Thesis for The Degree of Doctor of Philosophy

Requirements Engineering that Balances Agility of Teams and System-level Information Needs at Scale

Rashidah Kasauli Namisanvu

Division of Software Engineering
Department of Computer Science & Engineering
Chalmers University of Technology and Gothenburg University
Gothenburg, Sweden, 2020

Requirements Engineering that Balances Agility of Teams and System-level Information Needs at Scale

Rashidah Kasauli Namisanvu

ISBN 978-91-7905-331-4
Doktorsavhandlingar vid Chalmers tekniska högskola, Ny serie nr 4798.
ISSN 0346-718X

Technical Report No 184D
Department of Computer Science & Engineering
Division of Software Engineering
Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden

This thesis has been prepared using LaTeX.
Printed by Chalmers Reproservice,
Gothenburg, Sweden 2020.

To my children.
May you be inspired!

Abstract

Context: Motivated by their success in software development, large-scale systems development companies are increasingly adopting agile methods and their practices. Such companies need to accommodate different development cycles of hardware and software and are usually subject to regulation and safety concerns. Also, for such companies, requirements engineering is an essential activity that involves upfront and detailed analysis which can be at odds with agile development methods.

Objective: The overall aim of this thesis is to investigate the challenges and solution candidates of performing effective requirements engineering in an agile environment, based on empirical evidence. Illustrated with studies on safety and system-level information needs, we explore RE challenges and solutions in large-scale agile development, both in general and from the teams' perspectives.

Method: To meet our aim, we performed a secondary study and a series of empirical studies based on case studies. We collected qualitative data using interviews, focus groups and workshops to derive challenges and potential solutions from industry.

Findings: Our findings show that there are numerous challenges of conducting requirements engineering in agile development especially where systems development is concerned. The challenges discovered sprout from an integration problem of working with agile methods while relying on established plan-driven processes for the overall system. We highlight the communication challenge of crossing the boundary of agile methods and system-level (or plan-driven) development, which also proves the coexistence of both methods.

Conclusions: Our results highlight the painful areas of requirements engineering in agile development and propose solutions that can be explored further. This thesis contributes to future research, by establishing a holistic map of challenges and candidate solutions that can be further developed to make RE more efficient within agile environments.

Keywords

Requirements Engineering, Systems Development, Coordination, Large-scale Agile, Plan-driven, Empirical Research, Methods Co-existence, Safety-critical System Development

Acknowledgment

Working between two separate environments, Uganda and Sweden, the last four years have been quite an experience for me. I have interacted with many interesting, supportive and knowledgeable people that I would like to convey my gratitude to.

My heartfelt gratitude goes to my main supervisor Eric Knauss for the exceptional supervision that has changed my life in ways I had never envisioned. Through your positive attitude, rigor and confidence in my abilities, I have experienced the true spirit of research. Thank you!

To my co-supervisor Benjamin Kanagwa, I appreciate the encouragement, and calming sentiments that kept me strong. My examiner Ivica Crnkovic, thank you for guiding my research in a great and supportive way.

Special gratitude goes to all my 16 co-authors including Grischa Liebel, Jennifer Horkoff, Francisco Gomes, Rebekka Wohlrab, Jan-Philip Steghöfer, Salome Maro, Agneta Nilsson, Gul Calikli. It has been loads of fun working among you. Thank you for always sharing insightful thoughts and giving constructive feedback whenever asked.

Michel Chaudron, Engineer Bainomugisha and the SIDA BRIGHT project team in Uganda and Sweden. Thank you for making my PhD dream a reality. Special gratitude goes to Pär Meiling for always ensuring we have housing in Sweden. Thank you Pär.

To all colleagues and administrative staff at the software engineering division of Chalmers CSE department, thank you for the friendly environment. Léuson Mario Pedro da Silva, thank you for the relaxing jokes. Alessia Knauss, thank you for helping me stay healthy and sane.

My fellow PhD students on BRIGHT especially Hawa Nyende, Micheal Kizito, Dragule Swaib, David Bamutura and Grace Kobusinge. I have enjoyed the shared frustrations :) and happiness in travelling together.

Colleagues at Makerere SCIT especially Joseph Balikuddembe, Mary Nsabagwa, Jacob Katende, Hasifa Namatovu, Moses Ntanda, Joyce Nakatumba. Thank you for your motivation and support.

My father Hajji M.Z. Kasauli and mothers Cornety Kivumbi, and Rahma Makumbi. Thank you for your constant prayers. I know you are my miracle! My brothers Yasin, Shakie, Umar, Zed, and sisters Madiina, Aisha, Sharifah, Shamirah, Rukia, and Shakirah. The Kasaulis. The onus is now on you. :D

To my husband Bashir Mwebe, thank you for encouraging and supporting me to pursue this degree. You showed me that I could endure and never allowed me to give up. To my children: Nushin, Rahma, Rabiib, Rithwana, and Raihan, thank you for your prayers and endurance.

Alhamdulillah! Alhamdulillah! Alhamdulillah! None of this would be possible without your might Allah. Thank you Allah for putting me in the company of those wonderful and supportive people these last four years. May you continue to bless them all!

Funding: This research was funded in part by the Swedish International Development Cooperation Agency (Sida) and Makerere University under Sida contribution No: 51180060; Project No. 317; Building Research Capacity in Innovative Information and Communication Technologies for Development (ICT4D) for Sustainable Socio-economic Growth in Uganda (BRIGHT).

List of Publications

Appended publications

This thesis is based on the following publications:

[A] R. Kasauli, G. Liebel, E. Knauss, S. Gopakumar and B. Kanagwa, "Requirements Engineering Challenges in Large-Scale Agile System Development"
In IEEE 25th International Requirements Engineering Conference (RE'17), Lisbon, Portugal, September 4–8, 2017.

[B] R. Kasauli, E. Knauss, B. Kanagwa, A. Nilsson, G. Calikli, "Safety-Critical Systems and Agile Development: A Mapping Study"
In 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Praque, Czech Republic, August 29–31 2018.

[C] R. Kasauli, E. Knauss, J. Nakatumba-Nabende, B. Kanagwa, "Agile Islands in a Waterfall Environment: Challenges and Strategies in Automotive"
In Evaluation and Assessment in Software Engineering (EASE'20), Trondheim, Norway, April 15–17, 2020.

[D] R. Kasauli, E. Knauss, J. Horkoff, G. Liebel, F. Gomes, "Requirements Engineering Challenges and Practices in Large-Scale Agile Systems Development"
Journal submission: In the process of requested minor revision.

[E] R. Kasauli, R. Wohlrab, E. Knauss, J.P. Steghöfer, J. Horkoff, S. Maro, "Charting Coordination Needs in Large-Scale Agile Organisations with Boundary Objects and Methodological Islands"
Accepted In International Conference on Software and System Processes (ICSSP2020), Seoul, South Korea, 2020.

[F] R. Kasauli and E. Knauss. "A Simplified Guide to the use of T-Reqs"
Technical report Extending
E. Knauss, G. Liebel, J. Horkoff, R. Wohlrab, R. Kasauli, F. Lange, and P. Gildert. "T-Reqs: Tool Support for Managing Requirements in Large-Scale Agile System Development."
In IEEE 26th International Requirements Engineering Conference (RE'18), Banff, Alberta, Canada, August 20–24, 2018.

Other publications

The following publications were published during my PhD studies. However, they are not appended to this thesis, due to contents not related to the thesis.

[a] R. Kasauli. "Requirements Engineering for Large Scale Agile Systems Development."
In Requirements Engineering: Foundation for Software Quality (REFSQ) Workshops (Doctoral Symposium), Essen, Germany, February 26 – March 2, 2017.

[b] R. Kasauli, E. Knauss, A. Nilsson, S. Klug. "Adding Value Every Sprint: A Case Study on Large-Scale Continuous Requirements Engineering"
In 3rd Workshop on Continuous Requirements Engineering, REFSQ Workshops, Essen, Germany, February 26–March 2, 2017.

[c] E. Knauss, G. Liebel, K. Schneider, J. Horkoff, and R. Kasauli. "Quality Requirements in Agile as a Knowledge Management Problem: More than Just-in-Time."
In IEEE 25th International Requirements Engineering Conference Workshops (REW), Lisbon, Portugal, September 4–8, 2017.

[d] F.G. de Oliveira Neto, J. Horkoff, E. Knauss, R. Kasauli, and G. Liebel. "Challenges of Aligning Requirements Engineering and System Testing in Large-Scale Agile: A Multiple Case Study."
In IEEE 25th International Requirements Engineering Conference Workshops (REW), Lisbon, Portugal, September 4–8, 2017.

[e] R. Kasauli. "Requirements Engineering Challenges of Supporting Agile Teams in System Development."
Licentiate Thesis, Technical Report No 190L, ISSN 1652-876X, Department of Computer Science & Engineering, Chalmers University of Technology and Göteborg University, 2018.

[f] R. Kasauli, G. Liebel, E. Knauss, S. Gopakumar, B. Kanagwa, "Requirements Engineering Challenges in Large-Scale Agile System Development."
Multikonferenz Software Engineering & Management (SE), 2018

Research Contribution

All the included papers were published with collaborations from colleagues. I am the main author of five of the six included papers and as such responsible for the research design, dividing the work between co-authors and performing most of the writing. In particular, I have participated in the included papers as follows:

In papers A–C, the planning, design, execution of the research and writing were mostly done by me. For paper D and E, I participated in part of the data collection, participated and coordinated the writing of the papers. Paper F is a tool paper which I extend with a technical report expanding on the 2 paged document that was published. Here I, therefore, improve my contribution to include formal documentation of the tool.

Contents

Chapter 1

Introduction

"*It isn't just that businesses use more software, but that, increasingly, a business is defined in software. That is, the core processes a business executes–from how it produces a product, to how it interacts with customers, to how it delivers services–are increasingly specified, monitored, and executed in software. This is . . . a transition that is spreading to all kinds of companies, regardless of the product or service they provide.*' –Jay Kreps CEO of Confluent

Software has pervaded our lives and is continuously gaining importance. Seen as a driver for innovation, even the formally hardware-based systems like automobiles are becoming more software-oriented than before [1, 2]. This increased use of software has led to more software-intensive systems, i.e. systems that consist of software, hardware and possibly mechatronic parts defining the context in which they are used. Such systems include, e.g., telecommunications and automotive systems. At the large-scale, for such software-intensive systems requirements engineering is the key to success [3, 4].

Requirements Engineering (RE) is traditionally a sequential process where the execution of, for instance, software development requires indisputable completion of the requirements specification phase [3, 5]. This traditional approach to RE has formed the foundation on which many large-scale systems companies are built. These companies often have to deal with standards and regulations [6], along with parallel development of hardware and software. With advancement in software, and new players coming into the market, competition has increased and customer demands are evolving much faster, making reliance on traditional (plan-driven) methods, with their long lead times and lack of flexibility, less of an option. Thus large-scale systems development companies are seeking better approaches that allow flexibility, a characteristic of agile development methods.

Although initially meant for development at a small scale [7], agile development methods are increasingly adopted by large-scale systems development companies [8–10]. On top of the flexibility that agile methods provide, their adoption at scale is driven by reported success in handling changing customer demands, achieving shorter time-to-market and improved quality outputs [11]. However, their adoption at scale is challenging, not only because of the scale but also the foundation on which many of these companies are built that calls for a sequential adoption [12–14].

Agile development promotes customer collaboration which is in line with RE. Thus, RE and Agile development seem to support each other. However, long upfront analysis–a phase in RE commonly leading to extensive documentation–is considered anti-agile creating some friction between RE and agile methods. Existing work on this friction has addressed practices and their challenges [15,16] while also commenting on synergies and conflicts of traditional RE with agile [17], without focusing on RE in large-scale agile systems development. This research attempts to address that friction while focusing on large systems' development.

We approach the problem through a series of empirical studies that discover the information needs and related knowledge, pertinent to systems development. The overall aim of this thesis is to investigate the challenges and solution candidates of performing effective RE in an agile environment, based on empirical evidence. We explore RE challenges and solutions in large-scale agile systems development, both in the general and the teams' perspectives. The new knowledge and methods presented in this thesis can be used to inform process and tool design in large-scale agile system development. Once the gap between agile and traditional methods is addressed, many challenges relating to coordination and knowledge management will have been combated. Ultimately, large-scale companies have met their agile adoption goals.

The thesis is composed of two parts, the introduction part (Chapter 1) and the second part is an attachment of the included papers. The rest of this chapter is structured as follows: Section 1.1 presents the background and related work of the research presented in this thesis. Section 1.2 presents our research questions. The research methodology is described in Section 1.3 while Section 1.4 provides a synthesis of our research outputs. In Section 1.5, we give conclusions and future work. For the second part, we have Chapter 2 to Chapter 7 with Papers A–F respectively.

1.1 Background and Related Work

The adoption of agile methods has changed the way RE is interpreted in development. According to Leffingwell, "*No matter the specific method, agile's treatment of requirements is fundamentally different*" [18]. Whereas some argue that RE can be viewed from two different angles; 1) as a formal and structured transformation of information [17] (e.g. traditionalists) or 2) as a collaborative effort relying on the creativity and competence of the involved engineers [19] (e.g. the agilists), others seem to imply that RE ceased to exist with the introduction of agile methods (e.g. 'architecturalists'). The dispute on agile methods and RE existence is not something to argue for. In fact, as noted by Paetsch et al. [17], *the RE process phases of elicitation, analysis, and validation are present in all agile processes.* Thus in this thesis, we take the view that RE is a collaborative effort of which agile methods are an example.

This section discusses the background of RE and agile development in the systems development context. We start by detailing the RE process in systems development in which we describe the traditional RE process and fundamental RE terminology. We then review the literature on traditional methods and agile software development while discussing the documented comparison of the two methods. A discussion on large-scale agile systems development comes next followed by a discussion on RE in large-scale agile development. Since RE is a communication problem, we discuss the knowledge management literature before concluding the section with a discussion on safety as a cross-cutting concern in development. Safety systems' development is dependent on effective RE. Safety is used as a maximum formality example for performing RE in an agile development environment.

1.1.1 RE in Systems' Software Development

Systems development or engineering was initially about configuring hardware components into physical systems like ships or railroads [20]. The component parts would be produced once the configuration and the requirements specification are done. The production was thus a sequential process. As technology advanced and software began to appear in such systems, the same sequential process of development was naturally followed [20]. Over the years, even as software in systems (e.g. automotive [1,21]) increased, such sequential processes formed the basis for development. The dependence on software has led to software-intensive systems–systems that depend on software, hardware and the context in which they are operating for correct operations. It is important to note that for such software-intensive systems, software failures are commonly associated with RE challenges [4].

RE is defined as "*a systematic and disciplined approach to the specification and management of requirements with the goal to:*

[a] *Know the relevant requirements, achieve a consensus among the stakeholders about these requirements, document them according to given standards, and manage them systematically*

[b] *Understand and document stakeholders' desires and needs*

[c] *Specify and manage requirements to minimize the risk of delivering a system that does not meet the stakeholders' desires and needs*

All of which address important facets of RE: (1) process orientation, (2) stakeholder focus, and (3) importance of risk and value considerations [22]." RE activities typically include elicitation, analysis, specification, validation and management, with requirements prioritisation coming in to support elicitation and analysis by identifying the most valuable requirements [23].

