CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Incremental requirements engineering in a large-scale agile and safety-critical context: a case study

Master's thesis in Software Engineering and Technology

FELIX EHRNBERG, GUSTAV BLIDE

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2018

Master's thesis 2018

Incremental requirements engineering in a large-scale agile and safety-critical context: a case study

FELIX EHRNBERG
GUSTAV BLIDE

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2018

FELIX EHRNBERG, GUSTAV BLIDE

Supervisor: Jennifer Horkoff, Computer Science and Engineering
Advisor: Axel Franke, Robert Bosch AB
Examiner: Jan-Philipp Steghöfer, Computer Science and Engineering

Master's Thesis 2018
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2018

Incremental requirements engineering in a large-scale agile and safety-critical context: a case study
FELIX EHRNBERG, GUSTAV BLIDE
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Many software companies are turning to agile development, a philosophy that values "customer collaboration over comprehensive documentation". However, how can an organisation carry out an agile transition from a former waterfall environment, in spite of rigorous safety regulations and a non-agile business culture? In this case study, we examine a transition from waterfall to agile development at a business unit at Robert Bosch AB. Through observations, interviews and workshops we analyse their reasons for transitioning, as well as challenges related to the transition and their possible solutions. We conclude that it is necessary to compare the risks and costs with the expected benefits in order to take an informed decision of whether an agile transition is motivated, and provide such a reasoning for the case at Bosch.

Keywords: Agile, automotive, ISO 26262, large-scale, requirements engineering, safety-critical.

Acknowledgements

We would like to extend our appreciation to our supervisor at Chalmers University of Technology, Jennifer Horkoff, for giving us help and guidance throughout this thesis until the very end. We would also like to thank Eric Knauss for providing feedback.

We would like to extend our appreciation to Bosch for allowing us to interview their employees for data. A special thanks to our company supervisor Axel Franke for setting up interviews and the workshop and helping us in general while we were at Bosch's office in Lund.

Felix Ehrnberg and Gustav Blide, Gothenburg, May 2018

Contents

List of Figures

List of Tables

1 Introduction

When developing and maintaining software, more and more companies of today claim to be using an agile approach. Agile practises is by no means a new phenomenon to the software developing industry, but there is still a long way for many companies to implement these practises throughout the entire organisation.

Furthermore, changing the way of working is always a challenge in large organisations, and carrying out changes towards agile development has significant implications for the processes in such organisations. This is especially difficult in large organisations, due to for example difficulties in communication and knowledge management, tracing requirements, and aligning a holistic requirements model with agile practises (Kasauli et al., 2017).

The aim of this study is to examine requirements engineering related reasons for transitioning into agile, as well as challenges and solutions within large-scale agile requirements engineering in safety-critical systems.

The study was carried out as a case study at one of Bosch's offices in Lund. Recently the group developing Bosch's FOTA (Firmware Over-the-Air) solution for vehicles decided to adopt an agile way of developing, as opposed to the previous development process inspired by the waterfall model.

To discuss the reasons for the agile transition, as well as finding requirements engineering challenges and possible solutions to these challenges, several semi-structured interviews were carried out. Also, the current practices and problems in the organisation were observed.

1.1 Statement of the problem

Agile development sometimes face difficulties when carried out in a large-scale context (Kasauli et al., 2017), (Bjarnason et al., 2011). Advocates of agile methods highlight the flexibility and increased customer contact, but such an iterative approach is easier to implement seamlessly in projects and organisations on a smaller scale. Even though some previous studies indicate that the adoption of agile has improved the communication by using cross-functional teams (Bjarnason et al., 2011), Kasauli et al. (2017) have shown that this is not always true in large-scale environments, as all challenges they found in their study were related to communication and knowledge management.

Having functional safety as a critical attribute further increases the challenge of implementing agile, as development of safety critical software requires documentation and tracing by several standards (Kasauli et al., 2017), while agile prioritises “working software over comprehensive documentation” (Beck et al., 2001).

SAFe (Scaled Agile Framework) has become an established and common framework for scaling agile development (VersionOne, 2017), and the FOTA group was during this case study implementing this framework in their way of working. However, working with safety-critical systems puts the value of “working software over extensive documentation” in a different light, since standards like ISO 26262 require extensive documentation about the underlying system requirements, development processes, and traces between system artefacts. While SAFe provides an established framework for working agile in large scale, a more thorough process for maintaining traceability must be used when the project is also safety-critical (Cleland-Huang et al., 2012).

