Department of Computer Science
Series of Publications A
Report A-2019-2

On the Provisioning of Mission Critical Information Systems based on Public Tenders

Aapo Koski

To be presented, with the permission of the Faculty of Science of the University of Helsinki, for public criticism in Room 1 of Metsätalo building, University of Helsinki, on the 7th of August, 2019 at 12 o'clock noon.

University of Helsinki
Finland

Supervisors
Tommi Mikkonen, University of Helsinki, Finland
Tomi Männistö, University of Helsinki, Finland

Pre-examiners
Ivica Crnkovic, Chalmers University of Technology, Gothenburg, Sweden
Carlos Cuesta, Universidad Rey Juan Carlos, Madrid, Spain

Opponent
Kari Smolander, Lappeenranta University of Technology, Finland

Custos
Tommi Mikkonen, University of Helsinki, Finland

Contact information

Department of Computer Science
P.O. Box 68 (Pietari Kalmin katu 5)
FI-00014 University of Helsinki
Finland

Email address: info@cs.helsinki.fi
URL: http://cs.helsinki.fi/
Telephone: +358 2941 911

ISSN 1238-8645
ISBN 978-951-51-5324-1 (paperback)
ISBN 978-951-51-5325-8 (PDF)
Helsinki 2019
Unigrafia

On the Provisioning of Mission Critical Information Systems based on Public Tenders

Aapo Koski

Department of Computer Science
P.O. Box 68, FI-00014 University of Helsinki, Finland
aapo.koski@iki.fi
http://linkedin.com/in/aapokoski/

PhD Thesis, Series of Publications A, Report A-2019-2
Helsinki, July 2019, 96 + 79 pages
ISSN 1238-8645
ISBN 978-951-51-5324-1 (paperback)
ISBN 978-951-51-5325-8 (PDF)

Abstract

Large scale software-centric information system projects on public sector are often based on public tenders, in which the established request for quotation (RFQ) process is utilized in various forms. The way the information systems in these cases are procured is typically a long and energy consuming process. In this process, after the initial realization that there is a need for which the system is required for, the procuring organization, i.e., the future customer or an organization acting in behalf of the customer, first tries its best to determine, detail and document the need and then, based on received proposals, tries to evaluate the best candidate to fulfill the need with an proposed solution. In the past, these RFQ-based procuring processes aimed at and mostly also resulted in waterfall-type development processes, where again a considerable time was spent in specifying, implementing, testing and tuning the constructed information system before it finally was ready and accepted to be deployed for the operative use.

The approach utilizing well-documented needs, the tendering phase, and waterfallish development process after the awarding phase has numerous shortcomings. One of the most important problems is the strong dependency on the upfront designs and the implicit assumption that the need and the solutions to that need can be communicated effectively with tendering documents

and solution descriptions. Another major problem is the implicit assumption that the original needs do not significantly change during the process. Many development projects that are run in this way fail, either by exceeding their budgets, by experiencing considerable delays or just by being unable to provide a satisfactory solution to the actual need.

As we have entered the era of agility during the last 10-15 years, backed up by the agile values and principles, the incremental, iterative and customer-involving approaches have found their way into the RFQ-based tenders. This has certainly happened for a reason, the most likely reason being the hope the agility is the silver bullet that can cure the problematic RFQ process. The introduction of agility has certainly potential to solve some of the problems encountered in the RFQ processes, but at the same time, new challenges surface. Furthermore, recently many organizations, also on the public sector, have reassessed their position as both the information system user and the system's maintainer and have realized that it might be a good idea to let the software or IT companies to provide the needed systems as a service, thereby also transferring the risks and costs of running the system to external parties, and potentially to a party who has developed the system. This software-as-a-service (SaaS) paradigm opens up a whole new set of possibilities to develop information systems, get feedback and enable co-operation between the system owners or users and the system developers on improving the system.

This thesis studied the challenges encountered in mission critical information system projects in industrial setting, based on public tendering processes. The reasons for the challenges and why the challenges are difficult to tackle were analyzed from various points of view by the publications. The format of the study is that of the multiple-case study investigating and analyzing the causal links in the real-life projects. As a result, it seems that the traditional RFQ-based process, even when amended with some requirements calling for agile ways of working, does not provide us appropriate means to deliver high quality mission critical systems in an effective way and alternatives are needed. Based on the experiences gained from the system provision domain, it is evident that the SaaS model is able to save us from many of the short-comings of the RFQ process, by enabling agility and helping us, at least partly, to tackle the encountered challenges in the traditional RFQ process.

Providing something as a service is, however, far different from traditional information system development and deployment and thus new, user and customer-facing skills are required from the organization providing the service. In addition, several other improvements, like changes in the way the RFQ process is used, or even to the law governing the public tenders, would also be required to allow us to achieve the best possible outcomes and best achievable return-of-investment (ROI) for the information system projects in the future.

Computing Reviews (2012) Categories and Subject Descriptors:

Software and its engineering → Software creation and management → Software development process management → Software development methods
Software and its engineering → Software creation and management → Collaboration in software development → Programming teams
Information systems → Information systems application → Decision support systems

General Terms:
Mission critical, information system, information technology, public tender, software-as-a-service, service provision

Additional Key Words and Phrases:
Software architectures, requirement management, evolving architecture, continuous delivery, continuous improvement, customer care

Acknowledgements

During my already relatively long career in the software industry, I have been involved with the development of large-scale mission critical software-centric systems for the major part of my career. These systems are often fascinating as in them the coolness of the software-enabled malleability and the extremely high complexity of large software systems are contrasted by the strict requirements of no failures. In many cases, a system failure is just not an option and needs to be avoided at *any* cost (a bit more about that "*any*" in the discussion on the economics in Section 5.6).

This thesis is a direct outcome of the experiences I have gained first-hand while working in several mission-critical software projects. As one who does not have his background in the university, and who may not have the mindset of a true researcher, linking experiences into topics circulating in the scientific community and trying to document my ideas and insights as scientific papers, has given totally new perspectives to the performed work. These perspectives have enabled better understanding on what we are actually dealing with in software projects, when trying to respond to customer needs with information system solutions. The whole process of writing the publications, visiting several scientific conferences, getting to know people in the research community, getting updated on what is the state-of-art in the software engineering, and finally preparing the thesis, has been an exciting and fascinating adventure.

One could argue that this thesis does not bring anything truly new into the research of information systems and the provision of information systems with the SaaS paradigm; the ideas presented are not new. However, the unique combination of a mission critical system, public sector with its established procuring process and SaaS is still relatively rare and certainly deserves more attention. For that purpose, this thesis tries to do its part, by accentuating the opportunities the right combination of these elements offers.

As a doctoral thesis has one author, one may be inclined to think that the thesis is an outcome of a somewhat solitary effort: developing ideas in your head, elaborating and documenting the ideas into articles and bringing it all together by the creation of an introductory part. The truth is quite far from this.

I believe that no thesis, and especially not this one, would have seen the daylight without a great deal of input, contribution, co-operation, commitment, good will and support from a large number of people. While these people remain unnamed in the context of this thesis, I, nevertheless, see this thesis more like a group assignment than an individual work. I feel that most of the people I have been working and interacting with, during the last 20-25 years, in various companies and projects, have had a contribution to the finished thesis. That contribution has taken place either by arguing with me, agreeing with me, disagreeing with me or by any other means that has made me to realize something that I may not have understood before—like being wrong on my bold assumptions. The possibility to collaborate and share experiences when working together in a project are unique and working with people makes me feel that I'm not alone with my struggles. Software development is first and foremost teamwork and succeeds only if we work effectively as a team. The co-operative character of the software engineering is one of the main reasons I want to do this line of work. Therefore, I want to thank each of you for being there and doing your part. I remember you.

This work has been supported by the Department of Pervasive Computing, Tampere University of Technology, as well as by the Department of Computer Science, University of Helsinki. The work has also been supported by the companies that I have been working for and the organizations I have been working with during the last five years or so, for which support I am truly grateful. I wish also to thank all my co-authors of the included papers for excellent and enjoyable collaboration.

Finally, this thesis would not have ever been finished without the patience, constant and sustained encouragement and constructive support from (some might call it the stubbornness of) Professor Tommi Mikkonen. It would also have never got its final form without the effort Professor Tomi Männistö invested in it. Thank you both, Tommi and Tomi.

Helsinki, June 2019
Aapo Koski

Contents

List of Figures

List of Tables

List of Publications

This thesis is based on the eight peer-reviewed publications listed below. The publications are referred to in the text as Publications I-VIII. The publications are numbered in the chronological order of the publication date.

I: Koski, A. and Mikkonen, T. (2015) Rolling out a mission critical system in an agilish way: Reflections on building a large-scale dependable information system for public sector. In *Proceedings of the 2015 IEEE/ACM 2nd International Workshop on Rapid Continuous Software Engineering (RCoSE)* (pp. 41-44) IEEE.

II: Koski, A. and Mikkonen, T. (2015) Requirements, architecture, and quality in a mission critical system: 12 lessons learned. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE)* (pp. 1018-1021) ACM New York, NY, USA.

III: Koski, A. and Mikkonen, T. (2016) On the Windy Road to Become a Service Provider: Reflections from Designing a Mission Critical Information System Provided as a Service. In *Proceedings of the International Conference on Information Systems Engineering (ICISE)* (pp. 51-56) IEEE.

IV: Koski, A. and Mikkonen, T. (2016) Can we get some service here?: On the company transformation from a software vendor to a SaaS provider. In *Proceedings of the 12th International Conference on Web Information Systems and Technologies (WEBIST 2016)* (pp. 279-284) SCITEPRESS.

V: Koski, A., Kuusinen, K., Suonsyrjä, S., and Mikkonen, T. (2016) Implementing Continuous Customer Care: First-hand Experiences from an Industrial Setting. In *Proceedings of the 42th Euromicro Conference on*

Software Engineering and Advanced Applications (SEAA) (pp. 78-85) IEEE.

VI: Koski, A. and Mikkonen, T. (2017) What We Say We Want and What We Really Need: Experiences on the Barriers to Communicate Information System Needs. In *Requirements Engineering for Service and Cloud Computing* (pp. 3-21) Springer, Cham.

VII: Koski, A. and Mikkonen, T. (2017) Taming a Monster: Tackling the Emergent Issues Encountered in Mission Critical System Development. In *Proceedings of the 18th International Conference on Agile Software Development (XP 2017)* (pp. 1-8) Agile Alliance.

VIII: Koski, A. and Mikkonen, T. (2017) How to a Survive Mission Critical Systems Project Based on Public Tenders: Lessons Learned the Hard Way. In *Proceedings of the 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (pp. 35-39) IEEE.

The rights have been granted by the publishers to include the papers in the thesis.

Details on the role of the candidate per publication is described shortly in the following:

Publication I studied and analyzed real-life experiences on the design, implementation, testing and deployment efforts in a large mission critical information system project, executed in an incremental way. As the downtime of such a system is not acceptable, the system's extremely high availability requirement needs to be ensured by and taken into account in all the phases of the development effort. Furthermore, all parties involved need to understand the importance of and the criteria used for measuring the availability in the same way, as for the end-user the availability does not just mean that the services are up and responding swiftly. The basis for this publication is the first-hand experience candidate has gained while working as the chief architect of the project and the learnings presented in the publication are by candidate as well. The paper was based on the candidate's experiences and work within the information system project and the main ideas and the analysis presented are from the candidate. The candidate was the main author of the conference paper.

Publication II studied the way the mission critical information systems are created, typically starting from a set of written-down requirements and defining the architecture based on them. By the experiences gained in real software projects, this setup often leads to emerging architectures and unpredictable quality which, naturally, is not acceptable with any system and even more so with a mission critical system. To alleviate the situation, the publication proposes that such critical information system projects should start from the quality attributes and derive the required architecture from there on, instead of emphasizing the functional needs. As Publication I, this publication is also based on the experiences the candidate has first-hand gained from information system projects. The original idea is by the candidate and the candidate was the main author of the conference paper.

Publication III studied the challenges a company with project-based background faces when transforming to a more service-based business model. In case the system in question is a mission-critical one, the extremely high reliability and availability requirements make the service provision even harder, emphasizing further the need to have effective and wide-band communication between the service provider and the users of the service. Publication is based on the work candidate has personally done in a company performing the transition from a one selling software licenses to one providing software as a service. As with the previous two publications, the experiences behind this paper are candidate's own and the candidate was the main author of the conference paper.

Publication IV further analyzed in detail the knowledge that needs to be gained and the skills that a IT company needs to have to succeed with the service provision model. Contrasting the required skills and characteristics to the agile principles of the Agile Manifesto, the paper emphasizes that the SaaS model, if applied appropriately and with enough discipline, enables the companies to perform the work in more agile manner than with traditional system provisioning models. The presented work is based on first-hand experiences candidate has gained while working with the challenges of the service provision. The idea for the paper is by the candidate and the candidate was the main author of the conference paper.

Publication V analyzed the customer communication aspect of the SaaS model, proposing more effective ways and multiple channels to get feedback and to keep the customers informed as well as possible. The SaaS model provides means to understand the end-users by utilizing end-user monitoring

and analyzing the user's behavior in the operative environment in a systematic way. Moving away from formal communication into close, detailed monitoring of the end-user actions reveals the service provider aspects of user needs that otherwise would stay hidden. The presented work is based both on first-hand experiences candidate has gained while working with the customer care aspects of software system provided as a service as well as research done in the university project. The paper stems from candidate's experiences, enriched with the other authors' views and insights. The candidate was the main author of the conference paper.

Publication VI is related to the Publication V and highlights the problems we are facing when trying to communicate the information system needs as well as the problems we are having when trying to ensure that the communication has happened successfully. As the requirements form the basis on which the information systems are build, and the requirements are often communicated in written, condensed form, the shortcomings in the ways of communication result easily in misunderstandings or misinterpretations. Consequently, these misunderstandings lead to major unexpected additional costs, due to required extra work and delayed deliveries. With the SaaS model, the shortcomings become more prominent, as the shorter and more direct feedback loops are able to reveal the weaknesses in the communication more effectively. The study was published as a book chapter, of which the candidate was the main author.

Publication VII studied the complexities encountered in modern large-scale information systems and the often unforeseen consequences that these complexities introduce to the ways systems are designed, implemented, tested, deployed and operated. Publication lists important issues related to the quality assurance (QA) that might not be evident for companies who try to enter the service provision scheme. The presented problems and solutions to them are based on real-life project in which a mission critical system is provided as a service. The candidate was the main author of the conference paper, which was created through a shepherding process.

Publication VIII outlines the challenges commonly encountered in mission critical information system projects based on public tendering. The publication catalogues the reasons behind the challenges and addresses each of the challenges by providing possible solutions to the problems and mitigations to the risks involved. The publication also revisits the potential advantages an information system project could gain by employing the SaaS model for

the system delivery. The candidate has created the categorization of the challenges, based again on the real-life project experiences of his own. The candidate was the main author of the conference paper.

Abbreviations

CEM Customer Experience Management, also CXM: a collection of processes a company uses to track, oversee and organize every interaction between a customer and the organization throughout the customer lifecycle.

COTS Commercial Off-The-Shelf or Commercially available Off-The-Shelf products: packaged solutions which are then adapted to satisfy the needs of the purchasing organization, rather than the commissioning of custom-made, or bespoke, solutions.

DR Disaster Recovery: an area of security planning that aims to protect an organization from the effects of significant negative events.

DT Digital Transformation: not limited just to digital technology, but reflects also the fact that technology, which is digital, allows people to solve their traditional problems in different ways.

FOC Full Operational Capability: the completion of a development effort.

IaaS Infrastructure-as-a-Service: a cloud computing model in which some third-party provider hosts computing infrastructure which platforms can be built on.

IOC Initial Operating Capability: the state achieved when a capability is available in its minimum usefully deployable form. As IOC, the term is often used in government or military procurement.

IT Information Technology.

ITIL Information Technology Infrastructure Library: a set of detailed practices for IT service management that focuses on aligning IT services with the needs of business.

LOC Lines of Code.

MVA Minimum Viable Architecture: a set of design principles for architectural design, where prototyping, building just enough, and scaling are emphasized.

OOBE Out-Of-Box Experience: the experience a user has when preparing to first use a new product or a service.

OOBF Out-Of-Box Failure: the perceived failure of a product or a service that occurs immediately upon first usage.

PaaS Platform-as-a-Service: a cloud computing model in which some third-party provider hosts a platform on which applications can be run.

PDF Portable Document Format.

QA Quality Assurance.

RFI Request for Information: a standard business process whose purpose is to collect written information about the capabilities of various suppliers.

RFP Request for Proposal: a document that solicits proposal, often made through a bidding process, to potential suppliers to submit business proposals.

RFQ Request for Quotation: a standard business process whose purpose is to invite suppliers into a bidding process to bid on specific products or services.

ROI Return on Investment: a ratio between the net profit and cost of investment resulting from an investment of some resources.

RQ Research Question.

SaaS Software-as-a-Service: a cloud computing model in which some third-party provider hosts applications that customers of the service can access through the network. Typically, SaaS in build on top of the IaaS and PaaS models.

SDLC Software Development Lifecycle: a term used in software engineering to describe a process for planning, creating, testing, and deploying an information system.

SLA Service Level Agreement: a commitment between a service provider and a client.

SLI Service Level Indicator: a carefully defined quantitative measure of some aspect of the level of service that is provided.

SLO Service Level Objective: a target for performance of a particular metric (e.g., available 99.9% of the time).

UCD User Centered Design: a framework of processes in which usability goals, user characteristics, environment, tasks and workflow of a product, service or process are given extensive attention at each stage of the design process.

UI User Interface: the space where interactions between humans and machines occur.

UX User eXperience: refers to a person's emotions and attitudes about using a particular product, system or service.

Chapter 1

Introduction

Digital transformation (DT) that is happening with ever increasing speed and intensity everywhere around us (Bygstad, Aanby, & Iden, 2017), has brought along ever larger and more complex software systems. As a consequence, the complex software systems take care of the many mission critical needs of modern societies and organizations and we are more dependent on software than we have ever been in the past. Therefore, it would be natural to think that all the software running around us has been thoroughly tested before taken into use and developed in disciplined ways to ensure flawless operation in all circumstances (Fuchs & Hess, 2018).

Related to the digital transformation, the pace in which software and software-centric information systems need to be specified, developed, modified and deployed to use, has been forced to increase, concurrently with the strict requirements on the cost-efficiency and easy maintainability and modifiability. This evolution has resulted in systems with unforeseen number of features and lines of code (LOC). It has been reported that for instance in Google's code repository, there is some 2 billion LOC, in 9 million source files to which the company's 25000 developers commit 15000 changes per day (IT World, 2015). When systems are built in such sizes and with high speeds, one would hope that the need for which the system is build, is not a critical one and we can therefore fix the faults based on the failures encountered, without any dire consequences. However, if the system is a critical one, one would then hope that there exists fast and effective enough feedback mechanisms, in form of tests, reviews and audits, for instance, that correct at least the most severe faults before the system is deployed to production. Hope, however, cannot be a strategy and luck cannot be a factor when dealing with critical systems.

Despite the increased complexity and the speed in which the information systems are deployed into use, the development of software-centric information systems has still remained an inherently human, intellectual activity (Fagerholm, 2015). The need for which the software system is a solution for is specified and defined by human beings with their known limited, suboptimal capabilities to comprehend complex issues and communicate effectively (Wiio, 1978). Moreover, similar human beings with at least as limited capabilities try to design and implement the system based on the complex, but inevitably unexhaustive specifications. Thus, the increasing complexity of the developed information systems has not been much accompanied by a focus on the human cognitive skills or social sides of the software development, which should include the emphasis on the importance of right knowledge and skills of the people involved (Calefato, Iaffaldano, Lanubile, & Vasilescu, 2018). The needed skills, in the context of complex and rapidly changing environment, are not so called hard skills, like process and technology skills, but primarily "soft" ones, which call for a reformed understanding on what characteristics define a great modern software professional.

While dealing with ever more complex systems and along with the complexity, often long-lasting projects, the research suggests that we, as human beings, are not very good at handling complex things or keeping the focus when the goal is set far in the future (Dzubak, 2008; Konig, Buhner, & Murling, 2005). In addition, the importance of the motivation has a large effect on the effectiveness of human efforts (Senecal, Koestner, & Vallerand, 1995), which, again, is a challenge with complex and long-lasting projects. Consequently, these facts call for means to keep the perceived complexity at a given time as low as possible and effective ways to divide the work to be done into small pieces, the goals of which are always in the near future.

1.1 Context and Motivation

Large-scale information systems are often procured through long-lasting and energy consuming processes, the **public tenders**. These processes, at least by anyone observing them from outside, seem slow, as they in many cases last several years. One reason behind the apparent slowness must be that when one is to procure something that costs an extensive amount of money, take considerable amount of man-years to complete and may even be mission-critical, one should be extremely careful in what to do and in which order,

while thinking everything through in a pedantic way. Thus, when we want to be careful, we tend not do anything fast (Kahneman & Egan, 2011).