Requirements elicitation process includes users getting involved in gathering requirements [15]. During elicitation, the initial information regarding requirements and context is gathered. The *requirements analysis* phase then follows to check for consistency, completeness, necessity and feasibility of the requirements, thus creating an understanding of the requirements. The next activity is the *requirements specification* where the requirements are defined in terms of system behaviour, decomposing the problem into component parts and serving as input to design specification. The end of this process is marked with a requirements specification document where the agreed requirements are documented for communication with stakeholders and developers. The *requirements validation* is important for confirming customers' needs and correcting errors in the specifications to avoid rework which could be expensive. *Requirements*

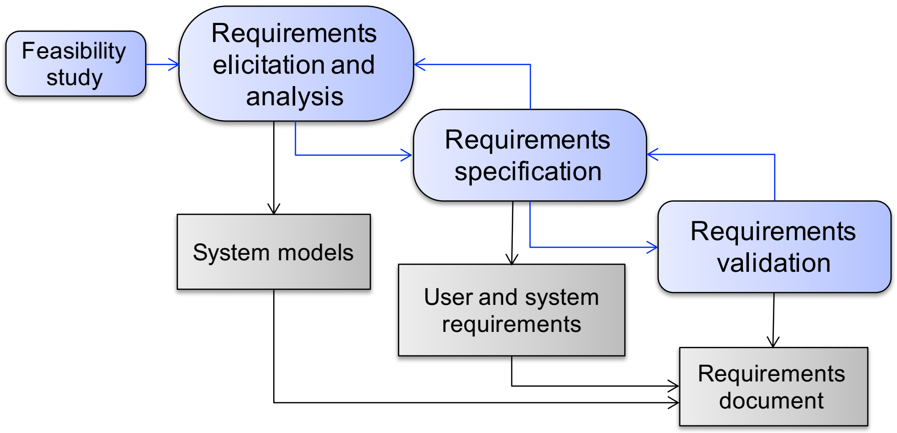
Feasibility study
Requirements elicitation and analysis
Requirements specification
Requirements validation
System models
User and system requirements
Requirements document

Figure 1.1: Requirements Engineering process flow. Adopted from [5]

management aims to keep the requirements' quality whenever there are updates or newly added requirements during the system implementation. The updating of requirements also means dependencies and relationships of different requirements documents must be managed, and all the requirements should also be traceable, which helps to investigate the impact of the changes [5]. The flow of the processes is as represented in Figure 1.1. The requirements flow through a sequential process similar to that followed in the waterfall or V-model methods that have existed in system engineering for many generations.

With increased emphasis on user value as well as increased pace of change, more strain has been put on the traditional, sequential approach. The strain came from the realization that requirements were more emergent with use than pre-specifiable, and thus traditional methods were not suitable for producing user valued products [20]. Alternative methods, like agile methods, were then devised and adopted.

1.1.2 Traditional Waterfall Process Vs. Agile Development in Systems Engineering

The traditional waterfall process is a classical systems engineering process that follows a sequential flow of activities, as shown in Figure 1.2. The phases are cascaded one after another with former processes being frozen when work is continued to later stages. As can be seen in Figure 1.2, once inconsistencies are noticed at the testing phase, then software requirements are revisited, which has proved expensive in the fast-paced change of today. As demonstrated in Figure 1.1, RE also follows a sequential process similar to the waterfall methods.

On the contrary, agile development is both iterative and continuous. According to the agile alliance [25],

> "*Agile is the ability to create and respond to change in order to succeed in an uncertain and turbulent environment.*"

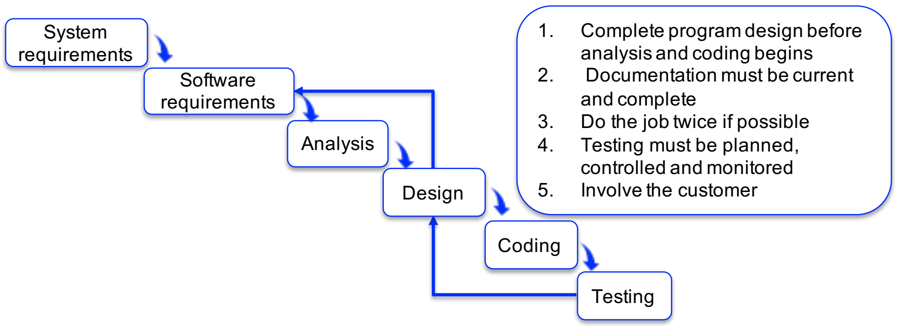
System requirements
Software requirements
Analysis
Design
Coding
Testing
1. Complete program design before analysis and coding begins
2. Documentation must be current and complete
3. Do the job twice if possible
4. Testing must be planned, controlled and monitored
5. Involve the customer

Figure 1.2: Waterfall process. Adopted from [24]

Concerning agile software development, Beck [26] notes that:

> *"Agile software development is an umbrella term for a set of methods and practices based on the values and principles expressed in the Agile Manifesto [26]."*

The agile manifesto identifies four values for agile development as shown in Table 1.1

Table 1.1: The four values for agile development

1.	Individuals and interactions	over	processes and tools.
2.	Working software	over	comprehensive documentation.
3.	Customer collaboration	over	contract negotiation.
4.	Responding to change	over	following a plan.

While the agile advocates acknowledged the items on the right as having value, they valued the items on the left more. The agile manifesto also connects 12 principles that attempt to make the agile values more concrete and deliver solid guidance for software development teams and their projects. The original agile principles are as presented in Table 1.2. These have been reviewed by William [27] but the basic concepts remain the same.

Agile methods like Scrum [28] and XP [29] are based on the above values and principles and encourage flexible, light-weight software development with short iterations [30] thus creating the ability to deal with changing requirements and fast time-to-market. In agile software development, requirements are allowed to evolve through collaboration between self-organizing, cross-functional teams utilizing the appropriate practices for their context.

In agile development, instead of fixing all plans at the project start, the project is broken into smaller sub-tasks which are implemented in short time-boxed iterations [18], commonly referred to as sprints. Sprint duration is also an agile variable with differing recommendations from the different agile methods but typically spans 2–4 weeks. The goal of each sprint is to produce shippable code incrementally. Sprints have the same pattern, which has three common phases illustrated in Figure 1.3).

Table 1.2: The 12 Agile Principles [25]

No.	***Principle***
1.	Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2.	Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3.	Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4.	Business people and developers must work together daily throughout the project.
5.	Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6.	The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7.	Working software is the primary measure of progress.
8.	Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9.	Continuous attention to technical excellence and good design enhances agility.
10.	Simplicity – the art of maximizing the amount of work not done – is essential.
11.	The best architectures, requirements, and designs emerge from self-organizing teams.
12.	At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

The first phase is a planning phase that includes a review of the sprint backlog that is later (re-)prioritized and estimation of the work to be done established [31]. The team–usually 5-9 people–then commits to the work. The second phase is the development phase in which the team takes responsibility for the requirements and elaborates them. Then the code implementation, building, and testing are done. The last phase is the delivery of the increment and assessment of the sprint [31], sprint review in Fig. 1.3.

For effective development in agile methods, communication, and collaboration among the team members are crucial. Thus the practice of face-to-face communication with team members is recommended [15], together with an on-site customer so that the developers will get quick clarifications on requirements. Most agile methods, such as scrum, have the practice of a daily stand-up meeting for the team. At this meeting, the team members give a report of what has been done, and what they plan to do, including the challenges that they could be facing. In this way, the sprint progress is tracked, and the problems are reported in time.

De Lucia and Qusef [32] state that the main difference between traditional and agile development is not whether, but when to do RE. In the traditional approach, RE is done only at the beginning of development (see Figure 1.2)

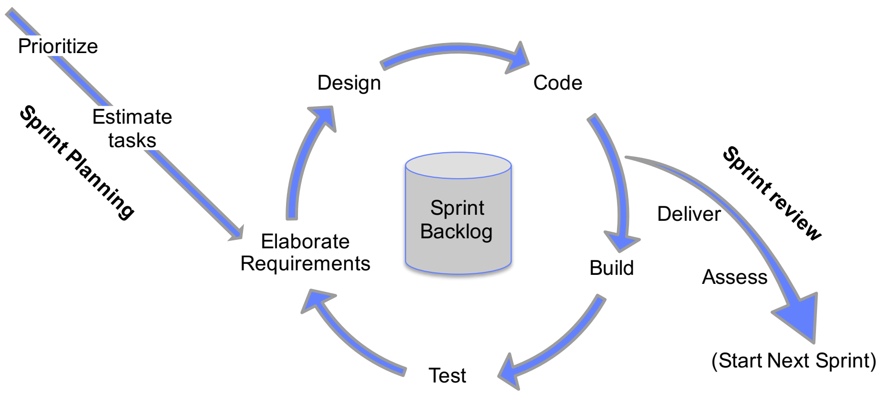

Figure 1.3: Typical sprint in an agile process. Adopted from [31])

while in agile development, RE is as continuous and incremental as the product being developed. Research on agile and traditional methods has varied considerably, starting from comparing characteristics of agile with those of traditional methods [33–35] to uncovering challenges of hybrid development in organisations [36].

The hybrid methods come from the fact that many large organisations are only still transitioning to agile and thus have both traditional and agile methods in operation. Also, many have safety concerns that call for a well-streamlined process and documentation that would otherwise be ignored in a fully agile process. Theocharis et al. [37] conducted a study to find out whether agility accelerated the extinction of traditional methods. They, however, found a mixed application of methods and concluded that hybrid approaches formed the standard of today's development. Research in this field has advanced to the point that researchers study how the methods have been combined in development [38], the challenges of having such combinations [39] and potential solutions to such challenges [40]. It should be noted that in large-scale organisations in practice, adoptions start with the software development teams using agile methods while the rest of the organisation works with traditional methods [41]. Whereas these studies are partially empirical, they lack the RE perspective, which gives a more general focus on communication and coordination in such environments.

1.1.3 Large-scale Agile Systems Development

Although initially meant for small collocated teams, the reported success of agile methods has led to their adoption at scale [8, 9, 42] and in systems development [9, 10, 43]. Large-scale companies are characterised by long lead times, many stakeholders with varying backgrounds and needs during (and probably after) the systems' development. Also being in the context of systems development, large-scale companies are also characterised by stable sequential processes [44]. Because of these characteristics, agile adoption at scale was received with skepticism, with most adoptions starting in software development teams. This selective adoption created pockets of agile teams within a larger ecosystem of plan-driven culture. We term these pockets of agile teams as *agile islands* (Paper A and C).

Large-scale agile development is a complex term that has been interpreted in various ways [6]. It has been used to refer to varying contexts starting from one large team in a project to large multi-team projects [45]. Dingsoyr et al. [46] provide a taxonomy of scale and categorise large-scale as a company or project with 2–9 development teams. We use the term "*large-scale agile system development*" to refer to large-scale agile development in the context of systems engineering and define it as follows:

> *Large-scale agile system development is the development of a product consisting of software, hardware, and potentially mechatronic components that include more than 6 development teams and is aligned with agile principles.*

Large-scale agile development has received considerable attention from the research community with many reporting successful adoption [9, 13, 42, 47] although challenges remain. Several challenges relating to e.g. coordination in a multiple team environment with hierarchical management and organisational boundaries [8] and coordinating work between agile teams [43] have been identified. As such, large-scale agile frameworks are being adopted to overcome the challenges [48].

SAFe framework has been reported by the state of agile report [49] as the most popular with 30% of the companies in the survey using it. These frameworks have received considerable attention from the research community recently, with some studies exploring how these frameworks have been adopted (e.g. SAFe [50]) and recommended guidance for clear adoption [51]. Others have explored the benefits and challenges of adopting these frameworks [52, 53]. It, however, remains unclear whether these frameworks do address the challenges identified by the introduction of agile methods at scale.

1.1.4 RE in Large-scale Agile Environments

"*No matter the specific method, agile's treatment of requirements is fundamentally different* [18]

Since many systems development companies are adopting agile methods, the existing techniques are proving inadequate. As a result, RE becomes an even more significant challenge for agile development companies, especially at scale and in systems development. The research on RE in agile development environments has thus attracted much attention from both research and practice, leading to a need for more empirical studies that devise working solutions collectively.

The process of doing RE in an agile development environment has been coined Agile RE, although it has no unanimously accepted definition [16]. Agile RE can however be weakly defined as the agile way of performing RE. Some of the existing research on agile RE has explored the benefits and challenges [16, 23, 54], together with the practices used in development [15, 19]. The challenges identified include, e.g. neglect of non-functional requirements, customer availability and minimal documentation. The consensus in all the studies is that agile RE addresses some of the classical RE challenges, e.g. communication gaps, while it also causes new challenges which call for new techniques.

Identified practices that apply to agile RE include; *face-to-face communication, customer involvement, requirements prioritization, review meetings and retrospectives, iterative/incremental development, user stories, test-driven development, acceptance tests, change management and code refactoring.* The practices adopted by teams vary depending on the agile method of development that is chosen. For instance, in XP, the planning game is used and begins with an on-site customer who writes the requirements. These are later prioritized by the development team together with the customer. However, experience with practitioners has shown that practices and methods have been adopted randomly according to the development needs.

Other researchers have compared the use of traditional RE and agile RE [17] while also identifying how agile development can benefit from traditional methods. Paetsch [17] note that the major difference is in the amount of documentation carried out in the development process. In summary, there is substantial amount of existing work on the use of agile RE and its practices. However, we notice a lack of empirical evidence in terms of large-scale agile systems development.

1.1.5 System Understanding in the Large

Software engineering is a knowledge-intensive endeavour [55, 56] with activities from requirements elicitation to the project coordination and management. It is unlikely for the team members to have all the knowledge obtained from those activities [56]. Agile development adds to the challenge with the idea of breaking down the system requirements into features to be developed every sprint, thus making the bigger picture of system understanding unclear. However, in order to deliver a quality product, the development team has to have a clear understanding of the system. Effective communication and knowledge management become an essential part of their development [6].

Knowledge management is crucial for proper system understanding and ensuring effective communication helps to transfer knowledge [57]. However, few existing works explicitly address communication in agile development [58]. Communication in agile projects has mainly focused on the impact of the agile practices on communication [59]. In a systematic review, Hummel et al. found reports that agile methods lead to improved communications in large-scale development projects. They also found that the informal communication on which agile methods depend may be problematic for large projects with many stakeholders and a lot of shared information. The studies present a broad understanding of the communication concept without focusing on the social interaction and behavior of teams [58]. Studies suggest synchronous and asynchronous communication means e.g. wikis and group e-mails in order to establish the multiplicity of social links between team members and to provide continual access to project information in large-scale settings [58]. While investigating inter-team coordination mechanisms, Nyrud and Stray [60] find ad-hoc conversations more beneficial than daily stand-up meetings that agile methods recommend. These were found to be the most time-consuming and involved less coordination. Such coordination can be tagged in artefacts which also could come from both agile and traditional methods [61].

For large-scale development, such artefacts are shared within the team and also across different teams. Some high-level artefacts, e.g. architectural models [62], and tools [63] are used as boundary objects . Boundary objects are defined by Star and Griesemer is as follows:

> *Boundary objects are objects which are both plastic enough to adapt to local needs and the constraints of the several parties employing them, yet robust enough to maintain a common identity across sites [64].*

Boundary objects are shared between several teams and each team can access what they need from it and thus helping in knowledge management across the system development. As Rolland et al. [6] state *"We believe there is a need to emphasize the boundary work and boundary infrastructures that are required for working across contexts resolving and coordinating complex socio–technical interdependencies."* This thesis explores the use of boundary objects in development.

Tools are essential to ensure effective communication and collaboration in large-scale development companies. In this thesis, the focus is on tools that help in effective communication of software requirements. Through a survey subjected to requirements tool vendors, de Gea et al. [65] provide an insight into the degree of support offered by requirements tools and the capabilities of these tools in supporting the RE process. They find a substantial number of tools supporting requirements elicitation but a poor representation of tools for requirements management. de Gea et al. argue that RE tools are traditionally oriented towards textual requirements and thus the reason for fewer modeling tools. These textual requirements tools are, however, used for elicitation processes. This thesis also explores the use of a text-based tool for requirements management.

