approach is often used approach for designing services. Patricio et al. (2009) present service experience blueprinting technique for identifying, developing and describing customer experience requirements. They extend service blueprinting method and propose service experience blueprinting for designing service experience of service. Service experience, i.e. the experience the user has when using the service, is one of the key factors in service design, also in case of IS-intensive services. Therefore, paying attention to these quality requirements is worthwhile. Patricio et al. (2009) propose a multidisciplinary method for the design of IS-enabled service systems. They combine goal-oriented modeling use case approach, and therefore the approach can be used to increase shared understanding of the requirements and a common vocabulary for different parties of the ISD process. The service experience blueprinting method is based on ideas and also tools from interaction design and requirements engineering: goal-oriented analysis, conceptual modeling, and service blueprinting. The method can be useful in acquiring customer experience requirements in service environments. This method can be used to better understand customer experience requirements and other non-functional requirements, which typically have gained less attention in ISD process that functional requirements. As both Komssi et al. (2015) and Kauppinen (2009) do, also Patricio et al. recommend the use of multidisciplinary team to enable an integrated design of the service.

Service blueprinting has been found useful way to specify the interactions between different actors in service interaction process (customer, ISs and actors

on service provider side). Further on, customer journey mapping, a customer experience design technique that has been much discussed and even debated on the industry domain during recent years, also builds on the ideas of service blueprinting and extends it to customer interaction design.

Service experience design and requirements acquisition in that context of IS-intensive service development has been on focus recently in the digital industry. This thinking extends also on the more traditional fields of industry, such as manufacturing. Especially platform-based models are under great interest concurrently. For instance, Ryynänen et al. (2016) present a recent study on the requirements for a collaboration driven product-service system. The researchers present that modern manufacturing business based on product service systems require software platforms, which can support offerings in over the service lifecycle. Ryynänen et al. (2016) integrate the Product Life-Cycle Management (PLM) and Service Life-Cycle Management (SLM) approaches in to develop collaborative Product-Service design and manufacturing engineering platform. Also Ryynänen et al. (2016) emphasize stakeholder involvement and importance of collecting and managing feedback from stakeholders. This is essential for designing and building functionalities and features that best enable efficient use and maintenance of the product service machinery.

Whatever requirements acquisition and analysis methods are selected, the organizations that develop ISs should also pay attention to measuring and developing their requirements management processes. For instance, Hooks & Farry (2001) propose several measures, such as number of change requests, number of error reports, etc., which can be used as performance indicators for measuring requirements engineering activity. The IS developers should develop a set of measures, which can be used as requirements management and engineering performance indicators.

3.3 Software product management and roadmaps

Product management, as defined by Ebert (2014), is the discipline and business process that governs a product from its inception to its delivery. Product management consists of activities that enable IS product to be commercially viable. The primary tools for product management are the business case, requirements acquisition and analysis methods and product roadmaps for IS-based product. Ebert (2014) also include managing risks and uncertainty, mastering stakeholder needs, and improving accountability toward business targets into product's life cycle management. All organizations developing IS-based products should implement product management in some form. Also the requirements risk issues should be covered in product management practices.

New mass-market software solutions and cloud-based business models bring new aspects to product management, as the solution can be delivered globally and developed rapidly. Poor product management causes insufficient pro-

ject planning, continuous changes in the requirements and project scope, configuration problems, and defects (Ebert 2014). Also, Ebert points out that often the focus in product management is too much on technology and on the features of the IS, but not enough on the value that it produces: extensive feature lists are collected, but only half of the originally listed requirements end up implemented in the final product release. This observation demonstrates one essential requirement engineering challenge: how to identify the right requirements?

To advance the product management activity in a product-based business model implementation in an organization, Ebert (2014) lists four success factors for product management: (1) a core product management team with reliable commitments from all functions; (2) a standardized product life cycle with clear interfaces, milestones, and governance; (3) prioritize the requirements that transport the customer value to ensure business focus; (4) and implementing portfolio management and roadmapping to facilitate transparency to the development and management of dependencies.

Ebert (2014) proposes introducing a standardized product life-cycle process to support product management activity in the organization. The life-cycle model should also describe the requirements engineering and management practices. Ebert describes the requirements as a contract mechanism for the project internally and a client externally, and proposes a structured and disciplined documentation and management of the requirements. Ebert also emphasizes the importance of enabling both technical and business evaluation of the requirements, conducted by the core product management team. This is to avoid making requirement changes without checking technical feasibility, evaluating business case and understanding downstream impacts of the change.

A standardized product life cycle, which sets the guideline for the product management and development process, should guide the development and maintenance of IS (Ebert 2014). Also, adequate risk management techniques must be implemented as part of the life cycle process. This should also cover risks related to requirements. For requirements risk management, multifaceted analysis and documentation of requirement changes helps in improving requirement risk management.

As pointed out earlier, the product management activities should focus to value production: this allows a solution to truly address a customer need and have a strong business goal. Therefore, the product and its market should be frequently assessed to judge on product value proposition and development priorities. Managing and maintaining roadmaps is in the core of the product portfolio management and their development activities (Ebert 2014).

Ebert (2014) proposes that the product team should consist of the product, marketing, project, and operations managers for each product (release). This core team decides about requirements and their priorities, test strategies (covering both technical testing and business case testing), increment planning, backlogging, and so forth.

3.4 From long-term roadmap to short-term backlog

To recognize new opportunities and demands set by the markets, competitors and new technologies, and to lead IS-intensive product and service development, companies implement roadmapping activity in some form. The purpose of roadmapping is to help focusing the development resources on sensibly on the long term. Product roadmaps, in general, are time-based charts, which includes at least commercial and technological view to the IS-intensive product or service development (Komssi et al. 2015). The roadmapping activity should focus on long-term planning, i.e. longer than "next release horizon" (Komssi et al. 2015), instead of short-term product planning.

Komssi et al. (2015) present an empirical study of two Finnish companies with software-intensive products and services. Their focus is on customer value creation in roadmapping process. They present the following key findings: 1) trouble with linking business strategy to solution planning; 2) feature-driven mind-set of developers hamper customer process understanding; 3) difficulties in defining and commercializing services, as business cultures and the personnel's profiles were technology-oriented; and 4) fragmented customer knowledge in the organization. The researchers point out that relevant knowledge about customers in the company resides in different functions within the organization, and roadmapping should allow collecting and merging that knowledge in a useful way. In addition, the researchers argue that the service development is typically neglected in roadmapping, and the focus is in product features.

Table 4. Six steps for improving customer understanding in roadmapping

	Step	Explanation
1	Form a cross-functional team	A cross-functional team should consist of participants with different functions, which more probably holistic understanding of customer processes and also business strategy of the company.
2	Examine the business strategy	The business strategy should be understood by the participants: especially target markets, business domains, technological drivers, and focus between existing and potential business (=customers) should be covered.
3	Select a customer segment	The customer segmentation should be done so that segments consist of similar kind of business process related to solution. The team should be familiar with the selected customer segment and strategically important.
4	Identify the customer's activities	The team considers customer's activities holistically segment by segment. The analysis should be kept on abstraction level suitable for roadmapping.
5	Analyze the customer's activities	To analyze current situation, analysis should focus first on current solution and then to customer benefits minus sacrifices for each customer activity. The business potential (sales opportunities, customer loyalty) at each customer activity is also analyzed here.

6	Linking the business potential of customer activities into a solution roadmap.	The team should prioritize the customer activities based for instance compatibility with the business strategy and the readiness of the existing solution. Based on the prioritization, customer activities can be placed in a solution roadmap.

Therefore, a functioning roadmapping process brings the relevant people together, and allows them to share and analyze information to better understand the value creation of their product to customer. The format of the roadmap is not the most relevant issue: Komssi et al. (2015) bring up that many of the benefits of roadmapping are derived from the roadmapping process rather than the roadmap itself. The researchers propose that software companies should shift their focus from the prioritization of software features to the analysis of customers' processes and the prioritization of customers' activities. In order to do that, a six-step process for improving customer understanding in roadmapping is proposed (see Table 4). These steps can be applied also in requirement engineering process for the requirements acquisition purposes also.

3.5 Summary

Project Management Institute (PMI) conducted a survey (Smith et al., 2014) to project management professionals to have a look at into the current practice of requirements management and its impact on projects and programs. Requirements management, as defined by PMI (Smith et al., 2014), is the discipline of planning, monitoring, analyzing, communicating and controlling requirements, including communication among developers and stakeholders, and requirements change management throughout the course of the development project (or program). Based on the survey, they argue that many organizations still miss sufficient capabilities for requirements management: they lack resources, relevant competences, and executive management commitment. It appears that in the industry the value of requirements management and engineering is not always seen, and making improvements on the area seem to demand big cultural, organizational and processual changes – and eventually remarkable financial contributions. This may lessen business managers' interest in requirements engineering development efforts.

The literature review of requirements engineering approaches shows that the problem area has been under active research both by industry practitioners and by scientific researchers. The overview of literature also shows that there are recurring principles of requirements engineering, which are summarized in Table 5.

Table 5. Requirements engineering principles

Principle	Explanation

1	Stakeholder collaboration	Active bi-directional collaboration between different stakeholders improve IS developers' understanding of the use context, user segments, stakeholder needs and requirements, value production and eventually also stakeholders' commitment to development efforts.
2	Business targets	The business targets (the business model and IS connection to strategy) should be understood by the IS developers and other stakeholders and lead the requirement development efforts.
3	User segmentation	The (intended) user base of the IS should not be handled as a one big lump. Proper segmentation helps in understanding the value creation, use contexts, usage drivers and use scenarios. Segmentation approaches may vary.
4	Consider use context	Understanding the context of use, including the user state and users' environment states, is important in understanding value creation and user experience design.
5	Requirements model	Select requirement model that is expressive enough for describing, visualizing and explaining the requirements (see also item 9 of this table).
6	Focus on value creation	Focus on value creation to users (and other stakeholders) instead of IS features or functionalities. This demands that IS developers understand what is valuable to whom (of the stakeholders and user segments) and how the IS can assist in creating more of it.
7	Experimentation and feedback	Making experiments and testing the design assumptions (such as prototyping or iterative implementation) helps in verifying requirements at early stage. Collecting specific enough feedback enables making analysis and design decisions. Developing the solution in increments and deploying increments into production use as soon as possible provides an ultimate testbed with real-life users and use contexts.
8	Multi-disciplinary design team	A multi-disciplinary design team has better chance to develop holistic understanding of the IS application area. Consider at least having different functions (marketing, sales, business lines, customers etc.) and domain-specific knowledge (technical competence, business competence, application domain knowledge, delivery competence) present.
9	Requirement characteristics	Requirements should be understood widely enough, and impact of other than functional and systems requirements should be assessed: how to analyze also quality aspects, emotional design, cultural values and experiential requirements.

It is notable, that the agile development methods propose many of the principles of the table above. In the next chapter an introduction to agile requirements is given.

4 Agile Development Methods

Agile IS development methods refer to a range of development frameworks, which implement agile principles and values. These frameworks, collectively known as agile, have been under development already from 1980s and 1990s. There are different agile methods and approaches; Scrum, XP, Kanban and Lean some to mention. However, they all borrow from same ideology and are based on the set of agile values and principles. In essence, agile ISD is about allowing continuous inspection and adaptation to changes in a dynamic environment. There is no single agile framework, which would provide everything that is required to deliver and govern information systems. Each agile framework has slightly different approach, and typically in real-life adaptations different frameworks are integrated with each other to provide an overall agile operating model for the organization. Therefore, a feasible agile solution is dependent on the area of business and the context within which the system is being implemented. (Measey 2015, 24-25).

In this chapter, we will have brief introduction to agile values and frameworks, but focus more on requirements analysis and risk management in agile ISD.

4.1 Values and Principles – The Agile Manifesto

The Agile Manifesto describes four values and 12 principles supporting the agile values. The values set out in the Agile Manifesto are (Measey 2015):

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

The 12 principles that the agile frameworks build on are expressed as follows (Agile Manifesto, 2017):

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Working software is the primary measure of progress.

6. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
7. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
8. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity - the art of maximizing the amount of work not done - is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

The agile values and principles foster practices, which helps in diminishing risks in requirement acquisition and analysis – such as focus on customer value, continuous feedback and improvement, and co-operation of business and technical people. We will discuss this more in chapters 5.4 and 5.5. As in this thesis the focus on studying how requirement risk handling is considered in ISD approaches, we do not go further in describing different agile frameworks.

4.2 Generic agile development framework

There are many agile frameworks that can nowadays called de-facto standards. As these standard frameworks all overlap and often use different terminology to describe the same thing, Measey (2015) describes a generalized agile process to describe a generic agile model. For the purpose of this thesis, we do not go in details of different agile methods, but instead refer to the generic agile model by Measey.

Figure 3 (Measey 2015, 38) illustrates elements that the generic agile process consists of. The agile process, as illustrated, wraps around product backlogs, iteration backlogs, release plans, agile artifacts, and agile activities performed by customer, team and other relevant stakeholders. The customer, often represented by product owner, develops the product backlog with help from the team and the stakeholders. The team is responsible for delivering product increments, each divided to iterations/sprints. Increments may constitute larger releases, which may require governance (projects or continuous delivery). The team performs planning at various levels, such as release and iteration/sprint planning. In scaled agile (such as SAFe), planning may occur across several agile programs (or release trains in SAFe terminology). The team and the customer maintain release plans, product backlog and iteration/sprint backlogs. Agile technical practices lay the foundation for technical implementation and deployment (delivery).

Team holds stand-ups (daily meetings, scrums) daily within iterations/sprints. The sprint/iteration and release status of work is made visible using visual boards. Risks, problems, dependencies and assumptions are stored and managed on the RAID log (risks, assumptions, issues, decisions). In the end of each iteration/sprint and release, the team and customer will demonstrate the delivered product increment (show & tell) to the stakeholders. Feedback is collected from the iteration demo about the feasibility of the product and prepare the next planning period. Detected risks are added to the RAID log. The team performs a retrospective learn from successes, failures and changes. The agile lead is responsible for facilitating and enabling the process gives coaching the team. Agile process can be scaled across many teams.

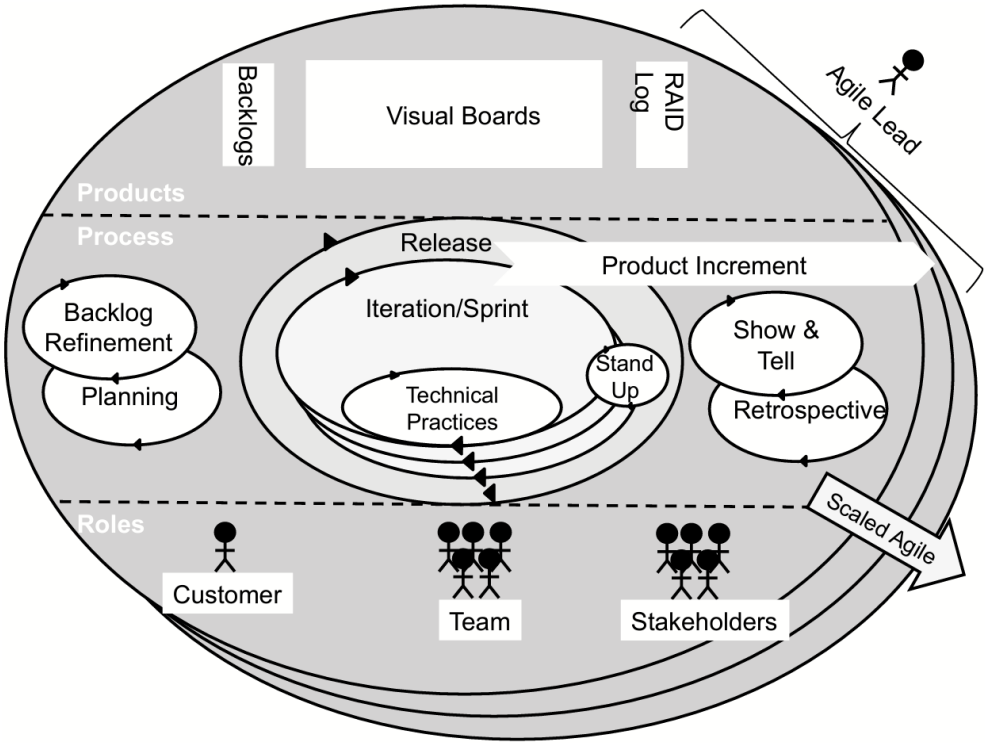

Figure 3. Generic agile process simplified.

Four key roles are generally recognized in agile development model, namely the customer, the agile lead, the agile team itself and stakeholders.

The customer refers to a role that is responsible for making decisions about the backlog contents and its prioritization. Therefore, the customer is not necessarily the actual customer of the product or project, but in agile the is customer is often represented by product owner or equivalent. In any case, the customer has to be able to represent the needs and priorities for the product, i.e. define the stories to be delivered, decide the order of the stories on the backlog, and approve each iteration/sprint or release (Measey 2015, pp. 43-44).

The agile lead refers to a role that is external to the team, but This role has different names in different agile frameworks, but the responsibility is somewhat

similar. The agile lead serves the team as a change agent, agile process coach, sponsor, facilitator or enabler within the organization – depending on the context. The agile lead's main responsibility is to enable the self-organization and continuous improvement of agile teams. (Measey 2015, pp. 48-50)

The term 'stakeholder' refers to people and organizations that have a real or perceived connection or interest towards the project outcomes. Identifying, analyzing the stakeholder network, and managing stakeholders is important for the product success (Measey, pp. 50-52). Stakeholder analysis helps in preparing a proper communications plan for the project.

Agile development teams and their operability are in the core of all agile frameworks. Agile teams are self-organized and carry the responsibility for delivering value to customer. Therefore, agile teams are typically responsible for deciding how to do the work, defining how long the work will take, deciding on division of work, and delivering the product. Depending on the project, various team types may exist, such as feature teams, component teams, core and support teams (Measey, pp. 45-53). The team has the authority to decide how to do the work, who does it and how long it should take. To ensure consistency in quality and delivery, proper standards and guidelines for the agile teams are defined – this is the responsibility of the agile lead. I.e. the agile teams have authority to decide how to best organize themselves to deliver maximum value to the customer within the boundaries set by the agile lead. Measey (2015, p. 51) states that teams would not have the authority to choose who is on the team, what business problem they are expected to help solve, or what budget they have. Also, technical and architectural restrictions may be set for the teams.

In agile, the teams also have an important role in planning and specifying the solution. The team refines the user stories with the customer and then dividing them into tasks. Based on the task breakdown done by the team, a proper estimate of the effort needed to complete the work. Team also divides the work, and follows the progress of each task in daily meetings (and decides how to intervene if problems occur) (Measey 2015, pp. 59-60).

4.3 Requirements and agile methods

As described in the previous chapter, agile development emphasizes customer value delivery. This reflects also on how the requirements development is approached in agile development. In the following subchapters, agile requirements specification principles are discussed.

4.3.1 Agile requirement specifications

User stories – or simply 'stories' – are commonly used in agile development to represent user needs and expectations towards a system. Stories define what is required from the IS development team by the customer and other stakeholders.

As the name suggests, user stories are written in a form of narrative description from the perspective of a user persona. This way the stories are understandable by the stakeholders and especially the customer (Measey 2015, 53). In agile development, user story is not a detailed specification of a system requirement, but instead it is a more like an instruction to the IS development team on what kind of feature or other requirement (such as fulfilling a standard or some other external limitation) is needed (Measey 2015, 52-53). For instance, Leffingwell (2017) states that "*User stories are not requirements. Rather, they're short, simple descriptions of functionality the user would like to have, usually told from their perspective and written in their language.*" Therefore, user stories in agile have a rather interesting distinction from the traditional requirements specification approach, where the requirements typically describe the expectations of the IS properties. On the other hand, some industry ISD practitioners argue that that in many real-life IS project environments this difference is rather theoretical, may not be very visible or existing at all. This is because also in agile environment the requirements may have to be fairly detailedly described because of other project reasons (for instance to fulfill the demands on estimating project timeline and budget or because of reasons associated with IS delivery contracts). In addition, also in agile development user stories are commonly complemented with other ways to describe mandatory requirements, restrictions or non-functional requirements. Nevertheless, in agile development the agile teams have lot of decision power on how the user stories will eventually be implemented.

User story defines what kind of feature is wanted, who wants a feature, and why they want it. In addition, properly written user story also includes acceptance criteria to describe when the implementation fulfills the story appropriately (Measey 2015, 53). The user stories should describe small, independent vertical slices of IS functionality, and be sized so that they can be completed within one iteration. Leffingwell (2017) also separates user stories (which deliver functionality to end user) and enabler stories (which define work items needed to support architecture, infrastructure, and other non-functional requirements).

Measey (2015) and Leffingwell (2017) both give very similar description for a good user story. A high-quality user story should be independent (of other user stories), negotiable (not an exact feature specification), valuable (for the customer), estimable (to sufficient level), small (fit within an iteration) and testable (has acceptance criteria defined). These criteria can be explained as follows:

- Independent stories are deliverable independently of each other, i.e. they are not connected. Often independence is easy to achieve on high level, but once the detail level of the user story gets deeper, dependencies may emerge.
- The idea of user story is that it is not a detailed requirement specification, but something the team can refine and negotiate until it gets designed and added into sprint plan.
- Valuable story is such that the customer agrees on the value of a story – only after that the story can be added to backlog.

- Estimable is understood by the team must be understood by the team as correctly. This often requires that the team is involved in the developing and refining of stories to have deep understanding of the story and to be able to estimate the stories.
- Stories should be small enough, which means that they should be implementable in one sprint.
- Stories should also include acceptance criteria against which the completion of the story can be tested. Criteria also guide the test planning.

The stories that do not meet these criteria need to be reworked or rewritten completely by the team (Measey 2015, 54). The criteria are summarized in Table 6.

Table 6. Qualities of a good agile user story.