At the same time, information system projects frequently fail (Krishna et al., 2018). Depending upon which study one wants to refer to, the failure rate of large software projects is reported as being between 50-80% (Dwivedi et al., 2015; Ismail, 2018; Florentine, 2016). However, because of the natural tendency of human behavior to hide the bad news or at least not be fully honest in assessing a project's state, the real percentage may be much higher than the reported one. On top of that, what we consider as a failed project depends a lot on our relation to it; are we with the project or outside it—being on time, on budget and on target cannot always be objectively measured (De Wit, 1988).

While in the cases that form the basis for this thesis the buying organization has been a public one, the private companies seem to behave in a relatively similar manner. Therefore, the topic of this thesis and the results obtained are not only limited to the public sector. However, there is an important difference between the public and private sectors when it comes to measuring the value gained from the procured system. In the private sector, where the stakeholders typically try and have to measure the return-on-investment (ROI) carefully and ensure that it is at an acceptable level for any procurement, the procurement processes are required to be swift and result in something concrete or get cancelled within relatively short periods of time. This is not always the case for the public-sector procurements, where the lost time and unexpected delays do not necessarily lead to unsatisfied stakeholders, as the ROI is hard, if not even impossible, to measure (Klijn & Teisman, 2003).

Independent of the domain, the information system procurement procedure starts by a specification phase, in which the buying organization spends a considerable time to produce a set of documents by which the system to be purchased is described. The short-comings of this thorough up-front specification process are numerous. First, for any larger scale system, with several hundreds of requirements, both functional and non-functional, the completeness and consistency of the set of requirements is extremely hard to achieve. Secondly, even within the group of people who have created the set of requirements describing the needs that must be fulfilled, the interpretation what a specific written requirement actually means, may vary. Consequently, when trying to communicate that complex set of requirements to any external party, like to a company who is awarded to build the needed system, mis-

understandings and misinterpretations are inevitable (Lutz, 1993; Zin & Pa, 2009).

By definition, an information system is **mission critical** if it is extremely important or necessary for an organization that the system operates successfully (Techopedia, 2018). The failure of such a system can lead to major economic losses or even to life-threating situations[1]. As the digital transformation progresses, more and more of the mission critical activities are handled by software-centric systems, that are procured by the same procurement procedures than any non-critical systems. This development may lead also to situations where a system that has earlier been a non-critical one, turns into a critical one as it is extended with some new mission-critical features.

As such, a careful and disciplined procurement procedure seems a perfect fit for the mission critical information systems. However, when taking into account the short-comings in the specification and the challenges in the communication of the actual needs, while taking into account the speed with which the modern system environment changes, a slow, strenuous and inflexible process may lead to solutions that would not fulfill the actual need, would not be safe to use and would probably be outdated already before taken into operative use.

To get better value for the information system investments, many organizations are looking into having the services they need provided as a service. The **software-as-a-service (SaaS)** paradigm opens a whole new set of possibilities to specify and develop the system, to give and get feedback and co-operate on the system development effort (Benlian & Hess, 2011).

The main motivation for this thesis stems from the frustration experienced when involved with large-scale information system procurement processes and undergoing some the inherent short-comings first-hand. The suboptimal outcomes, the insufficient quality and the failure to meet the real needs of the end user seem to take place all too often (Goatham, 2017; Fernández et al., 2017; Basili & Perricone, 1984). At the same time, considering the huge amount of knowledge already gathered and reported on how to build great software systems, it is quite obvious that we in the software engineering domain, be that in academia or in the industry, certainly could (and should) do a lot better.

The scope of the thesis is the mission critical information systems as with them the deficiencies of the public tendering process are most prominent and

[1]The notion of mission criticality is further addressed in Section 2.2

with them the risks involved are the biggest. With mission critical systems, also the proposed enhancements will have the biggest effect. The results are, nevertheless, for major part fully applicable to any information system procurement process, either in the public or private sector, and procured basically by any process.

1.2 Methodology and Research Questions

The research described in this dissertation has followed the principles of qualitative research (Sale, Lohfeld, & Brazil, 2002), although at the time the observations and learnings were made, a practical, rather than a research mindset, was assumed. This work started with a very simple question, asking *"what we can do better to help our customers to get their actual needs fulfilled?"*, which led, after some elaboration and analysis, to the writing of the publications, one by one, and finally to the compilation of this dissertation.

The driving force behind the publications and this thesis has been the urge to better understand the root causes for the problems often encountered in the communication and understanding of the viewpoints of the party on "the other side of the table". In the cases this thesis is based on, the other party has been a contracting authority or a system owner representing a public organization, while the viewpoint in the publications represent those on the system vendor side, i.e., the industry. The desire to look at, describe and understand the experiences gained and document the ideas for improvement, along with the assumptions, beliefs and values those ideas are based on, created the publications in this thesis.

This thesis is a collection of case studies, which focuses on an information system at the center of attention and aims to generalize the observations made over several systems (Stake, 2013; Gustafsson, 2017). An important thing to consider is thus the context (Yin, 2017). As this study includes multiple cases, we need to understand the differences and the similarities between the cases (Baxter & Jack, 2008; Stake, 1995). As all the cases the publications report on have been information system development projects that are based on public tenders, the similarities between the cases are evident. Multiple case studies can be used to either augur contrasting results for expected reasons or either augur similar results in the studies (Yin, 2017). In this way it can be clarified whether the findings are actually valuable or not and to what extent the experiences can be generalized (Eisenhardt, 1991).

Furthermore, one can argue also that this thesis has the similar goals as the design science research (Aken, 2004). Design research involves the analysis of the use and performance of designed artifacts, i.e., the information systems itself with all its services, to understand, explain and to improve on the behavior of aspects of information systems (Vaishnavi & Kuechler, 2004). One of the goals of this thesis certainly is to develop knowledge that all the professionals in procuring organizations, in the academia and in the software industry, can use to design better solutions for the problems at hand and, at the same time, with the knowledge to avoid some of the common pitfalls. Although design research is a powerful tool for addressing the needs to gain more knowledge, the design research brings with it serious challenges, which are addressed in Section 5.4.

The main research problem is quite generic, being "What are the main practical implications when a company is transforming to service-based model with the help of SaaS paradigm?". This research problem is supported by five research questions, addressing various aspects of or viewpoints to the domain under research, i.e., that of public tenders, mission critical information systems and the SaaS paradigm. In short, the five questions are

RQ1 What aspects need to be considered and emphasized in the RFQ tender process to support possible service provision and agile development?

RQ2 How the emphasis on different aspects differs between projects developing critical and non-critical systems?

RQ3 How mission criticality should be approached at the system specification and public tendering phase?

RQ4 What advantages the SaaS paradigm brings to the mission critical system domain?

RQ5 If a mission critical system is provided as a service, what points-of-view and requirements need to be focused on?

The research questions will be covered in more detail in Section 3.2 and answered in Section 5.1.

1.3 Overview of Results

The thesis contributes towards novel ways of procuring, designing, developing and providing for use large-scale information systems which, by their nature, are mission critical. The main contribution is threefold, as described briefly in the next three paragraphs.

First, the communication that takes place when large-scale information systems are specified and designed, suffers from several short-comings. The situation could be improved drastically by first admitting that we, the human beings involved in the information system procurement process, are not very good at communicating with other people and secondly, finding new effective ways to improve the communication. The enablement of various effective wide-band communication channels between the different parties involved in the information system development effort is a pre-requisite. Encouraging people to use them and ensuring that communication takes place as should, should be everyone's responsibility[2]. In many cases it is just as simple as to ask instead of to assume and if we have simple-to-use channels that we can use to ask questions, we tend to do that more often. This contribution is labeled as **"Bridging the Communication Gap"**.

Second, ways by which the critical information systems are developed should be disciplined processes which enable and encourage communication, provide reliable ways to assess the system quality and allow constant and frequent feedback. Feedback need to be given by the system's users and received by the system developers with ease, without bottle-necks, extensive delays or the broken phone effect. At the same time, while disciplined, the processes also need to be flexible enough to accommodate for the particular needs of the people and the project's context. There is no one and only way to act and behave, as the needs evolve over time with the increased understanding of the problem at hand. For these reason, the SaaS model is helpful in this as it enables the incremental development while providing effective feedback channels. This contribution is called **"Providing Efficient Development Process"**.

Third, as the SaaS model is seen as an attractive way to solve at least some of the problems often encountered in software-centric information system projects, there is a need for a transformation of a company from a software vendor to service provider. Unfortunately, this transformation does hap-

[2]Note that here communication includes also ensuring that the message is delivered and interpreted correctly.

pen neither overnight nor easily. A new service-oriented mindset and a whole new set of skills, practices and tools are needed to provide a software-based service effectively and to keep the users and customers satisfied. It may not be too far-fetched to claim that the change needed resembles the transformation from a manufacturing to service organization, where the most prominent differences are the tangibility of the output, production for inventory versus production on demand, COTS versus customer-specific production and the labor-intensive versus the automated operations. This contribution has the title **"Transformation to Service Provider"**.

1.4 Outline of the Dissertation

The thesis comprises two main parts, a summary and an appendix containing the eight original publications. The summary consists of six chapters including this one, the chapter Introduction.

Chapter 2, lays out the background of the thesis by presenting the inner workings of the public tendering process as they typically exist in the tenders on the public sector. In addition, Chapter 2 presents the concept of and challenges related to the mission critical information systems as well as describes the characteristics of the software-as-a-service (SaaS) paradigm. The SaaS paradigm is seen as a potential solution to many of the challenges encountered in the public tender-based information system projects.

Chapter 3 describes the research approach, including a short discussion on the applied research method and a description of the research process of this study. Chapter 3 also opens the background of the research questions and presents the relationship between the research questions to the published articles.

Chapter 4 gives an overview of each of the publications included in the thesis, summarizing the purpose of each publication, the main results and a description of the publication's relation to the whole. In the end of Chapter 4 a brief synthesis of the key findings is outlined, and the guidelines derived from the findings in the publications are listed.

Chapter 5 revisits the research questions and discusses the contributions and implications of the obtained results to theory and practice. The contributions are covered by the discussion of the various reasons that cause the problems in the public tender projects. In Chapter 5, the limitations and

validity of this research are also addressed, as well as some possible related future work and interesting associated research topics.

Finally, Chapter 6 summarizes the main findings. After the final chapter, the appendix comprises eight scientific publications in chronological order, the ones that construct this compilation.

In this thesis, the two parties of a service provision setup, i.e., the customer and the vendor or the buyer and the seller, are referred by several names, depending on their role in the particular context or at a particular phase in the procurement process. The roles, the responsibilities and characteristics may vary, but what remains the same is that these two parties are the ones that need to communicate, negotiate, co-operate with and understand each other. Although care has been taken to refer to the parties always with the same names, it should be noted that there may be, especially in the publications, some names used that are not fully aligned with the rest of the thesis. Therefore, to bring some clarity to the setup, the names that are used for the two parties, partly intermingled, when the parties act in their roles are listed below:

- **Customer:** client, (end-)user organization, buyer, system owner, tenderer.
- **Vendor:** tenderer, bidder, service provider, software company, seller.

The term "mission critical" is used throughout the thesis to refer to systems which must be highly reliable and retain the reliability as they evolve. An important aspect of mission critical systems is that they also need to be fail-operational, i.e., required to operate not only in nominal conditions, but also in degraded situations when some parts of the systems or some systems the critical system depends on, are not working properly (Bozzano & Villafiorita, 2010). Depending on the specific context, the same kind of system may be described as safety critical, business critical or even security critical, but despite of the term used, the guidelines outlined in this thesis apply to any critical system.

Chapter 2

Background and Related Work

The purpose of this chapter is to present the general view to the research domain and describe the central concepts which form the background of this thesis. The concepts, and how the concepts are treated here, are based on the experiences gathered by the participation to several software projects in the industry during the last decade.

The projects have all been on the public sector, procured based on the applicable laws governing the public procurement and been related to mission critical information systems. Thus, the presentation aims at describing the concepts from a practical point of view, as seen from the software supplier side, and the described challenges stem from authentic cases. The definitions used have been aligned to match as well as possible to those used in the related scientific research and literature.

In this thesis, the focus is placed on three key elements of mission critical system procurement and provisioning:

- **Public tendering process**, i.e., typically the RFI/RFQ process, which plays the key role at the initiation of the procurement program of an information system. The tendering process affects in a major way how any later specification, development, testing, deployment and maintenance phases will and can proceed (Section 2.1);

- **Mission criticality**, i.e., the special characteristics of certain critical information system, which considerably affect the way the system needs to be specified, designed, developed, tested and kept running (Section 2.2); and

- **SaaS paradigm**, i.e., a software licensing and delivery model in which software is licensed on a subscription basis and is centrally hosted. In this thesis SaaS model's role is seen as the enabler to develop and maintain mission critical systems in a way that drastically mitigates the major risks involved in the information system procurement process, if only applied appropriately (Section 2.3).

The focus of this thesis can be simplified as being in the intersection of the three key elements listed above, as illustrated by Figure 2.1. As the figure suggests, there are mission critical system that are procured by public tenders without any relation to the aspects of the SaaS, as well as system that are SaaS based and procured by public tenders but are not critical in any way.

As of the time being, there seems to be relative few operative information systems that fall into the category represented by the intersection of the three focus areas. However, during the recent years many organizations have seriously started looking for ways to outsource their IT system maintenance resulting in the success of the IaaS, PaaS and SaaS paradigms (M. J. Kavis, 2014) and as the shortcomings of the public tendering process get more and more attention, this focus area will undoubtedly gain more interest in the near future.

2.1 Public Tendering Processes

The mission critical information system projects the publications in this thesis are based on, have all been based on a request for information proposal (RFP) or request for quotation (RFQ) process. Both of these processes aim at describing a problem or a need the procuring organization or its customer has and outlining a project with which to solve the problem of fulfill the need (Mhay & Coburn, 2018). In many cases the difference between the RFP and the RFQ processes is in the eye of the beholder. However, the principal difference between the RFP and the RFQ is that commonly the RFP is used when the customer concentrates on describing the problem they have and want to hear in which ways it should be solved, whereas the RFQ is used when the stakeholders know what they want but need information on how the vendors would meet the requirements and how much it will cost (Ferriere, 2018). The goal of both the RFP and RFQ is that the potential suppliers are well informed on the need and get all the relevant and required information

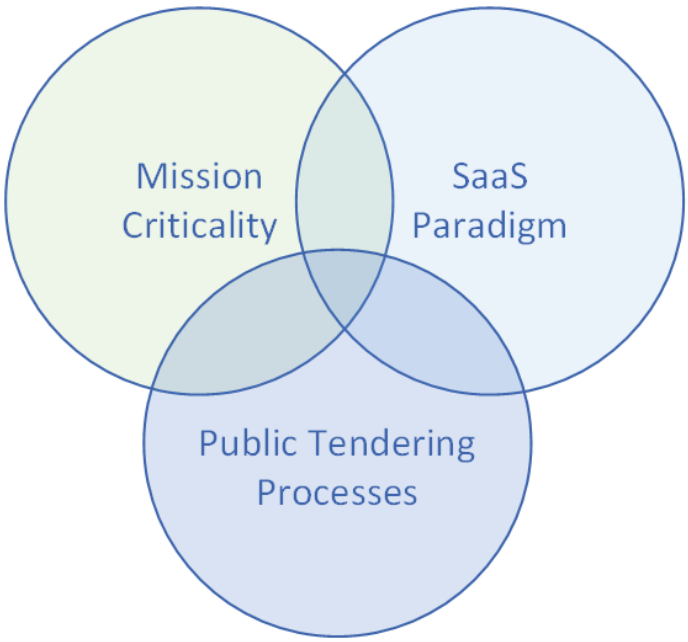

Figure 2.1: The key elements of this thesis: public tendering processes, mission criticality and SaaS paradigm.

on the need in a way that enables the potential suppliers to respond factually to the identified requirements with a solution.

Often, prior to the issuing of the RFQ or RFP to the potential providers, the standard business process of request for information (RFI) (Mhay & Coburn, 2018) is also executed. In the RFI process, the potential vendors are asked to determine what kind of services are available to meet the described needs. The RFI is typically used to be the source of information for a later issued RFP or RFQ process, thus enabling the resulting RFP or RFQ to be more realistic and precise than without the preceding RFI round. Of the projects that this thesis is based on, not all had a RFI phase but in the cases the RFI phase took place, it was used to refine and scope the requirements for the upcoming RFP/RFQ, and not just for the technical solution, but also to get a rough estimate on the price of the described solution.

The definitions of the processes (Mhay & Coburn, 2018) related to the procurement procedures are as follows:

- **Request for Information (RFI)**: An open enquiry that spans the market seeking broad data and understanding.
- **Request for Proposal (RFP)**: A business requirements-based request for specific solutions to the potential suppliers. Sometimes based on a prior RFI.
- **Request for Quotation (RFQ)**: An opportunity for potential suppliers to competitively price the final chosen solution or solutions.
- **Request for Tender (RFT)**: An opportunity for potential suppliers to submit an offer to supply goods or services against a detailed tender.

It is conventional to talk about RFQ process only, but it seems, based on the definitions above, that we should in many cases rather talk about RFP or a combined RFP/RFQ process, as typically the RFQ documents do not just ask for the solution description but also the detailed price of the solution and either commitment to a suggested delivery schedule or a proposal for one. Generally speaking, one could say that an RFQ is in question when the procuring organization knows exactly what it needs, and the organization is only asking for the price of the procured thing. Respectively, an RFP should be used when the procuring organization has a problem or a need, and it is asking potential suppliers to come up with solutions and identify

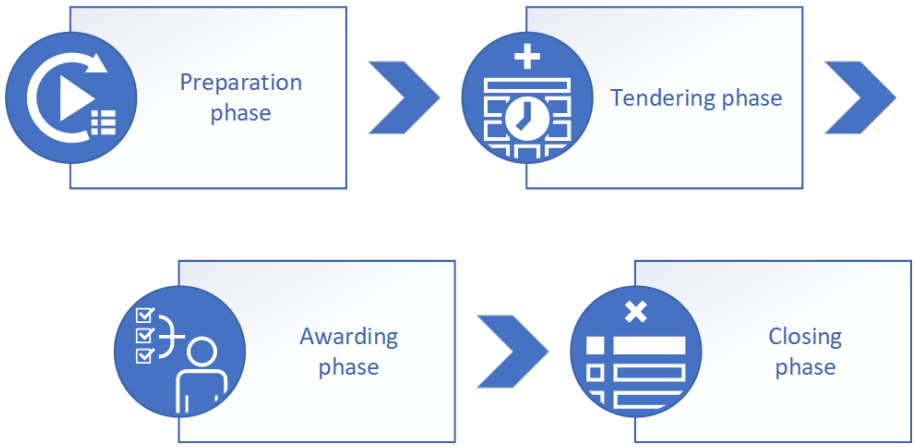

Figure 2.2: The tendering process phases.

the accompanying costs (Tomingas, 2018). On the other hand, many of the large-scale public tenders may also be more RFT-like and RFQs, but it seems that the terms are used intermingled. Consequently, and to cause as little misinterpretations as possible, in the following the term "tendering process" is used to refer to the procurement process, irrespective of the actual process details.

2.1.1 Phases of the Tendering Process

Figure 2.2 shows the phases of a typical tendering process. In short, the descriptions of the phases are, in the same order as in the figure, the following (Vaes, 2014):

- **Preparation phase**, in which the type and the characteristics of the tendering process to be run need to be decided and all the required documentation related to the process need to be prepared. For the success of the procurement process, this phase is an extremely important one and thus most of the time is consumed in this phase.
- **Tendering phase**, in which the tendering process is already underway and in which the most important task the procuring organization has is to treat the potential system suppliers equally and answer any posed questions in an appropriate and consistent way. During this phase, it happens often that some of the original requirements are further

clarified, modified or even discarded, as when the potential suppliers try to better understand the needs, they commonly make clarifying questions which affect the target of the process.

- **Awarding phase**, in which the procuring organization has received the proposals, assesses the proposed solutions and finally makes the decision on with which supplier candidate the process shall continue.

- **Closing phase**, in which the tendering process is ended and after which the consequent development project or equivalent is allowed to get started.

The tendering processes—disregarding if their domain is public or private sector and almost regardless of the target and characteristics of the to be developed system itself—have the habit of starting with an extensive preparation phase in which a comprehensive specification of the system to be developed is created. In the preparation phase, the procuring organization, often together with a hired group of consultants, spends a considerable period of time trying to determine the exact needs of the future system owner as precisely as possible. The future system owner and the users of the future system may or may not be from the procuring organization, as often in the public tenders a separate organization takes care of the procurement process on behalf of the "customer" organization.

Typically, the preparation phase results in a large number of documents, including a document which contains hundreds or even thousands of requirements, written carefully in, e.g., IEEE 830 requirement format (IEEE Computer Society, 1998). These requirements are reviewed many times, repeatedly, during the preparation phase, and finally, after several refinements and iterations, the requirements are accepted and approved by all the stakeholders representing the procuring organization. Thus, the core or the main artifact of the tendering process is a requirement document which is typically a Microsoft Excel Workbook, or worse, a similar looking sheet in a PDF format, not suited for effective analysis by spreadsheet tools or alike. In any case, the requirement document contains on its sheets or pages the crown jewel of the preparation phase: the requirements. In some cases, the requirements are short, written in a good understandable language and categorized and grouped intuitively. In some other cases, they are not.