1.1.6 Cross-cutting concerns in Development

We use the term cross-cutting concerns to mean the development concerns that are driven by RE and require system understanding. These are usually quality concerns or non-functional requirements of any software development. Existing research conducted in agile software development identifies challenges of dealing with non-functional requirements [66]. For the context of this thesis, the cross-cutting concern that raises interest is safety since it gives us a maximum formality example.

Safety raises concern since agile methods tend to have less favour for documentation and processes [67, 68] yet safety requires a well-defined process with extra documentation for certification [69]. Glinz [22] defines safety as the capability of a system to operate without resulting in harming people, property, or the environment [22].

Safety systems or Safety-critical systems (SCS) are becoming more prevalent in use with the advance of the digital era [70]. Many software applications are highly critical for safety, and are found in, for example, the avionics, medical, railway, and automotive sectors. Examples of such systems include flight control systems in avionics [71], automatic braking systems in automotive [72], Train Control Management System (TCMS) for a high-speed train in railway systems [73].

Requirements for the development of SCS (Safety requirements) are commonly stated as quality requirements and in some cases also stated in terms of functional requirements and thus usually follow the same development path as all other requirements. However, since these safety requirements have more stringent rules on the testing and validation, extra checks are put in place. With this extra effort demonstrated in safety development, using safety as a comparison gives us a maximum formality example to ensure we do not miss critical elements of RE in system development.

1.2 Research Focus

The overall goal of this thesis is to investigate the challenges and solution candidates of performing effective RE in an agile environment, based on empirical evidence. Considering that the use of agile methods suffers most at scale, we focus on large-scale systems development companies to achieve our goal.

The main goal is broken down to two sub-goals as follows:

- **G.1:** To gain insight in the current state of RE in large-scale agile systems development.
- **G.2:** To recommend solutions for addressing critical cross-cutting challenges of RE in large-scale agile system development.

To meet the set goals while scoping the thesis, research objectives (RO) were defined for each of the goals and addressed as presented in Table 1.3. The connection between the goals and objectives is further illustrated in Figure 1.4.

Table 1.3: Research questions for the respective aims

G.1: Gain insight in current state of Requirements Engineering		
RO.1A	Identify the Requirements Engineering challenges of using agile methods in large-scale systems development.	Paper A, D
RO.1B	To explore the state of the art on challenges of developing Safety-Critical Systems in agile environment.	Paper B
RO.1C	To identify challenges of using agile methods in structured environments.	Paper C
G.2: Explore solution space		
RO.2A	Examine and critique existing popular frameworks for managing scaled system development.	Paper D
RO.2B	To propose techniques to solve or overcome some of the identified challenges	Paper E, F

1.2.1 Gain insight in current state of RE

The first sub-goal G.1 was set to gain insight into the current RE challenges in the agile industry. The topic for *RO.1A* explores general challenges of performing RE in large-scale agile systems organisations. This topic was selected with request from participating companies that were in the process of adopting agile methods and thus formed the basis for the subsequent objectives.

Through *RO.1A*, the role of RE in large-scale agile development was re-emphasised and a range of challenges discovered. Since we aimed for empirical research, it was essential to perform relevant research while also keeping the practitioners interested. Thus, while *RO.1A* gave a broad scope of challenges, we explored the ones that geared interest for the participating companies and also proved less explored by academia. Without wanting to miss important RE related issues, we explored research on safety in agile development. In so doing, a study on the challenges related to safety-critical and agile development led to the second objective *RO.1B*. *RO.1B* aimed to find out which challenges have been identified in research that relate to safety systems in agile environments.

RO.1A also found different scopes of agile adoption in practice giving a hint of coexistence of both agile and plan-driven methods. This finding created a need to investigate the challenge of having both methods in industry. *RO.1C* based on one case company to explore the coexistence phenomenon. Challenges in this context were sought from the perspectives of the development teams.

1.2.2 Exploring solution space

After gaining insight and understanding our problem space, we aimed to explore the solution space as well. We explored the solution space in three separate ways which aimed to meet two objectives. *RO.2A* was set to explore the already designed frameworks and find out how, or if at all, they are addressing the identified challenges. Understanding which practices the scaled frameworks recommend for addressing the identified challenges also helped us to understand the challenges a bit more. With views from the practitioners, we also sought to understand whether the frameworks are helping and how it is that the challenges continue to surface.

RO.2B explored available techniques for overcoming (any of) the identified challenges. Through *RO.2B*, we identified other possible interventions that could be used to address some of the identified challenges. We started with a design and implementation of a tool that showed promising results for addressing some of the challenges. We also started to derive on a taxonomy that could help address coordination concerns in large-scale agile systems' development.

1.3 Research Methodology

This thesis builds on six studies (Papers A–E), i.e. four empirical studies that adopt a case study approach (Papers A, C, D and E), one secondary study (Paper B), and one tool study (Paper F). The work in this thesis follows an empirical research methodology. It collected, analyzed and evaluated empirical evidence on the phenomena of interest. This section describes the overall

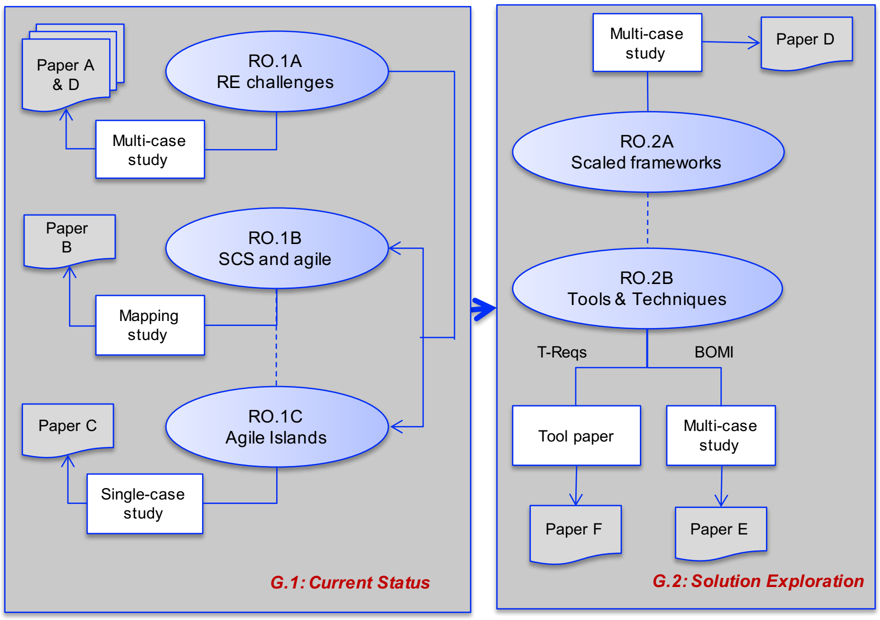

Figure 1.4: Research process of thesis

research approach of the thesis, followed by a description of the methods used, namely; a secondary study through mapping study, and case studies.

1.3.1 Research Approach

The overall research of this thesis was motivated by our industrial partners' enthusiasm to improve their RE management process. For this reason, the majority of the work was performed as qualitative empirical studies. The collaboration with industry partners allowed us to base our findings on empirical data from practitioners and also validate the relevance of our research questions.

Since software development is carried out by persons or groups in organisations [74], this makes it a multi-disciplinary area that also includes social boundaries. Thus, for this research, we investigate not only the tools and processes used by the development teams but also their social and cognitive processes [75]. For studying these real-life situations of practitioners in our partner organisations, the use of qualitative approaches became necessary.

The qualitative studies allowed us to get insight into the RE challenges that our industry partners encounter and also derive solution candidates. We also include a literature or secondary study paper that allows us to zoom out of RE to relate to the bigger context of safety, with which many large-scale system development companies are concerned. We elaborate on the qualitative methods used and the secondary study in the next sections.

1.3.2 Case Study Method

This thesis bases on a case study definition given by Runeson et al. [76]. According to Runeson et al., a case study is *an empirical inquiry that draws on multiple sources of evidence to investigate one instance (or a small number of instances) of a contemporary software engineering phenomenon within its real-life context, especially when the boundary between phenomenon and context cannot be clearly specified* [76]. Furthermore, case studies provide an in-depth understanding of why and how given phenomenon occur [75], thus giving opportunities to describe, explore and explain the studied phenomenon. This thesis includes studies that are mainly exploratory (especially Paper A and C), in that they seek new insights [76] and identify useful distinctions that clarify our understanding [75] about agile development and RE processes in industry. Paper C and D endeavour to describe the reasoning behind the identified challenges and identify solution candidates from literature and practitioners. Paper E also uses a case study approach and attempts to derive a taxonomy aimed towards improving the current situation. Paper F introduces and describes a Text-based Requirements system (T-Reqs) that was fronted as a solution to some of the identified challenges. T-Reqs is the open-source version we created based on a working tool at one case company.

Runeson and Host [74] present five steps to performing case study research and these include: (i) designing case study objectives and plan (ii) Defining data collection procedures and protocols (iii) Collecting data on the studied case, (iv) analysing the data and (v) reporting the data. The researcher participated actively in all these steps for the appended studies. For the first step, we had our research questions prepared before starting the study and purposely [77] identified the cases to use in our studies. For all the studies, we chose large-scale systems development companies that have agile development teams. While identifying the companies, we also discussed with the contacts in the different companies on the acceptable data collection procedures.

Case studies can be (i) holistic–with single or multiple case studies is the unit(s) of analysis, (ii) embedded–where one can either have a single case study with many units of analysis or multiple case studies each with multiple units of analysis. This thesis used two of those four described settings; holistic multiple case study for papers A, D and E, and one embedded single case study for paper C. Each setting was used to satisfy the aims we had in each study. For instance, in Paper A the aim was to find out the general challenges of doing RE in agile development companies and thus a multiple case study approach, with four cases of study, was chosen. The cases were all from different domains. This setting allowed us to discover as many challenges as there were, and we could attempt to generalise when we find challenges reoccurring in different domains. The findings of Paper A inspired the study that led to Paper B and C. For paper C, we used a single case study with two units of analysis as we aimed to find the challenges that individual agile teams faced working in a structured environment. The single case was viable since we needed to focus on the same environment setting and the two units of analysis shed more light on the actual situation enabling us to explain the phenomenon. For each of these case studies data was collected through interviews, focus group meetings and workshops where necessary.

Focus Groups and Workshops We used workshops as initial data collection instruments for most of our studies (Papers A, B and D) and backed them up with focus group meetings to get more detailed understanding and explanation. We differentiate between workshops and focus groups in that for a workshop, a group of practitioners–generally knowledgeable about agile development and RE–meets to work on creating a defined result jointly while focus groups involved representative stakeholders or experts that were invited to discuss the specific topic under investigation from all relevant perspectives. For instance, in Paper A, the workshops involved company contacts from all participating companies whereas focus groups were conducted at company sites with the representative practitioners of RE and agile development. Focus groups allowed us to dig deeper focusing on one company while workshops allowed us to triangulate our findings. In general, a total of 13 workshops with 79 participants and 7 focus group sessions with 26 participants were conducted. The totals are distributed in the different studies as shown in Table 1.4.

For all the workshops, at least three (3) researchers were always present whereas focus groups sessions were attended by at least two (2) researchers. The researcher was always one of those present and participating in workshops and focus group sessions. Prior to workshops and focus group sessions, a tentative agenda was shared with agreed participants. In many instances, the agenda included a presentation from the practitioners, detailing their experience with RE and agile development (Paper A, C and D), with safety (Paper B) or with islands and boundary-objects (Paper E). We would then present work related to the topic of discussion to ensure a common understanding of the topic. In most cases, this would raise discussion points for which notes would be taken. The researcher would present at some sessions (Paper A, B) and actively take notes while seeking clarifications where necessary at other sessions (e.g. Paper D, E). Follow-up questions and discussions were kept welcome and open. The sessions lasted three (3) hours in most cases (e.g. in Paper A, B, E) and full-day, at Chalmers university premises, in one paper (Paper D). We would summarise the session on one or more slides as a way to ensure we understood the participants' input correctly.

Interviews The researcher was actively involved in 18 of the 29 (unique, see Table 1.4) interviews that were conducted in this thesis. Interviews were used in studies leading to papers A, C and D. For Paper C, interviews were the primary source of data since it targeted development teams in particular. Semi-structured interviews, where one or more interviewers interacted with one interviewee based on an interview guide, were also used to collect data. For each interview study, an interview instrument with a structured set of questions was designed prior to the interviews. The interview instrument was created with recommendations from the contact persons in the respective companies. These recommendations allowed the researcher(s) to ask questions that interviewees would understand in their context.

The interviews started with an explanation to the interviewee about the focus of the study to ensure that the same topic was being discussed. During the interviews, the instrument acted as a guide and was kept open in such a way that there was no strict adherence to the structure and flow of the questions. Both the interviewer and the interviewee were free to ask for clarification and follow-on questions which allowed wide and deep inquiries into the reality of software development for the specific topics of interest. All interviews were audio-recorded with interviewee consent and later transcribed leading to textual analysis.

Data analysis For the data analysis, since we were dealing with qualitative data, we relied on a thematic coding approach [78]. Since we worked in groups, we had at least two researchers at each study to familiarize themselves with the data collected while highlighting noteworthy statements and assigning codes or labels to each. The researcher was always one of the two researchers at each coding phase. For the interview data, the researcher transcribed the data and performed the initial coding. The codes would then be discussed as a group and iteratively agree on the themes which we would discuss through workshops (Paper A, D, E) or through email and telephone calls (Paper B and C) for validation with the participating cases. Paper F describes T-Reqs which was inspired by a tool already in use in one of the companies as their in-house solution. Together with support from the pioneers of that in-house solution, we created an open-source version of T-Reqs through defining simple templates and scripts. For T-Reqs, we relied on feedback from the company contact to verify and improve its usefulness.

Table 1.4: Summary of Research Methods

Paper	*Data Collection*			*Analysis Method*
	Type		*No.of participants*	
Paper A	Holistic multi-case	5 focus groups 2 workshops 22 interviews	14 11 22	Thematic analysis
Paper B	Mapping Study			Mapping study
Paper C	Embedded single	1 focus group 18 interviews	9 18[1]	Thematic analysis
Paper D[2]	Holistic multi-case	5 focus groups 11 workshops 22 interviews	14 63 22	Thematic analysis
Paper E	Holistic multiple	1 focus group 2 workshops	4 16	Member checking
Paper F	Holistic single			Tool design

[1] 11 of the 18 interviews were used in Paper A.
[2] Paper D shares focus group, interview and 11 workshop participants with paper A.

1.3.3 Secondary Study

With the growing number of empirical studies in software engineering comes a necessity to construct an objective summary of the available research evidence to aid in decision making and formulation of research questions [79]. Using a systematic literature review is one way of obtaining the objective summary. Much as each study included in this thesis has a literature review section, a systematic review provides guidelines to follow while reviewing literature in order to avoid bias and ensure replicability [80]. Systematic literature reviews, however, require a considerable amount of effort [81].

A systematic mapping study provides a map of the results reported in literature, usually a more coarse overview thus often requires less effort than a systematic literature review [80]. The process followed in a mapping study is also systematic but more coarse than a systematic literature review, allowing to process larger numbers of papers.

System development companies are usually subject to standards and regulations, thus they aim for a transparent RE process. Also, with the increased digitalisation and interconnectedness of devices, safety becomes a concern in development. Following the findings from Paper A that identified challenges with safety, a mapping study (Paper B) was conducted to help us draw a map on the current state of affairs in as far as agile development and safety-critical systems development are concerned. The researcher performed the initial document search that gave 1986 documents and was able to reduce them to 69 documents when following the inclusion and exclusion criteria that were defined. Together with two of the other researchers, more studies were excluded giving a final total of 34 documents whose data was analysed and findings presented in Paper D. We aimed to classify and synthesize our findings from industry with those in published empirical studies. With this, we derived a viable research direction for managing requirements in large-scale agile development.