Quality	**Explanation**
Independent	Deliverable independently of each other
Negotiable	Refinable over time
Valuable	The value of a story is understood by the customer
Estimable	Understandable by the team to make the estimates, against which the plans are made
Small enough	Implementable in one iteration (roughly 1 to 5 days of work)
Testable	Stories must include testable acceptance criteria

User stories are by no means the only requirements specification technique applied in the agile ISD contexts. Others are used as well, such as use cases, textual requirement descriptions or process descriptions. However, user stories are commonly used in agile, and furthermore they build on something that is often considered to be in the core of agile requirements: the end user and his usage scenario.

By comparing the characteristics of a good requirement in Table 6Table 2 and characteristics of a good agile user story in Table 6, we can see that the characterizations are quite similar. Both recognize the importance of value for user (necessary vs. valuable), verifiability (verifiable vs. testable), understandability (estimable vs. comprehensible), and feasibility (feasible vs. valuable). Also, requirement independency, sufficient accuracy level, and avoidance of contradiction is considered in both lists. However, some differences exist. First, there are differences in how the ambiguity is handled: whereas 'a good traditional requirement' should be so clearly specified that ambiguity is removed, the agile developer believes more in negotiability of user stories to increase the detail level and remove ambiguity. Second, in the agile development, prioritization of the user stories is done on a continuous basis in the development process and it is not considered to be a characteristic of a user story. Third, in agile approach small enough user stories are preferred to support iterative implementation – no such criterion is described for traditional requirements.

4.3.2 Agile requirements development

From the requirements management perspective, it is important to understand that in agile the development team has lot of decision power (and responsibility) on how the IS is implemented so that it fulfills the user stories (requirements). As agile teams are self-organizing teams, they have authority to decide how the work gets done, who does it and how much time is needed for it. This process involves typically refining stories with the customer (possibly presented by product owner or equivalent), prioritizing them and dividing them into implementable tasks. Therefore, it is essential that the user stories describe real needs of the users. In agile, the customer representative, such as product owner in Scrum, has the responsibility over interpreting the end user needs and formatting them into user stories and further on backlogs.

In agile development, the user stories are processed in a continuous process of refinement and prioritization. This is being done throughout the lifetime of the IS. This means that stories will continue to evolve from inception of the product through to decommissioning the backlog (Measey 2015, 56-60). This practice is especially important in complex or rapidly changing environments: the user stories are developed in line with the evolving environment. To do this in practice, the IS developers should avoid the temptation to develop the details of all user stories in the early stage of the project (or release). If the stories have very much details, and/or they are created too early within the delivery process, it is laborious task to maintain them when the situation changes. Instead, in agile the details of user stories should be developed just-in-time, for instance for the next sprint (Measey 2015, p. 58).

Prioritization of user stories (i.e. requirements) is in significant role in all agile frameworks. The prioritization is connected to timeboxing: majority of agile frameworks implement timeboxing in some form, meaning that the development is divided to time periods (for example releases and sprints/iterations), and the team output is delivered within these stints in a prioritized order. Therefore, it is relevant that the user stories are prioritized and implemented according order. Typically, so-called MSCW prioritization (Must-have, Should-have, Could-have, Won't-have) is done for user stories allocated inside an iteration (Measey 2015, p. 59). That is, when the stories are added in the timebox (such as iteration, release, or project), then they are also arranged in a prioritized order within the timebox. Measey (2015, p. 60) points out that the prioritization in agile is closely associated with timeboxing.

MSCW prioritization requires that the user stories are defined in a detailed enough level. If the user stories are on too generic level, the customer has difficulties to give them priorities and then the user stories get easily defined as 'must haves'. Adding more details allows better evaluation of priorities (Measey 2015, 60). We see here slightly contradictory demands. On one hand, flexible change management requires that the user stories are not on a detailed level. On the other hand, the more detailed user stories are easier to prioritize. To solve this, a practical compromise must be found by the agile team.

4.3.3 SAFe Model and SAFe Requirements

Many agile practitioners have concluded that the most popular agile frameworks (Scrum or XP for instance) do not sufficiently address the problems faced by enterprises or large software organizations. In principle, majority of the agile frameworks can be scaled to fulfill the needs of larger enterprises, but most of the methods do not describe how it should be done (Measey 2015, 159-160). To bring the benefits of agile development methodologies into complex IS development context, a more comprehensive agile framework is needed. One of the most extensive agile models is probably Scaled Agile Framework (SAFe), which we use as an example of scaled agile framework in this thesis. According to Measey (2015, 159), *SAFe is the most fully featured, discussed and implemented agile scaling framework*.

It is apparent that the rather simplistic basic agile requirements modeling approach with user stories collected into a product backlog (even if structured with epics and divided into sprint backlogs) is not sufficient neither for larger businesses developing IS solutions nor for developers of complex information systems. The model is not expressive enough to describe various requirement types and their relationships and dependencies, which are unavoidable in complex scenarios. Therefore, also agile requirements models need to be scaled.

The SAFe framework consists of process model that reaching over different levels in an enterprise, a mid-level planning cycle (the program increment) covering the activities of many agile teams (an agile program), funding of the agile programs, means to align programs to deliver value to the customer (a value stream), associated agile values and practices, and four core values (code quality, alignment, program execution and transparency). SAFe directly borrow or refer to other agile approaches, namely Scrum, eXtreme Programming, Kanban, lean thinking, and the Agile Manifesto.

SAFe also extends the agile requirements specifications. Scalable requirements model by SAFe, illustrated in Figure 4, consists of epics, capabilities, features, stories, and nonfunctional requirements (NFRs), plus their sub-categories. In addition, the model describes their relationships between the elements. It also includes acceptance criteria for testing to support verification of non-functional requirements. (Leffingwell 2017)

The SAFe requirements model can be used to replace or complement more traditional requirement specifications, while still supporting the agile development paradigms. In the agile development the trust on building the correct solution rest on the idea that the understanding of the correct solution emerges during the development, and the decision-making in development process is guided by the vision of the IS. This is important distinction to the idea that the correctness could be ensured by increasing the requirements specificity in the IS specifications. Leffingwell (2017) argues that the SAFe requirements model leaves room for this process of emergent understanding of customer value development.

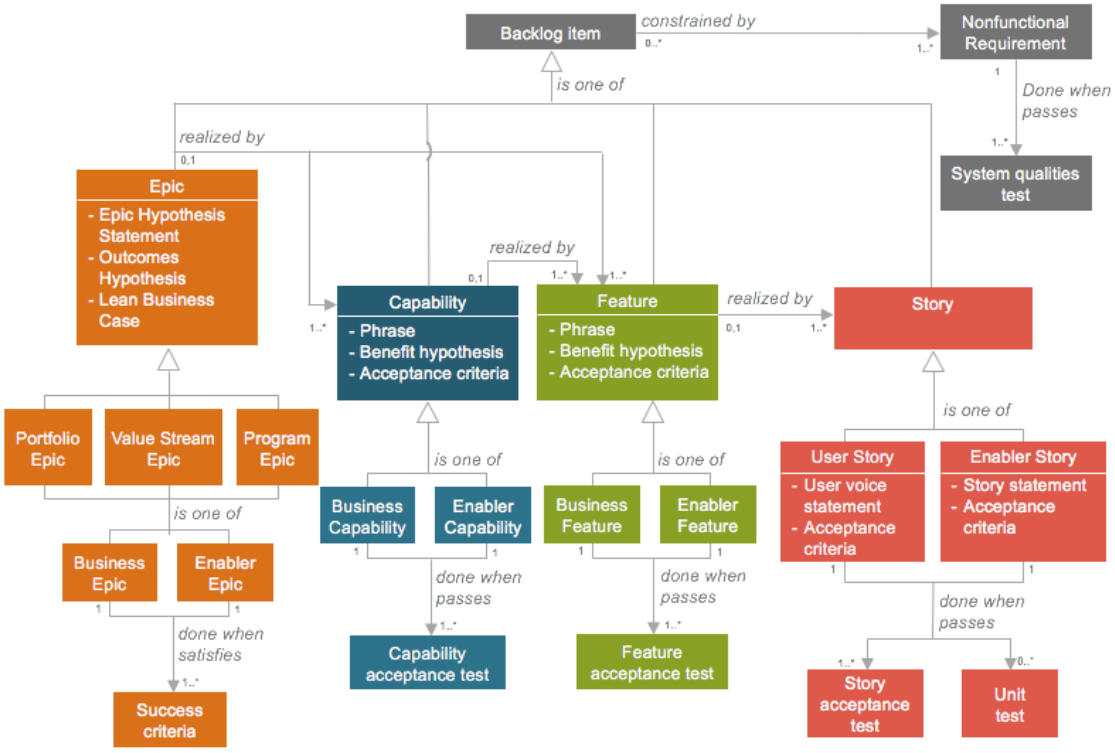

Figure 4: SAFe Requirements Model.

As in agile development generally, also in SAFe the requirement items are collected in backlogs. Depending on the selected SAFe configuration, there may be different types of backlogs, such as team backlogs, portfolio backlogs, program backlogs etc. Depending on the purpose of the backlog, the backlog items have different characteristics. The concepts of SAFe requirements model are explained briefly.

An *epic* is a portfolio backlog item, which defines characteristics of a larger entity, i.e. solution. As the term "solution" suggests, an epic has that much contents that it may require analysis phase and (financial) approval before entering the implementation phase. When implementing epic, also Minimum Viable Product (MVP) for the epic may be defined to help in user story prioritization. Epic is implemented over multiple increments. Leffingwell (2017) states that the epic implementation work continues until the optimum economic benefit is achieved. Epics can be portfolio epics, value stream epics or program epics, depending on the SAFe configuration and the level of backlog is discussed. Two types of epics are introduced in SAFe: business epics and architectural epics. (Leffingwell 2017)

As described in chapter 4.3.1, *stories* are brief descriptions desired functionality written in the user's language. In SAFe, *user stories* deliver functionality directly to the end user and *enabler stories* represent work items needed to support exploration, architecture, and infrastructure. Epics consists of set of stories.

SAFe requirements model also introduces *features* and *capabilities*, as illustrated in Figure 4. Feature is a system service that fulfills a stakeholder need. Each

feature defines what benefit it produces (benefit hypothesis) and acceptance criteria. Features consist of several stories, and should be sized to fit within single agile release train of a program increment. A capability, again, is a higher-level solution behavior that typically spans multiple agile release trains. Capabilities are split into multiple features. Features and capabilities are developed and managed, and they progress through the funnel of analyzing, backlogging, implementing, validating, deploying, and releasing phases. In SAFe, this process includes business and technical impact analysis, which stands for strategic direction setting for incremental implementation. The SAFe requirement model is hierarchical: epics > capabilities > features > stories. Capabilities and features are simply additional level of abstraction above stories, and they can be utilized if considered necessary in complex development scenarios. (Leffingwell 2017)

In SAFe model, non-functional requirements (NFRs) define system quality aspects of the system, such as security, reliability, performance, maintainability, scalability, and usability. NFRs set constraints and restrictions for the implementation across the entire system implementation. In agile development, NFRs are sometimes added in as definition of done statements for each iteration. In SAFe, NFRs exist at all backlog levels, that is team, program, large solution, and portfolio. Leffingwell (2017) emphasizes that *an adaptive and incremental approach to exploring, defining, and implementing NFRs is a key skill for agile teams*. Over-specifying NFRs lead to too complex solution, but under-specifying them will lead to inadequate solution.

The SAFe requirements model may appear complex. On the other hand, Leffingwell (2017) points out that most practitioners need only a subset of these items. An agile team can be running perfectly fine mostly using team backlog, user stories, acceptance tests, and NFRs. However, understanding the purpose of all elements help practitioners to choose correct requirements model to be applied in a more complicated scenario.

4.3.4 Agile requirements summary

Agile requirements come in many shapes and forms, but the most common form is user story. It is notable, however, that the agile methods tend to focus on how the work of the development team is organized and directed, and how the communication between the customer and the team occurs. The connection between the customer and the team is essential content in the models, but the agile models do not in fact give much on how to identify, acquire, and analyze the requirements (i.e. customer needs) from the stakeholders in practice. It is expected that the customer has representation in the agile development, but the question remains from where and through which process does the product owner generate the understanding about the end user value creation, IS vision and eventually the user story contents and their priorities. In a real-world IS development, processes such as product management and roadmapping, as presented in the chapter 3, need to be implemented in organizations to foster this process.

4.4 Agile principles in action

The following subsections describe main agile practices, by Measey (2015, pp. 69-90), which are generic for majority of agile frameworks.

Iterative development and short feedback loops
In agile, short feedback loops in different levels of operation are applied. The feedback loops refer to reflect-and-adjust approach to enable in continuous improvement. Furthermore, agile IS development is done in short iterations/sprints and the results are released frequently. This allows short feedback loops also for the delivered software, and enables quick corrections and steering the development accordingly.

Iterative development also means that working software is delivered in each iteration, i.e. each iteration/release should be deployable into use. Working software is the measure of progress. Short development iterations therefore allow continuous delivery of value to customer.

Efficient communication
Agile development emphasizes active communication between the relevant roles. This includes communication within the team, and especially between the team and the customer (as well as towards all stakeholders). In agile, business teams and development teams must work together. Face-to-face communication is preferred in agile, but in case of distributed agile teams, it is not always possible – then the means for team and stakeholder communication must be considered with exceptional care. The agile frameworks approve the position that communication is not only about transferring information, but also influences greatly on development of organization culture, team bonding and sharing the goals. Therefore, communication richness is valued highly.

Daily stand-ups
In agile, the progress of the tasks is monitored openly on a daily basis by the team in daily stand-ups. In these brief daily meetings, the team inspects the progress, synchronizes the work, and brings up corrective actions. The daily stand-ups are not project management reporting meetings, but focus is on synchronization within the team. Typically, in daily stand-up each team member answers three questions: (1) What did I do yesterday to progress sprint/iteration goals; (2) What will I do today to progress the sprint/iteration goal; and (3) Do I see any issues that prevents me from progressing towards the sprint/iteration goal? This allows frequent and continuous follow-up of the task progress and potential problems in implementation.

Show and tell
The results of the iterations (and releases) are made visible by demoing them to the customer and other relevant stakeholders. The purpose of the iteration/sprint

reviews is to demonstrate to the stakeholders completed stories, to receive feedback immediately from the stakeholders, and get recognition for their work. The demos/reviews are essential part of the agile inspect-and-adapt approach.

Retrospectives
Continuous improvement is enabled by frequently reflecting the working methods and team functionality, and adjusting the operation accordingly. There can be different types of retrospectives, depending on the purpose – sprint retrospectives, release retrospectives, project retrospectives, some to mention. Often at the end of the sprint/iteration, the team will together consider what went well, what didn't go well, and what will be done differently next time. Retrospectives form essential method for continuous improvement.

Emergent documentation
In agile, the focus is in delivering the working solution, which is then documented properly, not the other way around. Typical types of IT systems documentation include technical design documentation, code comments, test documentation (plans and test cases), user documentation, and operational documentation. The team produces the documentation in line with the IS implementation.

Visual boards
The status of work items within iterations (or releases) is constantly visible on the visual task boards (such as Kanban board) and burn-down charts. There are different methods and tools for this purpose in different agile frameworks. However, they all share the idea of making the tasks and their progress openly visible to all relevant stakeholders. Further on, a RAID log (or equivalent arrangement) is maintained to hold information about tasks that are not in the backlog. For example, risks, assumptions, issues/actions, and decisions/dependencies (the abbreviation 'RAID' derive from these terms).

Sustainable pace
In agile, the team decides what is released and when, which allows sustainable pace of work. In addition, the agile ideology encourages to maintain teams working hours reasonable. This is intended to improve team happiness and prevent overloading the team, which improves productivity on the long run.

Focus on quality
In agile, the functional quality and technical quality are in focus. Functional quality refers to delivering the features and functionality the customer needs. In agile, customer collaboration is in focus. To validate the user requirements, acceptance criteria for each feature is defined.

Technical quality practices that are expected to be applied by agile teams are refactoring, continuous integration with automated testing, and test-driven development.

4.5 Continuous integration and delivery

Continuous delivery (CD) refers to a software engineering approach in which agile teams produce software in short iterations and keep the software releasable at any time (Chen 2015). CD can be utilized with agile development frameworks, i.e. the development work is organized and directed applying agile framework and the team applies necessary practices to implement CD. In continuous delivery, the new versions of software can be released even daily. Continuous Integration (CI) is closely related to CD. It is a software development process where the developers frequently integrate their code in a code repository, and an automated build process is executed to find potential integration errors as quickly as possible (Claps et al. 2015). Continuous integration is often implemented to enable continuous delivery.

The business reason for CD implementation is that it gives organizations possibility to rapidly and reliably bring new product features and service improvements into market, which should give competitive advantage. Chen (2015) describes in his paper benefits experienced by a case company after implementing of CD. He identifies six general benefits of properly implemented CD: accelerated time to market; better confidence in building the right product; improved efficiency and productivity; improved product quality; reliable releases; and improved customer satisfaction.

CD allows rapid releasing and thus delivering business value in software releases to customers more quickly. This helps in staying competitive on the market. Frequent releases also allow collecting user feedback more quickly, which gives better possibility to evaluate value of added features and focus the work on the most useful features.

In CD, the release pipeline is highly automated, which means that after committing code in the repository, a release process is launched, and environments are set up automatically. In addition, the release pipeline also runs automated tests for the release. This saves lot of time from release engineers and testers, and allows this time to be used in more productive tasks (instead of routine release tasks).

Implementing CD requires that the release process is carefully designed and implemented to release pipeline tool. This improves the reliability of release process, decreasing the effort spent on troubleshooting and fixing release issues. The release risks decrease as the release process becomes more reliable. Furthermore, as releases are done more frequently, also number of code changes in each release decreases, making releases less error-prone. Moreover, CD allows easier rollback in case release fails.

In CD the whole codebase tested automatically in in every release. If the tests find a bug, it can be fixed before moving on to implementing new tasks. This allows eliminating bugs faster.

All in all, lessening number of quality and delivery problems increase trust between development team and the customer, which obviously impacts positively in customer satisfaction.

The benefits of CD appear obvious, but implementing the CD process is difficult and therefore true benefits may not always be easy to achieve — especially in the contexts where there is legacy release environment or large enterprise architecture (Claps et al. 2015). Challenges and risks of implementing CD are discussed in chapter 5.3.

4.6 Summary

Different agile frameworks and their adaptations are widely applied in the IS development. By ISD practitioners, agile development approach is considered better-suited for the rapidly changing business environment. The agile approach emphasizes continuous delivery of customer value, iterative development and delivery of software in small batches, constant customer feedback, continuous improvement of operations, focus on quality, specialist-driven development, open communication, and preparedness for change. These are principles highly valued in concurrent industry IS development.

Agile frameworks and their requirements models have several built-in characteristics, which diminish project risks associated with requirements. They are discussed more in chapter 5.4.

5 Requirement Risks

The previous chapters of this thesis present an overview of the common activities in requirements engineering and management. The fundamental intent of the product management, roadmapping, requirement engineering and requirement management activities represented earlier is to increase IS developers' understanding of the customer value creation enabled by the IS, and consequently their confidence in developing and implementing the right requirements. Most of the requirement engineering literature concentrate on studying what kind of factors must be considered and how to do it in practice. This is obviously important, but also begs a question 'What if developers fail in identifying the right requirements?' What are the risks associated with poor requirements engineering and how to handle those?

5.1 Business requirements and requirement risks

Risk management in ISD refers to proper handling of risks that projects can be exposed to. This covers assessment, avoidance and control of the IS project risks. In risk management process, project risks are identified, prioritized and measures for risk avoidance, mitigation or recovery are planned. The risks and the measures are often documented in a risk management plan and risk identification listing (Wiegers et al. 2013, pp. 538-542).

Typically risk management of IS projects focus on various external or internal factors that may affect the project. Risks in IS development have been considered in numerous studies (e.g. Wallace et al. 2004, Persson et al. 2009, Mathiassen et al. 2007, Keil et al. 1998, Chen et al. 2015, Barki et al. 1993). Wallace et al. (2004) present a software project risk categorization based on the earlier literature on the field. The categorization has six risk dimensions: team risks, organizational environment risks, requirements risks, planning and control risks, user risks, and project complexity risks. One of the risk categories is requirement risks, which contains risk items that are specifically related to requirements engineering process and requirements themselves. Therefore, Wallace et al (2004) name requirements risk as one risk category in ISD. They point out that changing requirements is not the only requirement risk type for a IS project, but name also incorrect, unclear, inadequate, ambiguous or unusable requirements as potential risks for an IS project success. The study argues, based on the cluster analysis on the earlier paper on software project risks, that requirements risk, planning and control risk, and organizational risk are the most prominent risks for high risk projects.

What is noteworthy in the risk categorization model by Wallace et al. (2004) is that business risks are not explicitly covered in the risk categorization. Obviously, risks that may have business implications are presented within several risk categories, but approaching the requirements from business risk perspective is

missing. One can also argue that all risks are eventually business risks, which may also be accurate in many cases. However, Wallace et al. (2004) approach the project risk mostly from the IS project management perspective, which may differ from for instance business management perspective. Perhaps symptomatically, the word "customer" is not used at all in the research report.

Various risk analysis techniques for software project risks have been introduced in research - see for instance top 10 risk checklist by Boehm (1998) or software project risk framework by Keil et al. (1998). However, they do not provide dimensions for construing the requirements risk model or method to analyze and prioritize them.

Researchers have also introduced contingency models that describe requirements development in specific contexts. The model by Mathiassen et al. (2007) takes into account also requirement risk handling. The model categorizes requirement risks into four categories: identity risks (the availability of requirements), volatility (the stability of requirements), and complexity of requirements. The model also describes how the risk level (high, medium, low) for a project can be estimated based on the identified requirement risks, and further on how to handle the requirement risks. Mathiassen et al. (2007) describe requirements risk categorization, which divides them into identity risks, volatility risks, and complexity risks of requirements. On top of that, Tuunanen and Vartiainen (2016) add requirements integrity risks as a new category. Table 7 summarizes and explains the four requirement risk categories proposed by Tuunanen and Vartiainen (2016). The risk categories are discussed in more detail in the chapter 5.6, where the requirement risk prioritization method is explained.

Table 7. Requirement risk categories.