The preparation of the tendering process often takes a considerable amount of time and a considerable amount of people is typically involved in the prepa-

ration effort (Rogerson, 1994). The length of the preparation phase on the public-sector tenders can be several years, and by rough estimate, this size of the effort can cumulate easily up to tens of man-years.

When the process enters the next phase, namely the tendering phase, the potential supplier candidates are informed on the existence of the tender either directly or through some public procurement journal. Despite of the relatively extensive effort in the preparation phase, the procuring organization responsible for the tender expects typically swift responses from the potential suppliers. Hundreds of requirements describing the future information system are given as a request to the potential providers who then have no other option than to lay out a technical solution fulfilling all or at least the most essential requirements in a hurry. Furthermore, the potential supplier is typically requested to give a binding price for the whole system, as well as describe, for example, project, quality and resourcing plans that fit, in many cases almost magically, the timetable requested or suggested in the tender documents by the procuring organization.

For the potential system providers, the details of the preparation phase of the tendering process remain mostly unknown. This is due to the common reckoning that by giving any of the potential system providers information on the work-in-progress or on the background of the specifications, would introduce an unfair situation and cause bias when the potential suppliers propose their solutions based on the tender.

The tendering phase is followed by a selection process, i.e., the awarding phase, where the customer organization tries to find out which one of the potential system providers is the fittest to deliver the requested system. The selection needs to be done among the successfully delivered proposals, i.e., the ones that were delivered before the deadline in the requested form and fulfilling all the mandatory requirements and other criteria. The selection is often based on some not-so-clear algorithm, details of which often remain partly unknown to the supplier candidates. In the tender documentation, the criteria of the selection of the best candidate is described, but as it is typically a combination of the price and the quality attributes of the proposal, it is almost impossible for anyone who is an outsider to the procurement organization to determine, what aspects of the proposals resulted in the selection or deselection of that particular candidate. Although the procuring organizations may not admit it, from an objective point of view, the award does not necessarily go to the vendor that has the best proposal, but to the one that has made the best impression by the delivered documents or in the

possible negotiation meetings. One should not underestimate the power of politics here, either. Often the best candidate is the one who represents an organization known to be a reliable partner or known to have capabilities to deliver similar systems.

In any case, and as a result of any procedure to determine the best candidate, at the end of the awarding phase one of the bidders is selected based on the price, on the assumed quality of the proposal, or on some weighted combination of the two. In practice, naturally, also other factors affect the end result of the selection process, like aspects related to politics or lobbying. In such cases the only problem the procuring organization has is that how to justify the selection they want to make without putting the process at risk of legal consequences.

After the awarding phase, the project is initiated, and the procuring process is closed. The closing means often that the responsibility of whatever was agreed is transferred from the procuring organization to a newly established project organization or some other organization taking the responsibilities of the customer from there on. After the closure of the tendering process itself, the development project is initiated, and time is spent on the further specification, design, development, testing, releasing and hopefully finally on the deployment of the system to its operative environment. The delivery of such a system to operative use happens, based on the experiences that form the background of this thesis as well as on the numerous reports on the failed software projects, with varying success rates.

2.1.2 On the Role of Requirements

The defined, refined, iterated, and finally accepted set of requirements, often in the form of a MS Excel Workbook or a PDF document, serves typically as the most valuable artifact in the tendering process. The requirements determine, at least for the major part, the costs stated by the proposals, the timeline within which the potential suppliers think the system can be developed and delivered and the quality to which the supplying party must aim to when developing the system. Therefore, any misunderstanding or misinterpretation potentially cause budget overruns, delays and unsatisfying perceived quality.

For the subsequent phases in the SDLC of the system, particularly in the design, development and testing phases, the requirements are treated in various occasions as the final "truth" to which one can safely refer to when

discussing system features or characteristics. The requirements are traced back to when determining is some feature or quality actually necessary in its designed format or at all. Furthermore, only issues that are not against the specifications, i.e., are not compliant with the requirements, are regarded as defects. By the same set of requirements, the customer organization and future system users assess the progress and the quality of the system during its development and deployment.

Although the software requirement specification document should list sufficient and necessary requirements for the development project, anyone involved with information system design and development can verify that the fixed set of requirements is already a problem in itself. The requirement set, which typically written in a way respecting the IEEE 830 standard, often cannot live up to the high expectations of being the solid basis for an agreement between the customer and the supplier on how the system should function and what quality it should possess. The requirements have, for example, subjective language, ambiguous adverbs and adjectives, superlatives and negative statements (Davis et al., 1993), all of which make the interpretation of the requirements hard and subjective. Thus, it is not a surprise that low-quality requirements have been found to be the most common reason for the failure of software projects (Hofmann & Lehner, 2001; Zowghi & Nurmuliani, 2002).

With the agile ways of working (Martin, 2002), it has been slowly admitted that if the software requirements are not adjusted and refined in a continuous manner throughout the SDLC, the implementation of a software system will lead to end results that do not fulfill the actual or original needs of the system owner and system users (Dybå & Dingsøyr, 2008). Moreover, even if one could assume that the requirements itself were at some point of the system procurement or development process complete, mutually exclusive and collectively exhaustive, and yet being able to reflect the true needs of the system owner, the form and the language used in the writing of the requirements leaves typically a lot of freedom to the interpretation of the requirement. This problem is emphasized when the project is large, related to multiple organizations and lasting for a long time, as the same requirements can be interpreted differently in different phases of the project or when they are interpreted by different people, with different background and history, and playing different roles in the development effort (Wiegers & Beatty, 2013). Moreover, ambiguities in the requirements leave the space for either party in project, the customer or the system provider, to interpret the wordings in a favorable way to themselves (Schmidt, Lyytinen, Keil, & Cule, 2001). Such

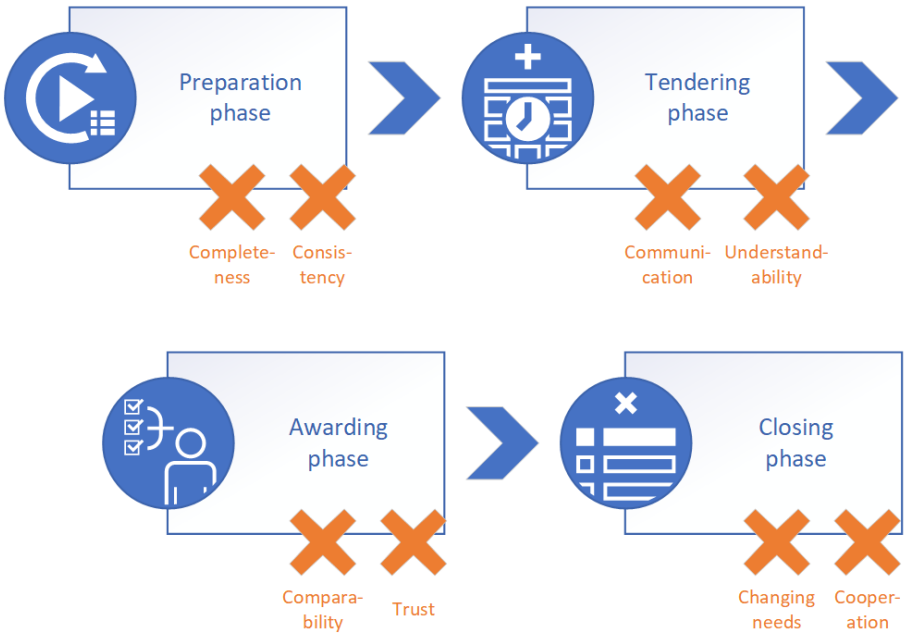

Figure 2.3: The tendering process phases with main pain-points highlighted.

interpretations result easily and fast in mistrust, which does not increase the probability of success for any project.

2.1.3 Pain points in the tendering process

To summarize the difficulties encountered when executing the tendering procedures, Figure 2.3 depicts the main pain points in the process. These pain points are also addressed in the Publications I, II, VI, VII and VIII.

Preparation phase problems

In the Preparation phase, the biggest challenge is to create documentation in which the requirements are complete and consistent and able to communicate the original ideas and needs to be potential suppliers. For any larger scale system, the completeness and consistency are next to impossible to verify, due to limitations of a human mind to grasp complex and multifaceted issues and to perform advanced high-level decision-making (Koechlin & Hyafil, 2007). It

is commonplace that the number of individual requirements is in the range of 500-1000.

The longer the preparation phase is, the more prominent the problems with the requirements become (Buzby, Gerstenfeld, Voss, & Zeng, 2002). On one hand, it is obvious that as time passes, many of the requirements become outdated. On the other hand, the people who have themselves created and refined the requirements, want to keep them as "already agreed" requirements and are reluctant to revisit these. This is natural behavior as one would like to see progress in the specification work and therefore consider some parts of the work as "done" and not to be reassessed. The written requirements are, however, rarely independent of each other to such degree that this kind of behavior is acceptable.

In the preparation phase, the team specifying the system to be procured also sets some goals for the schedule of the system development and target dates for the operational deployment. At the same time, there always exists also a budget within which the total price of the procured system should stay. The problem with this is, naturally, that how well the procuring organization is able to set restrictions on the budget and on the delivery schedule while specifying, and typically also extending, the scope (Lauesen, 2004). When setting the schedule, one should keep in mind the Hofstadter's Law: "It always takes longer than you expect, even when you take into account Hofstadter's Law" (Hofstadter, 1980).

Tendering phase problems

In the Tendering phase, the communication is still a challenge. The readability, clearness and even the used terminology may cause serious problems in the understanding and interpretation of what the requirements and other tendering documentation were actually trying to communicate (Lauesen, 2002). This challenge is not made any easier by the strict deadlines by which the responses to the tenders are often required to be delivered.

The supplier candidates are typically given a chance to ask questions and request clarifications about any of the aspects in the tender documents, which, at first, sounds like a good and fair idea. To keep the tendering procedure as equal as possible, all the parties in the process naturally need to be informed about all the questions asked and provided with all the answers and clarifications given. This setup poses again several challenges, the main three being:

- The same difficulties in the communication encountered with the preparation of the requirements, apply to the questions and answers as well. The information should be delivered in a form that is as hard as possible to misinterpret or misunderstand. If the answers given raise new questions, as they often do, the following questions can remain unanswered due to strict schedules.

- The questions are sometimes seen as critique to the tender contents and the answers to the questions are defensive and even dismissive. This is understandable based on the big effort already invested in the requirements and based on the fact that the people who have written the requirements may not be able to put themselves in the position of the supplier candidate. The former is already an expert in the area of the future information system, while the latter may not know next to nothing about the domain or the special terminology used in that field or about the original need itself.

- As all the supplier candidates know well that also all the other candidates will receive the questions and the answers, the candidates think twice what questions and in what form to raise. Some of the candidates may not even have realized that there were some issues that needed clarification. Often the candidates choose not to raise a question to avoid revealing some aspect of the tender that is considered to be an advantage for the candidate.

Awarding phase problems

In the Awarding phase, objective measures by which the best candidate should be selected are hard to determine. Furthermore, even if the solution descriptions and the information on the price of one of the potential suppliers seems to prevail the others, can the procuring organization trust blindly the material provided?

The awarding phase is particularly problematic, as now money comes into focus (Rogerson, 1994). There are many variations to the sayings "You never get something for nothing" or "Good isn't cheap, and cheap isn't good", and in the cases investigated within this thesis, these statements do hold the truth to large extent. In an effort to drive cost to the lowest value possible while trying to maximize the value, the silent assumption is that the structured pricing protocols of the RFQs are able to give the tendering organization

objectively comparable responses. However, in practice RFQs rarely offer fair comparisons for the potential vendors' ability to create the system and to take care that it can be deployed into operation. It is just naive to believe so.

There is also a big difference between the value and the price, though both are defined through the perceptions of the tenderer and potential supplier or vendor (Brown, Ashleigh, Riley, & Shaw, 2001). The most important distinction between price and value is the fact that price is arbitrary, and value is fundamental. For the tenderer or client, to extract value from the procured system, the expectation of the value must be defined, from which a ROI can be calculated. On the other hand, the vendor with the lowest price may not be the most valuable to their client[1].

Some potential mistakes that can take place in the Awarding phase are (Thai, 2001; Knight, Harland, & Telgen, 2012):

1. The tendering organization is under pressure budget-wise and as the price is one important criterion to choose the supplier by, the organization is tempted to select a provider with the lowest price without proper consideration of the risks involved and the cost of the consequences if the risks actualize.

2. The tendering organization has not done enough research to conclude what is a realistic price for the specified system and therefore is not able to be suspicious about proposals with an exceptional price.

3. The tendering organization has not requested any evidence that the potential supplier is actually able to perform the work described in the response to the proposal. To determine if a proposal is credible, extra effort is needed, as the credibility cannot be assessed by looking at the response material solely. Some other evidence, like completed works as references or working prototypes should be required.

4. The potential supplier overestimates their ability and is not aware of the real costs involved in the development effort of the system requested.

5. The tendering organization underestimates the costs needed to support the selected supplier. Even though the price of a certain supplier can-

[1]Note the difference between cost and price: cost is typically the expense incurred for a service being sold by a company. Price is the amount a customer is willing to pay for a product or service.

didate may be low, the overall lifecycle costs may favor some other candidate due to that candidate's superior competence, relations or better cultural match between the customer and vendor organizations.

Closing phase problems

Finally, in the Closing phase, the main problem is that how to transfer all the knowledge gained during the procurement process to the organization who will take care of the implementation process. Also, if and when the understanding or the possible solution has improved during the procurement phases, how the changed and changing needs should be taken into account when the development process takes over?

It seems that often the main focus in the closing phase is just to get over it, i.e., that the organization who was responsible for the procurement process can see that their job can be seen as completed and all the responsibilities set for the procurement process have been carried out.

2.1.4 On the Future of the tendering process

Even though it has been reported that in some cases the parties in the public tendering processes do not understand each other (Lauesen & Vium, 2005), the process seems to have a strong position as the default way public organizations procure the information systems they need. Whether this is because the public organizations are slow to change or because of the easiness of using habits that are established ones (like, "we have always done it this way"), is impossible to say. Nevertheless, claiming that the laws governing the public procurement do not allow the improvement of the process, is simply not true (Aukia, 2017).

Some of the reasons why the tendering process has stayed as it is, despite the inherent shortcomings, are possibly the high stakes involved. Many of the public-sector tenders are in the scale of million of euros, tens of millions euros or even more and thus are very attractive for companies that are familiar with the related domain and the tendering process itself. Many companies do know well how to play the game. The vagueness of the requirements allows also the supplier to make its own interpretations and as it is obvious that the project cannot be awarded fully objectively, there are possibilities to construct a proposal that looks good but is not fully based on facts. Thus,

to some extent, it seems to be ok for the parties in the tendering process to live in a situation where the requirements are somewhat semi-clear at best.

To alleviate the pain points intrinsic in the traditional tendering process and to increase the transparency, several concepts have been proposed in the recent years. These include, but are not limited to, the Public-private partnership, the Innovation partnership and the Reverse Auction model. Within the scope of this thesis, these models are discussed only in Subsection 5.6.3 and further analysis of the alternative models is left as future work.

The public tender projects that the publications of this thesis are about are mostly based on are Finnish procurement projects. The described tendering process is still not a national phenomenon: internationally the same procedure is utilized, and the pain-points are even more prominent by the misunderstandings fueled by the non-native speakers writing and interpreting the requirements stated in some lingua franca. The same pain-points exist, no matter what country the procuring and the potential supplier organization represent.

2.2 Mission Criticality

A mission critical system is a system that is so essential to the survival of an organization using it, that a failure of the system or even an interruption in some critical part of the system functionality impacts business operations significantly (Fowler, 2004; Techopedia, 2018). In many cases, a failure may cause a threat to human life or at very least cause significant human suffering and economic losses. Therefore, the downtime of such a system is not tolerable during the system's specified operational intervals. For public security systems, for instance, the specified operational interval is typically 24 hours per day, all year around. In the availability terminology, the requirement of being operational 24 hours a day corresponds to 100% availability level, which, for any practical system, is never truly achievable. Often mission-criticality is used to reference just to any application that must function continuously for a business to be successful. Thus, if a mission-critical system experiences even brief downtime, the negative consequences are likely to be financial. In addition to lost productivity, a mission-critical system's failure to function may also damage the business' reputation.

However, for the end user of a mission critical system, the mission criticality does not only mean that the services providing the system functionalities

are running and responding swiftly. From the end users' perspective, it is as critical that the user interface (UI) supports the user in executing her tasks and make her as effective and informed as needed or as is possible. In addition, as the systems are typically tightly integrated to other systems that are also more or less mission critical, the data flow in the system interfaces becomes critical. Thus, the mission criticality is contagious: in case the system at hand is not in itself a mission critical one, some mission critical system may be dependent on some data or functionality that the non-critical system possesses, making the non-critical also mission critical for that part.

For any information system, the quality and the positive user experience (UX) are important (Ferreira, Sharp, & Robinson, 2010). When mission critical systems are in the question, the importance of the quality and UX raise into a new level: the user must not just feel good about the system she is using, but to be able to trust that the system behaves as it should and guides and instructs the user to act in ways that are best suited for the current situation. Although the users of the mission critical systems are typically professional and trained users, they are also often under high stress, pressured to act fast and still required to do their job accurately (Thimbleby, 2007). On the other hand, the perception of the system availability may be different from the true availability (Deber, Jota, Forlines, & Wigdor, 2015; Lauesen, 2005). Any mission critical system should therefore ensure not only that the critical functions can and will be executed swiftly when needed, but also ensure that the user is kept up-to-date on what is happening and given the opportunities to act the right way no matter what the situation is. Often this implies that doing the wrong things should be made hard.

With a mission critical system, it is fair to ask that how much should be invested into the design and testing of it, because one cannot accurately know how much a failure of the system function will cost. In some cases reported, e.g., 4G network outage (BBC, 2018), British Airways computer system failure (Independent, 2018), or network problems in Finnish hospital (Aamulehti, 2017), it does not sound fair to just say that the organization responsible for the problems is sorry for not being able to provide the mission critical functionality when it was needed. Furthermore, it probably does not satisfy anyone that a promise is made to fix the problems as soon as possible: the problems should not have surfaced in the first place, at least not to the level observable by the users of the system.

Examples of mission-critical applications vary from domain to domain. For example, an automatic locator application may be mission-critical for a

public safety organization like police or an ambulance company. However, if a package delivery company uses the very same software for tracking its vehicles, it may be characterized as important or nice-to-have system, but certainly not essential without which the company cannot operate and perform its duties.

When deploying mission-critical software, IT administrators must determine exactly what support is necessary to ensure the systems' ability to function under optimal but also sub-optimal circumstances. For example, the servers running the services should have multiple, redundant network routers and power supplies to keep the services connected and servers powered in the event of a network or power outage. The servers need to have multiple independent connections to the outside world to stay connected when some of the connections fail. Depending on the available budget and physical infrastructure, mission-critical applications typically require at least N+1 redundancy (Gottesman, 2014), which certainly is not cheap. Furthermore, the redundancy mechanism themselves are critical and make the system even more complex. Even though the redundancy mechanisms are in place to ensure the availability of the service, how do we know that the mechanisms themselves won't fail?

Making sure that help desk support is available 24/7 can also help the system administrators to ensure that mission-critical applications are always available. Furthermore, frequent and automated backups are mandatory to protect applications from corruption or deletion. As a consequence, when deploying a mission-critical application, one needs to test and validate both the functionality of the applications and services as well as the capabilities of the infrastructure, both in normal and in all relevant exceptional conditions. Even in the cases where the disaster scenarios have been specified, disaster recovery (DR) plans are made and tested and appropriate actions are taken to mitigate the risks, problems do surface (Aamulehti, 2019).

The IT administrators often tier their DR plans to prioritize the restoration of mission-critical applications. Sometimes the administrators thus choose not to update mission-critical applications as frequently as lower priority applications in order to reduce the risk of introducing changes that might cause problems. In the era where continuous delivery and continuous deployment are seen as the way to go from the software vendors' point-of-view, this view point of an IT administrator may come as a surprise.

2.3 Software as a Service Paradigm

Companies have traditionally delivered services to their customer organizations in a product-centric operating model. In this model, the vendor creates a software bundle, often comprising several products, and the bundle is then installed at the customer organization's location. Here, the customer typically takes on the responsibility of managing both the infrastructure and the software within their data center. In this delivery scenario, no news is often good news, as if the vendor does not hear anything from the customer after the delivery, it most probably means that the customer and the system users are at least somewhat happy with the delivered system. However, the vendor cannot be 100% sure about this.

When software is delivered as a product, upgrades are made infrequently. Most often upgrades are made on yearly basis or even more infrequently because the customer must be able and willing to schedule and participate in the upgrade process. Often the customer organization also sees that the users need to be properly trained before any upgrades are made, so that the users understand the effects the changes in the new version have to their work.