1.3.4 Threats to validity

For this thesis, as with many empirical studies, there are validity threats worth discussing. We consider the four perspectives of validity threats as presented in Runeson and Host [74]and in Easterbrook et al. [75].

1.3.4.1 Construct validity

Threats to construct validity refer to the relations between the research method and the observations from the study [74]. With these threats, the question to answer is: Are the theoretical constructs interpreted and measured correctly? There is a threat that the interpretation of the questions asked at the interviews may be different for the researcher and the interviewees due to the use of different or abstract terms. To minimise this threat, we relied on our company contacts and selected participants who are knowledgeable in the subject of study. On the general scope, participants had to have knowledge on the constructs of agile development and RE. We collected data from multiple sources, including existing literature and different companies in varying domains. This diversity of sources helped us to ensure we got correct results. For the interviews, we shared and discussed the interview guide with the company contact persons in order to agree on the commonly understood/used terms at the company and also used literature to provide a link between our understanding and that of the interviewees. In cases where the interviewee did not understand the question, we endeavoured to rephrase and give explanations. We also asked interviewees for elaborations in case we got an ambiguous answer. Most of the data collected in interviews was validated in workshops and focus group meetings. These meetings always began with presentations from the company participants and also from one of the researchers. Judging from the presentations that were given by practitioners, we were always confident that the concepts were clearly understood. We also ensured that we had more than one researcher for data collection and analysis in all the studies. To combat the practitioners' fear to answer asked questions with honesty, we guaranteed anonymity and raw data was only to be used by the researchers. For Paper B in particular, we also calculated inter-rater agreement where there seemed to be some disagreements. This calculation was then followed with discussions, among the researchers, to resolve the disagreements.

1.3.4.2 Internal validity

Internal validity focuses on the research design and whether the results really do follow from the data [75]. For internal validity the question to answer is: Could external factors impact the results of the investigated factors? To minimise this risk, and with permission from the interviewee, we recorded all our interview sessions in order to ensure that each researcher gets the same message at data analysis phase. We also used data triangulation between interviews (Paper A and D), between the units of analysis (Paper C), and between the case companies (Paper A and E). Furthermore, the results of our case studies were discussed in workshops (Paper A, D and E) and at focus group meetings (Paper C). The workshops and focus groups included practitioners from these companies that were already involved in the respective studies. These practitioners were always allowed to discuss their (sometimes different) perspectives on the data we collected, thus increasing our confidence in the data. Through sufficient numbers of interviews and member checking, we made sure that we captured all important concepts in the scope of our inquiries. To avoid a too restricted view on smaller parts of a project or a product, we selected interviewees from different parts of the development. There might, however, still be a selection bias as the interviewees were selected through a convenience sample through our company contacts.

1.3.4.3 External validity

External validity relates to the ability to generalise the results beyond our case studies. By design, the external validity of our studies is low. Hence, generalisation of our findings to different domains or companies might not be possible. Easterbrook [75] notes that qualitative studies aim to understand and explain a given phenomenon rather than generalising. Understanding the investigated phenomenon in one setting may help to understand other situations. For instance, in Paper A, we designed our study to identify common challenges across participating companies. Thus, our research method does not support any deep reasoning about differences between companies, domains, and market positions. However, given that we found similar themes in all cases, we expect that these apply similarly to other companies or projects in large-scale systems engineering. By analysing data gathered from different companies with different characteristics, we believe that we have got results that can be extended in the future and were able to identify relevant categories that are applicable in other contexts.

1.3.4.4 Reliability

Threats to reliability refer to level to which the study results are dependent on specific researchers. We limit reliability threats by improving the interview instruments in multiple iterations and by conducting interviews in pairs of two researchers. Also, at least two researchers were involved in the focus groups and workshops in order to reduce the impact of subjectivity. We have continued involvement with our case companies and therefore a mutual trust among the parties exists. The data analysis was discussed and refined among the authors in several iterations. The results were discussed with the participating companies and also compared results obtained to available literature. We also tried to describe our data collection and analysis procedures and shared the instruments used for data collection in the studies conducted. The potential solutions proposed are based on our reasoning, claims in related work (that these solutions help with a specific challenge), and on discussions with the case companies. We have not applied the solutions from the literature in the case companies, or solutions suggested by one company in further companies. Further validation of the collected solutions is needed.

1.4 Research Synthesis

We summarise the main results and contributions of this thesis per research objective and reported study. The detailed descriptions of the results for each study can be found in the respective paper (Chapter 2–7).

1.4.1 Paper A: RE Challenges in Large-Scale Agile

RO.1A: To identify the RE challenges of using agile methods in large-scale systems development.

There is a substantial amount of research on agile methods from their definition to their adoption, even at scale. Research on RE in agile environments is, however, limited. Although there is available literature exploring challenges of agile RE, none has the context of scaled system development companies. Motivated by the limited empirical studies on RE challenges particular to large-scale agile systems development, we conducted the study leading to Paper A which also formed the basis for the subsequent studies. Paper A contributes to our first goal (G.1 – establishing the current use of RE) while meeting our first objective *RO.1A*, based on the following research questions:

RQ1A.a.: What are possible scopes of applying agile methods in large-scale system development?

RQ1A.b.: How is the role of requirements characterized in large-scale agile system development?

RQ1A.c.: Which requirements related challenges exist in large-scale agile system development?

Main findings: With the first question, *RQ.1A.a*, we aimed to establish the level to which our four case companies were agile. We found that the participating companies were in the process of adopting agility and thus had adopted agility in differing levels. Whereas one had adopted agile methods for the whole product development process, several of the companies had agile adoption only in the agile teams. This finding meant that the system-level was still following a plan-driven approach, which also gave slight differences in the challenges faced.

RQ.1A.b helped us confirm the importance of requirements, even in the face of agile methods development. We find that requirements are seen as an order while the teams prefer to work with user stories that are portrayed as goals and give them more autonomy. RE is still crucial for system development, and the study revealed that large-scale systems companies are struggling to perform RE in agile development to the level they are used to while they were using waterfall/traditional methods.

However, irrespective of the level of adoption, all companies exhibited challenges (*RQ.1A.c*) that were put under two particular groups: Shared understanding of User Value and Building and Maintaining System understanding. In regard to user value, most challenges were coming from teams struggling to understand customer value, writing meaningful user stories and feedback and requirements clarification from the user stories or features they develop. For system understanding, the challenges faced relate to informing and synchronising between teams, creating and maintaining traces, insufficient tests and user stories, agile tool chain establishment and, coming more from the traditional foundation, there was a gap between plan-driven development and agile development. Generally, findings demonstrate how system development is struggling with RE, and this has now become an essential topic for practitioners and researchers alike. Existing literature concerning RE in agile development does not provide approaches for both user and system requirements specification in an agile environment.

Potential application of the results: Paper A contributes a map of RE related challenges in scaled agile system development. Practitioners find this map useful to plan their adoption of agile methods as well as check their process improvement since it allows to avoid over-optimizing one aspect while negatively affecting another aspect. We hope that researchers benefit from our overview of challenges when conducting related studies. To follow up on this study, we conducted an independent study on challenges specific to safety (Paper B), since the companies having to deal with safety requirements seemed to show some differences and one following up on the teams being agile while system-level is not (Paper C).

1.4.2 Paper B: Safety-Critical Systems in Agile

RO.1B: To explore the state of art on the challenges of developing SCS in an agile environment.

Agile methods have been criticised for neglecting upfront requirements and extensive documentation [30] which are some of the defining features of safety systems' development. This criticism has thus created thinking that agile methods are not suitable for Safety-Critical Systems (SCS) development. Upon finding a challenge of developing SCS in agile development among companies dealing with safety systems in Paper A, we sought to understand what challenges have been identified from published empirical studies. Since we did not find a study with a comprehensive and recent overview to help us understand the context, we ventured to explore the domain through a mapping study. We included only published empirical studies from 2001 to 2017. Paper B meets this objective by answering the following research questions:

RQ1B.a.: What research exists about agile development of Safety-Critical Systems?

RQ1B.b.: What are the key benefits of applying agile methods and practices in SCS development?

RQ1B.c.: What challenges exist with agile development of SCS?

RQ1B.d.: What solution candidates (e.g. principles and practices) promise to address challenges with respect to agile development of SCS?

Main findings: The results obtained showed that the focus on SCS has changed considerably over the years from being positive in the beginning, i.e. identifying benefits, to the recent papers discussing more of solutions to overcome identified challenges. The reported benefits were in agreement with the ones reported in the non-safety development domains, for instance better test cases, improved quality and improved safety culture.

Apart from not trusting agile methods for safety development, the challenges faced in development of SCS using agile methods are more geared towards the certification and assurance requirements of safety systems. Assurance requirements, for instance, involve many stakeholders who come from different domains. Managing communication between such numbers proved challenging in the agile context where documentation is not used for communication or even given that much attention. Also, standards were written with a waterfall mindset which favour effectiveness of waterfall methods over the flexibility of agile methods. Obviously there were challenges with upfront planning, lack of documentation focus and the trade-off of flexibility vs safety which have always been a cause for the skepticism. Generally, safety requirements call for well-structured processes, and heavy documentation that is not clearly supported by agile development and agile teams find it hard and cumbersome to balance speed and flexibility with the need for documentation.

The solutions proposed relate to using much the same practices as in the non-safety development case but also include discussions relating to safety for instance in the daily meetings and sprint reviews. It was also encouraged to have safety experts as part of the team so that teams always have safety in their mindset(s). We also notice that some practices that have existed even in the traditional methods are highly recommended, e.g. setting high coding standards and relying on standard operating procedures to improve quality management.

Potential application of results: The results provide for potential generalisation to other areas. We know the research that exists and that allows us and the companies to systematically search for a setup on being large-scale agile using a map of these challenges. The results also indicate that the challenges faced in SCS development come from the need for structured processes that the standards impose. With the standards in mind, it becomes necessary for companies adopting agile methods to work on a coexistence plan that should have both practices in proper use. Results also provide a starting point for enabling us to accumulate more knowledge on which solutions could help. So future research can build on it rather than replicate it.

1.4.3 Paper C: Agile Islands in a Waterfall

RO.1C: To identify the challenges of using agile methods in structured environments.

Findings from paper A (and partly from paper B) show that there is a challenge of dealing with the gap between plan-driven and agile development and having 'agile islands in a waterfall' organisation. Paper A findings also indicate that some companies have agile adoption only in the development teams an occurring commonly noted in some studies but not yet explored further. Paper C sought to follow-up on that finding to reveal the challenges that agile teams in such traditional structures were facing. We explore the challenges from perspectives of two of the agile teams in one large-scale company. The research objective *RO.1C* was met by answering the following research questions:

RQ1C.a.: What are the perceived challenges when combining plan-driven and agile paradigms in large-scale systems engineering?

RQ1C.b.: What mitigation strategies exist for the challenges identified?

Main findings: The findings of this paper indicate a variation in challenges (*RQ1C.a*) faced for departments of the same company. The difference in challenges stems from the departments' different ways of working with agile development methods. We find that whereas both departments have adopted agile methods relating to Scrum, they have customised them to their specific needs and thus modified a few roles and practices that bring in the difference. In one department for instance, the role of product owner was not defined although requirement prioritisation and coordination seemed to be going on well. Also, the requirements management tools used in both departments differed considerably although for both departments developers had no access to the high-level or system requirements. In that context, we found challenges common to both departments, e.g. development teams not being aware of the high-level requirements, function owners over exposed to change requests and traceability issues mostly stemming from difference in tools and unclear responsibilities in such environments.

We also found challenges unique to each department. At one department, there was less focus on control and more ambition to facilitate autonomy of agile teams. This resulted in practices where, for example, anyone on the team could write a user story to the backlog. It was found that this could lead to temporal inconsistencies, between the user stories and implementation, that surfaced and had to be fixed during testing. At the other department, such challenges were not as evident as the challenge of not knowing what effects a change in one requirement could have on dependent units. Since developers in this department were working with a central platform that is the foundation for the work of several other departments, they felt the lack the complete picture of the function was a more pressing challenge.

Strategies to mitigate the identified challenges were obtained both from the participants and also from literature. On the one hand, participants felt it necessary to have (system) testers as part of the team, update requirements based on learning from sprint, create proper channels for writing user stories while also explicitly stating which user story must be traced. Literature, on the other hand, recommends cross-functional teams to manage requirements updates, increasing understanding of processes and roles in development and bringing testers closer to the requirement owner. However, these strategies remain abstract and empirical research on their effectiveness is currently lacking.

Potential application of results: The results suggest a need for a holistic company-wide approach to agile adoption and development to overcome some of the challenges. Although there exist studies that mention the possibility of different ways of working for sections of the same company, this is the only study, that we are aware of, that clearly documents it while providing the proof of what challenges could ensue. From this study, we also observe that while it might not be possible to have all parts of the company agile nor be desirable to have all parts of the company plan-driven, if different approaches must co-exist, one should find a way to integrate them. Future research may need to show how this integration can be achieved.

1.4.4 Paper D: RE practices in large-scale Agile

RO.1A: To identify the RE challenges of using agile methods in large-scale systems development and
RO.2A: Examine existing scaled frameworks

Many frameworks have been proposed to try to overcome the challenge of agile development at scale. However, their usefulness or ability to solve the challenges has not been explored as elaborated in Section 1.1.3. With Paper D, we explore what solutions two of the popular frameworks, SAFe and LeSS, offer in relation to our identified challenges. Paper D contributes to research objective *RO.1A* and satisfies *RO.2A* by answering the following questions:

RQ.2A.a: How pervasive are agile methods in large-scale system development?

RQ.2A.b: Which requirements-related challenges exist in large-scale agile system development?

RQ.2A.c: Which approaches have been proposed in popular literature and which approaches are used by practitioners to address the challenges identified in RQ.2A.b?

Main findings: Paper D reports on a three-year exploration that extends Paper A with an almost exhaustive catalogue of RE challenges from the participation of seven companies. Thus *RQ.2A.a* re-affirms that agile adoption still exists in different phases in large-scaled systems development companies, with the three more companies in perspective. We identify new challenges (*RQ.2A.b*) to those identified in Paper A and rearrange them to six categories which we briefly elaborate in the following:

1) The first category, *Build and maintain shared understanding of customer value*, comes from the core strength of agile methods – managing customer value. Challenges here include bridging gap to customer and building long-lasting customer knowledge. For these large-scale companies, the distance between customers and development is large, and it is difficult to break the features into meaningful packages that have customer value and can be delivered incrementally. The distance also makes the customer role unclear at scale.

2) For the second category, *support change and evolution*, we find challenges relating to managing requirements updates, management of experimental requirements, synchronisation of development and requirements re-use. We find that updates are manually done, leading to inconsistencies which are expensive to remove. This indicates that facilities for updating system requirements based on agile learning are currently missing.

3) *Build and maintain shared understanding about system* is our third category. With agile methods' focus on value, the system requirements knowledge is underrepresented. We find the need for documentation to complement test cases and user stories, which are common practices in agile development, as they are not sufficient for system understanding. Furthermore, the big picture of the system is not captured since teams focus more on features or components and thus, the creation and maintenance of traces becomes complex as well.

4) The fourth category *Representation of Requirements Knowledge*, denotes the shared responsibility of requirements knowledge. In particular, it is hard to manage the various levels of development in large-scale while giving meaningful decomposition of requirements in agile development. We also find that the current tools being used are not fit for agile development as they do not provide the flexibility needed to accommodate different representations of requirements that teams or individuals could have. Additionally, it is difficult to establish consistent requirements quality and also align or establish thresholds for quality requirements.