Requirement risk category	**Explanation and examples**
Requirements identity risks	The availability of requirements: high risk means that the requirement acquisition is difficult, or requirements are unknown or indistinguishable.
Requirements volatility risks	The stability of requirements: high risk means that requirements change easily.
Requirements complexity risks	Measures how easy it is to understand requirements: high risk means that requirements are difficult to understand, specify, and communicate.
Requirements integrity risks	Completeness and accuracy of the requirements elicited from end users.

In the chapter 2.2 listings of qualities of good IS requirements and requirements collections are given. As low-quality requirements pose a risk to the entire ISD project, assessing the requirements risks in ISD process should improve the end results. In Table 8, the requirements quality measures by Wiegers and Beatty (2013) and risk items from Wallace et al. (2004), Tuunanen and Vartiainen (2016), and Mathiassen et al. (2007) are put together and compared to see look for potential mismatches. One can notice that prioritization and verifiability are not present in the requirement risk categorization as such.

Table 8. Requirements risk items and quality measures

Quality measure	Related Risk
Completeness	Inadequate requirements (Wallace et al. 2004)
	Requirements integrity risks (Tuunanen and Vartiainen 2016)
Correctness	Incorrect (Wallace et al. 2004)
	Requirements integrity risks (Tuunanen and Vartiainen 2016)
Feasibility	Unusable requirements (Wallace et al. 2004), volatility
	Requirements integrity risks (Tuunanen and Vartiainen 2016)
Necessariness	Unusable requirements (Wallace et al. 2004)
	Requirements integrity risks (Tuunanen and Vartiainen 2016)
Prioritization	Requirements integrity risks (Tuunanen and Vartiainen 2016)
Unambiguousness	Unclear, ambiguous requirements (Wallace et al. 2004)
	Requirements integrity risks (Tuunanen and Vartiainen 2016)
Comprehensibleness	Complexity risks (Mathiassen et al. 2007)
Verifiability	Requirements integrity risks (Tuunanen and Vartiainen 2016)

For instance, Mathiassen et al. (2007) characterized risks related to requirements identity for situations where there is a communication gap between developers and would-be users, such as in development of generic applications or mass-market software. In another kind of IS project setting, for instance when developing business applications for specific user group and business process, the requirement risks may reside in other areas, such as in misunderstanding the customer value creation process. Also, the phase of the ISD process and requirements development process have effect on which requirement risks are emphasized: requirements elicitation/acquisition, requirements design, and IS implementation phases have different requirement risk profiles (Wiegers & Beatty 2013, Mathiassen et al. 2007, Tuunanen and Vartiainen 2016).

5.2 Practical means for requirement risk handling

Many project management specialists argue that most common culprits for project escalation are scope creep, lack of stakeholder involvement, poor stakeholder communication and inadequate support from the executive management (Smith et al. 2014). These phenomena involve or have impact on requirements development, since the process of identifying, defining, documenting and managing the requirements defines the solution, which a successful project should deliver.

Requirements engineering literature provides the ISD practitioners techniques and practices for crafting good requirements. The importance of reviews and assessments is also brought up by Wiegers & Beatty (2013, pp. 351-352), who propose that the requirements should be assessed by relevant persons in the requirements elicitation phase. The correctness of user requirements can be assessed with reviews by the users or user surrogates. Technical feasibility should be checked by a developer who participates during elicitation to evaluate the effort and cost required to implement requirements. However, the writers do not

propose a method for making the requirements review, but only remark that the reviewers should be trained for the work.

Wiegers & Beatty (2013, pp. 120-123) also emphasize proper tracing of requirements back to specific (user) input: it should be possible to connect each requirement to a business objective in order to indicate clearly why the requirement is necessary. Interestingly, the importance of traceability is questioned by some agile advocates, apparently because they see that the customer voice should be heard throughout the development in form of user stories.

Wiegers and Beatty (2013, p. 353) also bring up the importance of prioritization. The implementation priority to each requirement should be given. Requirements prioritization should be a collaborative activity involving multiple stakeholder perspectives. The writers also propose inspections to spot ambiguities in requirements. In a peer review each participant can compare his understanding of each requirement to someone else's, and whether the requirement is unambiguously expressed.

In requirements elicitation phase, i.e. the process of identifying the needs and constraints of the stakeholders of IS, the goal is to recognize and record the IS requirements. Requirements elicitation is perhaps the most error-prone phase of IS development, and decisions made there greatly influence the direction that the development process takes (Wiegers and Beatty 2013, pp. 119-120) Therefore, careful process of collaboration and analysis is advised for requirements elicitation.

Further on, incremental development approaches and proof-of-concept prototypes are two common ways to evaluate requirement feasibility.

The term 'scope creep' refers to a project phenomenon where the project contents keep expanding because of requirements and other demands are being added in the scope of a running project without leaving other contents out. Scope creep is common problem and often connected to changing requirements. The scope creep can be controlled by having a clear product vision and pre-defined project scope, which allow better change control (Wiegers and Beatty 2013, pp. 542-543).

Also, the completeness and correctness of requirements specifications improve the control over requirement risks, as preparing such require thorough analysis. Many methods can be applied from *prototyping* or *user group interviews* to *testing the assumptions*. Analysis requires usually customer involvement to receive input from real customers. Requirements for innovative products require especially careful consideration. Methods exist for helping in formulating the innovative product concepts – for instance, the Design Sprint concept by Google (Google 2017) introduces a one-week design and experimentation process for developing concept for a new product. Similar experimental approaches are feasible also when technically difficult features are present or unfamiliar technologies, methods, languages, tools, or hardware must be applied in the project. Creating models and prototypes is a way to prove the feasibility of the concept. In the elicitation process also non-functional requirements (for instance expectations of

quality, usability, security or reliability) need to be captured (Wiegers and Beatty 2013, pp. 543-545).

If the IS implementation phase is entered without prior knowledge about potential technically complex requirements, unpleasant surprises may occur as the effort and competences required for the implementation of such features do not become exposed until the feature is being implemented. Therefore, ensuring that technically difficult features and unfamiliar technologies are considered thoroughly enough is important (Wiegers and Beatty 2013, p. 544). For instance, *proof of concept implementations* may have to be made to gain better understanding on the technical solution and issues. In agile frameworks, *spike stories* may be added in the backlog early stages of the IS development process for this purpose. A spike story, as defined by Measey (2015, p. 58) is a story that drives technical or functional research effort or investigative work.

The customer collaboration is in the core of requirement elicitation, because it gives possibility to understand customer value creation. Therefore, engaging the customers to the requirements should be considered: how to ensure that the customers will remain committed in the requirements. This requires active collaboration with customer representatives (such as product champion) and potentially communication with wider user community (Wiegers and Beatty 2013, pp. 543). In a customer-provider IS projects, an agreement based on the requirements specification helps in keeping the scope.

The process of requirements elicitation requires *sufficient resourcing and time*. This must be considered when making schedules for IS development projects and making decisions on budgeting. Also, the requirements elicitation activities should be brought under the *continuous improvement* process (cf. agile retrospectives) to allow learning from experiences. Proper requirement elicitation efforts should be built into proper roadmapping process of continuously developed products. Business management needs pay attention to creating culture of collaboration within the organization and towards the customers to enable open dialogue and possibility for innovation to rise. Distrust and conflicts between different parties is poison for all collaborative activity also in requirements elicitation.

In requirement specification, the goal is to produce such specifications that requirements are understood similarly by all stakeholders. This requires avoiding ambiguous terminology, developing requirements specification to sufficient detail level, and handling open issues in requirements properly. The qualities of good requirements specifications are discussed in chapters 2.2 and 4.3.

Requirements validation is not a single, discrete phase of project where all the requirements are validated, but rather a set of activities accomplished in distinct phases of an IS project (Wiegers and Beatty 2013, 331-332). For instance, in agile the time for validating the implementation of a user story is in the iteration/sprint (or increment) review. Then again, many non-functional requirements must be validated differently: for instance, by running a load testing to validate that performance-related requirements are fulfilled. The IS development team must consider how to validate the requirements. In agile frameworks, user

stories should contain the acceptance criteria for validation and the tests should be designed accordingly. Validating non-functional requirements is done by integration testing, load testing and usability testing. However, testing the feasibility of the user requirements and validating business requirements is often requires testing in real-life environment with real users. Different methods are available, such as A/B testing, end user usability testing some to mention. Appropriate method depends on the IS and its stakeholders.

A common approach for validating the requirements is to inspect the deliverables in reviews. The reviews can be done either for requirements specifications or for the implementation (code reviews). If the reviews are done properly, they are an effective way to inspect the work and to engage stakeholders in the process. However, proper inspection of specifications may require training of the participants, which is often dismissed in real-life project world.

The changing environment and the resultant volatility in IS requirements is one of the biggest requirement risk in IS development. Therefore, preparing to change is probably the most important risk management activity in IS development. Consequently, proper change management practices should be implemented as part of requirements management.

In agile frameworks, one of the principles is to prepare for change, and the requirements analysis, prioritization and validation is an on-going process throughout the IS development life-cycle. If more traditional IS development framework is applied, change management process must be enabled, and it must include impact analysis, change control board, and tools to support the process (Wiegers and Beatty 2013, pp. 545-546). In any IS development model, clear communication of changes towards the affected stakeholders is essential.

The requirements risk management should be part of more generic project risk management activity. Wiegers and Beatty (2013, p. 546) point out that project managers should be able to identify requirements related risks (such as insufficient user involvement in requirements elicitation), record them into project risk analysis tool, follow the development of the risk, and escalate the risk when necessary. However, the writers do not present a model or tool specifically intended for identifying, evaluating and prioritizing the requirements risks as part of risk management efforts.

5.3 Challenges in continuous development environment

In agile development one of the key principles is to enable early and frequent delivery of valuable software for the customer. Continuous deliver (CD) approach aims at making the delivery of value continuous through automation of release and testing process. The benefits of CD are discussed in chapter 4.5.

However, implementing CD in real life IS development environment is not simple, especially in complex enterprise environments. The difficulties connected to CD implementation include relative easiness of deploying erroneous software and bugs, difficulties in reforming organization culture to support continuous

delivery, the need for adopting new agile principles in the organization, and increased need for cross-team collaboration (Claps et al. 2015).

When deploying CD into practice, technical difficulties are imminent and need to be solved. Rodriquez et al. (2017) conducted a systematic mapping study on the technical challenges in CD. They name and name the following technical challenges in CD: physical assets and hardware equipment must should also support automation; technical platform support for experimental scenarios; security and privacy issues require extra consideration; lack of trust in software quality; difficulty in managing various configurations and run-time environments; tensions between the desire to deliver functionalities quickly and the need for reliable products (Rodriquez et al. 2017). It seems that in comparison to more traditional ISD approach, CD brings new challenges to implementation, which may have to be considered in the requirements risk analysis.

Further on, Chen (2015) argues that a robust, out-of-the-box, comprehensive, and still customizable technical solution for CD doesn't exist, and each practitioner need to build own CD platform utilizing different tools and technologies. This can be both expensive and error-prone. Chen (2015) proposes building the solution on standards, using open APIs, and building an active plugin ecosystem. In addition, many applications may not even be amenable to CD (Chen uses large, monolithic applications as an example). Also Claps et al. point out that CD may not be suitable for all types of software, using enterprise software as an example of applications that are not well suited to CD. The technical implementation of CD is rather complex and time consuming, and require solving many practical difficulties, such as management of the changing code for frequent deployments (Claps et al. 2015).

However, despite all the technical challenges, the social and organizational challenges along the way appear even more important, as described by Claps et al. (2015), Chen (2015) and Rodriquez et al. (2017).

Chen (2015) points out that the biggest challenge in their case example was organizational, because the release activities involve so many departments in the company (from technical teams to product marketing and customer service). Each of the departments have their own targets and ways of working, and changing them is required to first implement and the get benefit from CD. Change towards CD requires that the organization and processes are redesigned to break down barriers among teams and promote a collaborative culture. This change process must be led by the leadership team. Chen calls for more research on understanding the challenges of adopting CD in more depth and developing strategies and practices for the change process.

Chen (2015) also argues that adopting CD also requires reformulating many of the old processes, because they hinder CD. Obviously, requirement management and change management process must be reconsidered to dismantle too heavy and too slow decision-making. However, usually CD has impact on business processes, software development processes, and operational processes also.

Change to CD also requires changes from individual developers: completely new mindset is required from the developers to release the new code directly into production environments. Change to CD may also have impact on external third-party developers because of the risk of non-compatibility of extension modules (Claps et al. 2015, Chen 2015).

Therefore, different social and human factors (including cognitive and learning aspects) in adopting continuous delivery must be carefully considered. Enough time must be given for the change process. Rodriquez et al. (2017) point out that for the adoption of CD, it is profoundly important to establish an agile thinking culture ranging from the individuals to teams and to upper management levels as well. In addition, transformation towards CD should be seen as an evolutionary change process: a direct transition to CD requires too many changes to handle at one time (Rodriquez et al. 2017).
Therefore, the when analyzing requirements for implementation in CD, the analysis efforts should also cater to the technical characteristics of CD.

The research and practical experience recognizes different challenging areas of CD deployment. Many of them are related to social aspects of organizations and have connection to requirements management engineering. In chapter 5.5 CD challenges from the requirements management perspective are discussed.

5.4 Managing requirements risks in agile development

Agile frameworks and requirements models have several built-in characteristics, which diminish project risks associated with requirements. First, agile methods focus on *continuous delivery of value* to the customer. To implement this, customer (or customer representative, actually) is continuously involved in the IS development process. This allows constant customer feedback, and gives better chances to detect unnecessary, misunderstood or faulty requirements.

The *iterative development* approach applied in all agile frameworks allows frequent collection of feedback from the stakeholders and also diminish the technical risk of the delivery.

In addition, the agile frameworks all *prepare the development team for change* in the environment or user requirements. Therefore, agile teams should be better equipped to handle changes in the environment or customer goals.

The agile models also propose defining a *vision statement* for the IS, which can be used to direct the work in rapidly changing environment and serve as a guideline for decision-making. Properly defined mission statement guides the agile team towards the most important business targets, making for instance requirement prioritization more accurate.

Further on, agile models also emphasize *active participation of stakeholders* and encourage *adoption of inclusive* working methods. Inclusive practices are methods, which make the requirement acquisition and analysis understandable for various stakeholders without wide knowledge about software engineering. Usage of inclusive methods is a prerequisite for active customer participation.

Applying these practices should improve the understanding of stakeholder requirements (including customer requirements) throughout the life-cycle of the IS. In agile models, the requirements are being *analyzed and prioritized* in a continuous manner. These agile development characteristics are well-suited for diminishing the risks related to requirement correctness, volatility and identity.

Table 9. Agile development requirement risk diminishing characteristics

Characteristic	Explanation
Emphasis on continuous value delivery	In agile, continuous delivery of customer value is paramount. To enable it, the IS developers must develop understanding on the customer value creation, and aim at continuous delivery of valuable software.
Iterative development	The iterative development approach applied in all agile frameworks allows frequent collection of feedback from the stakeholders and diminishes technical risk of the delivery. This also helps in diminishing risks related to requirement correctness and volatility.
Preparing for change	Agile attitude welcomes change, and prepares agile teams to handle changes in the environment or customer goals. This helps in diminishing risks related to requirement volatility.
Vision statement	Carefully considered, well defined and management-approved vision statement serves as a guideline in project decision-making, improving for instance requirement prioritization.
Active customer participation	Active customer participation aims at making the customer voice on requirements heard throughout IS development process. This helps in diminishing risks related to requirement correctness, volatility and identity.
Inclusive working practices	Inclusive working methods refer to practices that are simple and understandable to stakeholders, instead of using traditional software-based modeling tools. The usage of inclusive models allows the stakeholders to understand and analyze requirements. Common agile techniques are for instance user stories, free-form diagrams, use cases, user interface sketches, prototype, some to mention. This helps in diminishing the risks related to requirement correctness, volatility, and identity.
Continuous analysis and prioritization process	In agile models, the requirements are being analyzed and prioritized in a continuous manner, which helps in coping with requirements volatility.

However, agile models do not define how the customer requirements are identified and analyzed – for instance, how does the Scrum product owner develop the understanding about the customer value production to be able to describe and prioritize the user stories. Also with agile methods, requirements elicitation and analysis efforts must be carefully planned, and appropriate methods applied (such as proof of concepts, market research, end use studies etc.) To manage the process of developing customer understanding, product management and roadmapping activities may have to be implemented in the organization.

Even though agile development and continuous delivery may improve the confidence in building the right IS, the initial requirements acquisition and prioritization for a not-yet-existing IS remains a challenge. This is especially problematic for an IS that is developed for mass markets: there is not necessarily easily

identifiable customers, which could be used for requirements acquisition and prioritization.

It is also worth remembering that implementing agile development is not easy, and it requires commitment all the way from top management through entire organization. In addition, it demands a lot of skills and competence from the agile teams and product owners. Further on, agile is not for the lazy: once adopting agile, the agile process should be followed rigorously which requires remarkable efforts in leading the change. Slipping from the agile routines and principles raises the risk to slide from agile to ad hoc.

5.5 Risk management in continuous development environment

As IS development has been moving first towards agile and then further on towards continuous development approach, handling requirements and requirement risks demand new attention. First, it is important to understand that continuous development approach has built-in principles, which support IS developers in their effort to control requirement risks. One of the key benefits of agile software development and continuous deployment practice is shorter feedback loop, which allows diminishing the development of unnecessary and failed features. New features are developed and released frequently, and therefore feedback can be collected both on the implemented features and on what features the customers would like to add. This helps preventing waste software (Claps 2015, Chen 2015). This built-in feature of continuous development is also relevant from the requirement risk management: faster feedback loop allows quicker response to faulty or wrong features - if they just get detected. Detecting the faulty features or quality issues requires that feedback from the users is collected and the success of the IS features (also business success) is measured properly. Thus, evaluating and measuring the relevant system success factors should be covered in practical DevOps deployments. Research-based models for measuring IT success, such as the one by Delone & McLean (2003), could be applied in metrics design for the purpose.

Second, the agile principles that are applied in continuous development, emphasize strongly prioritization of the requirements. Requirement prioritization allows focusing the implementation efforts early on most important and valuable requirements, which lessens the total requirements risk in the ISD process. One of the quality criteria by Wiegers (2013) for requirements was prioritization.

Even though the benefits of CD are apparent, practical experiences show that many challenges remain. The faster feedback loop does not still seem to solve all the problems related to requirements and their correctness. Claps et al. (2015) identified, based on the case study, 20 social and technical challenges related to continuous delivery deployment. Part of them are also related to requirements engineering, if we consider requirements engineering being the mechanism for understanding the customer need, describing it coherently, and delivering IS fulfilling the needs. Issues directly connected to customer satisfaction identified by

Claps et al. (2015) were: demand for continuous product management documentation; difficulties in customer awareness and adoption of new features; testing and quality problems; and more complicated partner co-operation. These are summarized in Table 10.

Table 10. Customer satisfaction related challenges in continuous delivery

Challenge	Explanation and mitigation
Product marketing	Marketing continuously delivered product requires versionless product marketing and management.
Customer adoption	Not all customers are pleased to receive updates of features. This may require also maintaining multiple documentation for products with multiple offerings (e.g. a customer-hosted version and online-hosted version), where features may be released in one offering, but not in the other.
Customer feature discovery	Customers may not get the information or notice the new features. This requires new approach to deliver product update information.
Faster customer feedback	IS development team must be able to measure and analyze data, when justifying the deployment of a new feature. Deploying minimal changes quickly and then experimenting to decide, based on customer feedback, to update or remove.
Product quality	The quality of the software product may decrease, because errors may slip in since deployments occur more frequently. This issue is mostly mitigated by continuous delivery benefits: bugs can be fixed quickly and roll-back is easier
Testing	Automated testing must be built and rigorously applied, but having production quality tests and maintaining the process of 'code review' does not always work as wanted. Thus, ensure testing is thorough enough, and make sure the software releases are dependent on passed tests.
Partner plugins	Because SW keeps changing continuously, integration issues with partner systems get more complex (for example managing plugins or other integration solutions). This may require limiting the number of supported partner solutions to only those that can actively validate their compatibility.

In addition to above, Rodriquez et al. (2017) describe difficulties in release planning and managing the product roadmap in a fast-paced environment. Further on, there are risks associated with gathering user feedback in CD environment, since the user base is represented by a limited population, which may mislead product development or constrain the product evolution.

5.6 Requirement risk prioritization method

The previous chapters give an overview on how requirement risks are treated in contemporary ISD. As described, the risks associated with IS requirements are widely studied and their impact on IS success is recognized. The concurrent ISD approaches, such as agile or CD, advocate principles and practices that improve

the quality of the requirements engineering practices and the requirements themselves. However, it appears that the risk management approach towards the requirements is not ordinary, i.e. the requirements are not usually analyzed specifically from the risk identification and mitigation angle. Also, Tuunanen and Vartiainen (2016) argue that contemporary information science research does not provide many methods for prioritizing requirements risks, and that further emphasis should be given on how requirements risk prioritization is done in a way that takes account of agile and DevOps principles, while still supporting more structured information systems development approaches. To fill this gap, the researchers introduce a requirements risk analysis and prioritization method, which is applicable in IS development projects. The method is introduced in the following sub-chapters.

5.6.1 Overview of the method

The requirements risk analysis and prioritization method by Tuunanen and Vartiainen (2016) presents steps for identifying, assessing and eventually intervening the requirements risks in distinct phases of the ISD project. The method provides a list of potential requirements risk items for separate phases of ISD process, and general guidelines for identifying and prioritizing the risks. The risk prioritization is done by utilizing practical checklists to investigate how the risk profile of a project changes over the project life cycle. In addition, the method provides risk resolution patterns to assist IS project decision-makers to resolve the identified requirements risks. The method aims at helping IS developers to prioritize the overall requirements risks for ISD projects.

Method proposes four different requirements risk categories: identity, integrity, volatility, and complexity risks. The method is independent of ISD approach: it can be applied in projects that use traditionally structured ISD methods (waterfall), agile methods, or continuous development. The method only assumes that ISD approach has requirement elicitation, design and implementation phases.