In the traditional model, the upgrades are often relatively disruptive to customers. Due to the disruptive characteristics, the upgrades are deferred, even in the cases where the upgrades bring along much needed fixes and new valuable features. Infrequent upgrades lead to software releases that are the result of several months' worth of development. This, again, leads to large bundles of software to be deployed at one go, which, naturally, is far riskier than the deployment in smaller bundles. Finally, the riskier the deployment of a new version is seen, the less willing the organization is to agree on and to schedule any upgrade. Furthermore, and in addition to the risk, when a large upgrade is performed, there is often a relatively high cost associated with the upgrade and the upgrade often requires a scheduled outage, which further increases the costs. Outage may not even be an option.

The traditional model of acquiring an information system, i.e., buying software licenses and installing and managing the information system on the hardware owned by the organization consuming the service, has been the advocated approach for more 30 years. The three main recurring problems with this model are thus, from the customer point of view:

- unexpected and sometimes also uncontrolled costs, in the form of disruptions during the upgrades and in needs to keep the infrastructure updated,
- slowness and complexity of the commissioning, due to the infrequent but large upgrades and the required trainings needed before new version can be taken into use, and
- intolerable administrative burden due to the need to have a team responsible for running and maintaining the system.

Thus, the customers of information systems have been looking for a while already for better, more flexible and especially more cost-effective and reliable ways to get the needed information system services. To this need, the SaaS model provides a good alternative. The SaaS model is typically advocated with promises of higher adoption rates, lower initial costs, painless upgrades, better scalability and more effective and seamless integration between the systems, only to name a few (Turner, Budgen, & Brereton, 2003; Dubey & Wagle, 2007).

Looking back, it is surprising how much organizations have changed when it comes to provision of services, platform and infrastructure as a service (SaaS, PaaS and IaaS, respectively) (M. Kavis, 2014). The attitudes towards SaaS model and cloud computing in general have had a dramatic change within the recent years. The first responses to ideas on the cloud computing were from most organizations and companies fairly negative, as it was not seen as an applicable model to their current ways-of-working. The negative attitude quite soon turned into one with accepting the idea of private cloud, which was again, in a fairly short time, replaced by the acceptance of hybrid clouds for many purposes.

The types of workloads that organizations and companies are considering for deploying in cloud are changing as well. First, it was only non-mission critical and low risk applications. After that, it was test and development environments and cloud storage services that were taken into use but still the operative systems were run on dedicated services and maintained by the organization using the system. As of now, the cloud has become a viable target for almost anything, including the deployment of the mission critical services. Consequently, software vendors are realizing that their customers no longer want to buy packaged software and, instead, are expecting everything to be delivered as a service. Transformation to the service provision model

is expected to happen painlessly and to enable immediately frequent updates and better customer service. In other words, customers are expecting their vendors to swiftly become SaaS/PaaS/IaaS providers and being able to do the transformation in a flexible way and fast.

For any software vendor, the change from a license-based software offering to a service provider has also obvious advantages. Being able to provide services while having the monitoring and control of the SaaS environment fully available, sounds tempting in its simplicity and something to strive for (Waters, 2005). SaaS also allows us to respect the principles of XP and use the practices of XP (Beck & Gamma, 2000) effectively, as shown also in Publication IV.

SaaS has its darker side. From the customer's point of view the questions that need to be fully addressed when considering SaaS are, among others, related to the reliability of the service, the security of the data involved and the maintainability of the service in a way that has no negative consequences to the users. Although many organizations today use cloud storage to provide redundancy for their mission-critical systems, the question of whether to actually host the mission critical system services in the cloud is still quite controversial. The choice depends on many variables, including regulatory compliance requirements and trust in the cloud provider's ability to provide security and meet the appropriate service level agreements (SLAs). By placing the trust on the service provider's ability to provide an agreed-upon service, the customers certainly take a different kind of a risk than with the old model of acquiring the information system and running it by own organization. Whether this risk is somehow easier and cheaper to carry than the one organizations have been carrying with the traditional model so far, depends a lot on the principles and practices used in the development and deployment of the service and on the level and quality of the co-operation between the different stakeholders: the users, the customer's management and all the people in their roles on the system provider's side (Patel, Ranabahu, & Sheth, 2009).

From the system providers' point of view the SaaS model poses several challenges new to any organization not familiar with provisioning a service. These challenges call for organizational changes in the form of creation of a service organization, for mental changes to understand what it means to provide and especially maintain a service, and for process changes in the ways software is developed and deployed for SaaS users to use.

Even with these short-comings, SaaS can help organizations to quickly take into use new versions of services while avoiding large investments and investing fewer, if not at all, resources for system management and maintenance. The service provider carries the risks and responsibilities on behalf of the customer in releasing and running a service. Since the customer is subscribing to applications, infrastructure and environment all in one, without having real control over them, the customer places their full trust on the system provider's ability to provide an agreed-upon service (Waters, 2005).

Consequently, the very same qualities that make SaaS an attractive model for acquiring and consuming a system can introduce numerous new problems for the customer organization. One of the key issues is availability, which includes not only the reliability of the system, but also ensures the adequate performance, usability, scalability and maintainability of the system the users depend on. Another issue is security. With SaaS model, the customer trusts in the hands of the service provider the confidential data and the identity management, to name a few.

Chapter 3

Research Problem and Methodology

This chapter outlines the research problem and contains a description of the research methodologies used during this study. In addition, the chapter defines the appropriate research questions.

3.1 Research Problem

The research problem is the condition experienced in the procurement and creation of mission critical software systems and in its simplest form it could be stated as "why are we not doing better when creating large-scale software-centric systems by public tendering processes?". It is a troubling question that calls for meaningful understanding and deliberate investigation of the whole area of tendering procedures (Bryman, 2007). On the other hand, "not doing better" naturally has the other side as well: why are we not succeeding that well in the design, implementation, testing and deployment of the information systems, although one may argue that these phases of the SDLC are just execution after the system has been well specified during the tendering process and the best available candidate is awarded the project to develop the system.

As this research problem is wide and multi-faceted, within the scope of this thesis we approach the problem only in part, as specified by the research questions in Section 3.2. The publications do present also other points of view to the research problem than those stated by the research questions, but the elaboration and further processing of these viewpoints is left as future work.

3.2 Research Questions

The research problem (RP) in this thesis is: "What are the main practical implications when a company is transforming to service-based model with the help of SaaS paradigm?".

As sub-questions, to ease the development, deployment and operation of mission critical information systems, the following questions address the different aspects of the concept:

- RQ1: "What aspects need to be considered and emphasized in the RFQ tender process to support possible service provision and agile development?"
- RQ2: "How the emphasis on different aspects differs between projects developing critical and non-critical systems?"
- RQ3: "How mission criticality should be approached at the system specification and public tendering phase?"
- RQ4: "What advantages the SaaS paradigm brings to the mission critical system domain?"
- RQ5: "If a mission critical system is provided as a service, what points-of-view and requirements need to be focused on?"

Figure 3.1 shows the relations between the questions and the concepts or key elements handled in this thesis. Although the questions may be seen as addressing the intersection of the underlying key elements, that interpretation is not fully valid as the effect of the all three key elements is significant. This incapability to fully separate the responsibility areas of the questions and making them thus somewhat dependent on each other, is aligned well with the original objectives of this thesis. We were not aiming to analyze SaaS as a solution to information system needs specified by public tenders alone, but to analyze the relationship of the two when the system in question is a mission critical one.

The results presented in this thesis have been reported in eight separate publications. The research questions and their relations to the research questions are summarized in Table 3.1.

The first two publications (Publications I and II) focus mainly on the research questions RQ1, RQ2 and RQ3, where the research topic is the causes

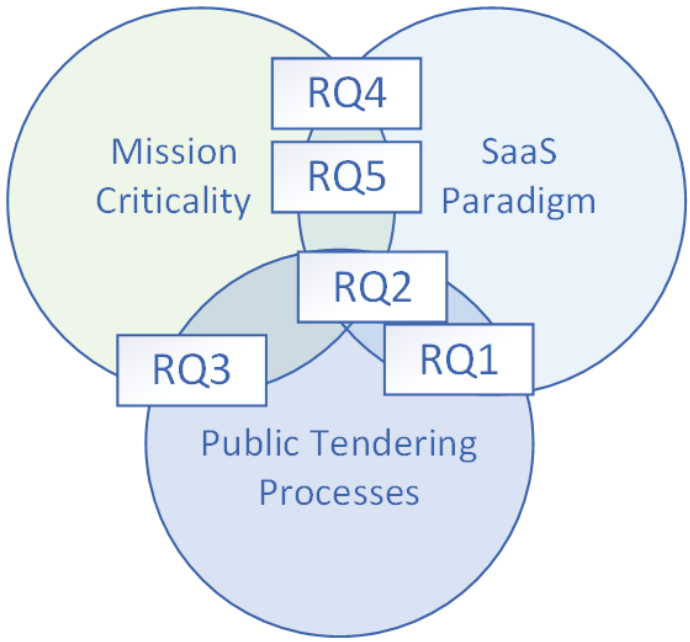

Figure 3.1: The research questions and their relation to the key elements in the scope of this thesis.

Table 3.1: Relations between the Publications (P), the Research Problem (RP) and Research Questions (RQ).

P	Objective	RP	RQ 1	RQ 2	RQ 3	RQ 4	RQ 5
I	To study and analyze the information system development process in the light of mission critical system needs.		x	x	x		
II	To study to roles of and relations between the requirements and the architecture in a mission critical system context.		x	x	x		
III	To study the challenges a company has when trying to transform from a software vendor to service provider.	x				x	x
IV	To study and analyze the skills needed to succeed in the role of a service provider.	x				x	
V	To highlight the different aspects of customer care when operating in a service provision context.	x				x	
VI	To study the problems in communicating the needs in service-based setup.		x	x	x		
VII	To analyze and understand the emergent issues encountered in the creation of large scale information system development.		x	x			
VIII	To study and outline the problem areas and possible solutions in the mission critical information system project based on a public tender.	x	x	x	x	x	x

and effects of the ways the systems are procured and means to ease the pains involved. The following three publications, namely Publications III, IV and V, focus more on the research problem as well as on the research questions RQ4 and RQ5. Here the research topic is the advantages of the SaaS model and the challenges a company faces when trying to transform its operational model to a service based one. Publication VI focuses again mainly on the research questions RQ1, RQ2 and RQ3, as it discusses and analyzes the difficulties we are having in communicating the needs and in clarifying them effectively, especially in a setup where communication needs to be frequent and wide-band and where misunderstandings and misinterpretations thus become more prominent. Publication VII focuses on the emergent complexity of critical systems and focuses mostly on the research questions RQ1 and RQ2. Finally, the Publication VIII is a summarizing publication and therefore deals with the research problem as well as with all the research questions.

3.3 Research Approach and Viewpoints

The research approach involves the major decision on how to conduct the research. According to Creswell (2003), there are three different research approaches, namely: qualitative, quantitative and mixed methods.

The qualitative data are descriptive and capture as well as communicate experiences of the field of study. Thus, qualitative data tell a story about the researched phenomenon. In addition, qualitative research relies on logical conclusions on the gathered data. In contrast to this, quantitative data seek numerical responses, and thus rely on quantitative measurement and mathematical models (Yin, 2017; Patton, 1990). The mixed method research approach, as the name suggests, employs data collection and analytical techniques associated with both qualitative and quantitative research methods by using multiple data collection methods (Easterbrook, Singer, Storey, & Damian, 2008).

As the data this thesis is based on communicated experiences, one can argue that the research is qualitative. On the other hand, the thesis does not base the results and conclusions on just one or a few software projects as the experiences stem from several tendering processes and development projects. Therefore at least to some extent the data is also quantitative. As a conclusion, the mixed method approach may be the best description of the approach used.

Chapter 4

Results

4.1 Summary of Contributions per Publication

This chapter presents an overview of the publications included in this thesis and the most important results of them.

Publications I, II, III, IV, V, VII and VIII have been published separately in peer-reviewed scientific conferences and journals. Publication VI has been published as a book chapter in a peer-reviewed book. In the following, each of these publications are discussed briefly, including the research objectives, main results, and relation to the overall goal of the thesis.

4.1.1 Publication I: Rolling out a mission critical system in an agilish way: Reflections on building a large-scale dependable information system for public sector

Publication Objectives

The objective of this publication was to document and analyze the experiences gained during the design, implementation, testing and deployment phases of a large mission critical information system project. The objective was to report experiences that have been gathered over several years, covering various aspects, such as contractual factors, agile adoption, process improvement, management, and end-user involvement in the development and deployment process.

For better or worse, the agile practices have become an essential part of building mission critical information systems. In the project that this publi-

cation derived its ideas from, the incremental and iterative development was a requirement, although it was not fully clear what the customer organization actually meant by that requirement and why that requirement was seen as an essential part of this project. One motivation of this publication was also to address this issue, i.e., to question and analyze why an organization wants the development effort to be incremental and what are the implications of that, although the project has a solid set of requirements as its basis.

Results

In Publication I, some real-life experiences from the design, implementation, testing and deployment of a multi-million euro, mission critical information system for emergency services were presented and analyzed. As of now, the system is operational and made available to the end users as a service.

As the downtime of such a mission critical system is not acceptable, the system's extremely high availability requirement needs to be ensured and considered in all the phases of the development. Furthermore, all parties involved should understand the criteria used for measuring the availability in the same way, as for the end-user the availability does not just mean that the services running on the servers in data centers are up and responding swiftly. From the end-users' perspective the user interfaces must effectively support the user in executing her tasks and make her as effective as possible and as informed as needed per situation at hand. This is possible only if the developers of the system fully understand the characteristics and the behavior of the end-users and the system.

Consequently, to ensure the preservation of dependability throughout the development, special attention must be placed on the mission criticality aspects in the design of both the system itself as well as tools, techniques and processes that are used during the development. One of the main findings in Publication I was the importance of the constant and continuous focus on improving the ways that the information in the development process flows. Figure 4.1 show the starting point, where the development process was based on the descriptions in some of the Scrum books or alike. Soon it was found out that such a process does not satisfy the needs of keeping all the stakeholders satisfied and the work flow was transformed to that of Figure 4.2.

Based on the experiences documented in Publication I, the most important issues that can make such a big project as the described one successful, are:

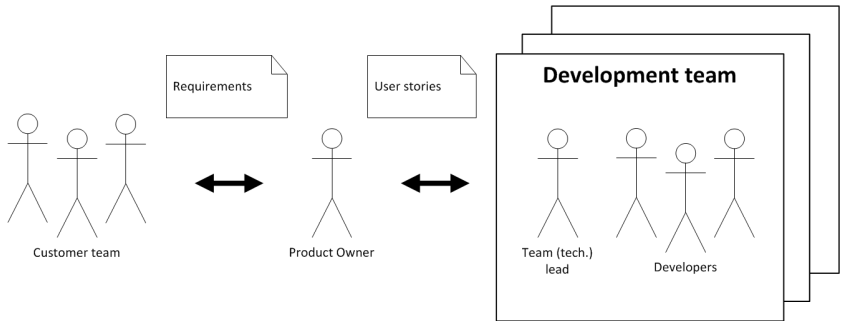

Figure 4.1: The development organization at the start of a project.

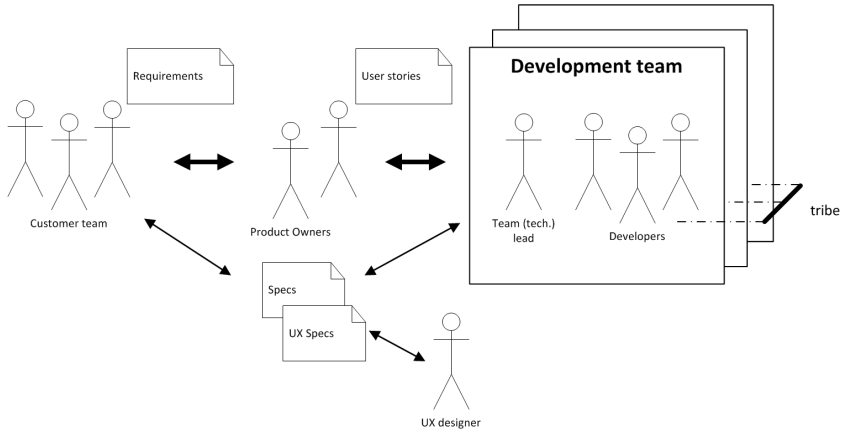

Figure 4.2: An improved development organization, at one phase of continuous improvement.

- Direct access to a real customer or at least a legitimate customer proxy. Access is needed for discussions, to make questions and to validate requirements and the end product.
- Installation of short and several feedback loops between the players, with the ability to change them rapidly in case priorities change.
- Putting up an environment supporting collaboration between all the different players. Development teams, QA, testers, people responsible for delivery and managers, all need instant access to up-to-date information related to the project and its status. People will communicate if they feel enabled and empowered to do so.
- Fostering the idea of quality driven development through wide-scale automated testing involving at least unit testing, integration testing, load testing, soak testing, and stress testing. The QA department must get a chance to have its voice heard and testing needs to be effective.

Furthermore, one important result is also that one should keep in mind that the items listed are only enablers of success, and by no means any guarantee. In any project that is larger than a few man-months, there can never be a guarantee that by following certain principles and best practices one succeeds. The complexity and the variety of the projects and the players involved make software development challenging, but also ever rewarding.

The experiences might be quite typical for such setup, where the parties involved are trying to do something they have not done together before. It is natural and thus should also be expected that it takes time to find effective ways of working. One important learning is also that the modification and improvement of the ways-of-working should never stop: there is, once again, no silver bullet.

Regarding the requirement for agile development approach, based on the experiences, one should simplify the requirement drastically. In order to be effective and contrasting to the mission criticality aspect, agility should mean three things only:

- User involvement.
- Iterative and incremental development.
- Constant adaptation to the situation at hand.

Key observation

The most appropriate tools and practices for developing software are the ones that work for the particular problem, for the particular people with their specific knowledge and skills and in the particular environment. The problem, the people and the environment do change continuously. Therefore tools, practices or even principles that have worked once in some setting may not anymore be the most suitable ones for the current setting.

4.1.2 Publication II: Requirements, architecture, and quality in a mission critical system: 12 lessons learned

Publication Objectives

With the public tender process of information systems, the representatives of the future owner of the system precisely define the exact needs the owner has in mind. Once requirements are fixed, it is up to the developers of the system, usually employees of another company, to create a design to meet the requirements. As the people involved in the specification are typically experts in the domain the system works in, the requirements concentrate typically a lot around the functionalities and the non-functional aspects, i.e., the quality aspects, are overlooked. This is not acceptable when the system in question is a mission-critical one.

The objective of Publication II was to analyze the roles the requirements, the architecture and the quality have in mission critical information system projects and propose improved ways to design an architecture that can effectively fulfill the needs.

Results

In the agile era, where interactions between developers and other stakeholders are advocated, the traditional way of specifying the requirements up-front and then holding to those is strongly challenged. To improve the situation, calls for different setups as well as different mindsets for the development. In general, the customer does not know what the right solution should be, and the developing organization does not have an off-the-shelf solution. Rather, joint work is required to find functional practices for cooperation and co-creation.

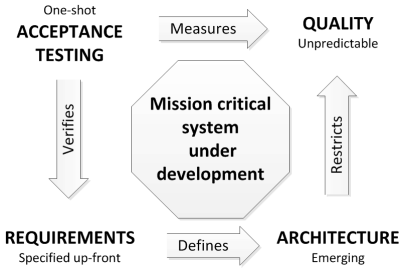

Figure 4.3: Architecture and requirements in "agilish" approach.

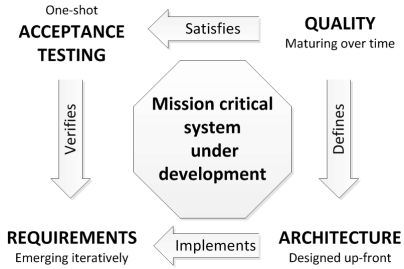

Figure 4.4: Requirements supported by Architecture.

The roles and the relations of the architecture and the requirements, as described in Publication II, in the traditional model and in the "optimal" case are presented in Figures 4.3 and 4.4, respectively.

The development project needs to start with some clear quality needs, consisting not only of functional quality aspects, but covering all the quality aspects the system owner and her representatives have in mind. The format of the quality need should state who needs something and why. The system architecture should build on these quality needs and reflect the up-to-date understanding of the required system quality.

The acceptance testing should be done against the required quality aspects as they are at the moment of testing. The testing should concentrate on the validation of the implemented quality aspects and give feedback on all tested quality features, guiding the consequent development and testing efforts. The requirements—if required for example for contractual reasons—can be defined and written on the basis of the implemented quality and the results from the acceptance testing. Important aspect is that the requirements here emerge iteratively and allow us to monitor and analyze the way the system is evolving, also quality-wise. This will not be trivial, but when the starting point is the required quality instead of the requirements related to the architecture itself, the quality must be given the focus it needs, as the quality requirements cannot be deemed as just another requirements among the functional ones.

The requirements, the architecture, and the original needs or vision of the system to be developed and taken into use give us three points of view into the system. These views are not similar, but complementary. Ideally, the three elements, augmented with the acceptance testing criteria or test cases, should support and reinforce each other and together give any stakeholder

a clear view on what are we developing, how the development is progressing and how well it satisfies the needs we have.