Instead of functional requirements, the SAFe Requirements Model advocates the use of “features”, that is “visible ’units’ of business intent that the customer recognizes” (Scaled Agile, Inc., 2018). In safety-critical projects like the one being examined, however, regulations dealing with traceability and documentation are often developed according to non-agile variants of requirements engineering (Varhol, 2012). For example, the ISO 26262 standard does not recognise “features” like the ones described by SAFe. These regulations usually also demand a document with a safety argument, in which it is described how the necessary safety characteristics are achieved.

The agility when managing requirements for a safety-critical project must therefore to some extent be compromised to comply with the standards and regulations. This study will focus on the problem of knowing which requirements engineering challenges will occur and what practices to implement to address these, when adapting to agile methods in a company like Bosch. It examines several challenges related to requirements engineering, which are likely to occur in this situation. Finding these challenges and ways to counteract them is difficult.

1.2 Purpose of the study

The primary goal of the study is to find and evaluate requirements engineering challenges and practices found in a large-scale agile development project where functional safety is critical. The study focuses on what implications a transition from the waterfall model to agile development will have on the requirements engineering activities for a company like Bosch, and to find and evaluate the practices used to counteract any negative implications, while still complying with the safety standard ISO 26262.

For the practitioners at Bosch the study will expose these implications, as well as evaluating potential solutions to these implications. Performed as a case study, it will give an opportunity to first-hand experience of a transition to agile requirements engineering.

On the way towards increased agile maturity, there are numerous mistakes and pitfalls to be made (Kasauli et al., 2017). This study aims to specifically address these pitfalls in the context of requirements engineering. While limited to only one software company, this study intends to draw conclusions that could be generalised and compared with similar studies.

The study also aims to find out what reasons the unit at Bosch has for transitioning into agile development in the given context, both from an "official" perspective but also related to the expectations and thoughts of those responsible for carrying out the transition. According to the findings of Gandomani et al. (2014), companies need to define clear business goals and convincing reasons for transitioning into agile, otherwise the process will be useless effort. This is a reason why discussing the reasons for the FOTA group to turn agile is important.

1.3 Research questions

- **RQ1:** *What are requirements engineering related reasons to expect for a transition from waterfall development into agile in a project like FOTA?*

  As mentioned in the previous section, it is vital for companies aiming to switch to agile development to define the reasons for doing so. This research question establishes a goal of the study: to help finding and defining the reasons for the agile transition.

- **RQ2:** *Focusing on incremental requirements engineering, which challenges need to be taken into account on the way to a successful transition into agile in a project like FOTA?*

  In the literature, multiple challenges related to large-scale agile transformations have been presented (Kasauli et al., 2017), as well as challenges related to complying with safety regulations while being agile. Which of them can be observed in the requirements engineering work in FOTA? Are there problems to be observed that have not been found elsewhere?

- **RQ3:** *What are solutions to the challenges mentioned in RQ2 seen by the practitioners in FOTA? To what extent are they expected to be useful in solving challenges related to requirements engineering?*

FOTA has already implemented a few agile practices, but they still need to comply with the regulations regarding the traceablity and documentation of requirements. The aim is to find and evaluate the solutions that FOTA have planned to implement, as well as capturing opinions from the practitioners regarding these solutions.

1.4 Significance of the study

Multiple important challenges linked to requirements engineering in large-scale agile organisations have been found (Kasauli et al., 2017). As mentioned, the study attempts to evaluate these challenges along with currently used practices.

Several studies call for more research on the role of requirements engineering in agile development. In a systematic literature review, Dikert et al. (2016) mention a lack of case studies about agile development in a large scale. Especially, there is a lack of studies that focus directly on agile transformation processes, to increase the knowledge of how they are done in practice.

This study will widen the research about requirements engineering in a large-scale agile context to include one more software company. It will contribute with finding requirements engineering related reasons for transitioning into agile, as well as challenges and solutions within large-scale agile requirements engineering in safety-critical systems, to the extent that the specific case can be generalised to a majority of such situations.