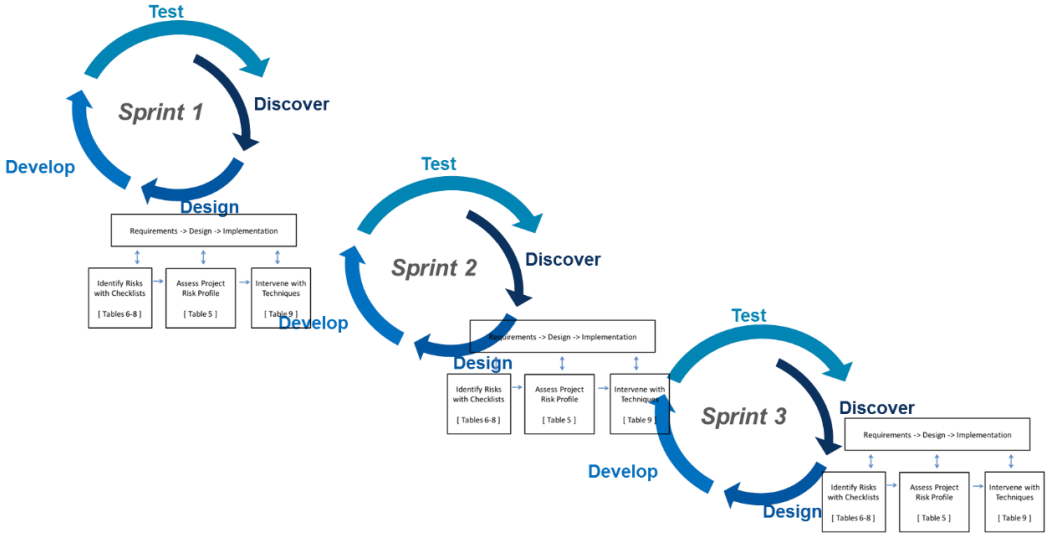

Figure 5: Requirements risk prioritization method in agile development

In Figure 5, an illustrative example of applying the method in agile or iterative ISD approach is given. In the example, the requirements risk prioritization method is applied in each iteration to evaluate the requirements stored in the backlog. The method could be applied as part of iteration planning session, when the backlog items are defined for implementation (for instance in Scrum terminology, in sprint planning sessions).

5.6.2 The requirement risk prioritization process

The requirements prioritization process by Tuunanen and Vartiainen consists of subsequent steps of risk identification, risk profile assessment and decisions on intervening techniques as follows (Tuunanen and Vartiainen, 2016):

1. Identify risks with checklists for each ISD phase. Nominal process starts with requirements phase and continue to design and implementation phases.
2. Assess project risk profile by recognizing individual requirements risks affecting the project. Use the indicative impact levels to prioritize requirements risks.
3. Intervene with requirements risk resolution techniques according to the risk resolution rules in following order using the appropriate techniques (see list in the appendix 1):
    - If identity risks are high, put high emphasis on discovery techniques.
    - If integrity risks are high, put high emphasis on prioritization techniques.
    - If volatility risks are high, put high emphasis on experimentation techniques.
    - If complexity risks are high, put high emphasis on specification techniques.
    - If three or more risk items are high, follow the above sequence of applying techniques (from 1 to 4).

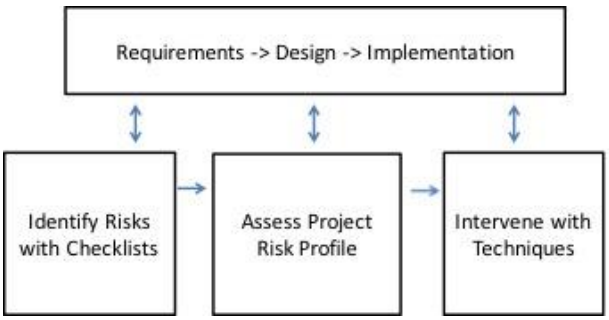

Figure 6: Requirements risk prioritization method

The method provides requirements checklists for use in requirements phase, design phase, and implementation phase, which are used in step 1. Further on, the method provides project risk profile assessment table, which can be used to determine the risk profile and risk impact to the project (step 2). Risk resolution rules (step 3) are introduced to support decision making in intervention decisions. The process is illustrated in Figure 6.

5.6.3 Summary of the method

The requirement risk prioritization method by Tuunanen and Vartiainen (2016) proposes a requirement risk prioritization method, which is not dependent on ISD approach being applied. The method provides a requirements risk item listing, which can be used in identifying the requirement risks in a ISD project. In addition, the method describes guidelines for project risk profiling, and proposes a how to prioritize the use of requirements risks resolution techniques. The requirement risk prioritization method can be applied by ISD practitioners to improve understanding about the risks related to requirements and making decisions on the requirements specification work.

5.7 Summary

One of the risk categories in IS development projects is requirement risks, which refers to risks that are specifically related to requirements engineering process and requirements themselves. The risk related to requirements can be divided to identity, volatility, complexity and integrity risks.

The requirements related risks can be managed by applying feasible requirements management practices rigorously throughout the life-cycle of the IS. The management of requirement changes improves the project control in changing circumstances. Requirements management includes proper requirement specification and documentation techniques. Multifaceted teams perform better in requirements elicitation and analysis, but customer involvement and engagement cannot be dismissed. In requirements development, the focus should be on customer value creation, not in features of the IS. Ensuring fast feedback improves the validation of requirements.

The development on IS should be based on product vision and business requirements. Product management and roadmapping activities help in keeping the IS requirements engineering process running throughout the lifecycle of the IS. Agile development methods have many built-in features, which help in diminishing the requirements risk. However, Tuunanen and Vartiainen (2016) point out that contemporary information science research does not provide many methods for prioritizing risks, and that further emphasis should be given on how requirements risk prioritization is done in a way that it takes into account differ-

ent information systems development approaches. They have developed a requirements risk prioritization method, which can be applied to ISD projects that use different ISD methods (Tuunanen and Vartiainen, 2016). The method presents steps for identifying, assessing and eventually intervening the requirements risks in different phases of the ISD project. The applicability of the method, however, has not been tested in real life ISD project setup, which leaves practical applicability of the method a question.

6 Empirical study: requirement risks prioritization method testing

6.1 Objective and methods for the empirical study

The empirical part of the study aims at testing and developing further requirement risk prioritization model described in chapter 5.6. The model is based on the work by Tuunanen and Vartiainen (2016). The empirical study will be conducted as an interpretive case study. The data is collected by making specialist interviews and observations in an IS development industry project applying agile development approach.

Case study research is the most common qualitative research method used in IS research. A case study is an empirical inquiry that investigates phenomenon within its real-life context. The case study approach is well-suited to IS research, because the boundaries between phenomenon itself and its context are not evident, and the focus of research is more on the social context of the phenomenon than in the technical issues (Klein et al. 1999).

There are different approaches to case studies: positivist, interpretive, or critical (Klein et al. 1999). In this thesis work, we take the interpretive approach. Interpretive approach is well-suited, if the studied phenomenon is such that the knowledge is gained through social constructions such as language, shared meanings, documents, tools, and other artifacts. Interpretive research attempts to understand phenomena through the meanings that people assign to them (Klein et al. 1999). This applies to the requirements and risk management in the context of the study – risks in practical ISD are most often identified, estimated, prioritized and managed mostly by specialist evaluations and reviews, since they are not often precisely measurable or modelable.

6.2 The case study implementation

The interpretive case study approach is used in the empirical study of the method. The data is collected by making specialist interviews and observations in an IS development industry project. The goal is to test applicability and usefulness of the requirement risks analysis and prioritization method is tested in case project.

6.2.1 Case project description

The case project customer a is large Finnish business operating on transportation industry. The customer company also manages large infrastructure and construction projects and other development projects. The company has about 8 600 employees (2016), and the turnover being about 1200 M€ (2016).

The IS in question is an investment management system, which is already in everyday use in the organization. The system is used by the customer employees for managing and reporting investments needed to implement diverse types of development projects in the organization.

The provider of the system is a small Finnish software development company – the company employs 18 persons (2017) and its turnover is 2 M€ (2016). The system has been originally implemented by the provider, and the case project is a further development project aiming at adding new features to the system.

In the case project, a bundle of new features will be developed into the IS. The development project is in the requirements elicitation and development phase, when the interviews are held. The applied ISD approach is incremental approach: the new features are specified, implemented and installed in releases, each containing a predefined set of features. A new release of this type is implemented in the case project, i.e. it is so-called 'release project'. In the project model, the requirements are specified and developed into use cases in the specification phase in the beginning of the project. The requirements are expressed in form of use cases.

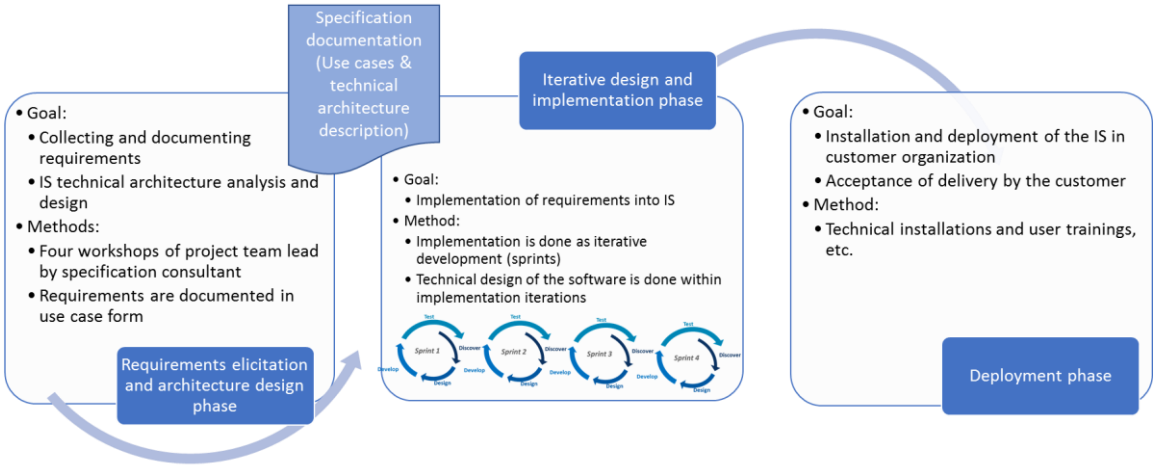

Figure 7: Overview of the case project phases.

Amount of the expected implementation work in the project corresponds to several development iterations, although the work amount and eventually the required number of iterations is not known before requirements analysis and design is done. However, the implementation work extends over multiple iterations (or sprints) and most likely require more than one delivery and deployment into use.

The project team in the specification phase consists of customer-side project manager and domain area specialists, and service provider's specification consultant and software architect. Customer-side project team consists of domain area specialists (business controllers), who present the key user groups and the project manager. All of them are users of the IS. In addition, the project's business owner (at the customer side) is part of the project team.

The service provider leads the specification work. The service provider team consists of leading consultant (business analyst), systems architect and project manager, although only the project manager is not actively involved in the specification work.

6.2.2 Testing the method in case project

The requirement risk analysis and prioritization method is tested in the requirement specification and analysis phase of the case project. The testing process consists of the following phases:

1. Observation in single specification workshop
   Goal: to observe how the requirements and specifically their risks are handled in the specification workshop situation. The specification consultant or the workshop participants do not get prior information about the method.
2. Testing the method with specialist interviews
   Goal: evaluation of the method and the results it delivers through specialist interviews
3. Analysis and summary of the interview and observation results

The interviewees are all members of the project team either on the customer side or provider side. The information about the interviewees is given in the Table 11.

Table 11. Interviewees in the case project.

ID	Job role	Role in the project and demographic information
H1	Controller	Customer-side project manager, specialist in the financial investment management. Customer side project management and specification lead. Male, age group 45-54.
H2	Director, business control	Customer-side business owner for the project Female, age group 45-54.
H3	Controller	Customer-side specialist in business control and finance. Male, age group 25-34
H4	Senior consultant	Service providers specification consultant, lead consultant in requirement specification and analysis. Male, age group 34-45
H5	Project manager	Service providers project manager. Project planning and management (including risk management) Demographic information: Male, age group 34-45

Originally the goal was to interview also service provider side software architect, but due to changes in the team the interview could not be held.

6.2.3 Interview structure

The interviews were structured in a following way. The interview questions are presented in appendix 1 (in Finnish).

1. Introduction to requirements risks in general and presentation of the requirement risks analysis and prioritization method
    a. Description of the method
    b. Requirement risk categories and some examples
    c. Brief presentation of the risk tables of the method
2. Case project requirement overview and requirement selection
    a. Case project use case documentation overview
    b. Selecting the use cases (maximum of three use cases) from the case project specification material: interviewee selected the use cases, which were used for testing the method
3. Question pattern 1: Review questions about the chosen use cases to ensure that the interviewee recalls the requirement in question properly
4. Question pattern 2: Analysis of the use cases using the method
    a. Identifying the risks by applying the check lists to the chosen use case(s):
        i. Requirements phase checklist
        ii. Design phase checklist
        iii. Implementation phase checklist
    b. Risk profile assessment:
        i. Project risk profile assessment table
    c. Risk resolution pattern evaluation
        i. Risk resolution pattern in the method
    d. Risk resolution techniques review
        i. Risk resolution techniques table overview and questions
5. Question pattern 3: Evaluation of the relevance of the results
    a. Questions to evaluate the results of the risk analysis and prioritization method and applicability of the method itself
6. Question pattern 4: Applicability of the method in projects and usefulness of the results
    a. The usefulness of the results given by the method: interviewee perspective.
    b. Applicability of the method in project: interviewee and customer view
    c. Possible additions or changes to the method
7. Summary and ending the interview

6.3 Analysis of the case study interviews

The interviews were recorded and transcribed. The interviews were analyzed, and results interpreted by applying appropriate success criteria for the method. We apply the IS success model presented by DeLone et al. (2003) in the evaluation of the method. The DeLone success model suggests a categorization for IS success criteria which can be applied also in the context of analyzing the successfulness of the method. Essentially, DeLone model is a process model explaining how the organizational impact is achieved through usage of a system (or the requirements analysis and prioritization method in this case study). DeLone's model points out that system and information quality attributes (i.e., the quality of the system and the quality of information handled in the system) must be on sufficient level before the system will get utilized and user satisfaction achieved. Further on, usage of a system is a prerequisite for achieving impact on individual user level and eventually on organizational level. Figure 8 illustrates the DeLone's IS success model.

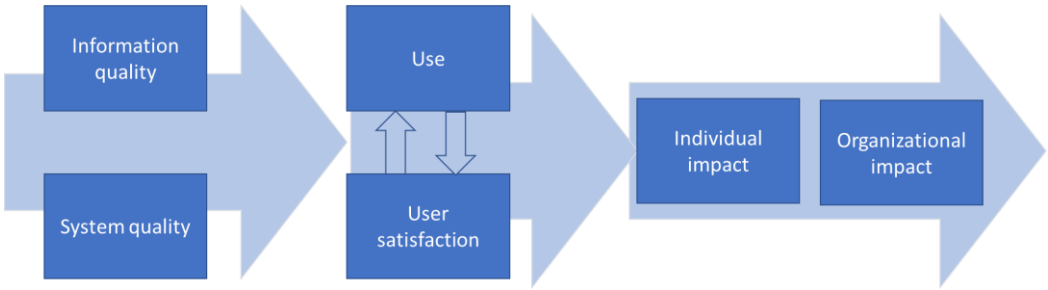

Figure 8: IS success model.

Borrowing from the DeLone's model, the following evaluation criteria for evaluating the requirement risk analysis and prioritization method is applied:

1. System (= requirement risk analysis and prioritization method) quality attributes: *The method must be applicable in practical project work*
   - Efficiency of the method: effort vs. results
   - Flexibility: applicability with different ISD approaches, modifiability of the model (such as adding new risks, new risk categories, new risk management methods)
2. Information quality measures: *The method must produce useful information about the requirements risks:*
   - Accuracy of the results
   - Completeness/adequacy of the results
   - Relevance of the results
3. User satisfaction: *The user of the method should be satisfied with the results:*
   - Decision satisfaction

4. Individual impact: *The method should provide support for (project) decision-making of project personnel:*
    - Decision effectiveness (correctness of decisions, confidence to decision-making)
5. Organizational impact: *The method should improve project decision-making and eventually IS success.*
    - Improved outputs (of requirements specification process)
    - Improved decision-making (in requirements specification process)
    - Overall project success (due to requirements risk control)

In the interview process, the aim is at finding evidence for the success factors set for the method. From the criteria above, the observation and interview process can mostly give input to the criteria 1-4. The criterion 5 is difficult to measure in this case study setting, because the interviewee cannot possess factual information about the organizational level influence at the specification phase of the project. The goal of the interviews is to study how the interviewees find the requirement risk prioritization method and the process of applying it in the case project: do they find it valuable, useful and therefore applicable in the context of the case project and in general.

6.3.1 Analysis of the answers to questions

In the following subsections, an analysis of the interview results is given. The analysis is divided to sections each presenting comments and analysis regarding the success factors presented in chapter 6.3. The citations from the interviews are presented in Finnish, and translations to English are given (after the "EN:" marking).

Remarks on applying the risk tables

The interviewees were presented the tables before applying them on requirements of a real-life project. Each risk item on the table was explained in the interview process, if the interviewee was not able to interpret it directly from the title.

One of the key questions for the applicability of the method is did the interviewees understand each risk item, and how did they interpret them. Based on the analysis of the answers from the interviews, it appears that most of the risk items were understandable as such for the interviewees. The interviewees could correctly interpret most of the risk items directly from the risk description (risk item title), and furthermore connect them to the case project context. However, some of the risk items were not self-explanatory, and required explanation from the interviewer. The meaning of the following risks required more specific clarification in the interviews:

- Unrated requirements – the meaning was not understandable
- Client Commitment – required clarification

- Constrained Users' Knowledge – easily misunderstood and interpreted as business analyst's lack of knowledge
- Delivering What the Client Requires – the meaning of the risk was not evident

It was also noteworthy, that some of the risk items got a bit different interpretation on the client side and on the service provider side. These were:

- Fixed Budget and Timelines - whose budget and timeline?
- Delivering What the Client Requires – who is 'the client'?

Some of the risk items on the risk tables also got several different interpretations depending on the interviewee:

- Underestimation of Change Magnitude – does this refer to technical change, cultural change, process change or combination of them?

The interviewees had also some difficulties to see the difference between certain risk items in real-life project context. These risk items were:

- Conflicting Requirements vs. Emerging Requirements Dependency – in theory the difference is apparent, but in the case project the difference was difficult to see.
- Knowledge Gap between Coworkers vs. Lack of Collaboration
- Constrained Users' Knowledge vs Knowledge Gap between Coworkers

Evaluating the risk tables and the resultant risk profile

Interview question: do you find the risk profile for the requirements meaningful?

All interviewees found the risk tables meaningful when evaluating the risks associated with the requirements. In addition to that, three of the interviewees (all from the customer side) pointed out that the risks brought up by this kind of process were real risks also in the case project. Both interviewees from the service provider side questioned the coverage of the tables, i.e. are all relevant risks covered with the tables.

All interviewees said that the risk profile for the reviewed requirement was descriptive to the real-life situation in the case project. This mostly points out that the resulting risk profile is what they anticipated, and therefore the answer may be somewhat predictable. The interviewer searched for comments about the applicability of the risk items and discussion around the topic. Two of the interviewees - both on the service provider side – were concerned whether every necessary risk area is covered with the requirements tables.

H1: Kyllä se kuulosti loogiselta ja järkevältä. Että ainakin se tukee sitä omaan käsitystä siitä, että mitkä niitä oli niitä vaikeita asioita siinä työpajassa. [EN: It did sound logical and reasonable. At least it supports my own impressions on which were the difficult things in the workshops]

H2: Kyllä se kuvaa sitä, joo. Molemmissa tapauksissa kuvaa tilannetta. [EN: Yes, it does describe it. In both cases it describes the situation]

H2: Siinä minun mielestä on kyse siitä, että tätä ei ole mietitty ihan kunnolla, ei ole mietitty loppuun asti, että mitä me sitten oikeastaan halutaan. Se lähti liikkeelle ihan järkevästä vaatimuksesta, mutta mitä se tarkoittaa ja mitä siitä seuraa, niin sitä ei ole ihan loppuun asti mietitty nyt. [EN: In my opinion, it is about not having properly thought it, not having thought through what we actually want. It started with a sensible requirement, but what does it mean and what incurs from it, well that has not been considered in the end.]

H4: Osin on [mielekäs]. Jäi epäilys, että onko kaikki riskit katettu. [EN: Partly it is sensible. A doubt remains that are all risks covered.]

H5: Vaikea monessa kohtaa arvioida asiakkaan puolella oleviin asioihin. Mutta niiltä osin kuin nyt voi arvioida, niin ihan olennaisia havaintoja riskeistä. Mutta sitten taas toimittajalle toteutusriski, tai tekninen riski, on iso ja sitä ei kyllä ole paljoa eroteltu tuossa. [EN: In many items it is difficult to evaluate the things that are on the customer side. But from those parts that can be evaluated, they are essential observations on the risks. But then again, service providers implementation risk, or technical risk, is big and that has not been sorted out there.]

It is important to notice that evaluating the risk categories for the project with the method in the interviews was restricted to individual risks. Because the interviews were time-limited, the arrangement did not allow analyzing every requirement (use case) of the case project. Therefore, the risk profile for the entire project could not be created, which also limits the reliability of the interview results.

It is also noteworthy that in one-hour interview only from one to three use cases could be analyzed with the risk lists. One can anticipate that for a project with large number of requirements, the risk prioritization method is quite heavy to implement thoroughly. On the other hand, implementing the process gets faster as the experience on the method accumulates. *However, the interviews suggest that the method is fairly time consuming to apply in real life project arrangement*. This was also brought up by one of the interviewees:

H3: Mutta tuota… Tämähän on peijakkaan raskas tämä menettely. [EN: Well… but my gosh, this is heavy, this procedure.]