The learnings from the analysis performed in this publication where gathered to a list of lessons in various categories. The lessons learned are presented in Table 4.1.

Key observations

The architecture of a mission critical system cannot be based on the stated requirements, as the set of requirements often does not contain the architecturally significant ones. Therefore, the specification of the mission critical system needs to start from the architecture that enables the functionalities, not vice versa. The concept of minimum viable architecture (MVA) helps a lot in designing the architecture.

4.1.3 Publication III: On the Windy Road to Become a Service Provider: Reflections from Designing a Mission Critical Information System Provided as a Service

Publication Objectives

The objectives of the Publication III were to document and analyze the experiences gathered from an implementation project where a mission critical information system was to be provided as a service by a company and project team that had a lot of experiences in the project-based deliveries but none from the service provision scheme. The publication focuses on the criteria by which the service provided is seen as a success and what are the reasons for the failures.

Publication III focuses on the specific skills needed from a company to succeed in the service provision model. These skills differ a lot from those of a software company that traditionally is regarded as a successful one. Traditionally, the focus area is the effectiveness and efficiency of the software delivery pipeline, and not necessarily and primarily the well-being of the end-user and the happiness of the customer.

Results

When assessing the perceived quality of an information system, the role of the end user and how the end user experiences the service have major impact.

Table 4.1: Twelve learnings of Publication II.

Category	**Lesson learned**
Needs and Requirements	Even if the client and the contractor both wish to execute a project in an agile fashion, joint practices and processes must be carefully agreed upon.
Needs and Requirements	The stakeholder that the developer hears best ("the loudest voice") may not be the most important stakeholder.
Needs and Requirements	Feature management system is not a replacement for interactions with end-users and other stakeholders
Specifying Requirements	New features are easy to invent; each feature should be associated with explicit stakeholder value.
Specifying Requirements	To avoid feature creep, one needs to perform rigorous and visible change management. Learning to say "No" in a nice way to end-users is obligatory.
Specifying Requirements	Criticality is a part of every requirement, not a part that can be isolated.
Architecture Design	A verifiably satisfying architecture for a mission critical system simply does not emerge; it must be explicitly designed and validated.
Architecture Design	Architecture is not a concern of any of the representatives of the system's owner. Therefore, the developers should be extremely concerned on its design.
Architecture Design	Limits of the architecture will be met in any case; create a strategy for managing overarching requirements.
Quality	Defect reports are no replacement for true interaction with representative end users.
Quality	Test automation is a necessity for agile development.
Quality	Quality in the large cannot be validated with technical tests only; instead, end-user involvement is needed.

The user wants and expects that the service is provided as well as possible, no matter what the circumstances are. Furthermore, in case there are some problems in the provided service, the user expects that all the relevant exceptional situations are well thought through in advance by the service provider. The user can justifiably assume that the system behind the service takes proper actions in due time to keep the services up and running. Furthermore, the user can assume that she does not need to face any unexpected situations when using the system. A user of a mission critical system is a trained professional, who has a strong need to get her job done. The user needs to be able to trust that any task she performs with the service is executed in a correct and best possible way.

A successful program that aim at providing good service needs proper tools. In addition to the tools associated with software—like user centered design (UCD) (Abras, Maloney-Krichmar, & Preece, 2004), agile practices (Martin, 2002) and continuous delivery (Humble & Farley, 2010), which are commonly used today—companies need to master some additional tools to meet the information system owners' and users' expectations on the service-based model.

The roll-out of the service needs to be carefully planned and executed, as with consumer products, one should not underestimate the power of OOBE (Ketola, 2006). Any difficulties experienced early in the service, will be remembered and they affect the users' confidence towards the system. Thus, when OOBE turns into an OOBF (out-of-box-failure), it can be highly detrimental to the future acceptance of the system, especially when the customer expectations for the system are high, as typical for a critical one. This applies to the user interface as well. Even if the service works perfectly from functional points of view, clumsy and not user-friendly UI's may result in bad user experience.

Service providers should hone their skills to take the full advantage of the fact that providing a system as a service, the software company is much closer to the customer and much closer to the customers' real problems and has potentially a much better visibility on what is happening when the users use the service and how the service is used. The skills required to take the software business to the next level are mostly soft ones: being interested in and respecting the customers, being open for changes, being good in communications and being willing to try new things and improve on everyday basis.

Key observation

The skill set and competencies needed to become a successful service provider differ a lot from those a company with a project or product development background has. It takes a considerable amount of time to transform to a service provider and the transformation is likely to encounter some failures along the road and wide-spread organizational change.

4.1.4 Publication IV: Can we get some service here?: On the company transformation from a software vendor to a SaaS provider

Publication Objectives

The objectives of the Publication IV were again to reflect the experiences on the transformation of a company from a license-based software development to SaaS offering. The main focus of Publication IV was to analyze and emphasize the opportunities the SaaS model offers for the service provider that enable the software to be developed and deployed in agile manner, respecting the agile principles.

Results

The transformation from the license-based software development to a SaaS offering poses challenges related not only to technical issues but to a great extent also to organizational and even mental issues.

The main takeaway is that with the SaaS model, many of the principles manifested by the agile movement can and should be followed closely and the advantages gained with the SaaS model are very close to the objectives set by the agile manifesto.

To enable effective service provision, the provider must have appropriate processes defined, accepted, and in active use. For each process, the owner and key individuals involved in and responsible for the process must be listed. The process framework that is fairly widely utilized is ITIL, which seems to be a good starting point that takes into account also business management.

Key observation

The SaaS model enables us to respect and follow the agile principles and use the agile practices. Companies that wish to continue to fulfill the changing

needs of the customers must adjust their practices to comply with the SaaS model.

4.1.5 Publication V: Implementing Continuous Customer Care: First-hand Experiences from an Industrial Setting

Publication Objectives

The objectives of Publication V were to look at the ways the end-user behavior, if made known to the developers, can affect the software development. With the traditional on-site installation model of enterprise software, end-user characteristics often remain at least partly unknown to the system providers. While users are typically modeled, and their personas described during the development, those models and personas are rarely verified or refined after installation, based on operative usage of the system.

The SaaS model introduces a drastic improvement to the situation, as the service provider can now have direct access to the user data and analyze it, if agreed appropriately with the customer. Publication V presents experiences regarding a company's transition from a provider of installed software systems to a SaaS provider from the viewpoint of customer care and related activities.

Results

The results presented in Publication V, although partly preliminary, show that many of the assumptions on the user behavior may not hold true in the operational use and substantial effort is required to understand various user needs thoroughly.

The service-based setup, where the end-users' behavior is easily available for the service provider, opens a whole new area for a company to work in, namely that of the Customer EXperience Management (CEM or CXM). In the broad sense, CEM refers to the collection of processes a company needs to track, oversee and organize every interaction between a customer and the company throughout the customer lifecycle. The goal of the CEM is, naturally, to optimize interactions from the customer's perspective, thereby increasing customer satisfaction and fostering customer loyalty. We also enter the concept of managing the customer experience, for which a company providing the service needs to create some sort of a customer-centric strategy that encompasses all interactions.

The findings of Publication V are also supported in other publication (Schmitt, 2010). To sum up, there are four critical steps to creating a successful customer experience strategy:

- Understand your customer. The first step in building a customer strategy is understanding customers' needs and behaviors and creating customer segmentation based on these factors.
- Create a customer vision. Once the target audience is identified, the next step is to create a customer journey map. This helps to identify customer touchpoints and anticipate how the customers will interact with the product or service. Eventually, the customer vision would help customer retention down the road.
- Develop an emotional connection. This step involves creating a brand personality that evokes positive consumer sentiment and helps establish a relationship between the customer and the company.
- Capture customer feedback. It is important to measure customer satisfaction in real time. Customer feedback can help the company track customer perceptions, enable quality monitoring and measure the success of the customer experience strategy.

Key observation

The SaaS model enables the service provider to monitor effectively the users of the service. The data gathered by the monitoring activity can be used to proactively enhance the user experience and improve the service, enabling close collaboration with the users and better understanding of their actual needs. Human touch is still needed: the customer satisfaction cannot be measured based on technology only.

4.1.6 Publication VI: What We Say We Want and What We Really Need: Experiences on the Barriers to Communicate Information System Needs

Publication Objectives

Information system requirements, as any requirements, are meant to communicate the relevant needs and intention to a wide range of stakeholders.

The requirements form the basis on which the tenders are issued, projects are agreed upon, progress is measured by and service level agreements are made. However, as the requirements state what the system owners—or the ones who are willing to pay for the system—want the system to achieve, they reflect only the owners' views and understanding.

This setup is plagued by many weaknesses and Publication VI aimed at listing and analyzing these weaknesses in the context of a SaaS-provided service.

Results

Based on the analysis and discussion in Publication VI, the main weaknesses in the information system requirements are the following:

- The system owners are seldom experts in the information system design and therefore they may be fully unable to state all the relevant requirements comprehensively.

- No matter how much energy and time is invested in the requirement definition and elicitation, many aspects of the requirements are only revealed during the development and deployment. These aspects remain unforeseen until later on, when the development is well under way and many parts of the system may already have been implemented.

- The required system architecture cannot be appropriately designed if we do not know the architecturally significant requirements at a sufficient level.

The traditional way of handling the requirements has proven to be insufficient in the analyzed projects. With the software as a service (SaaS) business model, where the goals are frequent releases and continuous delivery of ever-improved services, the associated weaknesses become even more prominent.

Publication VI encourages to employ practices which concentrate a lot more on true communication and discussion, focusing always on the most important issues and most important stakeholders only. This approach should aim for keeping the vision updated and clear for the whole duration of a system development project and also during the system maintenance period.

In the RFQ-based projects, the responsibility to communicate the needs is primarily based on some requirement document. As experienced, the communication seldom succeeds in conveying the information unaltered. We should admit that written documents are a great help in supporting what has been agreed and planned, but to ensure that as little misinterpretations and misunderstanding take place as possible, face-to-face communication needs to happen frequently.

Key observation

The human communication is error-prone and often unfounded assumptions are made on the misinterpreted information. Extensive documentation does not in itself enhance the quality of the communication, but in every case leads to misinterpretations. Instead of trying to improve the communication that takes place by written documents, other more effective communication channels, like face-to-face communication, should be encouraged and utilized more.

4.1.7 Publication VII: Taming a Monster: Tackling the Emergent Issues Encountered in Mission Critical System Development

Publication Objectives

The objective of Publication VII was to analyze and document the path often traveled in information system development projects, namely the one from a simple, well-behaving system to complex monstrous system, the behavior of which no-one understands anymore.

The interesting aspect of the observations on emerging issues is that the road to a challenged state is paved with good intentions.

Results

The creation of a system so complex that its behavior starts to be non-understandable, does not happen overnight. Nevertheless, by the ways of working we employ, we are able to create "monsters" instead of simple, well-behaving systems that stay maintainable and flexible for future needs. Much of the problems relate to how we test the system, since by testing we should

be able to collect enough information and gain the knowledge to understand how the system behaves and evolves.

There are many problematic areas in the information system testing, but with mission critical systems, some aspects of the testing stick out. These "pain points" are listed in Table 4.2 (table continues on the next page).

Table 4.2: Examples of the pain points in mission critical system testing in Publication VII.

Pain point	**Mitigation**
Test effort not focused optimally	A disciplined way to identifying in detail the environments needed for the testing efforts. Both the physical and virtual test environments and the production environment need to be analyzed and identified in detail, and the tools and resources required to be available to the testers in these environments agreed.
Non-functional tests do not get focus early enough	Need to take up the non-functional acceptance criteria with all the relevant stakeholders. The required response times, throughputs, and resource utilization goals and constraints need to be agreed and documented as well as all the failure scenarios and monitoring events. All the issues that criteria cannot be found for, need to be assigned on somebody's responsibility for clarification.
Test results do not reflect the real situation in operative environment	The test cases need to be planned and designed in close cooperation with the end-users, including the type and size of the test data into the test specification. All the key scenarios with determined variability, including all the failure scenarios and related redundancy mechanisms, need to be specified in detail.
Not understanding some configurational effects in test results	Preparing and configuring the test environment in a disciplined way and per documented process is a must. Furthermore, need to ensure that the test environment is instrumented for resource monitoring as necessary and that the monitoring system itself is working properly.

Energy wasted by performing tests that do not provide true value.	Performance tests should be developed in accordance with the other test designs and follow more closely the execution and monitoring of the tests. The tests, the test data, and results need to be validated to ensure that we have measured the right things and that the measurements collected are true measurements of real nature.
Test results not effectively used for improving the service.	Consolidation and sharing the test results to all relevant stakeholders is a must. There needs to be sessions for analyzing the results together with the customer, aiming for better understanding of and new points-of-view to the results obtained.

Related to the pain points listed in Table 4.2, the following issues should be taken into consideration:

- Having a thorough understanding of the entire test environment enables more efficient test design and planning and helps in understanding the testing challenges faced in the project. The process of analyzing and identifying the testing environments must be revisited periodically throughout the life cycle of the service.

- The response times are a concern of the users and should be analyzed in detail to enable good enough user experience. Without proper understanding on the user behavior good criteria on the response times are almost impossible to set. The throughput and resource utilization are in most cases not directly user concerns but business and system concerns, respectively.

- The test data must be realistic in form and in size and appropriate metrics to be collected need to be established. Also, the capabilities to simulate the specified variabilities need to be developed.

- The configuration of the test environment requires eye for details and discipline. The preparation of the test environment, all the required tools, and resources necessary to execute tests as new features and components become available for test need to be done in a documented and traceable way.

- The validated tests for analysis should be executed only while monitoring the test and the test environment. In case we observe any exceptional issues, these should be taken seriously and investigated thoroughly to understand the root causes of the observations.

- The analysis and reporting of the test results is the most crucial action of all. A test has been successful only if all the metric values are within accepted limits, none of the set thresholds have been violated, and all the desired information has been collected.

Testing a mission critical system needs to be done in a way where the mission criticality is fully considered. The role of the automated end-to-end testing of the system in an environment as close to the operative one as is possible should be emphasized. Focusing the testing efforts optimally and scheduling the efforts early enough in the development process saves us from many problematic situations in the operational environment.

Synthetic monitoring is the use of software to simulate user interaction with a given system. In contrast, passive monitoring uses data from actual transactions. Synthetic monitoring makes it possible to detect issues, so they can be dealt with before they arise with actual users or cause other problems with a system that could affect system performance or availability.

Key observation

When a system is under test, the tests should reflect the actual operative characteristics of the system as closely as possible. All the testing effort should be spent on testing the system in its operative environment, or in an environment indistinguishable from the operative one. If the tests do not reflect the actual use and the test environment is not like in the operative case, the testing effort is just waste.

4.1.8 Publication VIII: How to a Survive Mission Critical Systems Project Based on Public Tenders: Lessons Learned the Hard Way

Publication Objectives

The objective of Publication VIII was to further highlight the issues that are found to be hard in the mission critical information system creation and

development process. Publication VIII can be seen as a summarizing publication, deriving its main points from the other publications and focusing on the bigger picture.

Results

In Publication VIII it is shown that the key aspect of the mission critical system is that the users must be able to trust the service behavior, no matter what happens. Building such trust is difficult with the traditional RFQ-initiated provisioning model, where the needs of the users and customer remain at least somewhat unclear.

This condition calls for a new kind of customer-provider relationship that has its roots far deeper than what public tendering relationships can offer. Both parties must step away from the requirements and set the needs and disciplined processes first. These processes must be a supporting structure, not one that restricts the design and development or one that points who to blame when something goes wrong.

Key observation

The successful development of an information system requires constant focus on all aspects of the software development. The success does not result from doing things almost right, but exactly right and in continuous manner. Succeeding in development effort requires also full commitment to the work from all the stakeholders. To succeed, also new ways of communication need to be established between the parties and all participants need to ensure that information is delivered correctly and in timely manner.

4.2 Synthesis

As described in the previous sections, the publications look at the information system procurement process, including the development process, from different points of view and emphasize different aspects depending on the angle the individual publication has to the process. To get the overview on the relations of the publications to the whole, the relation of the publications and the software development lifecycle (SDLC) is shown in Figure 4.5.

In the following sections, some guidelines drawn from the results of the publications are presented from several viewpoints.

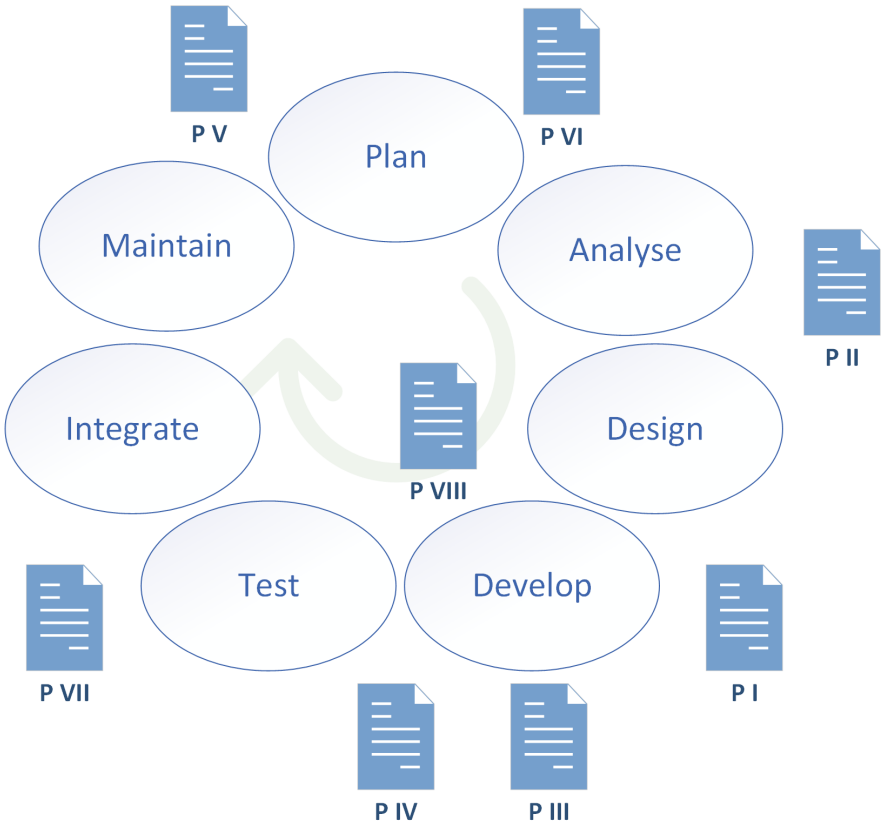

Figure 4.5: Publications I-VIII and the SDLC process phases.

4.2.1 Guidelines for Public Tenders on Mission Critical Systems

The main problem with the employment of public tenders on mission critical systems is that the short-comings of the communication as well as the fact that the needs do change, lead to end results where the deployed mission critical system does not fully meet the actual need in a reliable way. Naturally, it would be nice that all the information systems, critical or not, would better fulfill the needs they are built for, but with the non-critical systems the short-comings do not lead to dramatic consequences.

As guidelines to be followed when dealing with mission critical system, the public tender should be prepared and awarded in such a way that:

- The need is defined clearly and discussed to such extent that all the stakeholders understand and approve the need in a similar way.

- The need is stated in a language and form that leaves as little as possible room for misinterpretations. The possible ways to mitigate the risks of misinterpretations and misunderstanding are, for example, using user stories or use cases instead of IEEE 830 styled requirements and stating the non-functional requirements in a clear way upfront.

- Focus is set on the quality attributes of the system instead of the functionalities. Functionalities can be added later to a system as needed if there is a solid base with flexible architecture to build them on.

One main problem that still remains is that of the barriers in the communication as described in Publication VI. As long as we are unable to convey messages through so that it is understood the way it is meant to be understood, there is little chance that any co-operation succeeds. We may have extremely effective and streamlined software development process, waterfall or iterative, would it follow agile principles or some design method, and still we would do the wrong things or things in wrong order because we have not understood what was actually said or meant. In case we have already started the project without enough focus on the communication and understanding the actual needs, one should keep in mind the famous Turkish proverb: “No matter how far you have gone on the wrong road, turn back.”

4.2.2 Guidelines for Mission Critical Systems provided as a Service

To be able to successfully work in the service-provision scheme, a company must be able to transform itself to a service provider, which may sound straightforward and trivial, but most probably is not. The key requirements in a service-based scenario are:

- **Agility**: The company must be able to work effectively with the customer and the end-users to shape the service and to ensure that the original and any newly found needs are fulfilled. Agility means that work is done in increments and with iterations. Furthermore, the company needs to reflect its ways of working continuously and improve all the time.

- **Disruption**: The company must be willing and able to enable experimentation and innovation of new ways of working. Related to this, there may not be need for fixed established processes as those can limit the innovation. Not having processes does not, however, mean that one does not have to be disciplined, quite vice versa.

- **Extremely short feature to production cycle**: To support experimentation and enable fast service provision to the users the work processes must always aim for fast delivery to get feedback as fast as possible.