5) Our fifth challenge category, *process aspects*, relates to the process of working with requirements. We find challenges relating to prioritisation of distributed functionality, managing requirements completeness and consistency. We also find the common challenge of balancing between time-to-market and quality of the product.

6) *Organizational Aspects* formed our sixth and final category. For these

large-scale systems engineering companies, the overall organisation in which RE is practiced plays a vital role as well. Inherently, these companies perform (some) long-term planning, especially for facilities. Our challenges in this category relate to bridging between such system-level planning and agile work in software teams, planning of integrated system testing, managing research and pre-development, and identifying impacts on critical infrastructure in good time.

Practitioners made suggestions towards solutions for some of the identified challenges (*RQ.2A.c*). For instance, many of the practitioners agreed that it is a good practice to have the teams and PO update the requirements. In this way, the gap between plan-driven and agile development would be bridged, and the tooling not fit for purpose challenge addressed. They also shared a recommendation to move from project-focused development to product-focused development in order to create and maintain traces as well as encourage re-use of requirements. Here, practitioners also mentioned the Text-based Requirements system (T-Reqs) as a useful tool for updating and managing experimental requirements. Paper D gives more details on these challenges and solutions, while we expend on TReqs in Paper F.

A mapping of the challenges to the solutions provided by SAFe and LeSS frameworks (*RQ.2A.c*) revealed that these frameworks have in some cases concrete practices while in other cases, there is none. In relation to *Build and maintain shared understanding of customer value* category, we find that several practices have been proposed. We find solutions relating to concrete practices, (e.g. frequent demos, sprint review bazars, use models, continuous improvement, retrospectives). We also find more abstract guidelines. For instance SAFe describes techniques such as combining “weighted shortest job first”, “portfolio backlog”, and “program kan-ban” to support cross-cutting initiatives towards prioritization. It also advocates for a combination of other practices as elaborated in Paper D. However, there are no clear guidelines on how to combine all these suggestions into one coordinated process. The same can be said about many of the recommendations from LeSS framework. It is our understanding that such combinations would require specialised tools, or even just customised tools which do not yet exist.

Results from analysis of popular frameworks show a lack of concrete advice to manage some of the challenges (coming from the different categories) while for many others, the solutions from these frameworks are relatively underrepresented.

Potential application of results: The Paper presents a catalogue of RE challenges and their potential solutions both from practitioners and also recommendations from popular frameworks. This catalogue would help inform practitioners on which solutions they already have available through the scaled frameworks and what practitioners in similar settings have used to manage their situations. Additionally, in order to mitigate the identified challenges, we encourage future research to not only focus on producing further practices but also evaluate the existing and proposed solutions in large-scale agile settings.

1.4.5 Paper E: Charting coordination needs

RO.2B: To propose techniques to solve or overcome some of the identified challenges

Current research explored hybrid development in terms of how development methods are combined by teams during development, for instance, using a mixture of traditional methods with agile methods [38]. What does that mean for large-scale companies working on different projects with dependencies? Our previous studies showed issues with communication and documentation, which were affecting coordination between teams and high-level development. While motivated by the results of paper C and those leading to Paper D, in paper E we set out to search for solutions for identified challenges. We started with generalising the problem of agile islands to the whole system development and thus the whole organisation. We contribute to research objective *RO.2B* by answering the following research questions in paper E:

RQ2B.a: Which agile islands are repeatedly encountered in large-scale agile contexts?

RQ2B.b: Which boundary objects are repeatedly encountered in large-scale agile contexts?

Main findings: Our findings to *RQ2B.a* show that there are occurrences of varying methods being used in the companies but not only among those using agile methods but also non-agile teams. We thus started to think in terms of methodological islands. While evaluating which islands do exist, we realised that the level of abstraction for the island differed and these abstractions were extracted and grouped to give the drivers for methodological islands.

We identified three groups of methodological islands which occur on different levels in the company. At the very high level, we identified external *organisations* that companies have to work with. These include, for instance, suppliers, regulators and customers, all of which come with different ways of working that have to be accommodated. The next two groups come from within the companies. For companies using SAFe framework, agile release trains (teams of agile teams) exist, and companies have to coordinate tasks between the different release trains. Similarly, for companies without SAFe, departments still exist each with its teams. As has always been, product development can span several departments (e.g. marketing, hardware, software departments) and thus several disciplines in the company. Since departments and release trains, for instance, are a group of different teams, at the intermediate level we identified *groups of teams* as another category of methodological islands. At the lowest level in the company, we have the *individual teams*, for instance, component teams or integration teams, which also display different methodologies.

We identified drivers for such methodological islands as process-, business- and technology-related factors. Companies are business-oriented and thus have to watch different factors which are best addressed through having departments. For the large scale, some companies or departments are also distributed to different geographical zones which brings in time and cultural differences. This distribution creates different methods for the affected departments, thus the *business-related* drivers. *Process-related* drivers describe a mixture of development methods (SAFe [82], V-Model [83], Scrum [28]) and whether a company mainly works based on projects, or whether significant workflows in the continuous development of a platform. The *technology-related* drivers come from the systems being developed. Here, methodological islands will emerge depending on whether they are dealing with the same architectural decomposition, systems disciplines, platform and product-line, and the required time-scale of commitment.

The boundary objects related to the methodological islands were also identified and categorised depending on different viewpoints of the islands. Typically, for large-scale, before development commences, contracts, roadmaps and plans are created for proper project or product management. We categorised these as *planning boundary objects.* We further identified boundary objects that relate to tasks in the development effort, e.g. user stories and other backlog items as *task boundary objects.* Those that are concerned with technological aspects (e.g. tests or architectural objects) of the system being developed were categorised as *technology boundary objects. Regulation and standards boundary objects* are used to ensure that the company complies with regulations and standards through safety cases for instance. The regulations come with a demand for process definition and thus documentation which is achieved through *process boundary objects* like SAFe documentation. Boundary objects relating to the final customers' product were categorised as *product description boundary objects* and include, e.g., the technical documentation for the customer and the variability model. *Trace links* are a special category, as they represent the relationships between artefacts.

Potential application of results: Paper E presents a catalogue of methodological islands and the boundary objects between them. Our catalogue proved useful when discussing potential process improvements with companies as it gives them a mindset of planning their coordination efforts in relation to their distances and particular needs. So, this catalogue comes in as a starting point for a technique that can be used to identify gaps or distance in islands and also plan for possible intervention before it becomes costly. This catalogue for boundary objects is not exhaustive and future research could pursue creating a model or taxonomy that defines this technique fully. Also, thinking about tools as boundary objects is a viable next step.

1.4.6 Paper F: Tool support for managing requirements

RO.2B:To propose techniques to solve or overcome some of the identified challenges

We approached the other part of research objective *RO.2B* through developing the T-Reqs tool. The tool design was motivated by Paper A, C, D that reflect findings on the challenge of agile teams not being aware of or able to update the system requirements. Agile methods provide recommended practices which also need tools support to make them effective. To this end, requirements tools are abundant. However, most existing tools for requirements management do not support the autonomy of development teams, e.g., giving developers ability to view, and possibly update the requirements at system-level. We recommend

Table 1.5: Summary of challenges relating to the 'gap'

Category	*Related Challenge*
Organisational issues	Lack of trust in agile methods
	No time for invention and planning
	Hard to plan tests based on requirements
Process issues	Prioritisation of distributed functionality
	Managing different representations
	Tooling not fit for purpose
Technology issues	Lack of overall system knowledge
	Managing traceability
	Updating requirements
	Synchronisation of development

customisable tooling to address the gap between system-level and agile teams. With paper F, a new tooling concept is illustrated.

T-Reqs is a tool solution that gives the developers autonomy and ability to make changes to the system requirements when needed. It is text-based and works with Git version control system to enable agile teams to propose changes to the high-level requirements at system level. This tool is presented in Chapter 7 where it is further elaborated in a technical report.

Potential application The tool is applied in the development in one company with a few modifications. The tool works well with reviews and allows the developers to use it within their usual development environments and yet still maintain traceability. It, however, still has artefact (requirement, test case or user story) Identity (ID) generation and more ideas to enhance it with sophisticated capabilities such as models that will not make it more complicated are welcome. The tool highlights the text-based view on requirements and future research could explore which domains such specification works best and if it can be generalised.

1.5 Summary of Results

The appended papers each contribute incrementally to the thesis goal of providing an empirical investigation into the challenges and solution candidates of performing effective RE at scale and in systems development. In this section, we summarise the contributions of this thesis as per the sub-goals of the thesis.

1.5.1 Current status of RE (*G1*)

Findings re-affirm that RE in system development is still as important as in agile development as it is or has been in traditional development. However, the team focus on user stories that come with agile development has added on to the challenges of RE in systems development despite solving, e.g. a few previously known challenges. We explored the challenges of developing in a large-scale agile environment from three 'angles' in order to meet this goal. We began with the general large-scale organisation (*RO.1A*) that reveals challenges relating to user value and system understanding (Paper A). These findings in paper A are later expanded into six categories in Paper D. This thesis focuses on challenges that relate to system understanding. The studies addressing *RO.1B* and *RO.1C* explore system understanding yet further. The second 'angle' is the safety-critical domain, *RO.1B*. This research (Paper B) reveals that challenges in the safety domain stem from the recommendations provided by the standards and regulations, that companies have to fulfil. We find challenges relating to upfront, just-in-time, and long term or infrastructural aspects. However, the trade-off between upfront analysis and just-in-time development is still much of a pain point that is under researched. The third and final setting is the agile teams in a waterfall environment, *RO.1C*. This study (Paper C) highlights the challenge of the gap between system-level requirements and agile teams development. At the system-level, requirements are more plan-driven while in the development teams, the pace of change of requirements is easily accommodated. It is a challenge to balance the agility of teams and system-level information needs in large-scale agile system development. This challenge is a reoccurring challenge in all the studies and appears to be affected by the entire process of the organisation including business, process and technological aspects. Here we denote the challenges in those categories that also relate to or sprout from that major challenge and are also reported in our studies. Table 1.5 presents the visualisation.

Organisational issues In this category, challenges relating to organisational discipline are discussed. Agile at scale was received with skepticism from many project and/or product managers as they *lacked the trust in agile methods* since agile methods do not give clear guidelines for project monitoring and control. Also, large-scale organisations commonly invest much *time for inventions and planning.* However, with the introduction of sprints, it is not clear at what level to do it as it would slow down development if given to developers yet at system-level it would mean extensive documentation and handover not recommended in agile development. *Test planning (plan V & V based on requirements)* is another organisational issue that is affected by agility at scale. The agile teams with their product owners do not perform validation plans which are done by the system managers. This introduces a gap between the agile and system testing team. Also the rate of change of requirements could warrant an update to the testing infrastructure. However, the current setting does not give testers the ability to monitor changes and thus have a head-start in improving the test environment.

Process issues Here, we group challenges from the development process that relate to the gap. *Prioritisation of distributed functionality* is suffering at the gap since teams choose to work with simple tasks leaving out the high priority tasks with the excuse of not enough time to implement. This act frustrates system development as it becomes hard to follow development. Additionally, teams want to tailor requirements to their contexts but there is no support for *managing different representations* of requirements. At the same time, system-level development aims to keep artefacts consistent and manageable. This also brings in the *tooling not fit for purpose* challenge. Current tools limit developers' access to system-level requirements making updating of requirements slow and cumbersome. Also, since requirements are usually defined at the beginning of development, there is no obvious way to update them.

Technology issues In this category, both process and organisational aspects are at play, thus issues relating to technologies used. At the higher end, the organisations choose the technologies to use and at the lower end, the processes depend on (are driven by) the technologies used. Teams develop components that must fit into the full system but *lack the overall system knowledge*. They thus might sub-optimise with regard to the bigger system as they are thinking of component quality. This challenge can be motivated by the organisation improving its structures which should help the developers become more proactive than before. It is *hard to trace* user stories on development level to requirements on system-level since developers are also not motivated to create such traces. This challenge is also deepened by the usage of different tools at the development and system level. In the end, the *update to requirements* also suffers developers work with tools different from those at the system level. With large-scale development, there are usually many levels of requirements as they are decomposed making *synchronisation of development* hard as channelling the right information from system level down to the developers becomes time consuming and difficult.

1.5.2 Exploring solution space (*G2*)

The solution space is explored in three ways as well. First, we analyse two popular frameworks (SAFe and LeSS) to identify the solutions they offer for the identified challenges. In relation to balancing team agility and system-level information, scaled frameworks have recommended practices. However, there is no empirical proof of which we are aware. We find a lack of guidance on the tools to use, and the proposed practices are not concrete per se. With this finding, we explore alternatives for solutions. Second we start to recommend with a textual tool (T-Reqs) based on Git that allows the developers to propose changes to the high-level requirements. Lastly, we come up with a catalogue of methodological islands and boundary object types that practitioners should recognise in their organisations to help them try to address the coordination challenges between teams and system-level works. The proposed solution relies on artefacts, specifically boundary objects. In view of this aspect, T-Reqs is a boundary object that can also address the coordination needs of teams in large-systems' development. We recognise process, organisational and technical issues which relate to RE at scale both in the problem space and in the solution space. Our findings in the solution space imply that large-scale companies' RE needs for agile system development are driven by choices made at all those (process, organisation, and technical) levels. Thus, understanding the factors at play in each of the levels could help address the 'gap' challenge.

1.6 Discussion

Just as the technology-, process- and business-related (or organisational) aspects drive the occurrence of methodological islands, the current state of RE challenges has also been grouped in these categories allowing us to understand the challenges better in relation to the solution space. For organisations that have a set organisation structure and defined standards and processes to follow within the organisation, this grouping starts to appear at different levels. At the highest level is the business-related aspects, with processes- and technology-related aspects as subsets. The processes and technologies used are determined by the organisation and thus differences of methods used in teams are easily identified. In this way, organisations only take up processes if they have explored the different ways such processes can be applied to their contexts and thus expect less surprising challenges. In line with recommendations to tailor agile methods to their contexts [84], we further emphasis a company driven adoption.

This thesis expounds on the challenge of a gap between development and system-level requirements. The gap between agile teams and system-level development has been addressed through hybrid development in recent studies. Hybrid development in the most straightforward form consists of practices from different agile methods and also from traditional development methods. We have seen that this form of development has become the current norm for many companies, big and small alike [36]. For large-scale system development, however, hybrid development goes beyond the combination of different methods for development to having two separate sections in the organisation, one working in a plan-driven (traditional) and another working in an agile manner.

Results on the challenges arising from coexistence of traditional and agile methods reveal,e.g., inconsistencies between the implemented requirements and the requirements being tested (test planning), which comes from the development teams not being aware of the high-level requirements. These results are in agreement with, e.g., Kusters et al. [39] who identify a 'lack of linkage of the iterative development process to the test process'. This thesis identifies many challenges in this gap (see Section 1.5. In light of such challenges, the success of agile development at scale becomes questionable. Rolland et al. [6] while challenging assumptions of agile development at scale, posed the same question when they examined the concept of self-organising teams at scale. Although not focused on RE in system development, they pose this question *"Are there models of organizing the development process that can grant team-level autonomy and still ensure efficient inter-team coordination?"* Dealing with safety-critical components of the software and a solid foundation on waterfall methods limits the autonomy that teams can have at scale. We question how far team autonomy can be stretched in that context.

Additionally, our findings show that requirements change so fast during development that it is hard to track the change in dependencies between requirements among teams. At scale these challenges change form and become essential to address [6]. In the systems engineering context with parallel development of hardware and software, these challenges become critical, especially when we add the RE context as then the documentation and entire validation process could be terribly slowed. Struggles to find solutions for such challenges are still on-going.