Risk profile and requirements engineering techniques

The requirement risk prioritization method proposes that, based on the risk profile, different measures to handle the high risks should be considered depending on which risk types are on elevated level (identity risks, complexity risks, identity

risk, or volatility risks). The method includes a listing of requirements engineering techniques, which are categorized to specification, experimentation, discovery, and prioritization methods (see the appendix 3 for the list). The method proposes to emphasize discovery techniques for high identity risks, prioritization techniques for high integrity risks, experimentation techniques for high volatility risks are high, and specification techniques if complexity risks are high.

Interview question: based on you own experience, what kind of actions would you apply to the elevated requirement risks?

As explained earlier, due to the time-constrained interviews, the project level risk profile for the entire case project could not be assessed, which limits the testing of the risk resolution rules, as the project risk profile was not available in the interviews. However, the logic for selecting requirements engineering technique to resolve residual requirement risks was explained and discussed in the interviews. Particularly, the usefulness of the requirement engineering technique selection step of the method resulting technique proposals were discussed. *Remarkably, all interviewees commented the usefulness of this step of the method negatively: the value of the techniques selection was not seen by the interviewees.*

> H2: En osaa sanoa mikä menetelmä, mutta mun mielestä tässä käy selville se, että meidän pitäisi ensin se meidän oma case vielä sisäisesti kertaisesti keskustella. Me lähdettiin sitä nyt tuomaan ilman että me oltiin kunnolla itse valmisteltu sitä, että sehän siitä uupuu selvästi. Ja meidän pitäisi se vielä kerran sisäisesti keskustella, että mitä me halutaan. [EN: I cannot tell which techniques, but in my opinion this reveals that we should first internally discuss our own case through. We began to bring it now without having properly prepared it – that is what is missing clearly. And we should once more discuss internally what do we want.]

> H5: Teknistä suunnittelua pitäisi viedä pidemmälle. Moni asia on normaalin projektin riskienhallinnan piirissä muutenkin, niin kuin vaikka resurssiriskit. Asiakkaan puolen edustajuuteen liittyviä riskejä on paha arvioida, eihän siihen ole näkyvyyttä. [EN: Technical design should be advanced further. Many things are under normal project risk management anyways, for example resource risks. Customer-side representation issues are hard to evaluate, we have no visibility in there.]

Interview question: Risk resolution techniques list: is there techniques that you are familiar with?

When asked, the interviewees did not know or recognize majority of the techniques on the requirement engineering list. This was as expected. In the interviews, few exemplary requirements engineering techniques on the list were very briefly introduced.

> H5: No on tuossa tuttujakin menetelmiä joukossa, mutta ei kaikki ole. Käytännöllinen tuommoinen listaus ei ole. [EN: Well, there are familiar techniques in the group, but not all. That kind of listing is not practical.]

Interview question: considering the case project phase and situation in general, do you find it feasible to analyze the requirements using new technique or method?

Interviewees did not find the list of techniques very easily utilizable in the context of case project. This may be partly because the long listing of mostly unfamiliar techniques was considered somewhat overwhelming. The interviewees did not find it realistic that in the end of the specification phase after several workshops and use case documentation a completely new approach to requirements gathering and analysis would be applied. Instead, all the interviewees on the customer side would have focused on clarifying individual requirements having high residual risk level by analyzing the requirement with methods familiar from the earlier phases of the case project. Further on, interviewees on the service provider side brought up the possibility to use new techniques to clarify individual high-risk requirements, such as usability testing and prototyping. However, also they would have kept the focus in individual problematic requirements.

> H3: Mikä siinä oli se yksi, jossa oli näiden tuota... [näyttää materiaalista yksittäisen vaatimuskohdan]. Tämä. Tästähän meillä ei sitä konkretiaa ole näkyvillä, että tietysti tämmöisen tapauksen osalta voisi vielä piirtää sen mikä se lopputulema on. Nythän meillä on se ilmassa piirretty ja leijuva se, että jotakin tämmöistä meillä on tulossa. [EN: There was this one with the… *[points out individual requirement from the material]*. This one. We do not have concreteness for this one, so for this kind of case, the outcome could be illustrated. Now what we have is like drawn in the air and hovering, tha something like this we are going to have.]

> H4: Riippuu tilanteesta. Palataan suunnittelupöydän ääreen määrittelemään tai tutkitaan tarkemmin teknistä toteutusta. Ei tunnu todennäköiseltä, että otettaisi kokonaan uusia menetelmiä käyttöön. [EN: Depends on the situation. Get back to design board to specify, or study the technical implementation more thoroughly. It does not fell probable that we would take completely new techniques in use.]

> H5: Jos se sitä tarkoittaa, niin käytännössä ei voi kokonaan uutta määrittelymenetelmää ottaa koko paketille. Kyllä yksittäiset kohonneet riskit pitää käsitellä – jos nyt on vaikka kysymysmerkkinä, että tarvitaanko jotain ominaisuutta ollenkaan niin asiakkaan puolelta sitä pitäisi loppukäyttäjäpalautteen perusteella arvioida. Vaikka käytettävyystestein tai protoilemalla voisi sitä auttaa kyllä. Mutta että onko sitten siihenkään sitten budjettia, että kuka sen maksaa. [EN: In practice, you cannot take completely new specification method into use for the entire package, if that is what this means. Individual elevated risks must be handled – if the question is like whether some feature is needed or not, then the customer should evaluate it based on the end use feedback. For instance, usability testing or prototyping one could help it. But is there any budget for that – who pays for it.]

Taken all together, the requirement techniques selection process proposed in the method was not found practically feasible in the context of the case project. To improve it, it was proposed that organization should make a short listing of techniques, and appropriate methods would be selected from that lists.

Relevance of the results

Interview question: Is the prioritized requirement risk list relevant (i.e., are the risk real)?

All interviewees commented that the residual requirements risks were relevant and real risks for the project. At this point, two of the interviewees also brought up that the risk tables are generic, and that case project specific risks may be missing. The case specific risks, which were brought up, were associated with IS application area, applied project model and technical implementation complexity. All interviewees also agreed that the residual risks were meaningful.

> H1: Kaikki riskikohdat ei ole relevantteja, mutta se ei varmaan ole tarkoituskaan. [EN: All risks are not relevant, but probably that is not the idea anyway.]

> H3: Joo, sinällään riskejä ne on. Mutta onko ne yhteismitallisia ja onko se kattava paletti… [EN: Yeah, they are risks as such. But are they commensurate and is this a comprehensive package…]

> H4: Kyllä ne on ihan tunnistettavia, mutta herää kysymys että kattaako kaikkea. Sovellusaluekohtaiset riskit puuttuvat. [EN: Yes, they are quite identifiable, but there is a question whether it covers everything. Application area specific risks are missing.]

Even though the interviewees did not see the requirements engineering technique selection step very useful, they still found the risk tables and the risk items valuable.

Interview question: Have the risks been identified before? Did the risk tables bring up requirements risks, which have not been recognized before?

The interviewees pointed out that risks that remained high had been recognized in the requirement specification process, but they had not been handled as risks, but rather as difficulties in requirement specification process. On the other hand, all interviewees stated that requirement risks had not been handled in a systematic way before, and that the method brought a new way to organize this work. None of the interviewees evaluated that the method would have introduced completely new risks. Instead, it brings the requirements aspect to the project risk management.

> H2: No ei järjestelmällisesti. Ehkä jollain tasolla on tunnistettu, mutta ei järjestelmällisesti. [EN: Well, not systematically. Maybe on some level they have been identified, but not systematically.]

> H3: Voi olla, että tuo [uusia vaatimuksiin liittyviä riskejä]. Koska tässä on hyvää se, että on valmiita ajateltavia asioita. Kun meidän lähestyminen on ollut, että kun tehdään riskitaulua, niin lähdetään tyhjältä pöydältä miettimään. Ja aina se [riski] on sitten se resurssien riittävyys ja niin edelleen. [EN: Maybe it does *[bring up new risks associated with requirements]*. Because it is good that there are prepared risks to consider. Our approach has been, when making a risk table, that we start to think about the risks

from the scratch. And then the risks will always be the sufficiency of resources and so forward.]

H4: Suurin osa on [havaittu aiemmin], mutta oli hyviä havaintoja ja kysymyksiä riskilistoissa. [EN: Most of them have been *[identified before]*, but there were good observations and questions on the risk lists.]

H5: Pääosin kyllä on [tunnistettu]. [EN: Mostly, yes they have *[been identified before]*].

All in all, one can say that the issues brought up by the method were not completely new, but approaching them from the risk management perspective was considered new and useful approach. *Thus, the method was found useful in bringing structure to analyzing the risks related to project requirements and that was seen valuable by all interviewees*.

Interview question: What other requirements risks have appeared before during the specification? What about during the earlier development phases (projects) of this same solution?
When asked, the interviewees also brought up few new risk items that they think were missing:

- Practical difficulties in reviewing and the iterations in implementation phase of the project
- Emerging new requirements and new interpretations to existing requirements to leading to scope creep
- Development method/process and application area specific risks should be added
- Resource risk in project team (changes, availability, competence, attitude)
- Communication gap or communication difficulties between business analyst and specification project team, medium impact and valid in all phases

The interviewees pointed out that even though these risks would be covered by items in the risk tables, they are relevant and deserve own risk item.

Applicability of the method and usefulness of the results

Interview question: Does the requirement risk prioritization method produce information, which is useful in project decision-making?

The interviewees were asked whether the requirement risk prioritization method produces information, which would be useful in project decision-making. The interviewees were on the positive side, although a little bit hesitant.

H1: Ehkä. [EN: Maybe.]

H2: Ehkä. Ehkä tätä kautta tulee mietittyä niitä. Että miksei. [EN: Maybe. Maybe this way one really thinks about them. So, why not.]

H3: Mä en ole varma. Se panos-tuotossuhde… Periaatteessa tervetullut juuri sen ajatuksen laajentamisessa ja räjäyttämisessä, mutta sitten on se työllistävä vaikutus. Niin onko se panos-hyöty positiivinen vai negatiivinen, niin sitä en tiedä. Se pitäisi vielä testata ja katsoa. [EN: I am not sure. The input-output relationship… In principle, this is welcome in extending and exploding thinking, but then again it is laborious. IS the input-output relationship positive or negative, I do not know. That should be tested still.]

H4: Ehkä. [EN: Maybe.]

Interview question: Does the method help in construing project risk factors differently?

All interviewees said that they had not approached the requirements from the risk management perspective and that *the method brings a new way to analyze requirements and consequently also project success factors*.

H1: Tästä kulmasta ei ole aiemmin lähestytty. [EN: We have not approached the issue from this angle before.]

H2: No varmasti eri tavalla. [EN: Well, differently for sure]

H3: No en tiedä jäsentämään, mutta laajentaa ajattelua useammalle suunnalle. Tai laaja-alaisemmin. [EN: Well, I don't know about construing, but it expands thinking to multiple directions, or pervasively.]

H4: Eri tavalla kyllä. [EN: Different way, yes.]

H5: Kyllä – suoraan vaatimuksiin liittyviä riskejä ei aiemmin ole käsitelty. [EN: Yes, it does – specifically the risks associated with requirements have not been handled earlier.]

Interview question: Can you imagine utilizing the method when analyzing the requirements in new projects? If not, why?

H1: Ehkä. [EN: Maybe.]

H2: Joo, miksei. [EN: Yes, why not.]

H3: Mä voisin ajatella, ja ihan juuri sen takia, että olen huolestunut siitä, että lopputulokset on huolestuttavia useinkin. Epäonnistutaan turhankin usein, että kaikki se, millä pystytään parantamaan sitä, on hyödyksi. Ja miksi ei: se on aikamoinen punnerrus kuitenkin. [EN: I could imagine, and only because I am worried about the end results, they are often unsettling. Failures happen unnecessarily often, so everything that allows improvement is welcome. And why not? It is quite a big effort anyways.]

H4: Mahdollisesti. Vaikuttaa aika työläältä menetelmältä, joten voi olla, että ei ole aikaa käytännössä käyttää tällaista. Voisi myös asiakkaan kanssa pohtia tämän kanssa. [EN: Possibly. This seems to be fairly laborious method, so it may be that in practice there is no time for this. This could be also pondered together with the customer.]

H5: Jos tällainen analyysi liitettäisi osaksi vaatimusmäärittelyä ja tuotaisi projektin riskisuunnitelmaan, niin kyllä se varmasti kehittäisi positiiviseen suuntaan. [EN: If this kind of analysis would be added to requirement specification and brought to project risk plan, it would probably have positive impact.]

Interview question: In which phase of the project you would apply the method?

The interviewees were asked in which phase of the project they would apply the method. *All interviewees found that proper time for risk analysis and prioritization would be the specification phase at the early stage of the project*. Also, two interviewees pointed out that the method could be applied several times in different phases of the project (as proposed by the method itself).

H1: Menetelmän tekisin alkuun, että mitä me halutaan, ja sitten katsoisin, että miten ne [riskit] toteutuvat. [EN: I would implement this method in the beginning, once we know what we want. And then check how they [the risks] realize.]

H2: Tässä on tietysti viimeinen kohta implementation, niin sitähän me ei nyt vielä tiedetä. Että pitäisikö tää periaatteessa käydä useampaan kertaan. [EN: Obviously there us the last phase ”implementation”, which we do not know now. So maybe this should be done multiple times.]

H3: Siis alkuvaiheessa. Ehkä juuri tarvemääritysvaiheen jälkeen. Vai onko ennen koko tarvemääritystä, onko meillä edes semmoista tilannetta, että tätä voisi käydä läpi? Mutta jos meillä tämä olisi pohjana koko ajattelussa, niin voitaisiinko tehdä parempia [vaatimuksia]. [EN: Well, in the beginning. Maybe right after the requirements specification phase. Or do we even have a situation to go through this before the requirements specification? But if we used this as a basis for thinking, could we make better requirements?]

H4: Projektin määrittelyaiheen lopussa, kun dokumentoidaan niitä. Mutta ketterässä määrittelyssä missä oikea vaihe, kun jatkuvana prosessinahan tämä vaatisi paljon aikaa. [EN: In the end of the specification phase of the project. But what would be a right phase in agile specification, because this would require lots of time if it was done as continuous process.]

H5: Aika työläältä vaikuttaa. Mutta jos tästä tekisi check-list -tyyppisen version ja liittäisi vaatimusmäärittelyvaiheeseen, niin ehkä se toimisi siinä. Että aina kun vaatimusta kuvataan, niin mietittäisi nämä asiat [riskit] listalta. Olipa se mikä vaan se projektimalli. [EN: It looks fairly laborious. But making a check list version of this and adding it to requirement specification phase might work. When a requirement is being written down, these [risks] from the check list would be considered. Whatever is the project model.]

Interview question: are you satisfied with the information that the method produces? Does it help you in project decision-making?

The interviewees commented positively the output of the risk analysis with the method. It seems the they did not have very many expectations for the method. All interviewees commented that the biggest value from the method is that it forces to reconsider the requirements from the risk management perspective, and gives a framework for doing the risk analysis. The interviewees did not bring up that the techniques selection logic would directly help in project decision-making.

> H1: No varmaan ainakin pidemmässä juoksussa. […] Että varmaan pystytään parantamaan toimintaa, jos me pystytään määrittelemään paremmin niitä riskejä. [EN: Well, perhaps at least on the long run […] Perhaps operations could be improved, if we can define the risks better.]

> H2: Tämä tuo ehkä vähän uuden näkökulman. Ikään kuin pakottaa miettimään sitä casea. [EN: This may bring a new perspective. As it forces to reconsider the case.]

> H3: No olen siinä mielessä tyytyväinen, että… Jos minä olisin etukäteen ajatellut, että mikä tämän yksittäisen pienen määrityksen riski on, niin helposti sanoo, että siinä ei ole mitään riskiä. Mutta sitten kun sitä pohtii, niin kyllä sitä kuitenkin jotakin kohtia löytyy. [EN: Well, I am satisfied with that… if I would have beforehand thought about the risks in this one small piece of specification, I would have said there is no risk. But when you start to think about it, you still can find something.]

> H4: Ei ollut odotuksia, mutta ihan järkevältä tuo vaikuttaa. Mutta vaatii kyllä tilannekohtaista soveltamista. Menetelmää pitäisi höystää lisäämällä yrityksen menetelmään ja prosessiin liittyviä vaatimuksia. [EN: I had no expectations, but it seems sensible to me. But it demands case-specific adaptation. The method should be seasoned with the business and process specific additions.]

> H5: Riskien check listat olivat hyviä, mutta se riskiprofiilin ja menetelmän valinnan hyödyllisyys jäi minulle auki. [EN: Risk check lists were good, but usefulness of the risk profile and technique selection remained unanswered to me.]

6.3.2 Other comments and remarks from the interviews

This chapter provides a summary of comments and remarks about the method from the interviews.

New risk items proposed by the interviewees

The interviewees were asked to bring up risk items that they did not find from the list of the original method. According to one of the interviewees, *resource risk* is not sufficiently present on the risk lists. The risk list includes a risk item about project team member turnover, but other risks caused by other human factors in requirements acquisition and specification process may also be relevant. Deep understanding about the IS requirements is easily personified to one or two persons in the project team, such as business analyst, customer project manager or

software architect, which makes these persons crucial for the project success. Consequently, *human performance factors* (key persons' motivation, analysis skills, communication skills and attitude) and *human resource availability factors* (key persons' ability to contribute enough time and effort when required by the project) issues are necessary to consider. However, it is noteworthy that project human resource risks are often covered by project risk analysis, not at the requirements analysis.

Changing understanding about the requirements from the customer perspective was pointed out as a risk in the interviews. This is closely related to volatility risk in general, and may not therefore need an additional items on the risk identification lists. In addition, changing requirements and emerging new requirements during the project pose a risk about the *scope creep*.

One of the interviewees proposed adding a risk item "*Communication gap or communication difficulties between business analyst and specification project team*" to the risk tables. The interviewee argued that this risk is present in all phases of the project. The risk refers to difficulties within the project team in understanding each other. In the case project, the business analyst (service provider team member) acquires the requirements from the project team and other stakeholders. This is fairly common arrangement in similar business solution development projects in general. The risk actualizes in form of communication difficulties, when different parties in the project have different background, varying domain understanding, and use different vocabulary.

One of the interviewees argued that *budget and timeline risk* is worth evaluating for every requirement and in all phases.

Who should do the requirements risk analysis?

In the interviews, it became apparent that it there were no such person or position in the case project, who could assess every risk item presented in the risk tables. This was also pointed out by one of the interviewees: in the case project scenario, there is no single person who can evaluate every risk item, because many of them require insight from very different aspects of the project. For instance, in the case project, the customer's project personnel had no means to reliably assess risks related to technical implementation done by the provider team, because of missing technical competence and not having enough visibility to implementation progress. On the other hand, service provider's project staff was not able to completely and reliably evaluate for example risks related to customer's business strategy or sufficient end user representation in the customer's team. This raises a question how requirements risk analysis and prioritization should be conducted and by whom? For instance, would someone from outside of the project team be more objective? According to the experience from the interviews, it seems that *the requirement risk analysis should be a joint effort of project team and project business management*. It also seems that in the case project *the customer side project manager and/or business owner would benefit most from the method.*

Organization and domain specific adaptation on the method
Two interviewees pointed out that *organization and context specific adaptation of the requirements risk prioritization method* should be developed to make the method more useful in the case project. The two interviewees were concerned that the risk tables do not cover everything necessary, such as system architecture or software platform specific risks.

6.4 Summary of the case study interview results

In the following sections the interview results are summarized per success factor set in the chapter 6.1. Figure 9 illustrates the interview results and their input to the evaluation of the method following the success model.

The interviewees also gave improvement suggestions and made other remarks regarding the requirement risk analysis and prioritization method and its application. These are collected into the summary in the Table 12.

Method quality attributes
In the case study, we sought for evidence for the applicability of the method be applicable in practical project work. This means applicability with different ISD approaches, modifiability of the model (such as adding new risks, new risk categories, new risk management methods), and efficiency of the method. The key findings from the case study were:

- The requirement prioritization method was seen applicable in the case project arrangement
- The method appears as rather heavy to apply, which limits its applicability in a typical IS development project
- Selecting new requirements analysis methods, as proposed by the method, was not see feasible.
- The framework could be added to project toolbox for project risk management or requirements management purposes

In addition to the points above, we noticed challenges for the interviewees to understand certain risk items correctly. The meaning of the several risk items was not directly evident for the interviewees, they were interpreted differently by different persons, or were difficult to separate from one another. Any difficulty or ambiguity in risk items lessens the applicability of the method, as it increases the risk of misinterpretations and increases work load due to extra time required for clarifying the problematic risk items.

Risks that were difficult to understand:

- Unrated requirements – the meaning was not understandable without clarification

- Client Commitment – required clarification
- Constrained Users' Knowledge – misunderstood easily business analyst's lack of knowledge
- Delivering What the Client Requires – the meaning was not evident

Different interpretation on the client side and on the service provider side:

- Fixed Budget and Timelines - whose budget and timeline?
- Delivering What the Client Requires – who is 'the client'?

Risk items with different interpretations depending on the person:

- Underestimation of Change Magnitude – does this refer to technical change, cultural change, process change or all of them?

Information quality measures
To produce high quality information for project decision-makers, the method must produce useful information about the requirements risks. Regarding this quality attribute, the key findings from the case study were:

- All interviewees found the requirements risk analysis based on the risk tables meaningful when evaluating the risks tied to the requirements
- The risk tables provide useful framework for reviewing the requirements for risks that they may pose to the project

User satisfaction
The user of the method should be satisfied with the results, meaning that the users of the method consider the output as complete, adequate and relevant.

- All interviewees found the checklist approach useful, and the interviewees found the results useful
- However, none of the interviewees found the requirements analysis selection phase applicable and they were not able to apply it
- The method was found useful in bringing structure to analyzing the risks related to project requirements and that was seen valuable by all interviewees.

Individual impact
The method should provide support for (project) decision-making of project personnel Decision effectiveness (correctness of decisions, confidence to de-cision-making)

- The method brings a new way to analyze requirements and consequently also project success factors
- The interviewees commented positively the output of the risk analysis with the method

Organizational impact

The method should improve project decision-making and eventually IS success. The interview answers did not provide information to reliably evaluate the impact of the method in the project outcome, because of the project was in its early phase. However, interviewees did comment positively about the potential of the method for improving the risk management in their projects: the method provides a framework for requirement risks analysis, which usually is not thoroughly implemented in projects.