- **Ability to test effectively**: To enable reliable releases of new features and new versions of existing features, the testing must be fast and effective: no defects, i.e., zero, should be allowed in the production. This calls for high level of automation and test-oriented mindsets. The environment in which the tests are run cannot be anything else than the exact copy of the operative environment or, better yet, the operative environment itself.

How well these business requirements fit the strict availability and reliability requirements of a mission-critical service, depends a lot how the mission critical system owner sees the system's evolution needs and how much the system owner trusts the service provider to allow it for perform frequent updates and to experiment.

Discussion

It is impossible to run a service correctly, without thorough understanding on which behavior actually matters the most for that service and how that behavior should be measured to be able to evaluate the behavior in a way that tells us how the users are experiencing the service (Beyer, Jones, Petoff, & Murphy, 2016).

There is a fine balance between the agility and the required service levels as well. Often the service-level agreement (SLA) does not allow, or at least does not encourage, the service provider to be agile and to experiment and innovate on the system (Trienekens, Bouman, & Van Der Zwan, 2004). Any modification, a new feature or a change made for experimentation is a threat to the measured service level. Therefore, the SLA needs to be continuously improved and developed further to keep it effective in measuring the right kind of availability.

One should also distinguish between a SLA and a Service-Level Objective (SLO), although it seems that the SLA is the term used throughout when discussing the target service levels. In fact, when measuring the service levels, we should talk about SLOs, as they are the specific measurable characteristics of the SLA such as availability or response time. An SLA normally involves a promise to someone using the service that the availability of the service should meet some certain levels, i.e., the SLOs, over a certain period, and if it fails to reach that target, some kind of penalty will be paid. The idea is that going out of SLA is going to hurt the company providing the service, so the service providers are made to try their best keep the availability within the agreed SLA (Ross, Hilton, & Rensin, 2017).

With mission critical systems, the role of the SLA and included SLOs should be analyzed and rethought thoroughly. Are the penalties, if such exist, effective in making the service provider to try its best, or could there be better ways to do that? For the losses a failure in a mission critical system can cause, any penalty may not be enough.

To test effectively, the environment needs to be as close as possible to the operative one. Otherwise we end up with testing results that do not reflect the situation in the real environment and cannot thus be used to evaluate the relevance of the SLOs, for instance. A good guideline is that set by Boeing in the 90's (Spafford, 2015): when testing their avionics code, in addition to performing extensive design review and testing, all the software engineers were boarded to fly on some of the first test flights.

4.2.3 Guidelines for Service procured by Public Tender

When the public tenders aim for having the delivered system to be provided as a service, at least the following characteristics of the tender need to be ensured:

- The requirements, or for that matter, any other ways to communicate the need, must be as independent as possible and well prioritized. Otherwise, we will not be able to create the service in a way that enables us to adjust and pivot as needed.
- The tender should not state strict deadlines for some initial operating capability (IOC), i.e., the minimum usefully deployable form, or full operating capability (FOC), because in that way the ability to do iterations and change the scope of the incremental development bundles is made much harder.
- The tendering organization should take care that enough resources will be made available to the development project to allow appropriate user involvement and co-operation. These resources cannot naturally be just anyone, but people who understand the need and are able to vision the ways to fulfill that need.

4.3 Summary

In this Chapter, the objectives and the main results of all the publications in this thesis were presented. As already briefly noted in Section 1.3, the main result of the publications can be seen as threefold.

In software projects, we do have serious communication problems, which give us the challenge of **Bridging the Communication Gap**. The situation can be improved drastically by admitting that the communication taking place by formal documents and IEEE 830 style requirements is not effective. Other communication mechanisms should be utilized and one should not underestimate the power of face-to-face communication which is most easily achieved just by putting the right people physically in the same place for long enough period.

For any software project **Providing Efficient Development Process** is naturally important, but for mission critical one, efficient and effective process is a prerequisite for success. The development process should be

followed in a disciplined way while enabling and encouraging all the participants to communicate effectively. Communication should be reliable, ensured to have high quality and the mechanisms should allow constant and frequent feedback. At the same time, the processes need to be tuned and altered as needed to stay effective. The SaaS model helps in this aspect by enabling incremental development with effective feedback channels, if just implemented appropriately.

Transformation to Service Provider is a challenge for a company that has a history of a software vendor. A new service-oriented mindset and a whole new set of skills, practices and tools are needed in order to provide services effectively. The transformation is a concern of the whole organization and is not limited just to the development or operations offices.

Chapter 5

Research Contributions and Implications

This chapter discusses the main contributions of the publications presented in the previous chapter. The chapter revisits the research questions and gives implications for practitioners and researchers. The research is evaluated, and the limitations and validity of the study are discussed. The research questions are linked to the results.

Some topics for the potential future research are also outlined. These topics stem from the presented ideas and a more detailed discussion of the topics would require research efforts of their own.

5.1 Revisiting the Research Questions

In the following, the research questions are addressed and answered by linking the questions to the various aspects of the research area addressed in the previous chapters.

The research problem was: "What are the main practical implications when a company is transforming to service-based model with the help of SaaS paradigm?". The problem was supported by a number of research questions, each addressing a specific area in the research context. The findings related to the research problem are summarized in Table 5.1.

Although the research questions are in three categories, the findings related to the research questions are summarized individually in Tables 5.2-5.6.

Table 5.1: Summary of the findings, the research problem (RP).

RP: What are the main practical implications when a company is transforming to service-based model with the help of SaaS paradigm?	
Providing services	If the background of a company is project-based and the outcomes have been delivered so far as software bundles, the transformation into service profession is a huge step. Bigger than a company would assume.
Metrics	If accustomed to measuring hours spend on the construction of a software application and the number of defects found by testing it, the metrics need to be fully redefined in the service provision domain to reflect customer satisfaction.
Customer care	A company providing the service must know how the users of the provided service and the customer feel about it. There needs to be multiple channels to get the feedback and being reactive is not enough.

Table 5.2: Summary of the findings, RQ1.

RQ1: What aspects need to be considered and emphasized in the RFQ tender process to support possible service provision and agile development?	
Support the increments	Incremental and iterative approach should be required, including the reservation of enough time and resources to do the increments and iterations. It may seem waste to add a relative high amount of time and money to prototyping and redoing some things already implemented, but this is the only way the actual need is found and satisfied, instead of the assumed need.
Reserve the resources	To succeed, the development effort cannot happen in isolation. There needs to be enough resources on both sides to be available when needed for answering questions, for clarifying issues and for all kinds of discussions.

Table 5.3: Summary of the findings, RQ2.

RQ2: How the emphasis on different aspects differs between projects developing critical and non-critical systems?	
Start with simple	With a critical system, the importance of simplicity is emphasized. The less there are requirements, the easier it is for them to be mutually exclusive and collectively exhaustive, while allowing for elaboration and redefinition.
Limit the need	Much related to "Start with simple", with critical systems the process should not try to cover all the wishes of the stakeholders at single go. One should be able to limit the scope to the smallest thinkable and be stubborn enough to keep it that way.
Quality first	With critical systems the emphasis should be on the quality attributes. If we can build a solid and scalable architecture, the functionalities will be easy to add as needed.

Table 5.4: Summary of the findings, RQ3.

RQ3: How mission criticality should be approached at the system specification and public tendering phase?	
Knowing the domain	In order to achieve good quality, a pre-requisite for a mission critical system is that the system developers understand thoroughly what is important to the users and how do the users behave in different situations encountered while the system is in use. This is also on the responsibility of the organization buying the system.
Having the right people	There needs to be enough knowledge and understanding on what is realistic and what the required characteristics of the information system take time- and resource-wise. Unrealistic expectations result in dramatic failures.
Ensuring trust	There needs to be full trust on both sides; the vendor needs to be able to trust the client and vice versa. Trust is not build easily and it does not appear overnight.

Table 5.5: Summary of the findings, RQ4.

RQ4: What advantages the SaaS paradigm brings to the mission critical system domain?	
Enabling agility	SaaS model enable us to be agile in the information system development project if we want to be. By agility, we mean the close co-operation with the customer and end-users, incremental and iterative development and continuous improvement in what we do.
Enabling feedback	SaaS enables us to know more about the customers of the developed system through enhanced and fast feedback channels. Customer behavior can be analyzed in real-time and proactive measures can be taken.
Enabling transparency	SaaS enables frequent releases and transparency on what has been developed and how mature the service is.

Table 5.6: Summary of the findings, RQ5.

RQ5: If a mission critical system is provided as a service, what points-of-view and requirements need to be focused on?	
Focus on quality	Through early deployment and the ability to monitor and measure the system in production, the service provider can assess the service quality with confidence, without any bottlenecks in the channel providing information on the system quality.
Focus on architecture	Again, through the early deployment and ability to measure the quality, the service architecture can be evaluated and refined, if needed. This can take place already at early phases of the development effort and again, with the confidence that the environment the information comes from represents the operative environment, because it is the same.

5.2 Implications for Theory

The main implications for the theories are related to the SaaS paradigm when applied to the development and delivery of mission critical systems. The implications are, but not necessarily limited to:

- The SaaS model, and as well the IaaS and PaaS models, should be considered as an enabler of the agile practices and as an essential support to the agile principles.
- The maintenance phase in the SDLC needs to be thought as an integral part of the specification, design, testing and deployment of the system. This is naturally much related to the DevOps model that is gaining attention and the advantages of which are slowly materializing.

Supported by recent research (Rozental et al., 2018), it seems that we are not very good at delivering almost anything without strict deadlines. The further the deadlines are set, the more relaxed the development organization feels and consequently, may not feel the need to work in a disciplined way. Thus, the frequent deliveries, supported by the ideas of continuous delivery (CD), should be seen as the new standard way of accomplishing anything.

5.3 Implications for Practice

The main implications for the practice are related to the role of the trust in the information system procurement and development process, and to all the issues related to the transformation to the service provision model. These items are handled in the following sections.

5.3.1 Enabling Trust and Agility

Mission critical systems need to work despite of the circumstances, at least for the most critical part and in the best way possible. A key aspect of the mission critical system is that the users must be able to trust the service behavior, no matter what happens. Building such trust is difficult with the traditional RFQ-initiated provisioning model is hard for two reasons; first, the needs of the users and customer remain somewhat unclear to the developers of the system, and secondly, if the system is deployed at one go, it takes a lot of time to build any confidence on its reliability.

This situation calls for a new kind of customer-provider relationship that has its roots far deeper than usually gained with public tendering relationships. Both parties should step away from the requirements, as they are interpreted in the traditional way, and set the needs and disciplined processes first. These processes must be a supporting structure, not one that restricts the design and development or one that points who to blame when something, inevitably, goes wrong.

Trust is the main enabling driver. One of the reasons for not having the needed trust is that the parties involved do not know each other or have some sort of prejudice on each other. The same applies not just the relation between the organizations, but also within the organization. Often it seems that the management of a company or organization is not trusting the experts or developers in the same organization to do their job.

To enable the agility, the information system to be developed needs to be specified in such a way that allows for incremental development. In many cases when the public tenders wish or even require the iterative and incremental approaches, the incrementality means that the requirements are classified as ones belonging to the Initial Operational Capability (IOC) and Final Operational Capability (FOC). This does not much help the incrementality as it is basically just two up-front specified waterfalls in succession.

Incremental development is an enabler of improvement also due to human characteristics. The longer it is before we get the reward on something we do, the less motivated we are (Rozental & Carlbring, 2014). Breaking far-off deadlines and goals into smaller chunks forces us to remove things that are a distraction, work in a more focused way and forces us to get started. Incremental development also suits perfectly to software-centric systems, as software is never finished in the same way as some manufactured product is.

5.3.2 SaaS as the new operating model

SaaS solutions that already are commonplace are ones that are not the core to the business. Organizations have information system like customer relations management (CRM), travel management, payroll, invoicing and some human resources (HR) applications provided as a service (Seethamraju, 2015). Software system vendors in these areas have been early adopters of cloud computing and the SaaS delivery model, already for several years.

When it comes to mission critical applications, many organizations have started to realize that there are no good reasons for the mission critical sys-

tems stay away from the SaaS model. In order to stay competitive in today's fast paced business environment, the advantages of having the mission critical applications provided as a service, leading to ease of use, frequent updated service and better scalability, reliability and performance, are needed. The same arguments hold true for public organizations. Although public organizations are famous for being slow to transform, there does appear some degree of appetite for change. In a recent study even within the public security sector, a well-known slow adopter, 56% of organizations said they are planning to procure public cloud services within the next year (Donnelly, 2019).

Companies that have never experienced the traditional product-based software provision model are free of the legacy issues related to the transformation from product-centric model to SaaS. If the software a company offers to the market was built from scratch to be a SaaS solution, it is fairly easy for them then to get on the markets when the organizations start looking for SaaS solutions, even if the SaaS solutions do not have all the features the legacy solutions had. For this kind of behavior, there are many reasons. First, as the organizations are no longer responsible for managing and maintaining the software and infrastructure, the IT resources can be used to work on other, more valuable tasks. Second, the model enables new features and bug fixes to be made frequently and, if implemented correctly, without any downtime. Thus, the end result is that the cost to implement the SaaS solution is lower while the time to market is faster.

Agility is therefore an important driver for the SaaS model. In many organizations even more compelling fact is that the SaaS solutions can be seen as an operating expense instead of being capital expenditure.

5.3.3 Competing against the SaaS solutions

The first generation of service providers offering SaaS solutions saw an opportunity to deliver better customer experience in less time with decreased costs (Kim, Jang, & Yang, 2017). In many domains, once a SaaS pioneer company entered the market with a SaaS solution, the market was radically changed. Nowadays, even traditional on-premises applications, like CAD software or situational awareness and command-and-control solutions, (Lele, 2019), are being delivered or seriously planned to be delivered as SaaS.

Vendors competing in the tendering processes with legacy software and product-centric, license-based offering are facing difficulties in many domains by the SaaS solutions. The time and effort it takes to implement product-

centric software solutions is substantial and is now becoming unacceptable in the modern era of computing. Customers can get up and running with SaaS with a minimal upfront investment whereas in the product-centric model, they must make a big investment and spend a considerable amount of resources long before they are able to get the implemented software in use.

Moving to SaaS is becoming a survival strategy in many areas of industry. To alleviate this, more and more tenders nowadays mention SaaS as an alternative solution or even the preferred solution. Consequently, the vendors are required to be able to deliver their services as SaaS, there is no option. Vendors not being able to offer a SaaS service will be considered as not following the trends and will start to see problems in responding to the tenders demanding SaaS as the model the system needs to be provided.

5.3.4 Moving to SaaS

Despite the potential benefits of SaaS, it has to be emphasized that moving to a SaaS model is not an easy undertaking. Delivering a product released every six or twelve months, shrink-wrapped with a license file is very different from delivering services on continuous basis.

Becoming a service-centric organization is a transformational change that impacts every department within a company willing to do so. From a technology standpoint, the technical requirements in the areas of security, stability, scalability, availability, recovery, and so forth require much larger investments than in a product-centric model where many of the responsibilities are shifted to the customer to take care of. In many cases, the product-centric organization has no experience in running the product they have developed in the scales the operative environments are. The test environments, be that integration, system or even some demonstration environments, are often simplistic and not like the operative environment, with all its characteristics and peculiarities.

The operations in a service-centric model also require high investments since the default expectation is that the service is always on for all the customers it is provided for and updates are performed frequently without causing any disturbance or delays to the users. The operations teams need to be metrics focused, proactive, and more responsive than in the old model where support was often based on managing a queue of incoming tickets from a ticketing system. In the service-centric model, the mindset shifts to being

proactive, closely monitoring systems to avoid customer tickets from being created in the first place.

Actually, every department in the service-centric organization will experience changes due to the SaaS operating model. This need for change relates also to departments that are not so obvious ones, like sales and finance.

The sales process becomes radically different with the SaaS model. Instead of big up-front purchases of licenses, the pricing needs to be based of the delivery of the services. Furthermore, the annual maintenance fees have no place in the SaaS offering. With SaaS, pricing needs to be based on some form of a subscription model where the up-front cost to the customer is relatively low and then the up-front cost is accompanied by a subscription fee which depends on the actual services the customer has had in their use for some time period. This setup changes the dynamics of sales. The sales process can no longer be something that is done just once and the phrase "deal is closed" has no place in the service provision world. The customer must be kept happy all the time. Furthermore, the customers are not locked in by a big upfront expenses and ownership of infrastructure and software. Instead, as the customers are just "renting" the software, breaking up with the service provider or vendor is much easier to do than it has been in the past.

The financing is impacted as well as forecasting becomes more difficult. With SaaS, there is no large upfront purchases for software followed by an annual maintenance fees, and therefore forecasting becomes more speculative of how much the subscribers will be paying each month. Also, as the SaaS solution is commonly made up of a number of modules that the customer can pick from, the customer may decide to turn off some modules to decrease their costs. Furthermore, many SaaS models have some sort of penalties to be paid to the customer organizations in cases where the service provider has not been able to provide the service in a way meeting the SLOs. In such occasions, the flow of money can even reverse, and service provision may become highly non-profitable.

To conclude, many customer organizations start seeing SaaS as the way they want their services to be provided. Consequently, many companies realize that they must be able to turn into a SaaS company to stay competitive on a specific market. Companies that think that the cloud and SaaS are only for non-mission critical applications will fall behind, and as the shift from a product-centric to service-centric is a multi-year undertaking, the longer a company waits before starting the shift, the further behind it will fall.

5.4 Limitations and Validity

A common argument related to case study-based research is that generalization of its results is limited (Yin, 2017). In general, generalizability refers to the degree to which the original data are a representative of any larger population.

While the data gathered, and the results presented are based on different projects and the work has been done in different companies—thus respecting the principles of a multiple-case study research method—it still cannot be stated that the results can be applied generally to any company operating in the service provision scheme or with the mission critical information systems. Still, it is certainly fair to argue that the findings reported are valid and conclusions applicable to future projects, although any recommendations should be applied critically and with thorough thought.

Furthermore, one may argue that the results of this study are weakened by the fact that the projects and systems involved in this research were all Finnish projects and developed by Finnish companies. Moreover, the experiences were gained by the one individual person which would allude to confirmation bias, i.e., the tendency to search for and interpret information in a way that confirms one's preexisting beliefs. The empirical data are mainly based on qualitative experiences and reflect subjective perspectives.

As already mentioned, although being a powerful tool for addressing the needs to gain more knowledge, the design research, when applied to the systems developed, brings with it serious challenges, including the following (Collins, Joseph, & Bielaczyc, 2004):

- Difficulties arising from the complexity of real-world situations and their resistance to experimental control.
- Large amounts of data arising from a need to combine ethnographic and quantitative analysis.
- Comparing across designs.

The first aspect, that of the difficulties the complexity and the size of the analyzed projects gives rise to, certainly is a valid point. When participating to a number of large-scale information system projects and experiencing problems that tend to have similar root causes, one is tempted to think that the solutions to the problems are simple: just avoiding the root causes should

result in successful outcomes. The complexity of any larger projects makes it next to impossible to have any reliable measured results on the effects of any changes done and, in many cases, the only valid measurement is based on the overall feeling the people working on the project have. The large amount of data available does not make the assessment of any experimentation any easier. One could argue that by measuring the right things, almost any modification could be either seen as beneficial or detrimental.

The comparison of the experiences from different projects and making conclusions based on those can also be impeached. Any project has its own special starting point and goals, different people with different experiences and skills working on the project, several varying external aspects that can affect the project during the execution and different criteria by which the success of the projects are assessed.

Within the research done for this thesis, we have tried to be as neutral as possible and honest in admitting that some decisions made in the analyzed projects were not optimal ones. The point here is not to find the one to blame, but to analyze, as objectively as possible, what happened and why in order to learn and be able to avoid similar mistakes in the future.

5.5 Related Research

This section presents some related research on the topics discussed.

5.5.1 Public Tendering Process

Although the topic on public tendering is discussed widely in blogs, podcasts and in media, there seems to be little real scientific research on it. This may ne partly due to the sensitiveness of the topic; often companies want to keep all the details related to the tender preparation confidential and, at the same time, the organizations running the tenders may not want to be too transparent in their tender preparation and evaluation processes.

Public tendering processes are handled somewhat in other domains and the problems and challenges seem to be alike to those met in the software industry. In the construction industry, for instance, where the innovation partnership has gained some popularity lately (Carbonara & Pellegrino, 2018; Koivisto, 2018), the academic community has been active in analyzing and improving the procurement process (Lee & Kim, 2018; Cui, Wang, Liu, & Coffey, 2019).

Although the design and development processes are and have been in focus for a long time, it seems that the starting phase, which nevertheless has a dramatic effect on the following phases, is somewhat neglected when analyzing and improving the ways the information systems are created.

5.5.2 Mission Critical Systems

Due to the importance of the mission critical systems to a society that is ever more digitized, the reliability, maintainability and security aspects, among other qualitites, are on focus on constant basis in research. However, and perhaps due to the lack of visibility into the development of mission critical systems, the research on this field is scarce.

There exists a lot of research on various aspects on critical information systems (Ciccozzi et al., 2017; Carrozza, Pietrantuono, & Russo, 2015), but not that much in the relation to how the mission critical systems are often initiated, namely though the public tendering process (Russo, Taccogna, Ciancarini, Messina, & Succi, 2018).