Kuusinen et al. [40] recommend strategies that include convincing management to change their mindset about agile methods. Relating to our discussions in interviews and workshops, managers seem to be convinced but still challenges exist since, given the scale and related constraints, it is hard to be fully agile. We believe that buy-in to agile development at scale has to span to the known 'non-agile' departments like hardware departments as well. This thesis presents two fundamental attempts to solving the identified challenges. Coming from an RE perspective, development is turned into a communication problem and eventually into a coordination problem when we talk of software development among 6 or more teams. At that level, good strategies for coordination of development need to be devised. T-Reqs and the concept of Boundary Objects and Methodological Islands (BOMI) introduced in this thesis present ideas towards potentially working solutions.

With T-Reqs, the organisations do not change their infrastructure as such. The known rules and standards of development still apply with the flexibility of allowing the developers propose changes and work directly with the requirements. In this way, requirements updating challenges are reduced. Some studies have proposed having cross-functional teams [54,85] to help in updating requirements across the 'gap'. These solutions are applicable for intra-team coordination and communication, and become problematic across teams. T-Reqs ensures coordination across teams, and requirements are always to the most recent version across teams. In addition, the BOMI concept allows for an agile or autonomous way of identifying communication need and thus creating it. This suggestion relates to the practice of ad-hoc meetings [60] which was found effective. Boundary objects as artefacts can come up whenever they are needed and depending on the development aims, can be maintained or discarded afterwards. We, however, aim for reusable boundary objects to reduce workload and also carry over knowledge from previous development.

In short, T-Reqs and BOMI potentally address some of the identified challenges. T-Reqs potentially addresses the challenge of lack of agile tooling (tooling not fit for purpose), managing requirements updates during development and traceability management. BOMI concept is good for advising processes to synchrnonise development, managing different representations and traceability management. It is almost impossible to define a generic solution and thus we propose these two concepts as they can be customised to the particular company needs. Although they may appear constraining, these solution candidates offer teams an opportunity to be autonomous and efficient while working with system-level information that is process-driven. These two solution proposals call for a full organisational buy-in to agile development.

In the context of RE in large-scale agile systems development, whereas agile RE aims at using agile practices to do RE, we argue for RE for agile development (RE4Agile) where we perform RE to support agile development. This means that the known methods of doing RE would still hold. However, they would not be done one off as before and would also be done incrementally and concurrent to development. This thesis argues that, for large-scale systems development organisations, an approach that takes advantage of the speed and change handling of agile development while also building on the coordination provided by traditional methods would be more adequate. We thus recommend that such large-systems companies to aim for an informed hybrid approach that takes the best of both worlds.

1.7 Conclusions and Future Work

This research aimed to uncover RE challenges and solution candidates for the use of agile development in systems development companies. Through use of a series of qualitative empirical studies, we contribute a catalogue of RE challenges in large-scale agile systems development. We found challenges relating to maintenance and long-term product support which agile methods tend to ignore. Future research could explore that direction a bit more. This thesis dwells on the challenge of system understanding to highlight the challenges of interaction between agile teams and system-level requirements. While companies continue to struggle with the gap, we see a need for future research specifically aiming at investigation of trade-offs between effort done upfront and just-in-time. We also advocate for guidelines on how to shift more effort into just-in-time analysis as we aim towards RE for Agile development.

Through exploration of solution candidates, this thesis also provides a first attempt to skip the divide between upfront and just-in-time analysis. While uncovering the solution candidates, we noticed practices recommended in literature but did not yet find empirical evidence confirming their usefulness for large-scale systems development. We thus encourage future research to not only produce more practices to solve open challenges, but also focus on evaluation of existing large-scale agile system proposals from a requirements perspective. We also explored solutions offered by two of the popular large-scale frameworks, SAFe and LeSS. We found that whereas they have recommended practices, these are still abstract and not concrete enough for the practitioners to implement with certainty. Perhaps studies detailing the successful implementation of these frameworks while elaborating how the proposed practices were met could help in that aspect.

We introduce, T-Reqs, a custom made solution in one of the companies which we, together with its pioneers, created an open-source version of. Although the tool has not yet been tried in other environments or contexts, it is a first step towards addressing many of the identified challenges for large-scale systems development. We aim to continue adding functionality and exploring its usage and thus welcome contributions towards making the open source tool more usable.

We started to explore practical ways to address knowledge management in large-scale agile systems development. For that, we charted a landscape of methodological islands and boundary objects (BOMI) which practitioners found useful for discussing potential process and tool improvements. We are currently working on translating the landscape into a model that can benefit practitioners. Future research could use our landscape to prioritise and scope knowledge management needs. Additionally, a quantitative survey could provide information on which boundary objects and methodological islands are most frequent. With these contributions in the solution space, we provide a sketch of promising approaches, e.g. through agile tools and BOMIs, on how these challenges could be approached.

CHAPTER 2 - Papers A - F omitted

Bibliography

[1] U. Eliasson, R. Heldal, E. Knauss, and P. Pelliccione, “The need of complementing plan-driven requirements engineering with emerging communication: Experiences from volvo car group,” in *RE Conf.* IEEE, 2015, pp. 372–381.

[2] J. Pernstål, T. Gorschek, R. Feldt, and D. Florén, “Requirements communication and balancing in large-scale software-intensive product development,” *Information and Software Technology*, vol. 67, pp. 44–64, 2015.

[3] K. Pohl, *Requirements engineering: fundamentals, principles, and techniques.* Springer Publishing Company, Incorporated, 2010.

[4] T. Clancy, “The standish group chaos report,” *Project Smart*, 2014.

[5] I. Sommerville and P. Sawyer, *Requirements engineering: a good practice guide.* John Wiley & Sons, Inc., 1997.

[6] K. Rolland, T. Dingsoyr, B. Fitzgerald, and K.-J. Stol, “Problematizing agile in the large: alternative assumptions for large-scale agile development,” in *39th International Conference on Information Systems.* Association for Information Systems (AIS), 2016.

[7] M. Paasivaara and C. Lassenius, “Challenges and success factors for large-scale agile transformations: A research proposal and a pilot study,” in *Proc. of the Scientific WS Proc. of XP2016.* ACM, 2016, p. 9.

[8] K. Dikert, M. Paasivaara, and C. Lassenius, “Challenges and success factors for large-scale agile transformations: A systematic literature review,” *Journal of Systems and Software*, 2016.

[9] L. Lagerberg, T. Skude, P. Emanuelsson, K. Sandahl, and D. Ståhl, “The impact of agile principles and practices on large-scale software development projects: A multiple-case study of two projects at ericsson,” in *ACM / IEEE Int. Symposium on Empirical Software Engineering and Measurement*, 2013, pp. 348–356.

[10] C. Berger and U. Eklund, “Expectations and challenges from scaling agile in mechatronics-driven companies – a comparative case study,” in *Proc. of 16th Int. Conf. on Agile Processes in Software Engineering and Extreme Programming (XP ’15)*, 2015, pp. 15–26.

[11] T. Dybå and T. Dingsøyr, “Empirical studies of agile software development: A systematic review,” *Information and software technology*, vol. 50, no. 9, pp. 833–859, 2008.

[12] V. Vinekar, C. W. Slinkman, and S. Nerur, “Can agile and traditional systems development approaches coexist? an ambidextrous view,” *Information systems management*, vol. 23, no. 3, pp. 31–42, 2006.

[13] M. Lindvall, D. Muthig, A. Dagnino, C. Wallin, M. Stupperich, D. Kiefer, J. May, and T. Kahkonen, “Agile software development in large organizations,” *Computer*, vol. 37, no. 12, pp. 26–34, 2004.

[14] M. Laanti, O. Salo, and P. Abrahamsson, “Agile methods rapidly replacing traditional methods at nokia: A survey of opinions on agile transformation,” *Information and Softw. Techn.*, vol. 53, no. 3, pp. 276–290, 2011.

[15] I. Inayat, S. S. Salim, S. Marczak, M. Daneva, and S. Shamshirband, “A systematic literature review on agile requirements engineering practices and challenges,” *Computers in human behavior*, vol. 51, pp. 915–929, 2015.

[16] V. T. Heikkila, D. Damian, C. Lassenius, and M. Paasivaara, “A mapping study on requirements engineering in agile software development,” in *41st Euromicro Conf. on Softw. Eng. and Advanced Applications (SEAA '15)*, 2015, pp. 199–207.

[17] F. Paetsch, A. Eberlein, and F. Maurer, “Requirements engineering and agile software development.” in *WETICE*, vol. 3, 2003, p. 308.

[18] D. Leffingwell, *Scaling Software Agility: Best Practices for Large Enterprises.* Addison-Wesley Professional, 2011.

[19] B. Ramesh, L. Cao, and R. Baskerville, “Agile requirements engineering practices and challenges: an empirical study,” *Information Systems Journal*, vol. 20, no. 5, pp. 449–480, 2010.

[20] B. Boehm, “Some future trends and implications for systems and software engineering processes,” *Systems Engineering*, vol. 9, no. 1, pp. 1–19, 2006.

[21] J. Mössinger, “Software in automotive systems,” *IEEE software*, vol. 27, no. 2, 2010.

[22] M. Glinz, “A glossary of requirements engineering terminology,” *Standard Glossary of the Certified Professional for Requirements Engineering (CPRE) Studies and Exam, Version*, vol. 1, 2011.

[23] V. T. Heikkilä, M. Paasivaara, C. Lasssenius, D. Damian, and C. Engblom, “Managing the requirements flow from strategy to release in large-scale agile development: a case study at ericsson,” *Empirical Software Engineering*, pp. 1–45, 2017.

[24] W. W. Royce, “Managing the development of large software systems: concepts and techniques,” in *Proceedings of the 9th international conference on Software Engineering*, 1987, pp. 328–338.

[25] A. Alliance, "Home: http://www. agilealliance. org," Accessed 14th November 2018.

[26] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas, "Manifesto for agile software development," 2001.

[27] L. Williams, "What agile teams think of agile principles," *Communications of the ACM*, vol. 55, no. 4, pp. 71–76, 2012.

[28] K. Schwaber and M. Beedle, *Agile software development with Scrum.* Prentice Hall Upper Saddle River, 2002, vol. 1.

[29] K. Beck, *Extreme programming explained: embrace change.* Addison-Wesley, 1999.

[30] B. Meyer, *Agile! The Good, the Hype and the Ugly.* Springer, 2014.

[31] D. Leffingwell, "Mastering the iteration: An agile white paper," *Rally Software Development Corporation*, 2007.

[32] A. De Lucia and A. Qusef, "Requirements engineering in agile software development," *Journal of emerging technologies in web intelligence*, vol. 2, no. 3, pp. 212–220, 2010.

[33] B. Boehm and R. Turner, "Management challenges to implementing agile processes in traditional development organizations," *IEEE software*, vol. 22, no. 5, pp. 30–39, 2005.

[34] A. Sillitti and G. Succi, "Requirements engineering for agile methods," in *Engineering and Managing Software Requirements.* Springer, 2005, pp. 309–326.

[35] R. S. Carson, "4.2. 1 can systems engineering be agile? development lifecycles for systems, hardware, and software," in *INCOSE International Symposium*, vol. 23, no. 1. Wiley Online Library, 2013, pp. 16–28.

[36] M. Kuhrmann, P. Diebold, J. Münch, P. Tell, V. Garousi, M. Felderer, K. Trektere, F. McCaffery, O. Linssen, E. Hanser *et al.*, "Hybrid software and system development in practice: waterfall, scrum, and beyond," in *Proceedings of International Confonference on Software System Process (ICSSP).* ACM, 2017, pp. 30–39.

[37] G. Theocharis, M. Kuhrmann, J. Münch, and P. Diebold, "Is water-scrum-fall reality? on the use of agile and traditional development practices," in *Int. Conf. on Product-Focused Software Process Improvement*, 2015, pp. 149–166.

[38] P. Tell, J. Klünder, S. Küpper, D. Raffo, S. G. MacDonell, J. Münch, D. Pfahl, O. Linssen, and M. Kuhrmann, "What are hybrid development methods made of? an evidence-based characterization," in *Proceedings of the International Conference on Software and System Processes*, ser. ICSSP '19. Montreal, Quebec, Canada: IEEE Press, 2019, pp. 105–114.

[39] R. J. Kusters, Y. van de Leur, W. G. Rutten, and J. J. Trienekens, “When agile meets waterfall - investigating risks and problems on the interface between agile and traditional software development in a hybrid development organization.” in *Int. Conf. on Enterprise Information Systems (ICEIS)*, vol. 2, 2017, pp. 271–278.

[40] K. Kuusinen, P. Gregory, H. Sharp, and L. Barroca, “Strategies for doing agile in a non-agile environment,” in *ESEM*. ACM, 2016, p. 5.

[41] D. West, M. Gilpin, T. Grant, and A. Anderson, “Water-scrum-fall is the reality of agile for most organizations today,” *Forrester Research*, vol. 26, 2011.

[42] O. Salo and P. Abrahamsson, “Agile methods in european embedded software development organisations: a survey on the actual use and usefulness of extreme programming and scrum,” *IET software*, vol. 2, no. 1, pp. 58–64, 2008.

[43] U. Eklund, H. Holmström Olsson, and N. J. Strøm, “Industrial challenges of scaling agile in mass-produced embedded systems,” in *Proc. of Int. WS on Agile Methods. Large-Scale Dev., Refactoring, Testing, and Estimation*, 2014, pp. 30–42.

[44] J. Pernstål, A. Magazinius, and T. Gorschek, “A study investigating challenges in the interface between product development and manufacturing in the development of software-intensive automotive systems,” *International Journal of Software Engineering and Knowledge Engineering (IJSEKE)*, vol. 22, no. 07, pp. 965–1004, 2012.

[45] T. Dingsøyr, N. B. Moe, T. E. Fægri, and E. A. Seim, “Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation,” *Empirical Software Engineering*, vol. 23, no. 1, pp. 490–520, 2018.

[46] T. Dingsøyr, T. E. Fægri, and J. Itkonen, “What is large in large-scale? a taxonomy of scale for agile software development,” in *Proceedings of 15th International Conference on Product-Focused Software Process Improvement (PROFES ’14)*, A. Jedlitschka, P. Kuvaja, M. Kuhrmann, T. Männistö, J. Münch, and M. Raatikainen, Eds., 2014, pp. 273–276.

[47] M. Jørgensen, “Do agile methods work for large software projects?” in *Agile Processes in Software Engineering and Extreme Programming*, J. Garbajosa, X. Wang, and A. Aguiar, Eds. Cham: Springer International Publishing, 2018, pp. 179–190.

[48] C. Ebert and M. Paasivaara, “Scaling agile,” *IEEE Software*, vol. 34, no. 6, pp. 98–103, 2017.

[49] V. One and C. Net, “13th annual state of agile survey,” *Survey. Accessed online*, vol. 15, 2019.

[50] M. Paasivaara, “Adopting safe to scale agile in a globally distributed organization,” in *2017 IEEE 12th International Conference on Global Software Engineering (ICGSE)*, 2017, pp. 36–40.

[51] O. Turetken, I. Stojanov, and J. J. Trienekens, "Assessing the adoption level of scaled agile development: a maturity model for scaled agile framework," *Journal of Software: Evolution and process*, vol. 29, no. 6, p. e1796, 2017.

[52] K. Conboy and N. Carroll, "Implementing large-scale agile frameworks: Challenges and recommendations," *IEEE Software*, vol. 36, no. 2, pp. 44–50, 2019.

[53] M. Laanti and P. Kettunen, "Safe adoptions in finland: a survey research," in *International Conference on Agile Software Development.* Springer, 2019, pp. 81–87.