Table 12. Summary of remarks on the method.

Interpretation of the risk items	**Risk analysis - who should implement it?**
Several risk items were difficult to understand, and removing ambiguity requires additional effort in the process. Some risk items got different interpretation on the client side and on the service provider side. Several risk items got different interpretations depending on the interviewee.	There was no such person (or position) in the case project, who could assess every risk item presented in the risk tables. The requirement risk analysis method should be implemented as joint effort of project team and project business management. It appears that the customer side project manager would benefit most from the method.
Adaptation to context	**Proposed new risk items**
The interviews brought up a concern whether the risk tables cover everything necessary in the context of the IS and the project. Organization and context specific adaptation of the requirements risk prioritization method may be necessary. Instead of using the requirement engineering techniques selection process proposed in the method, the organization could create a shortlist of practically available techniques, and make selection from the listing.	Risks associated with human performance factors and human resource availability factors. Changing understanding about the requirements. Communication gap or communication difficulties between business analyst and specification project team. Adding the budget and timeline risk to all project phases and all requirements.

Evaluation criterion	Method quality attributes	User satisfaction attributes	Individual impact attributes
Description of the criterion	The method must be applicable in practical project work: efficiency, flexibility, modifiability	The user of the method should be satisfied with the results: decision satisfaction	The method should provide support for the project personnel in decision-making: decision effectiveness (correctness of decisions, confidence in decision-making)
Interview results summary per criterion	The requirement prioritization method was seen applicable in the case project arrangement. The method was considered rather laboursome to apply, which limits its applicability in a typical IS development project with limited time and resources. Selecting and applying new requirements analysis techniques based on the risk profile, as proposed by the method, was not found feasible. Most of the risk items in the risk tables were understandable as such for the interviewees. Few risk items were not directly evident for the interviewees, they were interpreted differently by different persons, or they were difficult to separate from one another.	All interviewees found the risk checklist approach useful, and the interviewees also found the results of the checklist analysis useful. None of the interviewees found the requirements analysis techniques selection phase applicable and they were not able to apply it in the case project. The method was found useful in bringing structure to analyzing the risks associated with project requirements and that was seen valuable by all interviewees.	The method brings a new way to analyze requirements and consequently also project success factors. The interviewees commented positively the output of the risk analysis with the method. The interviews did not give direct input for evauating the method's impact on the correctness of decisions. The requirement risk analysis list could be added to project toolbox for project risk management or requirements management purposes.
Evaluation criterion	**Information quality measures**		**Organizational impact attributes**
Description of the criterion	The method must produce useful information about the requirements risks: accuracy of the results, completeness/adequacy of the results, and relevance of the results.		The method should improve project decision-making and IS success: improved output, improved decision-making, and improved overall project success.
Interview results summary per criterion	The requirements risk analysis based on the risk tables were found meaningful for evaluating the risks associated with requirements. The risk tables provide a useful framework for reviewing the requirements for risks that they may pose to the project.		The interview answers did not provide information to reliable evaluate the impact of the method, because of the project was in its early phase. The interviewees commented positively about the potential of the method

Figure 9: Interview results summary per success criterion.

7 Discussion

The literature review of this thesis has focused on IS requirements and their characteristics, requirements development techniques and models, and agile development and continuous delivery principles. The review also introduced the concept of requirement risk, and how requirement risks are approached in agile development and in IS development approaches in general. Further on, requirements risk analysis and prioritization method by Tuunanen & Vartiainen (2016) has been represented as a tool for improving requirement risk management in the context of contemporary IS development. In this thesis, the method was tested in a case project to gain understanding about applicability and value of the method. In this chapter, we will describe how the research questions are addressed by this thesis and provide both practical and theoretical implications that the results have.

7.1 General Discussion

In this thesis, the goal is to study how requirement risks and their management are considered in contemporary information system design approaches. Requirements risks refer to IS development project risks that are specifically associated with requirements themselves or requirements engineering and management activities. The primary research question of this thesis was set as follows: *How the requirements risk management and prioritization is approached in the context of contemporary ISD methods, such as agile or continuous delivery?* The primary question has the following sub-questions: *(1) How the contemporary ISD methods, such as agile and continuous delivery, approach requirements acquisition and management* and *(2) How requirement risks are handled in these methods?* In the following sections, the contribution to answering the research questions is presented. First, we provide the contribution to sub-questions, which build ground for the answer to main research question.

Research sub-question 1: How the contemporary ISD methods, such as agile and continuous delivery, approach requirements acquisition and management?

IS requirements can be categorized by applying different models and expressed in many different formats (Wiegers 2013, Pohl 2010). The traditional practitioners' approach to requirements model is to have three requirement categories: business requirements, functional requirements and non-functional requirements. Even though this categorization still has its place in practical IS development, more user-centric techniques are often called for to express the desired IS functionality. Contemporary ISD approaches propose utilizing user-centric requirements models: in agile development requirements are often expressed as user stories or use cases (Leffingwell 2017, Measey 2015). For more complex agile scenarios, the SAFe requirements model (Leffingwell 2017) may be better applicable. Where most requirement models may come short, is when expressing quality-related attributes, experiential user needs (Patricio et al. 2009, Tuunanen & Govindji 2016), or cultural aspects (Tuunanen & Kuo 2016) of requirements. The research literature provides means for analyzing the before-mentioned aspects of IS requirements.

In this thesis we do not go into details of various requirements acquisition and development methods and techniques (which are numerous) but summarize the principles behind requirement engineering and management practices in contemporary ISD. The concurrent ISD approaches emphasize user-centric thinking, stakeholder collaboration and the importance of understanding the customer value creation. In general, the concord is that by applying user-centric requirements acquisition and development approaches, and by focusing on customer value creation, correctness of the IS requirements can be improved. Kauppinen et al. (2009) point out that IS developers should develop their customer process understanding by doing proper customer segmentation, creating direct contacts between IS developers and customers, and collecting customer information actively. Also, Komssi et al. (2015) report similar outcomes on their study on value creation in IS-intensive product roadmapping: identifying IS requirements that are valuable for the stakeholders require collaboration processes, which enable the developers to learn from the users. Importance of understanding various stakeholder groups and their interests is brought up in many sources in the literature review: Kauppinen et al. (2009), Ebert (2014), Komssi et al. (2015), Ryynänen et al. (2016), Wiegers (2013), Pohl (2010), Nemoto et al. 2015, and Tuunanen & Kuo (2015), Tuunanen & Peffers 2016. The requirements engineering literature, such as Wiegers & Beatty (2013), Pohl (2010) and Hooks & Farry (2001) also introduce typical techniques for collecting and prioritizing requirements.

It also is notable that value for a user is created in the certain use context. Therefore, Nemoto et al. (2015) discuss the use contexts in value creation, and describe a framework for understanding context-related attributes of user value creation, and process of utilizing the model in user persona-based requirements acquisition process. Nemoto's model could be well applied in agile ISD or IS-intensive product roadmapping context as well. Also, Kauppinen et al. (2009),

Ebert (2014), Komssi et al. (2015) and Ryynänen et al. (2016) argue that IS developers should invest more in analysis of customers' processes, prioritization based on customers' activities and basing the IS development decision-making on the customer value analysis, instead of setting the focus of analysis and prioritization on features of the IS. The process of product feature prioritization based on customer value may not be straightforward to apply in practice.

Various user-centered solution design approaches, like Life-Based Design (Leikas 2009), service blueprinting technique (e.g. Patricio et al. 2009), or Google's Design Sprint (Google 2017), have been introduced to guide the process of identifying, understanding and analyzing user requirements and designing IS-based solutions. Further on, Peffers et al. (2003) describe how critical success factors of an organization can be utilized in designing IS solutions starting from the factors that are vital for the organization competitive performance. This kind of design method should help IS developers to craft IS solutions that are valuable to its users.

Appropriate requirement acquisition, elicitation and presentation techniques depend on the purpose of the IS and the context in which it is used. The IS requirements must be acquired from the relevant sources and expressed to the IS developers in appropriate form. The requirements acquisition and analysis activities may consist of very different methods and techniques, depending on the context of the IS. When developing a mass-market IS for product-based business model the requirement acquisition methods are very different than when developing IS for business systems for professional users (so called management information systems). Therefore, selecting a requirements development method, which fits the scenario, can be crucial for the IS success. The SRAM method (Tuunanen & Peffers 2016) is represented to select and design methods for requirement acquisition for different contexts.

Especially the agile literature emphasizes stakeholder collaboration, continuous experimentation and feedback from stakeholders. Making proper user segmentation and considering the context of IS use helps in implementing collaboration and experimentation. The requirement model that is applied in IS development should be selected so that it supports expressing and analyzing potentially versatile characteristics of requirements. Further on, the IS requirements development should build on business targets, i.e. business strategy and IS's position in implementing the strategy. In the process of requirements development and management the focus on customer value creation is emphasized. The importance of allocating a multi-disciplinary IS design team to develop holistic understanding of the solution area is also emphasized in several sources. These principles are summarized in Table 5 in chapter 3.5.

Agile IS development methods are concurrently probably the most commonly applied ISD approach. The agile development refers to a range of development frameworks, which implement agile principles and values. There is no single 'standard' agile framework, but they all have slightly different approach

and purpose (for instance, XP, Kanban, Scrum). However, they all build on similar set of agile values and principles (Measey 2015). Therefore, in this thesis we have used the generic agile framework by Measey to represent 'The Agile'.

One of the most profound principles in agile is concentrating on continuous delivery of customer value by implementing valuable IS. This reflects also to agile requirements engineering, which builds on shared understanding of the customer needs and customer value production (Measey 2015, Leffingwell 2017). This kind of value-oriented software delivery ideology stems with what Kauppinen et al. (2009) and Komssi et al. (2015) describe about the need for the IS developers to focus on the customer value creation.

In agile ISD, understanding of customer needs is shared between the stakeholders, the key roles in requirements development being the product owner and development team. In agile development, product owner should focus on higher-level requirements and leave implementation details to the development team. In agile requirements specification many different techniques are applicable, but user stories and its variations are the most common technique. Requirement specifications in agile represents the customer needs, and many of the proposed requirement techniques are essentially customer centric (Measey 2015). As the agile methodology in general has evolved during the recent years to cover more complex scenarios, also requirements models have been developed. The SAFe model (Leffingwell 2017) describes a comprehensive agile requirements model, which is better suited for scaling into IS development in large organizations.

Because in rapidly evolving business environment changes are unavoidable, also the expectations towards IS solutions change may change even during the implementation of IS. In agile ISD one of the key principles is to accept the change and prepare for it, instead of attempting to prevent the changes from happening, remove the causes of change, or predict the changes. By implementing agile ISD practices, the organization prepares for changing IS requirements. This has implications on requirements development: the concept of emergent requirement design and documentation is commonly associated with agile (Measey 2015, Leffingwell 2017). In agile development, the user stories are continuously refined and prioritized throughout the development of the IS, and in practice the agile teams develop the details of user stories just in time before the entering the implementation phase of the user story. This reveals a difference in thinking to traditional approach: whereas 'a good traditional requirement' should be so clearly specified and thoroughly analyzed that ambiguity is removed, an agile developer believes more in negotiability of user stories and fast feedback to remove ambiguity while progressing in development.

The continuous prioritization of user stories (i.e. requirements) is present in all agile frameworks. The agile methods also bring the element of timeboxing to requirement prioritization: the development work is divided into time periods (sprints/iterations), and prioritization also the output that the team produces. As Measey (2015) puts it: "*the best way to think about prioritization is 'Within this specific time frame, we must have/should have/could have/won't have this feature.'*". In agile, the prioritization is therefore connected with the time available for delivery. In

addition, distinctively to agile, prioritization of the user stories is done continuously in the development process. Therefore, the priority of a user story is not considered to be a characteristic of a user story: priority is dependent not only on the other requirements, but also on available delivery time and resources.

Measey (2015) and Leffinwell (2017) also point out the importance of crafting non-functional requirements, i.e. restrictions, technical and architectural expectations. In software development, it is important to establish solid foundation for the solution implementation by architectural design, which requires that there is sufficient understanding about factors that have impact on the design, such as restrictions, constraints, or demands on reliability, usability, maintainability or performance. Leffingwell (2017) emphasizes that exploring, defining, and implementing non-functional requirements is a key skill of a successful agile team.

The agile ISD models also encourage use of multi-disciplinary IS design team to make business people and IS developers work together throughout the development. The agile encourages the co-operation to occur on daily basis and to communicating the development status and decisions openly. Introducing inclusive working practices fosters the culture of co-operation.

Agile ISD aims at delivering working software frequently and developing the solution iteratively. This approach supports collecting feedback from the stakeholders quickly, since the new versions of software are frequently available for testing or even for delivery to production use. Benefiting from this obviously assumes that the feedback collection is arranged appropriately. Collecting feedback may not always be straightforward to implement – consider for instance adding new features to mass-market software. In requirements development, this iterative development requires that the requirements (epics, user stories) are sliced so that they can be implemented within a single iteration.

Further on, the continuous development approach, which also belongs under the umbrella of agile approaches, brings new requirement engineering challenges that are specifically connected to phenomena associated with CD. As CD aims at very rapid delivery of changes to ISs, product management, release planning and management of product roadmap requires innovative approach in a fast-paced environment. The issues, as pointed out by Claps et al. (2015), Chen (2015) and Rodriquez et al. (2017) are not only technical, but the cultural and social challenges may be even more troublesome. CD requires that other parts of the organization, such as marketing, sales and business management, must run in the same cycle with the IS development. CD changes the way how the software is delivered to customer: continuously delivered product requires versionless product marketing and management. Making this happen may require changes in processes, competences and eventually organizational culture, which would require implementation of remarkable change program.

The literature review on agile methods reveals that the agile methods tend to focus on how the work of the development team is organized and directed, how the communication between the customer and the team occurs, and how the delivery of the software is paced. In addition to this, agile models encourage open

communication, continuous improvement of operations, open and efficient communication, focusing on high quality, and self-organization of development teams. But still the agile models do not actually give many tools for the agile designers for identifying, acquiring, and analyzing the customer needs, even though the interaction between the customer and the team is emphasized in agile development. It is assumed that the customer has representation in the agile development, but the question remains: from where and through which process the product owner generates the understanding about the end user value creation, set the vision for the system, and eventually create the user stories and set their priorities. Therefore, agile models as such leave space for various kinds of requirement elicitation, development and prioritization techniques.

Research sub-question 2: How requirement risks are handled in these methods?

Risks and risk management in IS development has been studied in numerous studies, for instance Wallace et al. 2004, Persson et al. 2009, Mathiassen et al. 2007, Keil et al. 1998, Chen et al. 2015, Barki et al. 1993, and Tuunanen and Vartiainen 2016. For instance, Wallace et al. (2004) present a software project risk categorization, which builds on the earlier research on the field. The categorization has six risk dimensions: team risks, organizational environment risks, requirements risks, planning and control risks, user risks, and project complexity risks. The category 'requirement risks' covers risk items, which are specifically related to requirements engineering process and requirements themselves. Wallace et al. (2004) point out changing, incorrect, unclear, inadequate, ambiguous or unusable requirements as potential risks for IS project success. The study argues, based on the cluster analysis on the earlier research on software project risks, that requirements risk, planning and control risk, and organizational risk are the most prominent risks for high risk projects.

The ISD practitioners also seem to agree on the relevance of the risks associated with requirements. For instance, Smith et al. (2014) report in Project Management Institute's Pulse of the Profession study results of the survey made to over 2000 project and program management professionals globally: "inaccurate requirements gathering" was seen as a primary cause of project failure by the respondents of the survey.

The requirements engineering literature, for instance Wiegers & Beatty (2013), Hooks & Farry (2001) and Pohl (2010), discuss the risks associated requirements and recognize their potential impact on the project success. The literature discusses characteristics of high quality requirements, and in general gives rather coherent description on the qualities of good IS requirement. As a summary: each requirement should have the following qualities: complete, correct, feasible, prioritized, necessary, comprehensible and unambiguous. Wiegers & Beatty (2013) point out the importance of requirement risks and own a chapter in their book for representing requirements risks and their impacts. The writers also propose techniques, albeit rather generic ones, for improving the quality of requirements. Increasing the requirement quality diminishes risks associated with requirements. The techniques mentioned in the literature (Wiegers & Beatty 2013, Pohl

2010, Google 2017, Measey 2015) are requirement reviews and assessments by multidisciplinary design team and user representatives, design team brainstorming and mind mapping, end user usability tests with prototypes, A/B testing, conducting proof-of-concepts to test technical feasibility, ensuring bi-directional requirement traceability, arranging focus group interviews, performing end user observations in use context, and utilizing questionnaires to stakeholders. In general, in testing of ISs also the requirement validation aspect must be incorporated: it is not sufficient to test only technical correctness, but also the fulfilment of the user need must be validated.

Agile methodology approaches the requirements risks a bit differently when compared to traditional requirements management thinking. In agile ISD, preparing for of the most requirement risks are taken as part of the ISD process – this is the case especially with regards to requirements volatility, emerging new requirements and changes in IS goals. In general, agile *prepares the development team for changes* in operating environment and user requirements, and therefore, agile teams should be better equipped to handle the changing situation. The agile models also propose defining a *vision statement* for the IS, which helps in managing the change and decision-making in prioritization.

All commonly applied agile development methods have many built-in features, which help in diminishing the requirements risk exposure of an IS project. Therefore, one can argue that by applying agile, organization is better positioned to handle changes and therefore also requirements risks. Agile methods focus on *continuous delivery of value* to the customer, *constant customer feedback*, and *iterative development approach*. These practices give the IS developers better chance to detect unnecessary, misunderstood or faulty requirements rapidly and have better control over technical risks. Agile also emphasizes *active participation of stakeholders* and encourages *adoption of inclusive* working methods. Furthermore, the requirements are being *analyzed and prioritized* in a continuous manner throughout the agile ISD process. These agile principles, which are summarized in Table 9 of the chapter 5.4, diminish the risks associated with requirement correctness, volatility and feasibility.

The plentiful literature on the IS project risks in general and the requirement risks indicate that the importance of requirements engineering and management efforts in risk handling is recognized. Exemplarily, requirement risk management is added on the project manager's risk management agenda by Wiegers & Beatty (2013): they point out that project managers should be able to identify requirements related risks, record them into project risk analysis tool, follow the evolution of the risk, and escalate the risk if considered necessary. However, the literature does not really provide the IS developers with techniques or methods for analyzing the requirement risk exposure of project or prioritizing the requirements risks. Therefore, it appears that specifically the risk management approach to requirements is not commonly applied in practice. Further on, not very much research has been done on the field of requirement risk prioritization and management in continuous ISD context, as also suggested by Tuunanen and Vartiainen (2016). For this argument, we found support from the literature review.

Research question: How the requirement risk management and prioritization is approached in the context of contemporary ISD methods, such as agile or continuous delivery?

Both IS research efforts and experiences from practical IS development projects have unveiled ISD project risks, which are associated with requirements set for IS. These so called requirements risks, as described for instance by Mathiassen et al. (2007), Tuunanen & Vartiainen (2016), and Komssi et al. (2015), result from lack of understanding of customer value production, misunderstanding the importance of customer-side culture (be it organizational culture or national culture), low quality of requirements (ambiguity, incompleteness, complexity), missing requirements, unsuccessful or missing requirements change management, misinterpreting the stakeholder groups, insufficient end user participation/representation in requirements elicitation, changes in requirements (volatility), technical complexity, or unanticipated implementation costs. Both research literature and practical experience tell that remarkable share of unsuccessful IS development projects fail because of shortcomings in requirements development and requirements management. Therefore, paying attention to the quality of requirements and the quality of requirements engineering techniques is worthwhile.

In general, the requirements engineering and requirements management literature propose that requirements risks are managed by producing high quality requirements by applying best practices in requirements elicitation, requirements engineering and requirements management rigorously throughout the life-cycle of the IS, all the way from setting the vision for the yet non-existing IS through specification and implementation phase to further development.

The requirements risks have been categorized by Mathiassen et al. (2007) and Tuunanen & Vartiainen (2016) into four categories. First, requirements identity risks refer to the availability of requirements: high risk means that the requirement acquisition is difficult, or requirements are unknown or indistinguishable. Second, requirements volatility risks refer to the stability of requirements: high risk means that requirements change easily. Third, requirements complexity risks measure how easy it is to understand requirements: high risk means that requirements are difficult to understand, specify, and communicate. And fourth, requirements integrity risks refer to completeness and accuracy of the requirements elicited from end users. Each of these risk categories contain different types of risk items, which may threaten the success of requirements development efforts. The researchers have attempted to chart typical risk items per category.

Further on, Tuunanen & Vartiainen (2016) describe a requirement risk categorization, which takes ISD project phase into account. According to the researchers, different requirement risk items are emphasized in each phase of the project (specification, design, and implementation). Therefore, the risk model proposed in the paper introduces requirement risk tables for each IS development phase, which can be used in analyzing the risk exposure, prioritizing the risks and making decisions on the requirement risk mitigation activities.

Even though there has been research activity on the requirement risks, approaching IS requirements specifically from risk management perspective is not that commonly applied in practical IS development. Risk management, in general, is a process of identifying risks, assessing them, and executing actions to reduce risk to an acceptable level. Requirements risk management means reviewing requirements for potential risks, analyzing risk exposure and making decisions on how to mitigate the risks and improve the requirements.

The requirements engineering literature also touches the topic. For instance, Wiegers & Beatty (2013) discuss requirement risks in their book, and propose techniques to improve requirement quality. Still they do not provide the reader with techniques for analyzing the requirement risk exposure or prioritizing the requirements risks. Similarly, Pohl (2010) describes requirement engineering and management techniques and quality criteria for requirements artefacts for implementing high quality requirements development and validation practices, but risk assessment approach to requirements is not covered. Also Hooks & Farry (2001) present in their book checklists for checking completeness of the requirement collections and evaluating the quality of requirements, but these techniques do not help in requirements risk prioritization either. Furthermore, the requirement risk checklists also appear narrow, when compared to the research literature on the topic.