5.5.3 Software-as-a-Service Paradigm

The research on the SaaS model has been a hot topic for some time already, mainly due to the advantages businesses see in employing it for all information system deployments (Seethamraju, 2015; Alotaibi, 2016). Main concerns related to the wide-scale use of the paradigm are reliability, scalability and in particular, security (Tiwari & Joshi, 2016; Chouhan, Yao, & Sezer, 2015).

What is less analyzed and researched are the aspects of SaaS paradigm handled in this work, namely its ability to provide close communication, enhance the feedback mechanisms and make the co-operation between the various stakeholders more effective.

5.6 Future Work

In this section some topics that could be the targets of future research are discussed. These include, in no particular order, the importance of so called soft skills and discussions on the economics of the mission critical systems and on the possibilities related to the enhancing the tendering process.

Furthermore, the continuous operations model, enabled by the SaaS model, is discussed from the angle of how the continuous operations transform and

change the landscape of software development and delivery. This change should be seriously taken into account on both the customer and the vendor side.

Each future work topic starts with a proposal for a "research problem", a simple question that potentially could be a research question if and when some further research is performed on the topic.

5.6.1 Softer Sides of Software Development

Proposal for a research problem: "What kind of skills become the most important ones when developing and providing services?"

As discussed when addressing the research questions and implication to the practice, to enable success in the information system development projects and especially in agile development projects, one needs to have the right people with the right competences on the work. A good mix of solid competencies and soft skills enables us to build a relationship of mutual trust and respect. With mission critical systems the people matter even more than with non-critical ones. Good, disciplined people can struggle, even under wretched conditions, and produce good results (Fowler, 2004).

Providing good customer service is a prerequisite for making the users and customers happy, and while not probably being the most usual skillset mentioned when talking about people working in the software domain, the soft skills make the difference when assessing the customer service (Flood, 2011). By the soft skills we mean here practicing some simple practices when working with the customer:

- Being honest: Being honest and direct on the capabilities of the service is of utmost importance. Consumers of the service should not be surprised in a negative way when taking the service into use—be it functions or costs that might be involved in using the service. The pricing of the service must be understandable and accepted by the customer before using the service. The service provider must do proper research to present a detailed description of services and qualities of them in a language well understood by the customer. Moreover, if a customer needs something that is not available in the service, the customer should be properly informed, with an honest answer indicating when the required kind of service will be available, if ever.

- Being patient: Holding hand sounds a bit far-fetched, but it is an important part of the customer service process. If the client has never used services in the past or they have new versions of the services in their use, they need to be educated thoroughly and the service provider must provide extensive customer support as long as required.

- Providing options: The service must be configurable so that multiple options can be provided, indicating versatility to consumers. The users have different kinds of hopes and expectations, and these should be fulfilled or met at least to some extent to make users happy.

- Following up: Once the service is in place and users have started to use it, the service provider must follow the users up to see if they are satisfied with the overall experience. If they are not, then immediate action needs to be taken to shape the service up and to inform the users that the service provider is doing all things possible to make the users satisfied. The communication here needs to be timely, understandable, and solution-focused.

- Being on the same page: All team members on the service provider's side need to be on the same page in regard to customer service. There is no room for individual black sheep that neglect the users' view and do not do their share of the customer care.

One thing that is common to the items listed above, is that excelling in those does not require any real talent at all. Traditionally, in the education of the software engineer, not much emphasis has been put to the social skills, which, nevertheless, define to great extent how well a newly graduated software engineer will perform in the industry. To succeed in the career of a software professional, more and more skills like listen above are needed. In recent studies, like (Frezza et al., 2018), this aspect has gotten much more attention and support the views presented here.

The users of a service will be dealing with multiple persons in the service provider's organization and the level and quality of the customer service should not depend on the individual providing the service. Ensuring that the message received by a customer is always consistent and that no false promises or misunderstandings take place is a difficult issue. However, the service provider has no other option than to try its best to provide customer service.

It is obvious that in order to provide satisfying service, the people who are to do it need to be service-oriented. When looking at the ways the software engineering is taught in universities, the service-mindedness is not in the foreground. A combination of a great developer, a great visionary, a great salesman and a great key account manager with a right attitude is what we should be looking for instead of proficiency in programming languages or some hyped technologies.

5.6.2 Economics of the Mission Critical Systems

Proposal for a research problem: "On what basis should the costs of a mission critical system be assessed? What is the right price for a mission critical system?"

One aspect that is not dealt with in the publications but is of high relevance, is that of the price of the mission critical system or service. As in the tendering process the price of the procured service always plays an important role, it must naturally be taken into account also when assessing the SaaS-based solutions to the identified needs.

With non-critical systems, the price of the procured system can be relatively easily compared to the value provided by the system by computing the return of investment (ROI). With mission critical systems, the computation of the ROI becomes a bit more complicated: how much should a system cost, an error in which may cause the loss of a human life? Compared to some non-critical systems, how much more should one invest in the design, testing and verification of a mission critical system? And are there some hard criteria that when met, the mission critical system can be deemed as good enough for the operative use? If such criterion exists, it is likely a hard one to measure, like does the new system, if taken into operative use now, cause directly less than n (additional) deaths during its lifetime?

As a side note, the cost of the human life has, naturally, been assessed when weighing benefits versus costs in various contexts. As of 2011, the Environmental Protection Agency (EPA) set the value of a human life at 9.1 million US dollars (USD). Meanwhile, the Food and Drug Administration put it at 7.9 million USD and the Department of Transportation around 6 million USD (Partnoy, 2012). To contrast these American figures, the Finnish Transport Agency has calculated that the value of a Finnish human life is just a little below 2 million euros (Raivio, 2012).

The biggest problem in assessing the right cost is naturally time-related. We cannot know in advance the costs involved and therefore to determine the right price is impossible. Other criteria than money must be employed.

5.6.3 Enhancements to the tendering process

Proposal for a research problem: "What kind of changes are needed to the tendering process to make a successful system procurement more likely?"

The tendering process that is described in this thesis is an established process and based on the prevailing laws and regulations. Thus, one cannot expect that process itself will be changed in any dramatic way in the near future. However, to alleviate the problems related to the process, several aspects of the process can certainly be modified. These are discussed in the following.

Better Awarding

To potentially avert the risks associated with selecting the service provider, the buyer organization should consider at least these points (Malhotra, 2017):

1. Does the service provider have industry experience in relation to the expected outcomes? Do they know the domain the service is about?

2. Can the service provider provide evidence on some previous work or product to assist in alignment of value?

3. Does the service provider have industry affiliations and recognition by peers?

4. What kind of documentation is the service provider capable of providing? How the service provider can prove its operational efficiencies related to scope and development process?

5. Does or has the vendor actually completed similar scopes and provided similar outcomes with success in the past, and if there exist such claims, can those be substantiated?

6. What differentiates the vendor from the other vendors?

These questions will allow a better judgement of cost versus the value of a service provider. In general, the buyer organization should always be suspicious about the vendors unless there is enough evidence to think otherwise.

It is also essential for both the customer, i.e., the buying organization, and service providers to break the current ways of working and work through the tendering process together. Often there is a need for a reality check by either or both parties, though once an understanding is attained the outcome becomes clear and expectations can be aligned. This will allow for a better experience for all involved and build the trust in between.

Reverse Tendering

One option, that could be a good candidate to alleviate the problems involved with the RFQ-based traditional tendering process, is the so-called reverse tendering.

A reverse auction (Chen-Ritzo, Harrison, Kwasnica, & Thomas, 2005) is a type of auction in which the roles of buyer and seller are somewhat reversed. In an ordinary auction, also known as a "forward auction", buyers compete to obtain goods or services by offering increasingly higher prices. In a reverse auction, the sellers compete to obtain business from the buyer and prices will typically decrease as the sellers underbid each other.

The reverse tendering may bring some help in setting the correct price for the solution, but the problems related to the credibility of the sellers remain. To buy a mission critical system from anyone who is willing just to state the lowest price might not be the best decision. Nevertheless, any improvement that makes the buyers and the sellers to communicate more before decisions on awarding are made, is a step towards the right direction.

Innovation Partnership

As the traditional way to run the procurement procedures that combine both development and purchase elements, like those typically involved in any iterative and incremental development processes, will lead to a number of difficulties already in the phase where the tender is structured, a new method called the "Innovation Partnership" has emerged (European Commission, 2016).

The innovation partnership allows for the combination of the development and purchased elements tailored to the requirements. The innovation

partnership process takes place in three phases of the procurement procedure (European Commission, 2016):

- The competitive phase at the very beginning of the procedure, when the most suitable partner or partners are selected based on their skills and abilities. The contracts establishing the innovation partnership are awarded using the criteria of the best price-quality ratio proposed.
- The development phase, in which the partner or partners will develop the new solution in collaboration with the contracting authority. This research and development phase can be divided into several stages during which the number of partners may be gradually reduced, depending on whether they meet predetermined criteria or not.
- The commercial phase, in which the partner or partners provide the final outcomes.

The innovation partnership process should, however, only be used in limited circumstances where the following two criteria are met (European Commission, 2016):

- the goods, works and services that are sought are innovative,
- there is an intention to include both the development and purchase elements in the procedure, provided they correspond to agreed performance levels and maximum costs.

The SaaS model, at its best, could resemble a lot the innovation partnership, if just applied correctly.

5.6.4 Continuous-*

Proposal for a research problem: "How to do a smooth transformation to continuous operations?"

Related to the SaaS paradigm, numerous initiatives building on the notion of continuity have been proposed as promising future software engineering practices. These include continuous integration, testing and releasing, continuous deployment and delivery, which finally culminate in "continuous-*".

Although the advantages related to continuous processes and operations are clear and achievable in many software organizations, there are also a

number of challenges. One challenge is related to the ability of the customer organizations and the end-users therein to adapt to the drastically changed ways of working. Not all customers are happy with one day, one week, or one-month release cycles, and therefore, to be able to serve also these customers, the continuous delivery model needs to be partially reconsidered and made flexible enough to suite such customers as well.

The reasons behind the problems related to the continuous ways of working can be various, but at least four categories of root-causes can be named:

- The customer expectations are not matching the goals of implemented continuous operations.
- The processes on the service provider side are unfit for the continuous models of working.
- The processes of the customer organization are unfit for the continuous models.
- The overall maturity of the implemented continuous delivery model is insufficient.

Another challenge is that even though SaaS enables us to get feedback and empower us to improve, provided that we are measuring the right things in a right way, the importance of commitment in doing this cannot be underestimated. Studies have shown that nearly two-thirds of all software process improvement (SPI) efforts fail or at least fall short of expectations and commitment has been argued to play an important role in determining the outcome of an SPI endeavors (Abrahamsson, 2001). So, even if we get feedback and try to improve based on the information received, it is quite likely that many of the efforts for improvements will fail. This should by no means to be considered as a sign of doing the things wrong, but as a natural phenomenon related to the trial-and-error process. On the contrary, we should admit that there cannot be any real improvements if we do not fail every now and then.

Chapter 6

Conclusions

> What's next is to stop thinking about software development as a delivery process and to start thinking of it as a problem-solving process, a creative process. Time and again we run into software delivery organizations—IT departments operating as cost centers and software firms working under contract—whose job is to turn someone else's requirements into delivered software. Agile practices have helped these organizations handle requirements in smaller batches, reduce work-in-progress, and speed up software delivery. But unfortunately, agile practices do not address the underlying problem: the very idea of a software delivery organization is flawed.
>
> *from "The Lean Mindset: ask the right questions", Mary and Tom Poppendieck, 2013.*

The mission critical systems need to work despite of the circumstances, at least for the most critical part and in the best way possible. One of the key aspects of the mission critical system is that the users must be able to trust the service behavior, no matter what happens. Building such trust is difficult with the traditional RFQ-initiated system provisioning model, where the actual needs of the users and customer may remain at least somewhat unclear. The tendering processes focus typically a lot to the traditionally formatted requirements from which it is often hard to see the true needs, use cases and usage scenarios—the elements which most probably have been in the minds of the people creating the tender documents and requirements therein.

Principles that enable the provision of a mission critical system include strong, wide-band and continuous customer collaboration. To be able to respect these principles, a new kind of customer-provider relationship, something that has its roots far deeper than gained with public tendering relationships, is required. In that relationship both parties, the provider as well as the customer, must step away from looking at and playing with the requirements and set the needs and disciplined processes first. The processes involved must be a supporting and helping structure, not one that restricts or limits the design and development or one that is to blame when something goes wrong. In other words, processes need to be seen as a living thing; they evolve over time as needs evolve and if the process does not work for some part, it should be changed. Another important aspect of working trustful collaborative vendor and customer relationship is the continuous improvement in which both sides need to be committed to.

The SaaS paradigm helps a lot in providing a development and collaboration model where incremental development, customer feedback, user monitoring and system testing in the operative environment are much easier to do and natural parts of the model. However, the SaaS model is not, once again, the silver bullet and will not help if we are not otherwise willing or capable to do the required thinking and communication. **Transformation to Service Provider** is not an easy undertaking but a mandatory one to enable success in the foreseen future and the sooner companies start this transformation journey, the sooner we start to see the positive outcomes resulting from it.

There seems to be a paradox when we seem to know precisely how to develop large mission critical information system while often failing in the efforts. This behavior may relate to the same phenomenon as doing sports or eating healthy. Everyone knows that one should avoid non-healthy food and exercise, but few actually do so. Other possible reasons for the existence of this paradox might be wrong people doing the job, lack of understanding of the true nature of system development processes or lack of effective communication in general. To improve, every person involved, both on the system provider but also on the customer side, needs to understand that every day every decision, even the small ones, do influence the outcomes and therefore the success of the project. There is no easy part in the mission critical system development effort: everything is crucial, and one cannot leave some aspects out of focus. Success is never a coincidence (Kahneman & Egan, 2011), at least when dealing with large information systems, and success, or perhaps just survival, is not about doing most of the things about right but doing each

one of the things exactly right. The SaaS paradigm, again, by enabling incrementality, enhanced feedback channels in both direction and transparency, supports us in keeping the focus and remembering how even the small things matter.

After all this, one could ask, is it plausible to believe that in the near future we see better-constructed public tenders and streamlined tendering processes which lead to deployment of systems in time and on budget? Based on the long history of the RFI/RFQ processes and slowness in the changes of the related legislation, it is hard to believe that such changes happen is the near future. Even so, the first step in the path of improving the current situation is to admit that there is a problem, and that is also partly what this thesis aims for; to point out the fact that we should not continue as-is.

As a take-away, and in addition to the aspects related to the transformation of companies to service providers, there are two points which stem from the findings reported in the publications. These two points will affect a lot how well an information system project succeeds. When simplified a bit, those can be stated as:

- There needs to be enough focus on the details: discipline in the ways of working is crucial and there is no room for unfounded optimism. We need to be almost paranoid on checking that everything goes well and as planned, we measure the right things in a right way and that everyone understands what is required and when. **Providing Efficient Development Process** is certainly a trivial goal, but to achieve that constant focus and continuous improvement are needed. It is not enough to do the tasks casually, everything needs to be done as well as we can and even that might not be enough.

- There needs to be a thorough understanding on the domain and the users on both sides of the table. There is not much sense in developing anything if we are not 100% sure that what is specified and developed will fulfill the needs it is developed for. If any assumptions are made, those need to be documented and discussed thoroughly, and revisited over and over again to verify that the assumptions still hold. **Bridging the Communication Gap** remains as the challenge and the first step to overcome it is to admit that communication just does not often work very well. After that, we can start finding the solutions that work the best in a particular situation and for particular set of people.

The points above seem trivial and something that should be considered self-evident on any project. But if anyone who has been involved in any larger-scale public tender-based information system can claim that both of these points were handled appropriately, without room for improvement, and still failing, it is hard to state what else we can do. Maybe complex information systems just cannot be developed effectively?

To put it simply, the main claim in this thesis is that we are not doing the best we can, and we certainly know better what we should do when specifying and creating information systems. The end users of the systems deserve better.

References

Aamulehti. (2017). *Tampereen verkkohäiriön syy selvisi: Olemme erittäin pahoillamme keskiviikon aikana tapahtuneesta.* Retrieved 2018-11-20, from `https:// www.aamulehti.fi/kotimaa/tampereen-verkkohairion-syy-selvisi-olemme-erittain-pahoillamme-keskiviikon-aikana-tapahtuneesta-24316222` (in Finnish)

Aamulehti. (2019). *Sairaanhoidon tietoverkon kytkimet pettivät niin Pirkanmaalla kuin Helsingissäkin - Pirkanmaan häiriö oli hyvin vakava siksi, että se oli aivan totaalinen tietoliikennekatkos.* Retrieved 2019-01-10, from `https:// www.aamulehti.fi/a/201398970` (in Finnish)

Abrahamsson, P. (2001). Commitment development in software process improvement: critical misconceptions. In *Software engineering, 2001. icse 2001. proceedings of the 23rd international conference on* (pp. 71–80).

Abras, C., Maloney-Krichmar, D., & Preece, J. (2004). User-centered design. *Bainbridge, W. Encyclopedia of Human-Computer Interaction. Thousand Oaks: Sage Publications*, *37*(4), 445–456.

Aken, J. E. v. (2004). Management research based on the paradigm of the design sciences: the quest for field-tested and grounded technological rules. *Journal of management studies*, *41*(2), 219–246.

Alotaibi, M. B. (2016). Antecedents of software-as-a-service (saas) adoption: a structural equation model. *International Journal of Advanced Computer Research*, *6*(25), 114.

Aukia, P. (2017). *Hankintalaki on tekosyy - 7 neuvoa päättäjälle.* Retrieved 2019-01-10, from `https://www.codento.fi/2017/08/hankintalaki-7-neuvoa-paattajalle/` (in Finnish)

Basili, V. R., & Perricone, B. T. (1984). Software errors and complexity: an empirical investigation0. *Communications of the ACM*, *27*(1), 42–52.

Baxter, P., & Jack, S. (2008). Qualitative case study methodology: Study design and implementation for novice researchers. *The qualitative report*, *13*(4), 544–559.

BBC. (2018). *O2 4G data network restored after day-long outage.* Retrieved 2019-01-09, from `https://www.bbc.com/news/business-46464730`

Beck, K., & Gamma, E. (2000). *Extreme programming explained: embrace change*. Addison-Wesley Professional.

Benlian, A., & Hess, T. (2011). Opportunities and risks of software-as-a-service: Findings from a survey of IT executives. *Decision Support Systems*, *52*(1), 232–246.

Beyer, B., Jones, C., Petoff, J., & Murphy, N. R. (2016). *Site reliability engineering: How Google runs production systems*. " O'Reilly Media, Inc.".

Bozzano, M., & Villafiorita, A. (2010). *Design and safety assessment of critical systems*. Auerbach Publications.

Brown, D. C., Ashleigh, M. J., Riley, M. J., & Shaw, R. D. (2001). New project procurement process. *Journal of management in engineering*, *17*(4), 192–201.

Bryman, A. (2007). The research question in social research: what is its role? *International Journal of Social Research Methodology*, *10*(1), 5–20.

Buzby, C. M., Gerstenfeld, A., Voss, L. E., & Zeng, A. Z. (2002). Using lean principles to streamline the quotation process: a case study. *Industrial Management & Data Systems*, *102*(9), 513-520. Retrieved from `https://doi.org/10.1108/02635570210450190`

Bygstad, B., Aanby, H.-P., & Iden, J. (2017). Leading digital transformation: The Scandinavian way. In *Scandinavian conference on information systems* (pp. 1–14).

Calefato, F., Iaffaldano, G., Lanubile, F., & Vasilescu, B. (2018). On developers' personality in large-scale distributed projects: The case of the Apache ecosystem. *arXiv preprint arXiv:1803.01126*.

Carbonara, N., & Pellegrino, R. (2018). Delivering innovation in public infrastructure through public private partnerships. In *Geography, open innovation and entrepreneurship*. Edward Elgar Publishing.

Carrozza, G., Pietrantuono, R., & Russo, S. (2015). Defect analysis in mission-critical software systems: a detailed investigation. *Journal of Software: Evolution and Process*, *27*(1), 22–49.

Chen-Ritzo, C.-H., Harrison, T. P., Kwasnica, A. M., & Thomas, D. J. (2005). Better, faster, cheaper: An experimental analysis of a multiattribute reverse auction mechanism with restricted information feedback. *Management Science*, *51*(12), 1753–1762.

Chouhan, P. K., Yao, F., & Sezer, S. (2015). Software as a service: Understanding security issues. In *2015 science and information conference (sai)* (pp. 162–170).

Ciccozzi, F., Crnkovic, I., Di Ruscio, D., Malavolta, I., Pelliccione, P., & Spalazzese, R. (2017). Model-driven engineering for mission-critical iot systems. *IEEE software*, *34*(1), 46–53.

Collins, A., Joseph, D., & Bielaczyc, K. (2004). Design research: Theoretical and methodological issues. *The Journal of the learning sciences*, *13*(1), 15–42.

Cui, C., Wang, J., Liu, Y., & Coffey, V. (2019). Relationships among value-for-money drivers of public–private partnership infrastructure projects. *Journal of Infrastructure Systems*, *25*(2), 04019007.