[54] E. Bjarnason, K. Wnuk, and B. Regnell, "A case study on benefits and side-effects of agile practices in large-scale requirements engineering," in *Proc. of 1st WS on Agile Reqts. Eng.*, 2011.

[55] K. Schneider, *Experience and knowledge management in software engineering.* Springer Science & Business Media, 2009.

[56] T. Chau, F. Maurer, and G. Melnik, "Knowledge sharing: Agile methods vs. tayloristic methods," in *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on.* IEEE, 2003, pp. 302–307.

[57] K. Petersen and C. Wohlin, "The effect of moving from a plan-driven to an incremental software development approach with agile practices," *Empirical Software Engineering*, vol. 15, no. 6, pp. 654–693, 2010.

[58] M. Hummel, C. Rosenkranz, and R. Holten, "The role of communication in agile systems development," *Business & Information Systems Engineering*, vol. 5, no. 5, pp. 343–355, 2013.

[59] M. Pikkarainen, J. Haikara, O. Salo, P. Abrahamsson, and J. Still, "The impact of agile practices on communication in software development," *Empirical Software Engineering*, vol. 13, no. 3, pp. 303–337, 2008.

[60] H. Nyrud and V. Stray, "Inter-team coordination mechanisms in large-scale agile," in *Proceedings of the XP2017 Scientific Workshops*, ser. XP '17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3120459.3120476

[61] G. Wagenaar, S. Overbeek, G. Lucassen, S. Brinkkemper, and K. Schneider, "Working software over comprehensive documentation–rationales of agile teams for artefacts usage," *Journal of Software Engineering Research and Development*, vol. 6, no. 1, p. 7, 2018.

[62] R. Wohlrab, P. Pelliccione, E. Knauss, and M. Larsson, "Boundary objects and their use in agile systems engineering," *Journal of Software: Evolution and Process*, vol. 31, no. 5, p. e2166, 2019.

[63] M. Barrett and E. Oborn, "Boundary object use in cross-cultural software development teams," *Human Relations*, vol. 63, no. 8, pp. 1199–1221, 2010.

[64] S. L. Star and J. R. Griesemer, “Institutional ecology,translations’ and boundary objects: Amateurs and professionals in berkeley’s museum of vertebrate zoology, 1907-39,” *Social studies of science*, vol. 19, no. 3, pp. 387–420, 1989.

[65] J. M. C. De Gea, J. Nicolás, J. L. F. Alemán, A. Toval, C. Ebert, and A. Vizcaíno, “Requirements engineering tools: Capabilities, survey and assessment,” *Information and Software Technology*, vol. 54, no. 10, pp. 1142–1157, 2012.

[66] W. Alsaqaf, M. Daneva, and R. Wieringa, “Quality requirements in large-scale distributed agile projects–a systematic literature review,” in *International Working Conference on Requirements Engineering: Foundation for Software Quality.* Springer, 2017, pp. 219–234.

[67] B. Fitzgerald, K.-J. Stol, R. O’Sullivan, and D. O’Brien, “Scaling agile methods to regulated environments: An industry case study,” in *Proc. of 35th Int. Conf. on Software Engineering (ICSE)*, 2013, pp. 863–872.

[68] G. K. Hanssen, B. Haugset, T. Stålhane, T. Myklebust, and I. Kulbrandstad, “Quality assurance in scrum applied to safety critical software,” in *Proc. of Int. Conf. on Agile Software Development (XP)*, Edinburgh, Scotland, UK, 2016, pp. 92–103.

[69] ISO, “Road vehicles – Functional safety,” 2011.

[70] L. T. Heeager and P. A. Nielsen, “A conceptual model of agile software development in a safety-critical context: A systematic literature review,” *Information and Software Technology*, vol. 103, pp. 22–39, 2018.

[71] E. S. Grant, “Requirements engineering for safety critical systems: An approach for avionic systems,” in *2nd Int. Conf. on Computer and Communications (ICCC), 2016.* IEEE, 2016, pp. 991–995.

[72] F. McCaffery, M. Pikkarainen, and I. Richardson, “Ahaa –agile, hybrid assessment method for automotive, safety critical smes,” in *Proceedings of the 30th International Conference on Software Engineering*, ser. ICSE ’08. New York, NY, USA: Association for Computing Machinery, 2008, p. 551–560.

[73] L. Provenzano and K. Hänninen, “Specifying software requirements for safety-critical railway systems: An experience report,” in *Int. Working Conf. on Requirements Engineering: Foundation for Software Quality.* Springer, 2017, pp. 363–369.

[74] P. Runeson and M. Höst, “Guidelines for conducting and reporting case study research in software engineering,” *Empirical software engineering*, vol. 14, no. 2, pp. 131–164, 2009.

[75] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian, “Selecting empirical methods for software engineering research,” in *Guide to advanced empirical software engineering.* Springer, 2008, pp. 285–311.

[76] P. Runeson, M. Höst, A. Rainer, and B. Regnell, *Case study research in software engineering: Guidelines and examples*, 1st ed. Hokoben, New Jersey: John Wiley & Sons, 2012.

[77] L. A. Palinkas, S. M. Horwitz, C. A. Green, J. P. Wisdom, N. Duan, and K. Hoagwood, "Purposeful sampling for qualitative data collection and analysis in mixed method implementation research," *APM&MHSR*, vol. 42, no. 5, pp. 533–544, 2015.

[78] G. R. Gibbs, *Analysing qualitative data.* Sage, 2008.

[79] D. Budgen and P. Brereton, "Performing systematic literature reviews in software engineering," in *Proceedings of the 28th international conference on Software engineering.* ACM, 2006, pp. 1051–1052.

[80] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering." in *EASE*, vol. 8, 2008, pp. 68–77.

[81] B. Kitchenham, "Procedures for performing systematic reviews," *Keele, UK, Keele University*, vol. 33, no. 2004, pp. 1–26, 2004.

[82] D. Leffingwell *et al.*, "Scaled agile framework 3.0," 2014.

[83] K. Forsberg and H. Mooz, "The relationship of system engineering to the project cycle," in *INCOSE International Symposium*, vol. 1, no. 1. Wiley Online Library, 1991, pp. 57–65.

[84] A. S. Campanelli and F. S. Parreiras, "Agile methods tailoring – a systematic literature review," *Journal of Systems and Software*, vol. 110, pp. 85 – 100, 2015.

[85] G. Liebel, M. Tichy, E. Knauss, O. Ljungkrantz, and G. Stieglbauer, "Organisation and communication problems in automotive requirements engineering," *Requirements Engineering*, vol. 23, no. 1, pp. 145–167, 2018.

[86] T. Kahkonen, "Agile methods for large organizations-building communities of practice," in *Agile Dev. Conf., 2004*, 2004, pp. 2–10.

[87] K. Beck, *Extreme programming explained: embrace change.* addison-wesley professional, 2000.

[88] J. Savolainen, J. Kuusela, and A. Vilavaara, "Transition to agile development-rediscovery of important requirements engineering practices," in *18th Int. Req. Eng. Conf.* IEEE, 2010, pp. 289–294.

[89] K. Wiklund, D. Sundmark, S. Eldh, and K. Lundqvist, "Impediments in agile software development: An empirical investigation," in *Proc of Product-Focused SW Process Impr.*, 2013, pp. 35–49.

[90] T. Chow and D.-B. Cao, "A survey study of critical success factors in agile software projects," *Journal of Systems and Software*, vol. 81, no. 6, pp. 961–971, 2008.

[91] D. Stahl and J. Bosch, "Modelling continuous integration practice differences in industry software development," *Systems and Software*, vol. 87, pp. 48–59, 2014.

[92] R. Hoda, J. Noble, and S. Marshall, "Self-organizing roles on agile software development teams," *IEEE Transactions on Software Engineering*, vol. 39, no. 3, pp. 422–444, 2013.

[93] F. Evbota, E. Knauss, and A. Sandberg, "Scaling up the planning game: Collaboration challenges in large-scale agile product development," in *Proc. of 17th Int'l Conf. on Agile Softw. Dev. (XP)*, Edinburgh, UK, 2016.

[94] R. Kasauli, E. Knauss, A. Nilsson, and S. Klug, "Adding value every sprint: A case study on large-scale continuous requirements engineering," in *Proc. of 3rd WS on Cont. Reqts. Eng.*, Essen, Germany, 2017.

[95] E. Bjarnason, M. Unterkalmsteiner, M. Borg, and E. Engström, "A multi-case study of agile requirements engineering and the use of test cases as requirements," *Information and Software Technology*, vol. 77, pp. 61–79, 2016.

[96] R. Wohlrab, J.-P. Steghöfer, E. Knauss, S. Maro, and A. Anjorin, "Collaborative traceability management: Challenges and opportunities," in *Proc. of 24th Int. Reqts. Eng. Conf. (RE)*, 2016, pp. 216–225.

[97] P. A. Laplante and J. F. DeFranco, "Software engineering of safety-critical systems: Themes from practitioners," *IEEE Transactions on Reliability*, vol. 66, no. 3, pp. 825–836, 2017.

[98] J. Hatcliff, A. Wassyng, T. Kelly, C. Comar, and P. Jones, "Certifiably safe software-dependent systems: challenges and directions," in *Proceedings of the on Future of Software Engineering.* ACM, 2014, pp. 182–200.

[99] M. Vuori, "Agile development of safety-critical software," *Tampere University of Technology*, vol. 14, 2011.

[100] J. P. Notander, M. Höst, and P. Runeson, "Challenges in flexible safety-critical software development – an industrial qualitative survey," in *Proc. of Int. Conf. on Product Focused Software Process Improvement (PROFES)*, Paphos, Cyprus, 2013, pp. 283–297.

[101] X. Ge, R. F. Paige, and J. A. McDermid, "An iterative approach for development of safety-critical software and safety arguments," in *Proc. of AGILE Conf.* Nashville, TN, USA: IEEE, 2010, pp. 35–43.

[102] P. Pelliccione, E. Knauss, and et al., "Automotive architecture framework: the experience of volvo cars," *Journal of systems architecture*, 2017.

[103] M. Kaisti, V. Rantala, T. Mujunen, S. Hyrynsalmi, K. Könnölä, T. Mäkilä, and T. Lehtonen, "Agile methods for embedded systems development-a literature review and a mapping study," *EURASIP Journal on Embedded Systems*, vol. 2013, no. 1, p. 15, 2013.

[104] O. Cawley, X. Wang, and I. Richardson, “Lean/agile software development methodologies in regulated environments–state of the art,” in *Proc. of Conf. on Lean Enterprise Software and Systems (LESS)*, Helsinki, Finland, 2010, pp. 31–36.

[105] B. Kitchenham and S. Charters, “Guidelines for performing systematic literature reviews in software engineering,” in *Technical report, Ver. 2.3. EBSE.* sn, 2007.

[106] P. Diebold and M. Dahlem, “Agile practices in practice: A mapping study,” in *Proc. of the 18th Int. Conf. on Evaluation and Assessment in Software Engineering*, ser. EASE ’14, 2014, pp. 30:1–30:10.

[107] C. Wohlin, “Guidelines for snowballing in systematic literature studies and a replication in software engineering,” in *Proceedings of the 18th international conference on evaluation and assessment in software engineering.* ACM, 2014, p. 38.

[108] K. Łukasiewicz and J. Górski, “Agilesafe - a method of introducing agile practices into safety-critical software development processes,” in *Proc. of Federated Conf. on Computer Science and Information Systems (FedCSIS)*, Gdansk, Poland, 2016, pp. 1549–1552.

[109] T. Stålhane and T. Myklebust, “Early safety analysis,” in *Proceedings of the Scientific Workshop Proceedings of XP2016.* ACM, 2016, p. 19.

[110] J. Schmidt and K. Weyrauch, “Getting agile with medical device development,” *Biomedical Instrumentation & Technology*, vol. 47, no. 3, pp. 221–223, 2013.

[111] Y. Wang and S. Wagner, “Toward integrating a system theoretic safety analysis in an agile development process.” in *Proc. of Software Engineering (Workshops)*, Wien, Austria, 2016, pp. 156–159.

[112] T. Myklebust, T. Stålhane, and N. Lyngby, “An agile development process for petrochemical safety conformant software,” in *Proc. of Annual Reliability and Maintainability Symposium (RAMS)*, Tucson, AZ, USA, 2016.

[113] G. K. Hanssen, G. Wedzinga, and M. Stuip, “An assessment of avionics software development practice: Justifications for an agile development process,” in *Proc. of Int. Conf. on Agile Software Dev. (XP)*, Cologne, Germany, 2017, pp. 217–231.

[114] L. T. Heeager, “How can agile and documentation-driven methods be meshed in practice?” in *(XP)*, Rome, Italy, 2014, pp. 62–77.

[115] R. Rasmussen, T. Hughes, J. Jenks, and J. Skach, “Adopting agile in an fda regulated environment,” in *Proc. of AGILE Conf.*, Chicago, IL, USA, 2009, pp. 151–155.

[116] T. Stålhane, G. K. Hanssen, T. Myklebust, and B. Haugset, “Agile change impact analysis of safety critical software,” in *Proc. of Int. Conf. on Computer Safety, Reliability, and Security (SAFECOMP)*, Firenze, Italy, 2014, pp. 444–454.

[117] S. Vost and S. Wagner, "Keeping continuous deliveries safe," in *Proc. of 39th Int. Conf. on SW Eng. (ICSE)*, Buenos Aires, Argentina, 2017.

[118] M. McHugh, F. McCaffery, and G. Coady, "An agile implementation within a medical device software organisation," in *(SPICE)*, Vilnius, Lithuania, 2014, pp. 190–201.

[119] J. Górski and K. Łukasiewicz, "Assessment of risks introduced to safety critical software by agile practices-a software engineer's perspective," *Computer Science (CSCI)*, vol. 13, no. 4, pp. 165–182, 2012.

[120] P. A. Rottier and V. Rodrigues, "Agile development in a medical device company," in *Proc. of AGILE Conf.*, Toronto, ON, Canada, 2008, pp. 218–223.

[121] S. Wolff, "Scrum goes formal: Agile methods for safety-critical systems," in *Formal Methods in Software Engineering: Rigorous and Agile Approaches (FormSERA)*, Zurich, Switzerland, 2012, pp. 23–29.

[122] R. F. Paige, A. Galloway, R. Charalambous, X. Ge, and P. J. Brooke, "High-integrity agile processes for the development of safety critical software," *Int. Journal of Critical Computer-Based Systems (IJCCBS)*, vol. 2, no. 2, pp. 181–216, 2011.

[123] Y. Wang, J. Ramadani, and S. Wagner, "An exploratory study on applying a scrum development process for safety-critical systems," in *Proc. of Int. Conf. on Product-Focused Software Process Improvement (PROFES)*, 2017, pp. 324–340.

[124] Y. Wang, I. Bogicevic, and S. Wagner, "A study of safety documentation in a scrum development process," in *Proc. of XP Scientific Workshops (XP WS)*, Cologne, Germany, 2017.

[125] H. Jonsson, S. Larsson, and S. Punnekkat, "Agile practices in regulated railway software development," in *Proc. of 23rd Int. Symp. on Software Reliability Engineering, Workshops (ISSREW WS)*, Dallas, TX, USA, 2012, pp. 355–360.

[126] W. Kuchinke, C. Krauth, and T. Karakoyun, "Agile software development requires an agile approach for computer system validation of clinical trials software products," in *Proc. of eChallenges Conf.*, Belfast, UK, 2014, pp. 1–8.

[127] M. McHugh, F. McCaffery, and V. Casey, "Barriers to adopting agile practices when developing medical device software," in *Proc. of Int. Conf. on SW Process Impr. and Capability Determ. (SPICE)*, Palma de Mallorca, Spain, 2012, pp. 141–147.