Based on the literature review, it appears that existence of requirements risks is widely recognized, and requirement risks are also broadly studied topic. However, in IS development the requirements risks are mostly approached specifically from the requirement development perspective keeping the focus in developing high quality requirements. Paying attention to requirements quality obviously decreases risks associated with requirement. However, risk control and management approach to requirements is not conventional: requirement risks are not usually handled with risk management methods, i.e. analyzed, prioritized and mitigated as part of project risk management process.

Tuunanen and Vartiainen (2016) have proposed requirements risk analysis and prioritization method for improving the requirement risk management in ISD projects. In the empirical part of this thesis, a case study was used to test applicability of the method. In the case study, we noted that requirement risk analysis approach was found novel by the case study participants. Further on, the case study interview results suggest that the risk analysis and prioritization method was applicable in the case project, and that it provides valuable information about the requirements risks. Especially, the risk tables provide a useful framework for reviewing the requirements for risks that they may pose to the project. This indicates that even the risk tables as such could be added to project risk management toolbox and applied with different ISD approaches to bring structure to project requirements risk analysis.

7.2 Theoretical implications

In this thesis, we have studied requirement risk prioritization and management in concurrent ISD approaches based on the literature. In addition, to evaluate and develop requirement risk analysis and prioritization instrument for IS developers, the requirement risk analysis method (Tuunanen & Vartiainen 2016) was tested in a case study arrangement.

The literature review on requirement risks and their handling in modern ISD approaches show that existence of requirements risks is widely recognized, and requirement risks are also broadly studied topic. Literature on the IS project risk management propose generic project risk categorizations, and identify requirements risks as one risk category (for instance Wallace et al. 2004, Persson et al. 2009, Keil et al. 1998, Chen et al. 2015, Barki et al. 1993). Further on, many researchers (see for instance Mathiassen et al. 2007, Tuunanen & Vartiainen 2016, Wiegers & Beatty 2013) have contributed to creating categorization and analysis tools for requirements risks, and therefore models to categorize the risks associated with requirements exists.

In addition, the requirements engineering literature introduce different models for categorizing requirements (see Wiegers et al 2013, Measey 2015, Leffingwell 2017), models for expressing various types of requirements (Nemoto et al. 2015, Patricio et al. 2009, Tuunanen & Govindji 2016, Tuunanen & Kuo 2016), and also describe characteristics of high quality requirements (Measey 2015, Wiegers & Beatty 2013). Still, in concurrent IS development the requirements risks are commonly approached specifically from the requirement development perspective, keeping the focus in developing high quality requirements. The concurrent ISD approaches emphasize user-centric thinking, stakeholder collaboration and feedback, and the importance of understanding the customer value creation. The idea is that by applying user-centric requirements acquisition techniques and iterative development approaches, and by focusing on customer value creation, the correctness of the IS requirements can be improved. Applying these principles contribute positively to the quality of the requirements development. The case study also supports the conclusion of the literature review: the requirement risk analysis and prioritization is not commonly applied in requirements development. Therefore, a following implication is recorded: *based on the literature review, approaching requirements risks specifically from the risk management approach is not conventional even in concurrent ISD frameworks.*

Further on, the literature review shows that the appropriate requirement acquisition, elicitation and presentation techniques depend on the purpose of the IS and the context in which it is first developed and then eventually used (see Tuunanen & Peffers 2016, Komssi et al. 2015, Peffers et al 2003, Leikas 2009, Patricio et al. 2009, Measey 2015, Leffingwell 2017, Google 2017). The requirements acquisition and analysis activities may consist of very different methods and techniques, depending on the context and stakeholders of the IS. *This implies that also the risks associated with IS requirements have IS context and goal dependent factors.*

ISD practitioners may sometimes utilize their own checklists for reviewing quality of requirements, but research-based methods are not common. The literature on the ISD risks discusses also requirements risks, and indicates that depending on the IS context and IS development project setup different requirement risk types are emphasized. This underlines the importance of developing for example organization-specific and application area specific requirements requirement risk analysis techniques. Therefore, a following theoretical implication to the research is recorded: *Concurrent ISD approaches do not approach requirement risks from the risk management perspective, but focus on producing high quality requirements. Typically the methods lack integrated risk analysis and prioritization methods. Furthermore, context-specific adaptations of requirement risk analysis methodology were not found.*

Therefore, a further research item might be to study what kind of industry, organization, project model or technology dependent factors for requirements risks there are, and how they should they be represented in the risk item lists.

Further on, continuous development of operations is one of the cornerstones in agile methods. The continuous improvement should cover also development of requirements analysis and development practices (Measey 2015, Claps et al. 2015). This calls for methods for measuring and evaluating quality of requirements and requirement management processes. Risk management approach would be one idea to improve quality control of the requirement engineering processes. The assumption would be that if the requirements process produces high-risk requirements, the process must be adjusted. Requirement risk analysis tools could provide a tool for evaluating the risk level of new requirements, which could be then used to measure the progress.

Many of the implications of this thesis work derive from the case study about feasibility and applicability of the requirements risk prioritization method (by Tuunanen and Vartiainen, 2016). We present here critique towards the method and suggestions for the further development of the method.

Importantly, the case study suggests that the requirement risk prioritization method (by Tuunanen & Vartiainen, 2016) was considered applicable and to give valuable information about the requirements risks in the case project arrangement. Therefore a following theoretical implication is recorded: *The case study suggests that the requirement prioritization method was seen applicable in the case project arrangement. The requirements risk analysis and prioritization based on the risk tables was found meaningful for evaluating the risks associated with requirements. In addition, the risk tables provide a useful framework for reviewing the requirements for risks that they may pose for the project.*

On the other hand, all elements of the method were not found applicable by the case study participants. The usefulness of the requirement analysis technique selection based on the requirements risk profile was challenged in the interviews. The value of the selection logic was not truly perceived, and the interviewees did not find it possible to introduce completely new requirement analysis techniques into the project. However, it is noteworthy that the limitations of the case study might have impacted the evaluations given by the interviewees, and when drawing conclusions on the limitations must be considered.

The case study interviews also brought up a question whether the method covers all necessary risk areas in the context of case project. Exemplarily, IS users and stakeholders (such as user groups and other interest groups) and their representation in the project, IS application area (the area of industry where the solution will be applied), applied technologies, and system architecture are all factors, which affect the risk position. Other relevant factors may exist as well.

The case study results also show some difficulties for the participants in understanding and communicating the requirement risks. The purpose of some risk items was not directly evident for the interviewees, the risk items were interpreted differently by different persons, or were difficult to separate from one another. Therefore, applying the method requires rather good understanding of risks and IS requirements in general.

In addition to above, the method appears as rather heavy to apply, which limits its applicability in a typical IS development project. Therefore, finding a more straightforward approach might improve its applicability.

Drawing from the observations above, the idea of developing the risk prioritization and analysis method to adapt not only to ISD approach, but also to project context may be worth more thorough analysis. Organizations could also develop their own version of the method by including risk items specific to the organization and IS context, and restricting the range of requirements analysis techniques. Deriving from earlier, the following implications regarding context-specific adaptation of requirement risk analysis tool is recorded: *Context-specific requirements risk tables should be considered, i.e. studying the possibility to add organization and IS context specific risk items into risk tables. In addition, the practitioners may consider adapting the risk tables to match the context and project organization to improve comprehensibility of risk items, to add missing risk items, and to cover potential gaps in risk tables. Similarly, developing context-specific requirement risk analysis techniques should be considered: requirement analysis techniques selection phase, which restricts the range of requirements analysis techniques to only techniques applicable in the context and for the project organization.*

Testing the requirement risk prioritization method by Tuunanen and Vartiainen (2016) in the case project gave support for the presumption that analyzing the requirements risks properly requires wide knowledge both about the IS in question and the context of the IS. Evaluating risk items reliably and making meaningful risk level estimations requires thorough understanding about the IS's technological solution, users and other stakeholders of the solution, application environment, and the process of requirements acquisition and specification. This means that there may not be single person in a project team who is able to make the requirement risk analysis reliably. The case study gives support to this assessment: depending on a role in a project (for instance, customer or service provider side, project manager or specialist role), slightly different risk items were emphasized. Furthermore, it was also explicitly pointed out in the interviews that interviewees were not always able to evaluate certain risk items. Therefore, a following implication is recorded: *the requirement risk analysis should be conducted as team work by a multidisciplinary project team to improve the accuracy of the risk assessment and to divide the workload.*

A common arrangement in IS development projects is that the customer (i.e. the eventual owner of the IS under development) has outsourced the technical specification, design and implementation work to external service provider, such as software vendor or systems integrator. This was the situation also in the case project. It is important to understand that in a customer-provider project arrangement, the customer project team often has poor visibility to project risk factors associated with technical design, implementation and quality issues. This is because the technical design and implementation work is done by the provider team and information flow regarding the technical design and implementation details may be hindered. The risk prioritization method may alleviate the customer project manager's pain, if the method is applied in co-operation between the parties. Based on the case study interviews, it appears that the risk prioritization method was found especially valuable by the customer side project personnel, because it gives them a new tool to evaluate the risk position of the project. Especially, the customer-side project owner and the project manager found the method promising. This is interesting observation, since the presumption was that the method would bring value especially for the provider-side project manager in evaluating project risk position. *Drawing from the earlier, the results of the case project interviews suggest that the method would provide most value for the customer-side project management, because it provides them with a new tool for estimating project risk elements associated with requirements.*

7.3 Implications to practice

The literature review on requirements risks gives quite a persuasive view on the importance of requirements engineering and management activities for the success on IS development. Both research literature and practical experience show that shortcomings in requirements engineering and management is a major reason for project failure. Therefore, paying attention to the quality of the requirements and the quality of the requirements engineering techniques is worthwhile. IS developers should invest in developing requirements acquisition, development and management practices. It requires also cultural, organizational and processual changes in the organization. As change processes are laborious, expensive and risky to implement, it may lessen business managers' interest in requirements engineering development efforts. This calls for practical techniques and tools for helping organizations in first understanding the real value of requirement engineering practices in their own business context, then understanding their own organization's current state of requirement management practices, and eventually improving their requirements management capabilities. The IS developers should be aware of different requirements development methods and how they are applied, and select methods that are best applicable to the situation to produce and maintain high-quality and low-risk requirements. Various capability maturity models, such as CMMI, and wealthy literature on the area are available to support in the development process.

In this thesis we have studied how the requirements risk are considered in modern ISD approaches. The conclusion is that the ISD methods, such as agile and continuous development, address requirements quality issues, but the risk management approach to requirements is not common. By applying techniques for assessing and prioritizing requirement risks the IS developers could take better control over project risks. For example, in case of product development, where the roadmapping process including requirement analysis is conducted in a continuous manner, risks associated with product requirements could be assessed by the product management team frequently by applying the risk identification and profiling. This would allow the product team to point out high-risk requirements and propose alternative requirements analysis techniques before moving on in a IS development process. On the other hand, many IS practitioners, such as software vendors or system integrators, have developed their own requirements quality analysis checklists and included them as part of their project model. For instance, project milestone reviews may contain requirements-related checkpoints. In addition, common project management standards (such as IPMA or PMP) and capability maturity models (for instance CMMI) cover also requirements development and requirements management. However, they do not directly provide techniques for requirement risk prioritization. Literature also proposes measures, which can be used as performance indicators for requirements engineering activity, and utilized by IS practitioners to measure and develop requirements management processes. Therefore, a following practical implication is recorded: *The organizations that develop information systems should invest sufficiently in requirements analysis and management activities to enable high quality of requirements-related decision making in IS development. The organizations should implement the requirements acquisition, analysis and prioritization processes, which are suitable in their own operating context. The organizations should also measure both quality of requirement management processes and quality of the requirements that the process produces, and improve the applied methods according to the measurement results*.

The research literature represents a method for requirements risk assessment and prioritization method (Tuunanen & Vartiainen 2016), which was tested in a case study arrangement for its applicability and feasibility. As a result, several practical implications can be made.

As explained earlier, the case study results suggest that the risk tables provide useful framework for reviewing the requirements for risks of a project. The method could bring structure to project requirement risk analysis. On the other hand, the method has certain steps, which feasibility were not evident for the case study participants: specifically, applicability of the selection of new requirements analysis techniques based on the risk prioritization, as proposed in the method, was challenged. As a practical implication, instead of using the requirement analysis techniques selection logic proposed in the method, the organization could create a shortlist of practically available techniques, and make technique selection from the shortlist.

In addition, the case study results also raise a question whether the requirements risk listings per project phase proposed in the method (Tuunanen & Vartiainen 2016) covered all relevant risk items in the context of the project. Different

risk factors may be emphasized, depending on project characteristics, such as IS stakeholders (such as user groups), user representation in the project, IS application area (the area of industry where the solution will be applied), applied technologies and system architecture. This raises a question what kind of industry, organization or technology dependent factors for requirements risks there are and how they should they be represented in the risk item tables. Consequently, organizations could study the possibility to adapt the method in their context. Parallelly, the ISD practitioners may as well develop their own variations of the risk prioritization method by including organization and IS context specific risk items in the risk tables and restricting the requirements analysis techniques. Therefore, we record the following practical implication: *The organizations could study possibility to make a context-specific adaptation to the risk tables and requirements analysis technique selection. Instead of using the requirement analysis techniques selection logic proposed in the method, the organization could create a shortlist of practically available techniques, and make selection from the listing. Second, the risk tables could be reviewed for and update with organization and context specific risk items.*

The case study results also show the difficulty in both understanding and communicating the requirement risks, as explained in the chapter 7.2. Applying the method requires rather good understanding of risks and IS requirements in general, which may require participation of multiple people from different parts of the organization to take part in risk assessment. In addition, the case study suggests that the risk prioritization method appears heavy to apply. *Practical implication from this is that the requirement risk analysis should be conducted as team work by multi-disciplinary project team to improve the accuracy of the risk assessment and to divide the workload.*

As explained in the chapter 7.2, it seems that the risk prioritization method was found especially valuable by the customer side project personnel. However, also the case project's service provider's project personnel found potential from the method. It is worthwhile to notice that sometimes in real life IS development projects, service provider project manager and customer side project manager may have some contradictory interests, which derive from different business goals of the involved organizations. In principle, the IS service provider is responsible for technical implementation project in IS development, but the customer carries the responsibility (and costs) about the entire deployment of the IS and about making the most out the investments in IS. The service provider's project manager may have reservations about using the method in their projects, because project risk profile analysis may lead customer to require for more analysis on certain requirements or even demand applying completely new requirement analysis techniques. This may be a setback for the service provider project manager, if his goals are set to run the project through with the predetermined requirements, budget and timeline. Therefore, making full use of the requirement risk analysis method requires that it is applied in correct a phase of the project, and that project contracts (and other managerial or administrative agreements) allow executing the operations proposed by the method. Consequently, a following practical implication is made: *The project setup should support using the method:*

project contracts, project change management, project resourcing, budgeting must be considered.

The requirement risk analysis and prioritization method is intended to be independent of the ISD approach. In the example scenario presented in this thesis (chapter 5.6.1), the requirements risk prioritization method is applied with agile Scrum development framework: in each iteration the method is applied to evaluate the requirements stored in the sprint backlog. The method could be applied as part of iteration planning session, when the backlog items are defined for implementation (for instance in Scrum terminology, in sprint planning sessions). The example is illustrative, and an appropriate way to apply the method is worth more investigation. *The organization implementing the method should fit the steps of the method in their ISD model.*

7.4 Summary

The requirement risks have been studied in numerous studies and they are also discussed in the requirements engineering literature. Therefore, the importance of paying attention to IS requirements has been acknowledged, but still requirements risk management perspective is not commonly applied IS development. In general, requirements engineering literature propose that requirements risks would be managed by producing high quality requirements and applying in requirements elicitation, requirements development and requirements management activities rigorously. The concurrent ISD approaches, such as agile frameworks, have built-in characteristics which help IS developers to keep challenges of requirements development in control. In addition, the literature presents techniques to improve requirement quality, outline different quality criteria for requirements and requirements artefacts, and measures to help implementing high quality requirements development and validation practices. The assumption is that by applying user-centric requirements acquisition and development approaches, and by understanding IS customer value creation, correctness of the IS requirements can be improved and therefore confidence in building a right IS improved. This approach decreases risks associated with requirements, but does not give IS developers tools for managing requirement risks. The requirements risk analysis and prioritization method (Tuunanen and Vartiainen 2016) was proposed to improve requirement risk management in ISD projects. The method was tested in a case study for applicability of the method, and building on the case study results and literature review, theoretical and practical implications were made to improve applicability of the method.

8 Conclusions

In this chapter the conclusions of the thesis work are presented. The following subchapters provide a summary of the thesis, contribution of the work, limitations of the study, and future research items.

8.1 Summary of the study

The academic motivation for this thesis is to study how the risks associated with IS requirements are considered in contemporary ISD approaches. The theoretical part of the thesis consists of a review of research articles and literature on the field. The literature review covers description of concurrent approaches to requirements acquisition and management, with special emphasis on agile ISD and continuous development. Thesis also discusses how requirement risk management and prioritization realize in ISD methods. In addition, the literature review section presents how requirements risks are handled in contemporary ISD methods, such as agile and continuous delivery. The goal of the literature review is to find definition for requirement risks, discuss requirement risk handling techniques and methods, and consider how they are applied agile development methods. Further on, the model for requirements risk analysis and prioritization, which is applied in the case project, is represented based on the literature.

In the empirical part of the thesis, the requirements risk prioritization method presented by Tuunanen and Vartiainen (2016) is tested in the case study for applicability and usefulness of the method. The goal of the case study is to test requirement risk analysis and prioritization method, and propose topics further research. The empirical study was conducted as an interpretive case study. The data was collected from specialist interviews in an IS development industry project, and the interview results and their conclusions are summarized and reported.

8.2 Contribution

The requirement risks have been studied in numerous studies and they are also discussed in the requirements engineering literature. The importance of correctness and high quality of IS requirements has been acknowledged, but approaching IS requirements specifically from risk management perspective is not very common in IS development. Requirements risk management means reviewing requirements for potential risks, analyzing risk exposure and making decisions on how to mitigate the risks and improve the requirements.

This thesis provides a literature-based overview on the concurrent requirements acquisition and development approaches and categorization of characteristics of high quality requirements. In general, the review on requirements engineering and requirements management literature propose that requirements risks in IS development are managed by trying to produce high quality requirements by applying best practices in requirements elicitation, requirements development and requirements management throughout the life-cycle of the IS. The requirements engineering literature proposes many techniques to improve requirement quality, define quality criteria for requirements artefacts, and measures to help implementing high quality requirements development and validation practices. The concurrent ISD approaches emphasize user-centric thinking, stakeholder collaboration and the importance of understanding the customer value creation. The assumption is that by applying user-centric requirements acquisition and development approaches, and by understanding IS customer value creation, correctness of the IS requirements can be improved and therefore confidence in building a right IS improved.

In this thesis, an analysis of a generic agile model's requirement risk mitigating characteristics is presented. The concurrent ISD approaches, such as agile frameworks, have many characteristics which help IS developers in keeping challenges of requirements development under control. However, the agile models do not dictate methods for identifying, acquiring, and developing the customer needs into requirements, but leave space for various kinds of requirement elicitation, development and prioritization techniques.

In IS development literature, the requirements risks are mostly approached specifically from the requirement development perspective keeping the focus in developing high quality requirements. This approach decreases risks associated with requirements, but does not give IS developers tools for measuring and managing requirement risks – 'if you can't measure it, you can't manage it'. Even though there has been research activity on the requirement risks, approaching IS requirements specifically from risk management perspective is not that commonly applied in practical IS development. This thesis also describes the concept of requirement risk and requirement risk categorization, and discusses practical means for requirement risk handling, and provides the reader with set of principles which help in mitigating requirement risks. Specifically, agile development and continuous development environment and requirements risk management is discussed.

The requirements risk analysis and prioritization method (Tuunanen and Vartiainen 2016) is proposed for improving the requirement risk management in ISD projects. The method was tested in a case study for applicability of the method. Based on the case study results and literature review, theoretical and practical implications were made.

The case study results suggest that the requirement risk analysis approach was found novel by the case study participants. In addition, the method was found applicable in the case project and provided valuable information about the requirements risks. The requirements risk analysis and prioritization based on

the risk tables was found meaningful for evaluating the risks associated with requirements. Especially, the risk tables provide a useful framework for reviewing the requirements for risks that they may pose for the project.

The implications from the study also indicate that context-specific requirements adaptation on the risk tables should be considered to improve comprehensibility of risk items, to add missing risk items, and to cover potential gaps in risk tables. Context adaptation to the requirement analysis techniques selection phase of the method was also brought up.

The implications also contain remarks on applying the method in ISD projects. First, it is suggested that the requirement risk analysis should be conducted as team work by a multidisciplinary project team to improve the accuracy of the risk assessment and to divide the workload. Second, the project arrangements (project contracts, project change management, project resourcing, budgeting) should not be in contradiction with the method.

As an interesting implication, the results of the case project interviews suggest that the method would provide most value for the customer-side project management, because it provides them with a new tool for estimating project risk elements associated with requirements. This notice may be relevant when making decisions about the direction of development.

8.3 Limitations

This study consists of two main parts: the literature review and the empirical part. In the literature review, an overview of the requirements risk management advances in concurrent ISD approaches are presented. The empirical part consists of requirement risk review method evaluation. The limitations of the study are discussed in this chapter.

The limitations of the literature review are connected to sufficiency of the review coverage, i.e. is the review coverage wide enough. In the review, we specifically aimed at finding relevant information science research literature on the requirement risks. Especially finding practitioners solutions to requirement risk handling is difficult, as they are often intellectual property of private businesses, such as consultants or software vendors, who do not publicize details of their methods. Also, a wider analysis of IS project management literature and project management certification and standardization literature might have given additional insight to the topic.

Similar critique can be made towards the literature review coverage of the concurrent ISD and requirement analysis approaches. In the thesis, we chose to use the generic agile model by Measey (2015) to represent agile approach, which may leave out some agile techniques from the review.

The empirical study has also its limitations. Even though the case study was carefully prepared to match with the research targets, the case study arrangement had some limitations, which limit the both reliability and generalizability of the results.