Davis, A., Overmyer, S., Jordan, K., Caruso, J., Dandashi, F., Dinh, A., ... others (1993). Identifying and measuring quality in a software requirements specification. In *Proceedings first international software metrics symposium* (pp. 141–152).

Deber, J., Jota, R., Forlines, C., & Wigdor, D. (2015). How much faster is fast enough?: User perception of latency & latency improvements in direct and indirect touch. In *Proceedings of the 33rd annual acm conference on human factors in computing systems* (pp. 1827–1836).

De Wit, A. (1988). Measurement of project success. *International journal of project management*, *6*(3), 164–170.

Donnelly, C. (2019). *Emergency services organisations ranked slowest public sector cloud adopters in Eduserv poll.* Retrieved 2019-01-24, from `https://www.computerweekly.com/news/252456374/Emergency-services-organisations-ranked-slowest-public-sector-cloud-adopters-in-Eduserv-poll`

Dubey, A., & Wagle, D. (2007). Delivering software as a service. *The McKinsey Quarterly*, *6*(2007), 2007.

Dwivedi, Y. K., Wastell, D., Laumer, S., Henriksen, H. Z., Myers, M. D., Bunker, D., ... Srivastava, S. C. (2015). Research on information systems failures and successes: Status update and future directions. *Information Systems Frontiers*, *17*(1), 143–157.

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and software technology*, *50*(9-10), 833–859.

Dzubak, C. M. (2008). Multitasking: The good, the bad, and the unknown. *The Journal of the Association for the Tutoring Profession*, *1*(2), 1–12.

Easterbrook, S., Singer, J., Storey, M.-A., & Damian, D. (2008). Selecting empirical methods for software engineering research. In *Guide to advanced empirical software engineering* (pp. 285–311). Springer.

Eisenhardt, K. M. (1991). Better stories and better constructs: The case for rigor and comparative logic. *Academy of Management review*, *16*(3), 620–627.

European Commission. (2016). *Innovation partnerships keep public services up to date.* Retrieved 2018-12-02, from `https://ec.europa.eu/growth/content/8699-innovation-partnerships-keep-public-services-date`

Fagerholm, F. (2015). *Software developer experience: Case studies in lean-agile and open source environments* (PhD thesis). Helsingin yliopisto.

Fernández, D. M., Wagner, S., Kalinowski, M., Felderer, M., Mafra, P., Vetrò, A., ... Wieringa, R. (2017). Naming the pain in requirements engineering. *Empirical software engineering*, *22*(5), 2298–2338.

Ferreira, J., Sharp, H., & Robinson, H. (2010). Values and assumptions shaping agile development and user experience design in practice. In *International conference on agile software development* (pp. 178–183).

Ferriere, T. (2018). *Writing an RFQ vs RFP: The difference.* Retrieved 2018-12-20, from `https://tenderspage.com/writing-rfq-vs-rfp-difference/`

Flood, T. (2011). The soft side of SaaS: Implications for IT in higher education. *ECAR Research Bulletin*, *1*(6), 1–12.

Florentine, S. (2016). *More than half of IT projects still failing.* Retrieved 2018-12-29, from `https://www.cio.com/article/3068502/project-management/more-than-half-of-it-projects-still-failing.html`

Fowler, K. (2004). Mission-critical and safety-critical development. *IEEE Instrumentation & Measurement Magazine*, *7*(4), 52–59.

Frezza, S., Pears, A., Kann, V., Kapoor, A., Peters, A.-K., Sabin, M., ... Wallace, C. (2018, 11). *Modelling competencies for computing education beyond 2020 a research based approach to defining competencies in the computing disciplines.* doi: 10.13140/RG.2.2.11776.48642

Fuchs, C., & Hess, T. (2018). Becoming agile in the digital transformation: The process of a large-scale agile transformation. In *Thirty ninth international conference on information systems* (pp. 1–17).

Goatham, R. (2017). *Catalogue of catastrophe.* Retrieved 2019-01-09, from http://calleam.com/WTPF/

Gottesman, S. (2014). *What does N+1 redundancy mean?* Retrieved 2019-01-20, from https://searchservervirtualization.techtarget.com/answer/What-does-N1-redundancy-mean

Gustafsson, J. (2017). *Single case studies vs. multiple case studies: A comparative study.* Retrieved 2019-01-12, from http://www.diva-portal.org/smash/get/diva2:1064378/FULLTEXT01.pdf

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

Hofstadter, D. R. (1980). *Gödel, Escher, Bach.* Vintage Books New York.

Humble, J., & Farley, D. (2010). *Continuous delivery: reliable software releases through build, test, and deployment automation.* Pearson Education.

IEEE Computer Society. (1998, Oct). IEEE recommended practice for software requirements specifications. *IEEE Std 830-1998*, 1-40. doi: 10.1109/IEEESTD.1998.88286

Independent. (2018). *British Airways chaos at London Heathrow as cancelled and delayed flights affect 10,000 passengers.* Retrieved 2019-01-09, from https://www.bbc.com/news/business-46464730

Ismail, N. (2018). *Why IT projects continue to fail at an alarming rate.* Retrieved 2018-12-31, from https://www.information-age.com/projects-continue-fail-alarming-rate-123470803/

IT World. (2015). *That's one big repository: Here's how many lines of code Google has.* Retrieved 2018-11-09, from https://www.itworld.com/article/2985099/application-management/thats-one-big-repository-heres-how-many-lines-of-code-google-has.html

Kahneman, D., & Egan, P. (2011). *Thinking, fast and slow* (Vol. 1). Farrar, Straus and Giroux, New York.

Kavis, M. (2014). *Cloud's next big wave: Mission critical applications.* Retrieved 2018-11-30, from https://www.forbes.com/sites/mikekavis/2014/06/27/clouds-next-big-wave-mission-critical-applications

Kavis, M. J. (2014). *Architecting the cloud: design decisions for cloud computing service models (SaaS, PaaS, and IaaS).* John Wiley & Sons.

Ketola, P. (2006). On out-of-box experience and online support. In *Proc. 20th international symposium on human factors in telecommunication, Sophia Antipolis, France.*

Kim, S. H., Jang, S. Y., & Yang, K. H. (2017). Analysis of the determinants of software-as-a-service adoption in small businesses: Risks, benefits, and organizational and environmental factors. *Journal of Small Business Management*, *55*(2), 303–325.

Klijn, E.-H., & Teisman, G. R. (2003). Institutional and strategic barriers to public-private partnership: An analysis of Dutch cases. *Public money and Management*, *23*(3), 137–146.

Knight, L., Harland, C., & Telgen, J. (2012). Public procurement in perspective. In *Public procurement* (pp. 44–52). Routledge.

Koechlin, E., & Hyafil, A. (2007). Anterior prefrontal function and the limits of human decision-making. *Science*, *318*(5850), 594–598.

Koivisto, J. (2018). Co-designing an outcome-based public procurement: Early involvements, participations and orderings. *Journal of Public Procurement*, *18*(4), 323–335.

Konig, C. J., Buhner, M., & Murling, G. (2005). Working memory, fluid intelligence, and attention are predictors of multitasking performance, but polychronicity and extraversion are not. *Human performance*, *18*(3), 243–266.

Krishna, R., Agrawal, A., Rahman, A., Sobran, A., & Menzies, T. (2018). What is the connection between issues, bugs, and enhancements?: lessons learned from 800+ software projects. In *Proceedings of the 40th international conference on software engineering: Software engineering in practice* (pp. 306–315).

Lauesen, S. (2002). *Software requirements: styles and techniques.* Pearson Education.

Lauesen, S. (2004). COTS tenders and integration requirements. In *Proceedings of the 12th ieee international Requirements Engineering conference* (pp. 166–175).

Lauesen, S. (2005). *User interface design: a software engineering perspective.* Pearson Education.

Lauesen, S., & Vium, J. P. (2005). Communication gaps in a tender process. *Requirements Engineering*, *10*(4), 247–261.

Lee, H., & Kim, K. (2018). Traditional procurement versus public–private partnership: A comparison of procurement modalities focusing on bundling contract effects. *Asian Development Bank Economics Working Paper Series*(560).

Lele, A. (2019). Cloud computing. In *Disruptive technologies for the militaries and security* (pp. 167–185). Springer.

Lutz, R. R. (1993). Analyzing software requirements errors in safety-critical, embedded systems. In *Proceedings of IEEE international symposium on Requirements Engineering* (pp. 126–133).

Malhotra, G. (2017). *There is always someone cheaper.* Retrieved 2019-01-07, from `https://www.linkedin.com/pulse/always-someone-cheaper-gagan-malhotra/`

Martin, R. C. (2002). *Agile software development: principles, patterns, and practices.* Prentice Hall.

Mhay, S., & Coburn, C. (2018). *What's the difference between contract RFT RFQ RFP RFI?* Retrieved 2018-11-20, from `https://www.negotiations.com/articles/procurement-terms/`

Partnoy, F. (2012). *Wait: The art and science of delay.* Public Affairs.

Patel, P., Ranabahu, A. H., & Sheth, A. P. (2009). Service level agreement in cloud computing. *KNO.E.SIS PUBLICATIONS*.

Patton, M. Q. (1990). *Qualitative evaluation and research methods.* SAGE Publications, inc.

Raivio, J. (2012). Hengen hinta. *Suomen Kuvalehti*. Retrieved from `https://suomenkuvalehti.fi/nurkanvaltaaja/paljonko-maksaa-yksi-kolarissa-kuollut-ihminen/`

Rogerson, W. P. (1994). Economic incentives and the defense procurement process. *Journal of Economic Perspectives*, *8*(4), 65–90.

Ross, A., Hilton, A., & Rensin, D. (2017). *SLOs, SLIs, SLAs, oh my - CRE life lessons.* Retrieved 2018-12-09, from `https://cloud.google.com/blog/products/gcp/availability-part-deux-CRE-life-lessons`

Rozental, A., Bennett, S., Forsström, D., Ebert, D. D., Shafran, R., Andersson, G., & Carlbring, P. (2018). Targeting procrastination using psychological treatments: A systematic review and meta-analysis. *Frontiers in psychology*, *9*.

Rozental, A., & Carlbring, P. (2014). Understanding and treating procrastination: a review of a common self-regulatory failure. *Psychology*, *5*(13), 1488.

Russo, D., Taccogna, G., Ciancarini, P., Messina, A., & Succi, G. (2018). Contracting agile developments for mission critical systems in the public sector. In *Proceedings of the 40th international conference on software engineering: Software engineering in society* (pp. 47–56).

Sale, J. E., Lohfeld, L. H., & Brazil, K. (2002). Revisiting the quantitative-qualitative debate: Implications for mixed-methods research. *Quality and quantity*, *36*(1), 43–53.

Schmidt, R., Lyytinen, K., Keil, M., & Cule, P. (2001). Identifying software project risks: An international Delphi study. *Journal of management information systems*, *17*(4), 5–36.

Schmitt, B. H. (2010). *Customer experience management: A revolutionary approach to connecting with your customers*. John Wiley & Sons.

Seethamraju, R. (2015). Adoption of software as a service (SaaS) enterprise resource planning (ERP) systems in small and medium sized enterprises (SMEs). *Information systems frontiers*, *17*(3), 475–492.

Senecal, C., Koestner, R., & Vallerand, R. J. (1995). Self-regulation and academic procrastination. *The Journal of Social Psychology*, *135*(5), 607–619.

Spafford, G. (2015). *Short, random thought on testing.* Retrieved 2019-01-13, from `https://www.cerias.purdue.edu/site/blog/post/short_random_thought_on_testing`

Stake, R. E. (1995). *The art of case study research*. Sage.

Stake, R. E. (2013). *Multiple case study analysis*. Guilford Press.

Techopedia. (2018). *Mission critical system.* Retrieved 2018-12-08, from `https://www.techopedia.com/definition/23583/mission-critical-system`

Thai, K. V. (2001). Public procurement re-examined. *Journal of public procurement*, *1*(1), 9–50.

Thimbleby, H. (2007). User-centered methods are insufficient for safety critical systems. In *Symposium of the Austrian HCI and usability engineering group* (pp. 1–20).

Tiwari, P. K., & Joshi, S. (2016). Data security for software as a service. In *Web-based services: Concepts, methodologies, tools, and applications* (pp. 864–880). IGI Global.

Tomingas, P. (2018). *The CPO's guide to strategic sourcing, Chapter 7: RFQ management process.* Retrieved 2018-12-07, from `https://www.deltabid.com/rfq-process/`

Trienekens, J. J., Bouman, J. J., & Van Der Zwan, M. (2004). Specification of service level agreements: Problems, principles and practices. *Software Quality Journal*, *12*(1), 43–57.

Turner, M., Budgen, D., & Brereton, P. (2003). Turning software into a service. *Computer*, *36*(10), 38–44.

Vaes, K. (2014). *Best practices regarding the creation of an "RFP" (aka "Request For Proposal").* Retrieved 2018-12-20, from `https://kvaes.wordpress.com/2014/11/07/best-practices-regarding-the-creation-of-an-rfp-aka-request-for-proposal/`

Vaishnavi, V., & Kuechler, W. (2004). *Design research in information systems.* Retrieved 2019-01-02, from `http://www.desrist.org/desrist/content/design-science-research-in-information-systems.pdf`

Waters, B. (2005). Software as a service: A look at the customer benefits. *Journal of Digital Asset Management*, *1*(1), 32–39.

Wiegers, K., & Beatty, J. (2013). *Software requirements.* Pearson Education.

Wiio, O. A. (1978). *Wiion lait — ja vähän muidenkin.* Weilin+ Göös. (in Finnish)

Yin, R. K. (2017). *Case study research and applications: Design and methods.* Sage publications.

Zin, A. M., & Pa, N. (2009). Measuring communication gap in software requirements elicitation process. In *Proceedings of the 8th WSEAS international conference on software engineering, parallel and distributed systems* (pp. 66–71).

Zowghi, D., & Nurmuliani, N. (2002). A study of the impact of requirements volatility on software project performance. In *Ninth Asia-Pacific software engineering conference* (pp. 1–9).

TIETOJENKÄSITTELYTIETEEN OSASTO
PL 68 (Pietari Kalmin katu 5)
00014 Helsingin yliopisto

DEPARTMENT OF COMPUTER SCIENCE
P.O. Box 68 (Pietari Kalmin katu 5)
FI-00014 University of Helsinki, FINLAND

JULKAISUSARJA **A**

SERIES OF PUBLICATIONS **A**

Reports are available on the e-thesis site of the University of Helsinki.

A-2013-1 M. Timonen: Term Weighting in Short Documents for Document Categorization, Keyword Extraction and Query Expansion. 53+62 pp. (Ph.D. Thesis)

A-2013-2 H. Wettig: Probabilistic, Information-Theoretic Models for Etymological Alignment. 130+62 pp. (Ph.D. Thesis)

A-2013-3 T. Ruokolainen: A Model-Driven Approach to Service Ecosystem Engineering. 232 pp. (Ph.D. Thesis)

A-2013-4 A. Hyttinen: Discovering Causal Relations in the Presence of Latent Confounders. 107+138 pp. (Ph.D. Thesis)

A-2013-5 S. Eloranta: Dynamic Aspects of Knowledge Bases. 123 pp. (Ph.D. Thesis)

A-2013-6 M. Apiola: Creativity-Supporting Learning Environments: Two Case Studies on Teaching Programming. 62+83 pp. (Ph.D. Thesis)

A-2013-7 T. Polishchuk: Enabling Multipath and Multicast Data Transmission in Legacy and Future Interenet. 72+51 pp. (Ph.D. Thesis)

A-2013-8 P. Luosto: Normalized Maximum Likelihood Methods for Clustering and Density Estimation. 67+67 pp. (Ph.D. Thesis)

A-2013-9 L. Eronen: Computational Methods for Augmenting Association-based Gene Mapping. 84+93 pp. (Ph.D. Thesis)

A-2013-10 D. Entner: Causal Structure Learning and Effect Identification in Linear Non-Gaussian Models and Beyond. 79+113 pp. (Ph.D. Thesis)

A-2013-11 E. Galbrun: Methods for Redescription Mining. 72+77 pp. (Ph.D. Thesis)

A-2013-12 M. Pervilä: Data Center Energy Retrofits. 52+46 pp. (Ph.D. Thesis)

A-2013-13 P. Pohjalainen: Self-Organizing Software Architectures. 114+71 pp. (Ph.D. Thesis)

A-2014-1 J. Korhonen: Graph and Hypergraph Decompositions for Exact Algorithms. 62+66 pp. (Ph.D. Thesis)

A-2014-2 J. Paalasmaa: Monitoring Sleep with Force Sensor Measurement. 59+47 pp. (Ph.D. Thesis)

A-2014-3 L. Langohr: Methods for Finding Interesting Nodes in Weighted Graphs. 70+54 pp. (Ph.D. Thesis)

A-2014-4 S. Bhattacharya: Continuous Context Inference on Mobile Platforms. 94+67 pp. (Ph.D. Thesis)

A-2014-5 E. Lagerspetz: Collaborative Mobile Energy Awareness. 60+46 pp. (Ph.D. Thesis)

A-2015-1 L. Wang: Content, Topology and Cooperation in In-network Caching. 190 pp. (Ph.D. Thesis)

A-2015-2 T. Niinimäki: Approximation Strategies for Structure Learning in Bayesian Networks. 64+93 pp. (Ph.D. Thesis)

A-2015-3 D. Kempa: Efficient Construction of Fundamental Data Structures in Large-Scale Text Indexing. 68+88 pp. (Ph.D. Thesis)

A-2015-4 K. Zhao: Understanding Urban Human Mobility for Network Applications. 62+46 pp. (Ph.D. Thesis)

A-2015-5 A. Laaksonen: Algorithms for Melody Search and Transcription. 36+54 pp. (Ph.D. Thesis)

A-2015-6 Y. Ding: Collaborative Traffic Offloading for Mobile Systems. 223 pp. (Ph.D. Thesis)

A-2015-7 F. Fagerholm: Software Developer Experience: Case Studies in Lean-Agile and Open Source Environments. 118+68 pp. (Ph.D. Thesis)

A-2016-1 T. Ahonen: Cover Song Identification using Compression-based Distance Measures. 122+25 pp. (Ph.D. Thesis)

A-2016-2 O. Gross: World Associations as a Language Model for Generative and Creative Tasks. 60+10+54 pp. (Ph.D. Thesis)

A-2016-3 J. Määttä: Model Selection Methods for Linear Regression and Phylogenetic Reconstruction. 44+73 pp. (Ph.D. Thesis)

A-2016-4 J. Toivanen: Methods and Models in Linguistic and Musical Computational Creativity. 56+8+79 pp. (Ph.D. Thesis)

A-2016-5 K. Athukorala: Information Search as Adaptive Interaction. 122 pp. (Ph.D. Thesis)

A-2016-6 J.-K. Kangas: Combinatorial Algorithms with Applications in Learning Graphical Models. 66+90 pp. (Ph.D. Thesis)

A-2017-1 Y. Zou: On Model Selection for Bayesian Networks and Sparse Logistic Regression. 58+61 pp. (Ph.D. Thesis)

A-2017-2 Y.-T. Hsieh: Exploring Hand-Based Haptic Interfaces for Mobile Interaction Design. 79+120 pp. (Ph.D. Thesis)

A-2017-3 D. Valenzuela: Algorithms and Data Structures for Sequence Analysis in the Pan-Genomic Era. 74+78 pp. (Ph.D. Thesis)

A-2017-4 A. Hellas: Retention in Introductory Programming. 68+88 pp. (Ph.D. Thesis)

A-2017-5 M. Du: Natural Language Processing System for Business Intelligence. 78+72 pp. (Ph.D. Thesis)

A-2017-6 A. Kuosmanen: Third-Generation RNA-Sequencing Analysis: Graph Alignment and Transcript Assembly with Long Reads. 64+69 pp. (Ph.D. Thesis)

A-2018-1 M. Nelimarkka: Performative Hybrid Interaction: Understanding Planned Events across Collocated and Mediated Interaction Spheres. 64+82 pp. (Ph.D. Thesis)

A-2018-2 E. Peltonen: Crowdsensed Mobile Data Analytics. 100+91 pp. (Ph.D. Thesis)

A-2018-3 O. Barral: Implicit Interaction with Textual Information using Physiological Signals. 72+145 pp. (Ph.D. Thesis)

A-2018-4 I. Kosunen: Exploring the Dynamics of the Biocybernetic Loop in Physiological Computing. 91+161 pp. (Ph.D. Thesis)

A-2018-5 J. Berg: Solving Optimization Problems via Maximum Satisfiability: Encodings and Re-Encodings. 86+102 pp. (Ph.D. Thesis)

A-2018-6 J. Pyykkö: Online Personalization in Exploratory Search. 101+63 pp. (Ph.D. Thesis)

A-2018-7 L. Pivovarova: Classification and Clustering in Media Monitoring: from Knowledge Engineering to Deep Learning. 78+56 pp. (Ph.D. Thesis)

A-2019-1 K. Salo: Modular Audio Platform for Youth Engagement in a Museum Context. 97+78 pp. (Ph.D. Thesis)