[128] K. Trektere, F. McCaffery, M. Lepmets, and G. Barry, "Tailoring mdevspice ® for mobile medical apps," in *Software and System Processes (ICSSP)*, Austin (TX), USA, 2016, pp. 106–110.

[129] O. Doss and T. Kelly, "Addressing the 4+1 software assurance processes within scrum," in *Proc. of XP Scientific Workshops*, Edinburgh, Scotland, UK, 2016, 5 pages.

[130] T. Stålhane and T. Myklebust, "The agile safety case," in *Proc. of Int. Conf. on Computer Safety, Reliability, and Security (SAFECOMP)*. Trondheim, Norway: Springer, 2016, pp. 5–16.

[131] A. Wils, S. V. Baelen, T. Holvoet, and K. D. Vlaminck, "Agility in the avionics software world," in *Proc. of (XP)*, Limerick, Ireland, 2006, pp. 123–132.

[132] O. Doss, T. Kelly, T. Stålhane, B. Haugset, and M. Dixon, "Integration of the 4+1 software safety assurance principles with scrum," in *Proc. of European Conf. on Software Process Improvement (EuroSPI)*, Ostrava, Czech Republic, 2017, pp. 72–82.

[133] A. Abdelaziz, Y. El-Tahir, and R. Osman, "Adaptive software development for developing safety critical software," in *Proc. of Int. Conf. on Computing, Control, Networking, Electronics and Embedded Systems Eng. (ICCNEEE)*, Khartoum, Sudan, 2015, pp. 41–46.

[134] J. Górski and K. Łukasiewicz, "Towards agile development of critical software," in *In Proc. of Int. Workshop on Software Engineering for Resilient Systems (SERENE)*, 2013, pp. 48–55.

[135] P. E. McMahon, "Cmmi the agile way in constrained and regulated environments," *Journal of Defense Software Engineering (Crosstalk)*, vol. JulAug, pp. 10–15, 2016.

[136] K. Gary, A. Enquobahrie, L. Ibanez, P. Cheng, Z. Yaniv, K. Cleary, S. Kokoori, B. Muffih, and J. Heidenreich, "Agile methods for open source safety-critical software," *Journal of Software: Practice and Experience (SPE)*, vol. 41, no. 9, pp. 945–962, 2011.

[137] G. Van Waardenburg and H. Van Vliet, "When agile meets the enterprise," *Information and software technology*, vol. 55, no. 12, pp. 2154–2171, 2013.

[138] R. Kasauli, G. Liebel, E. Knauss, S. Gopakumar, and B. Kanagwa, "Requirements engineering challenges in large-scale agile system development," in *25th Int. Requirements Engineering Conference (RE)*. IEEE, 2017, pp. 352–361.

[139] R. Wohlrab, P. Pelliccione, E. Knauss, and S. C. Gregory, "The problem of consolidating re practices at scale: An ethnographic study," in *REFSQ*. Springer, 2018, pp. 155–170.

[140] F. Almeida, "Challenges in migration from waterfall to agile environments," *World Journal of Computer Application and Technology*, vol. 5, no. 3, pp. 39–49, 2017.

[141] S. Bannink, "Challenges in the transition from waterfall to scrum–a casestudy at portbase," in *20th Twente Student Conference on Information Technology*, 2014.

[142] S. Theobald and P. Diebold, “Interface problems of agile in a non-agile environment,” in *International Conference on Agile Software Development*. Porto, Portugal: Springer, 2018, pp. 123–130.

[143] A. Bucaioni, A. Cicchetti, F. Ciccozzi, M. Kodali, and M. Sjödin, “Alignment of requirements and testing in agile: An industrial experience,” in *Information Technology - New Generations*, S. Latifi, Ed. Cham: Springer International Publishing, 2018, pp. 225–232.

[144] C. Khalil, “The state of the practice of agile and plan-driven approaches in ict development projects: An exploratory research study,” in *Digital Technology and Organizational Change*. Verona, Italy: Springer, 2018, pp. 25–33.

[145] M. Kuhrmann, P. Diebold, J. Munch, P. Tell, K. Trektere, F. McCaffery, V. Garousi, M. Felderer, O. Linssen, E. Hanser, and C. R. Prause, “Hybrid software development approaches in practice: A european perspective,” *IEEE Software*, vol. 36, no. 4, pp. 20–31, July 2019.

[146] R. K. Yin, “Case study research: Design and methods 4th ed,” in *United States: Library of Congress Cataloguing-in-Publication Data*, vol. 2, 2009.

[147] A. Alliance, “Glossary: Scrum master,” 2019, last visit: 2019-Oct-18. [Online]. Available: https://www.agilealliance.org/glossary/scrum-master/

[148] E. Knauss, A. Andersson, M. Rybacki, and E. Israelsson, “Research preview: Supporting requirements feedback flows in iterative system development,” in *Proceedings of 21st International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ? 15)*. Essen, Germany: Springer, Cham, 2015, pp. 277–283.

[149] S. Balaji and M. S. Murugaiyan, “Waterfall vs. v-model vs. agile: A comparative study on sdlc,” *International Journal of Information Technology and Business Management*, vol. 2, no. 1, pp. 26–30, 2012.

[150] E. Knauss, G. Liebel, J. Horkoff, R. Wohlrab, R. Kasauli, F. Lange, and P. Gildert, “T-reqs: Tool support for managing requirements in large-scale agile system development,” in *2018 IEEE 26th International Requirements Engineering Conference (RE)*. Banff, Alberta, Canada: IEEE, 2018, pp. 502–503.

[151] E. J. Uusitalo, M. Komssi, M. Kauppinen, and A. M. Davis, “Linking requirements and testing in practice,” in *16th RE Conf.* IEEE, 2008, pp. 265–270.

[152] F. G. de Oliveira Neto, J. Horkoff, E. Knauss, R. Kasauli, and G. Liebel, “Challenges of aligning requirements engineering and system testing in large-scale agile: A multiple case study,” in *REW*. IEEE, 2017, pp. 315–322.

[153] G. Kalus and M. Kuhrmann, “Criteria for software process tailoring: A systematic review,” in *Proceedings of the 2013 International Conference*

on Software and System Process, ser. ICSSP 2013. New York, NY, USA: Association for Computing Machinery, 2013, p. 171–180.

[154] C. Larman and B. Vodde, *Large-scale scrum: More with LeSS.* Addison-Wesley Professional, 2016.

[155] R. Knaster and D. Leffingwell, *SAFe 4.0 Distilled: Applying the Scaled Agile Framework for Lean Software and Systems Engineering.* Addison-Wesley Professional, 2017.

[156] Ö. Uludag, M. Kleehaus, C. Caprano, and F. Matthes, "Identifying and structuring challenges in large-scale agile development based on a structured literature review," in *IEEE 22nd International Enterprise Distributed Object Computing Conference (EDOC)*, Oct 2018, pp. 191–197.

[157] D. Leffingwell, *Agile software requirements: lean requirements practices for teams, programs, and the enterprise.* Addison-Wesley Professional, 2010.

[158] Campbell-Pretty, *Tribal Unity: Getting from Teams to Tribes by Creating a One Team Culture.* CreateSpace Independent Publishing Platform, 2016.

[159] M. Khurum, T. Gorschek, and M. Wilson, "The software value map—an exhaustive collection of value aspects for the development of software intensive products," *Journal of Software: Evolution and Process*, vol. 25, no. 7, pp. 711–741, 2013.

[160] J. Horkoff, J. Lindman, I. Hammouda, and E. Knauss, "Experiences applying e^3 value modeling in a cross-company study," in *Conceptual Modeling*, J. C. Trujillo, K. C. Davis, X. Du, Z. Li, T. W. Ling, G. Li, and M. L. Lee, Eds. Cham: Springer International Publishing, 2018, pp. 610–625.

[161] O. Batsaikhan and Y.-C. Lin, "Building a shared understanding of customer value in a large-scale agile organization: A case study," 2018, master thesis, Chalmers | University of Gothenburg, Dept. of Computer Science and Engineering.

[162] A. Cockburn, *Agile software development: the cooperative game.* Pearson Education, 2006.

[163] I. Sommerville, *Software Engineering*, 10th ed. Pearson, 2015.

[164] S. Lauesen, *Software Requirements.* Pearson / Addison-Wesley, 2002.

[165] ——, *Guide to Requirements SL-07: Problem-oriented requirements v5*, 2017, the course book (Lau).

[166] S. Ambler, *Agile modeling: effective practices for extreme programming and the unified process.* John Wiley & Sons, 2002.

[167] B. Rumpe, “Agile modeling with the uml,” in *Radical Innovations of Software and Systems Engineering in the Future*, M. Wirsing, A. Knapp, and S. Balsamo, Eds. Berlin, Heidelberg: Springer, 2004, pp. 297–309.

[168] R. Wohlrab, P. Pelliccione, E. Knauss, and M. Larsson, “Boundary objects and their use in agile systems engineering,” *Journal of Software: Evolution and Process*, vol. 31, no. 5, p. e2166, 2019, e2166 smr.2166.

[169] J. Lindman, J. Horkoff, I. Hammouda, and E. Knauss, “Emerging perspectives of application programming interface strategy: A framework to respond to business concerns,” *IEEE Software*, vol. 37, no. 2, pp. 52–59, March 2020.

[170] R. Wohlrab, E. Knauss, and P. Pelliccione, “Why and how to balance alignment and diversity of requirements engineering practices in automotive,” *Journal of Systems and Software*, vol. 162, p. 110516, 2020.

[171] G. Lucassen, F. Dalpiaz, J. M. E. M. van der Werf, and S. Brinkkemper, “Forging high-quality user stories: Towards a discipline for agile requirements,” in *IEEE 23rd International Requirements Engineering Conference (RE)*, Aug 2015, pp. 126–135.

[172] A. M. Davis, *Just Enough Requirements Management: Where Software Development Meets Marketing.* Dorset House, 2005.

[173] E. Hadar and A. Hassanzadeh, “Big data analytics on cyber attack graphs for prioritizing agile security requirements,” in *IEEE 27th International Requirements Engineering Conference (RE)*, Jeju Island, South Korea, Sep. 2019, pp. 330–339.

[174] T. Sedano, P. Ralph, and C. Péraire, “The product backlog,” in *IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, Montreal, QC, Canada, May 2019, pp. 200–211.

[175] R. Kasauli, R. Wohlrab, E. Knauss, J.-P. Steghöfer, J. Horkoff, and S. Maro, “Charting coordination needs in large-scale agile organisations with boundary objects and methodological islands,” in *In International Conference on Software and System Processes (ICSSP ’20).* Seoul, Republic of Korea: ACM, New York, NY, USA, 2020.

[176] R. Kasauli, E. Knauss, B. Kanagwa, A. Nilsson, and G. Calikli, “Safety-critical systems and agile development: A mapping study,” in *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA).* IEEE, 2018, pp. 470–477.

[177] J.-P. Steghöfer, E. Knauss, J. Horkoff, and R. Wohlrab, “Challenges of scaled agile for safety-critical systems,” in *Product-Focused Software Process Improvement*, X. Franch, T. Männistö, and S. Martínez-Fernández, Eds. Cham: Springer International Publishing, 2019, pp. 350–366.

[178] M. Paasivaara, B. Behm, C. Lassenius, and M. Hallikainen, “Large-scale agile transformation at ericsson: a case study,” *Empirical Software Engineering*, vol. 23, no. 5, pp. 2550–2596, 2018.

[179] A. Scheerer, T. Hildenbrand, and T. Kude, “Coordination in large-scale agile software development: A multiteam systems perspective,” in *47th Hawaii international conference on system sciences.* Waikoloa, HI, USA: IEEE, Jan 2014, pp. 4780–4788.

[180] A. Qumer and B. Henderson-Sellers, “A framework to support the evaluation, adoption and improvement of agile methods in practice,” *Journal of Systems and Software*, vol. 81, no. 11, pp. 1899–1919, 2008.

[181] L. R. Vijayasarathy and C. W. Butler, “Choice of software development methodologies: Do organizational, project, and team characteristics matter?” *IEEE Software*, vol. 33, no. 5, pp. 86–94, Sep. 2016.

[182] E. Bjarnason and H. Sharp, “The role of distances in requirements communication: a case study,” *Requirements Engineering*, vol. 22, no. 1, pp. 1–26, Mar 2017.

[183] R. Abraham, “Enterprise architecture artifacts as boundary objects - a framework of properties,” in *Proceedings of the 21st European Conference on Information Systems (ECIS 2013).* AIS Electronic Library (AISeL): Association for Information Systems, June 2013, p. 120.

[184] A. Zaitsev, B. Tan, and U. Gal, “Collaboration amidst volatility: The evolving nature of boundary objects in agile software development,” in *Proceedings of the 24th European Conference on Information Systems.* Istanbul, Turkey: AIS, 2016.

[185] J. K. Blomkvist, J. Persson, and J. Åberg, “Communication through boundary objects in distributed agile teams,” in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI’15).* New York, NY, USA: ACM, 2015, pp. 1875–1884.

[186] R. K. Yin, *Case Study Research: Design and Methods (Applied Social Research Methods)*, 4th ed. Thousand Oaks: Sage Publications, 2008.

[187] P. Liamputtong, “Qualitative data analysis: conceptual and practical considerations,” *Health Promotion Journal of Australia*, vol. 20, no. 2, pp. 133–139, 2009.

[188] R. E. de Souza Santos, F. Q. B. da Silva, and C. V. C. de Magalhaes, “Member checking in software engineering research: lessons learned from an industrial case study,” in *ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM).* Toronto, ON, Canada: IEEE, 2017, pp. 187–192.

[189] D. Yang, D. Wu, S. Koolmanojwong, A. W. Brown, and B. W. Boehm, “Wikiwinwin: A wiki based system for collaborative requirements negotiation,” in *Proceedings of the 41st Annual Hawaii International Conference on System Sciences.* Waikoloa, HI, USA: IEEE, Jan 2008, pp. 24–24.

[190] R. Jain, L. Cao, K. Mohan, and B. Ramesh, “Situated boundary spanning: An empirical investigation of requirements engineering practices in product family development,” *ACM Transactions on Management Information Systems*, vol. 5, no. 3, pp. 1–29, dec 2014.

[191] M. Hertzum, "Small-scale classification schemes: A field study of requirements engineering," *Computer Supported Cooperative Work (CSCW)*, vol. 13, no. 1, pp. 35–61, 2004.

[192] G. C. Bowker and S. L. Star, *Sorting things out: Classification and its consequences*. Cambridge, Mass.: MIT Press, 1999.

[193] M. Barrett and E. Oborn, "Boundary object use in cross-cultural software development teams," *Human Relations*, vol. 63, no. 8, pp. 1199–1221, 2010.

[194] M. Kalenda, P. Hyna, and B. Rossi, "Scaling agile in large organizations: Practices, challenges, and success factors," *Journal of Software: Evolution and Process*, vol. 30, no. 10, p. e1954, 2018.

[195] R. Kasauli, E. Knauss, J. Nakatumba-Nabende, and B. Kanagwa, "Agile islands in a waterfall environment: Challenges and strategies in automotive," in *Proceedings of the Evaluation and Assessment in Software Engineering*, ser. EASE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 31–40.

[196] D. Fucci, C. Palomares, X. Franch, D. Costal, M. Raatikainen, M. Stettinger, Z. Kurtanovic, T. Kojo, L. Koenig, A. Falkner *et al.*, "Needs and challenges for a platform to support large-scale requirements engineering: A multiple-case study," in *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 2018, pp. 1–10.

[197] M. G. Gebremichael, "Requirements engineering for large-scale agile system development: A tooling perspective," 2019.