First, the case project characteristics bring certain predetermined limitations. As the case project was a further development project for a management information system that is already in production use, the results may not be directly generalized for instance to development of systems intended for mass markets or to development of a completely new (non-existing) systems. The case project did not involve adding new user groups to the system either. The user requirements tend to be better understood in further development projects than in designing completely new solutions, because both the customer and IS developers have experience in the system in its real use context.

Second, it is noteworthy that the IS of the case project was a customer-specific tailored IS-based solution for a specific user group within customer organization. In this situation, the customer has fairly good understanding about the goals of the IS per user group. Therefore, the case project is not comparable to product development type of ISD projects, as in IS product development key challenges are identifying such requirements that are common across different customers, and developing a solution that is feasible for many different user groups.

Due to the limitations described above, applicability of the risk prioritization method in product development type of IS development could not be properly assessed. However, developers of an IS-based product or service (consider for instance SaaS solutions) may find the risk prioritization valuable, because the product developers seek maximum return on the investments put in product development. If the requirement risk analysis improved understanding about requirements risks, it certainly would give relevant input for making good product management and roadmapping decisions. And further on, if the requirements risk analysis and prioritization was conducted as part of product management and roadmapping activity frequently, would that improve the product owner's confidence in implementing right features in the IS-based product or service? The case study of this thesis did not give much input to answer this question.

Another limitation of the case project results from the fact that the interviews were made only in one case project. Therefore, issues that are specific to this single project and opinions of individual interviewees may get too much emphasis. This must be remembered when drawing conclusions from the case study results: the generalizability of the results is therefore limited.

Certain case study limitations arise from the constraints of the interviews. Because the interviews were time-limited and applying the method was time consuming, it was not possible to analyze every requirement (use case) in the requirement specification of the case project in the interviews. Therefore, it was not possible to create full requirement risk profile for the case project based on the risk analysis of the entire requirement collection, which means that the requirement analysis technique selection with the risk resolution rules could not be fully tested. Instead, this phase of the method was discussed with interviewees to explain the logic and the purpose of the technique selection. Therefore, there is a risk is that the value of the techniques selection logic was not properly per-

ceived by the interviewees. This limits the reliability of the case study results regarding the requirement analysis technique selection. It worth noticing that the interviewees did not find it possible to introduce completely new requirement analysis techniques into the case project, which indicates that this phase in the method would be worth further analysis.

8.4 Future research

In this chapter the topics for future research are presented. The areas for further research focus especially on the development ideas for the requirement risk analysis and prioritization in IS development. The literature review revealed that approaching requirements from risk management perspective is not common in IS development. However, methods and techniques for the purpose exist, but their practical adaptations require further work. The conclusion from the case study is that the requirement risk prioritization method was found to give valuable information to project team members in the case study arrangement about the requirement risks. However, several issues for further research were brought up during the study.

One of further research items is associated with the case study observation about the potential need for context specific adaptations of the requirement risk analysis and prioritization method. The interviewees brought up that organization and project context specific adaptation of the risk tables might be needed to increase reliability of the risk analysis and prioritization. Therefore, an interesting future research item would be to study what kind of industry, organization or technology dependent factors there are for requirements risks, and how they should they be represented in the risk item listings. It might be valuable to first construct a model for describing IS project contexts for requirement risk analysis purposes. The model would describe common IS project factors, which are relevant from the requirement risk analysis and prioritization perspective. The model could be utilized in adding context-adaptive requirements risk approach to the requirements risk analysis and prioritization method.

Also, practical application of the requirement risk prioritization method with different ISD approaches and in different project setups could be developed further. The method is intended to be independent of the ISD approach, but practical adaptation must be done in the ISD context. In the example scenario presented in this thesis (chapter 5.6.1), the requirements risk prioritization method is applied with agile framework, namely Scrum. In the example, it is proposed that the method is applied as part of iteration planning sessions, when the backlog items are groomed and prepared for implementation (e.g. in Scrum terminology, in sprint planning sessions). The example is superficial and mostly illustrative. Practically applicable way of implementation of the method is worth more study: for example, which risk item lists should be actually be applied in sprint planning and sprint review sessions, or how to deal with different types of risk requirement risk profiles in agile environment.

Also the limitations of the case study arrangement pointed out in the chapter 8.3 leave several open issues for further research efforts. Testing the method in different ISD development scenarios would be worthwhile. For instance, developing the requirement risk control in continuous development environment of a mass-market software or service is worthwhile. The case study of this thesis did not give input in this questions, and further research on the topic would be interesting to evaluate applicability of the method for this purpose. In addition, certain steps of the method, especially the requirement analysis technique selection step, require further analysis, since its applicability was challenged in the case study.

As described earlier, the case study had its limitations, which limit the reliability and generalizability of the results. Additional empirical research would be valuable to study the method in alternative contexts. Further on, in this thesis case study the interviews were conducted in the early phase of the case project. Therefore the interviews did not provide sufficient information to reliably evaluate the impact of the method on organization level performance, because the project outcome is not known. Based on the case study, the method shows promise, but proper impact analysis would be also worth further research for instance in a longitudinal study arrangement.

8.5 Summary

In this chapter the findings of the study are summarized, and the conclusions are presented. The chapter also highlighted the limitations of this study and further research items identified during the work.

REFERENCES

Barki, H., Rivard, S., and Talbot, J. (1993). Toward an Assessment of Software Development Risk. Journal of Management Information Systems (10:2), pp 203-225.

Boehm, B.W. (1991). Software risk management: principles and practices, IEEE Software 8 (1), 1991, pp. 32–41.

Chen, L. (2015). Continuous Delivery: Huge Benefits, but Challenges Too. IEEE Software (32:2), pp. 50-54.

Claps, G.G., Berntsson-Svensson, R., and Aurum, A. (2015). On the Journey to Continuous Deployment: Technical and Social Challenges along the Way. Information and Software Technology, vol. 57, 2015, pp. 21–31.

Delone, W.H. & McLean, E.R. (2003). The DeLone and McLean Model of Information Systems Success: A Ten-Year Update. Journal of Management Information Systems Vol. 19, Issue 4, 2003.

Ebert, C. (2014). Software Product Management. IEEE Software, vol. 31, no. 3, May-June 2014, 21-24

Google Ventures. (2017) Design Sprint Kit https://designsprintkit.withgoogle.com/, accessed 15.10.2017.

Hooks, I.F. & Farry, K.A. (2001). Customer-centered products: Creating successful products through smart requirements management. AMACOM, New York, USA, 2001.

ISO/IEC 24765 (2010). Systems and software engineering — Vocabulary. ISO/IEC/IEEE 24765:2010(E), 1st edition. 15 December 2010. Online publication, available: http://www.iso.org. Accessed: 22nd December 2016.

Iversen, J. H., Mathiassen, L., & Nielsen, P. A. (2004). Managing risk in software process improvement: an action research approach. *MIS Quarterly*, 395-433.

Kauppinen, M., Savolainen, J., Lehtola, L., Komssi, M., Töhönen, H. & Davis, A. (2009). From feature development to customer value creation. 17th International requirements engineering conference RE'09, 275–280.

Keil, M., Cule, P. E., Lyytinen, K., & Schmidt, R. C. (1998). A Framework for Identifying Software Project Risks. *Communications of the ACM*, (41) 11, pp. 76-83.

Klein, H. K. and Myers, M.D. (1999). A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems, MIS Quarterly, Special Issue on Intensive Research (23:1), 1999, pp. 67-93.

Komssi, M., Kauppinen, M., Töhönen, H. Lehtola, L. & Davis, A.M. (2015). Roadmapping problems in practice: value creation from the perspective of the customers. Requirements Engineering (2015) 20: 45-69.

Leffingwell, D. (2017). Scaled Agile Framework (SAFe). Available online at www.scaledagileframework.com. Accessed 28 September 2017.

Leikas, J. (2009). Life-Based Design - A holistic approach to designing human-technology interaction. VTT Technical Research Centre of Finland, VTT publications 726, 2009.

Mathiassen, L., Saarinen T., Tuunanen T., and Rossi M. (2007). A Contingency Model for Requirements Development. Journal of the Association for Information Systems (8:11), 2007, pp. 569-597.

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?docID=1759633.

Nemoto, Y., Uei, K., Sato, K. & Shimomura, Y. (2015). A Context-based Requirements Analysis Method for PSS Design. Procedia CIRP 30, 42-47.

Patrício, L., Falcão e Cunha, J. & Fisk, R.P. (2009). Requirements engineering for multi-channel services: the SEB method and its application to a multi-channel bank. Requirements Engineering 14:3, 209-227.

Peffers, K., et al. (2003). Extending Critical Success Factors Methodology to Facilitate Broadly Participative Information Systems Planning. Journal of Management Information Systems 20(1): 51-85.

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), pp. 508-532.

Pohl, K. (2010). Requirements Engineering: Fundamentals, Principles, and Techniques, 1st ed. Springer Publishing Company, Inc. 2010.

Smith, A., Bieg, D.P. & Cabrey, T.S. (2014) Requirements Management — A Core Competency for Project and Program Success. PMI's Pulse of the Profession In-

Depth Report, August 2014. Online publication, available http://www.pmi.org/ Accessed 22nd December 2016.

Rodriguez, P., Haghighatkhah, A., Lwakatare, L.E., Teppola, S., Suomalainen, T. Eskeli, J., Karvonen, T., Kuvaja, T., Verner, J.M. & Oivo, M. (2017). Continuous deployment of software intensive products and services: A systematic mapping study. Journal of Systems and Software, Volume 123, January 2017, Pages 263-291.

Ryynänen, T., Karvonen, I., Korhonen, H. & Jansson, K. (2016). Collaboration Driven Requirements for a Product-Service Engineering Platform. Collaboration in a Hyperconnected World. In the proceedings of 17th IFIP WG 5.5 Working Conference on Virtual Enterprises, PRO-VE 2016, Porto, Portugal, October 3–5, 2016, 340-349.

Tuunanen, T. & Govindji, H. (2016). Understanding flow experience from users' requirements. Behaviour & Information Technology 35:2, 134-150. Online publication date: 27-Feb-2015.

Tuunanen, T. & Kuo, I-T. (2015). The effect of culture on requirements: a value-based view of prioritization. European Journal of Information Systems, May 2015, Vol. 24 (3) 295-313.

Tuunanen, T. & Peffers, K. (2016). Stakeholder Requirements Acquisition Methodology, submitted to an international peer-reviewed journal. 2016.

Tuunanen, T. & Vartiainen, T. (2016). Development of Requirements Risk Prioritization Method, submitted to peer-reviewed journal. 2016.

Wallace, L., Keil, M. & Arun, R. (2004). Understanding software project risk; a cluster analysis. Information & Management, (42) 1, pp. 115-125, 2004.

Wiegers, K. & Beatty, J. (2013). Software Requirements. Microsoft Press. Redmond, Washington, USA, 2013.

APPENDIX 1 - INTERVIEW QUESTIONS

The interviews are done using the following interview structure and questions. The interviews are done in Finnish, as all the interviewees are native Finnish speakers. Therefore, also the questions are presented in Finnish.

Aloitus

Haastattelijan esittäytyminen, tutkimusaiheen kertominen sekä haastattelukäytännön kuvaaminen: vaatimusriskien priorisointi- ja hallintametodin soveltuvuus projektityöhön.

Kysymykset:

- Haastattelu kestää noin tunnin – onhan se edelleen sopiva aikataulu?
- Voinko nauhoittaa haastattelun?
- Onko sinulla kysymyksiä haastatteluun liittyen?
- Kerrotko seuraavat tiedot: ikä, tehtävä yrityksessä, tehtävä projektissa

Menetelmän kuvailu haastateltavalle tapahtuu käyttäen apumateriaalia (joko paperilla tai sähköisenä kannettavan näytöltä):

- Vaatimusriskien kategoriat sekä esimerkkejä niistä
- Riskitaulukoiden esittely
- Menetelmän kulku

Menetelmän testaaminen case-projektissa: kuvaus miten testaus aiotaan tehdä

Case-projektin vaatimusten katsaus

Yleiskatsaus case-projektin vaatimuslistaan (use caset ja niitä tarkentavat materiaalit, kuten UI-wireframet).
Käsiteltävien use casejen valinta (max 3 kpl): haastateltava valitsee use casen, johon liittyviä riskejä haluaa tarkastella

Kysymyspatteri 1: Valittujen käyttötapausten / vaatimusten arviointikysymykset

Kysymykset haastateltavalle, kun tarkastellaan vaatimusdokumentaatiota / use casea:

- Kuvaile omin sanoin käsiteltävää vaatimusta?
- Minkä käyttäjäryhmän tai -ryhmien tarpeeseen vaatimus vastaa?
- Minkä sidosryhmien työhön tai toimintaan vaatimuksen toteuttaminen vaikuttaa?

Kysymyspatteri 2: Riskien käsittely menetelmän mukaan

1. **Riskien tunnistaminen**

Riskien tunnistaminen valituille vaatimuksille menetelmän riskitaulukoiden mukaisesti (Requirements phase checklist, Design phase checklist, Implementation phase checklist):

- Haastattelumenettely:
    - Yksi taulukon rivi kerrallaan pyydetään haastateltavaa arvioimaan, että toteutuuko kyseinen riski (vastaus: ”kyllä / e i /EOS / ei relevantti”)
    - Perustele vastauksesi

2. Riskiprofiilin arviointi

Soveltaen projektin riskiprofiilin arviointitaulukkoa, valituille vaatimuksille laaditaan riskiprofiili käyttäen ennakkoon valmisteltua taulukkoa, jossa riskikategoriat ja impaktin taso myös nähtävissä.
Esitellään haastateltavalle riskiprofiili vaatimuksille. Kysymykset:

- Onko riskiprofiili mielestäsi mielekäs kyseisille vaatimuksille?

3. Riskien painopisteiden (kategoriat) arviointi

Risk resolution pattern -taulukon avulla arvioidaan perusten riskiprofiiliin, että mihin riskeihin tulisi erityisesti kiinnittää huomiota. Kysymykset:

- Onko saatu riskiprofiili mielekäs?

4. Riskienhallinnan menetelmäehdotukset

- Millaisia toimenpiteitä kohdistaisit riskeihin - perustuen omaan kokemukseesi?
- Risk resolution techniques -taulukon menetelmien lyhyt katsaus: kerrotaan olemassa olevista menetelmistä
    - Onko tässä tuttuja menetelmiä?
- Pidätkö projektin vaiheen ja käytännöt huomioon ottaen mahdollisena vaatimusten tarkempaa tarkastelua jonkin uuden menetelmän avulla?

Kysymyspatteri 3: Priorisoitujen riskien oikeellisuus, relevanssi ja kattavuus

Kysymykset mallin antamien tulosten tarkkuuden, kattavuuden ja merkityksellisyyden arvioimiseksi:

- Onko priorisoitu lista relevantti (eli ovatko riskit oikeita)?
- Onko vastaavat riskit tunnistettu aiemmin?
- Ovatko havaitut riskit merkityksellisiä?
- Toivatko riskitaulukot esiin vaatimuksiin liittyviä riskejä, joita ei aiemmin ole käsitelty?
- Millaisia muita vaatimuksiin liittyviä riskejä määrittelyn aikana on ilmennyt?

- Entä saman järjestelmän edellisten kehitysprojektien aikana? (Esimerkkejä voivat olla vaikkapa ”unohtuneet” sidosryhmät, väärin tulkitut vaatimukset tai käyttämättä jääneet ominaisuudet)

Kysymyspatteri 4: menetelmän soveltuvuus projektikäyttöön ja tulosten hyödyllisyys

- Tuottaako tämä menetelmä sellaista tietoa, joka auttaa sinua projektiin liittyvässä päätöksenteossa?
- Auttaako vaatimusten käsittely menetelmän mukaan jäsentämään projektin riskitekijöitä eri tavalla kuin aiemmin?
- Löytyikö aiemmin havaitsemattomia projektiriskejä?
- Voisitteko ajatella hyödyntävänne menetelmää uusien projektin vaatimuksia analysoitaessa?
    - Jos ei, niin miksi ei?
- Missä vaiheessa projektia soveltaisitte menetelmää?
- Oletko tyytyväinen tietoon, jota menetelmä on tuottanut? Auttaako se sinua projektiin liittyvässä päätöksenteossa?

Yhteenveto ja lopetus
Onko sinulla vielä asioita, jotka haluaisit nostaa esille?

APPENDIX 2 – RISK TABLES

Impact to ISD Project	Risk Items	
High	1. Absence of project sponsor	Identity
	2. Access to clients	Complexity
	3. Ambiguous requirements	Identity
	4. Client commitment	Identity
	5. Delivering what the client needs	Identity
	6. Incorrect stakeholder	Identity
	7. Missing requirements	Identity
	8. Misunderstood business needs	Identity
	9. Unrated requirements	Volatility
Medium	10. Change in business strategy and direction	Volatility
	11. Change in external regulations	Volatility
	12. Compliance with external regulations	Identity
	13. Conflicting requirements	Integrity
	14. Emerging requirements dependency	Volatility
	15. Fixed budget and deadline	Integrity
	16. Hostile users	Identity
	17. Knowledge gap between coworkers	Complexity
	18. Lack of collaboration	Complexity
	19. Project team member turnover	Volatility
	20. Underestimation of change magnitude	Volatility
Low	22. Constrained by users' knowledge	Complexity
	23. Technology changes	Volatility

Requirements phase checklist

Risk	Risk Type	Project is/can be exposed?
Absence of Project Sponsor	*Identity*	
Access to Clients (Proximity to Source)	*Complexity*	
Ambiguous Requirements	*Identity*	
Change 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*	
Any other risks that could affect design and implementation		

Design phase checklist

Risk	Risk Type	Project is/can be exposed?
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 Gap between Coworkers	*Complexity*	
Lack of Collaboration	*Complexity*	
Technology Changes	*Volatility*	
Underestimation of Change Magnitude	*Volatility*	
Unrated Requirements	*Volatility*	
Any unresolved risks from requirements and risks that could affect implementation		

Implementation phase checklist

Risk	Risk Type	Project is/can be exposed?
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*	
Any unresolved risk items from design and requirements		

Step 2: Assess Project Risk Profile:

Requirements Phase Specific Risks	Impact	Design Phase Specific Risks	Impact	Implementation Phase Specific Risks	Impact
Absence of project Sponsor	High	Missing requirements	High	Hostile users	Medium
Access to clients (proximity to source)	High	Delivering what the client requires	High	Project team member turnover	Medium

Incorrect Stakeholder	High	Compliance with external regulations	Medium		
Misunderstood business needs	High	Conflicting requirements	Medium		
Change in business strategy and direction	Medium	Emerging requirements dependency	Medium		
Constrained by users' knowledge	Low	Knowledge gap between coworkers	Medium		
		Lack of collaboration	Medium		
		Technology changes	Low		
Risks Affecting All Phases					
Ambiguous requirements	High	Ambiguous requirements	High	Ambiguous Requirements	High
Unrated requirements	High	Unrated requirements	High	Unrated Requirements	High
Client commitment	High	Client commitment	High	Client Commitment	High
Change in external regulations	Medium	Change in external regulations	Medium	Change in external regulations	Medium
Underestimation of change magnitude	Medium	Underestimation of change magnitude	Medium	Underestimation of change magnitude	Medium
Fixed Budget and Timelines	Medium	Fixed budget and time lines	Medium	Fixed budget and timelines	Medium

APPENDIX 3 – REQUIREMENTS ENGINEERING TECHNIQUES

Listing of requirement specification, experimentation, discovery and prioritization techniques by Tuunanen & Vartiainen (2016).

Name	Specification	Experimentation	Discovery	Prioritization
Affinity technique			*	
Aspect mining in requirements specification			*	
Attributed goal-oriented analysis	*		*	
Behavior analysis	*		*	
Box structure specification and design	*			
Brainstorming			*	
Business information analysis and integration technique	*		*	
Business process planning (BSP)	*		*	*
Card sorting			*	*
Cognitive mapping			*	
Contextual design	*		*	*
Cooperative prototyping		*		
CREV	*		*	
CREWS	*		*	
Critical success factors			*	*
Data flow diagram	*			
Decision analysis			*	
Delphi method			*	*
Deriving requirementsfrom existing system			*	
Domain specific modeling	*			
EasyWinWin			*	*
Email/bulletin board			*	
Ends/means analysis			*	
Entity-relationship modling	*			
Facilitated team			*	
Focus group			*	
Future analysis			*	
Goal modeling oriented requirements elicitation	*		*	
Goal oriented approach	*		*	

Group support systems and strategic business objectives			*	*
Guided brainstorming			*	
Human, social and organizational requirements elicitation			*	
Inquiry cycle model – structure and describe requirements discussions	*		*	
Joint application design		*	*	
KAOS	*			
Laddering			*	
Lyee	*			
Machine rule induction	*			
Marketing and sales			*	
MIS intermediary	*		*	
Multidimensional data models	*			
Multidimensional scaling	*			
Nominal group technique			*	*
Normative analysis			*	
Object oriented Z	*			
Open interview			*	
Open systems task analysis			*	
Participatory design		*	*	
Petri nets	*			
Petri nets combined with use cases	*		*	
Precision model			*	
Prime-CREWS	*		*	
Process analysis			*	
Protocol analysis			*	
Prototyping		*		
Quality function deployment	*			*
Repertoire grids			*	
Requirements generation model			*	
Requirements prototyping		*		
Requirements workshops			*	
Rich pictures	*		*	
Scenario-based requirements elicitation	*		*	
Semantic maps			*	
Socio-technical analysis			*	
State charts	*			
Strategic business objectives			*	*
Strategy set analysis			*	

Structured group elicitation method			*	
Structured interview			*	
Structured walkthroughs			*	
Support line			*	
Surveys			*	
Teach-back interview			*	
Testing		*	*	
Text analysis			*	
Trade show		*	*	
Usability lab		*		
Use cases			*	
Use of video in requirements elicitation			*	
User group			*	
User-interface prototyping		*	*	
Variance analysis			*	
VDM ++,VDM-SL	*			
Warnier-Orr diagrams	*			
Z	*			