2 Background

This section begins with a review of the literature where the most important concepts and their interactions are presented. After that a collection of similar case studies are shown. These case studies discuss challenges, success factors and recommendations with working agile in a large-scale and safety critical context. Finally the case on which this study is based will be presented.

2.1 Review of the literature

In this section several concepts essential for this study will be presented. It will begin with requirements engineering, which is a discipline within the software engineering community, including the specification of software systems and the eliciting, validating, verifying and tracing of stakeholder requirements. This is followed by three different software development processes: Waterfall development, which is the process being used before the transition; The V-model, which has been and will only be used as a way to structure software artefacts in FOTA; Agile development, which is the process which FOTA is transitioning into. Further background on the combination of these concepts with each other and with other concepts, such as large-scale development and safety-critical systems, will be presented.

2.1.1 Requirements engineering

Requirements engineering is a discipline within the software engineering community, which includes the specification of software systems and the eliciting, validating, verifying and tracing of stakeholder requirements (Lauesen, 2002). In the traditional waterfall model, as well as the V-model, requirements engineering is pictured as the very first step in the development process.

Nuseibeh and Easterbrook (2000) describes the core requirements engineering activities as eliciting requirements, modelling and analysing requirements, communicating requirements, agreeing requirements, and evolving requirements.

A software requirement is a description of a condition or capability of the system that is needed by a user, or needed to satisfy a contract or a specification (IEEE Xplore, 1990). The written list of software requirements is commonly referred to as the Software Requirement Specification (SRS). Its purpose is to describe the externally observable behaviour that is expected of the system (Davis et al., 1993).

Understanding requirements is an important factor in order to successfully develop software systems (Lawrence Pfleeger and Atlee, 2009), but deficient requirements pose a threat to software projects in a majority of all enterprises (Hofmann and Lehner, 2001). There are different factors in the requirements engineering process that contribute to the success of a project, for example the amount of resources spent on the requirements engineering process and the amount of domain knowledge in the project team (Hofmann and Lehner, 2001).

2.1.2 Waterfall development

The waterfall model describes software development as a linear or sequential series of phases. These phases commonly described are, in order: requirements analysis, design, implementation, integration and system testing, and maintenance. (Cusumano and Smith, 1995)

The process of integration and system testing often requires rework to correct problems due to mistakes, miscommunication and changes to the design due to changing requirements. The waterfall process is suited to projects where changes to design details can be controlled and unplanned rework can be incorporated with little interference from customers or competitors. The waterfall model is not suited for projects with many uncertainties where changes to design is inevitable or desired. (Cusumano and Smith, 1995)

2.1.3 The V-model

The V-model describes the process of developing a system, from the collection of requirements to the final user acceptance testing. The left side of the V refers to the development of requirements, system architecture and more detailed design. The right side refers to integration and testing. The bottom between these sides represents the coding phase (Mathur and Malik, 2010).

The second half of the development process is thus lined up against the first half, linking the development phases from the left side to the corresponding testing phases on the right side (Mathur and Malik, 2010).

In this original description of the V-model, the requirement engineering happens in the very first part of the development process. However, it is relevant through later parts of testing and integration, when the system is tested for compliance with the previously specified requirements (Mathur and Malik, 2010).

The V-model can be used to illustrate the dependencies between different artefacts in a software product, regardless of whether developed according to waterfall or agile processes. Its purpose is to provide an illustration of the connection between certain parts of the process, for example between user requirements and acceptance tests.

2.1.4 Agile development

Agile development is a term that refers to a framework of methods and principles for software development (Beck et al., 2001). Many of these methods were developed and described before the term "agile" was popularised in this context, for example Scrum (Takeuchi and Nonaka, 1986) and Extreme Programming (Beck, 2000).

The agile values described in the original Agile Manifesto are:

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

There is no full consensus in the software engineering community about which agile practises and values are desirable in which situations, to which extent they should be implemented, or even if all the agile principles are really new (Meyer, 2014).

Ge et al. (2010) describe agile processes as a family of development processes, common agile development methods being eXtreme Programming (XP), Feature Driven Development (FDD), and Scrum. It is possible to produce a generic software development life-cycle model for agile methods, based on the commonality between these methods and the tasks and activities in each phase of them. The generic model consists of these four phases: preparation, planning, short iterations to release, and integration. (Ge et al., 2010)

According to the yearly State of Agile report made by VersionOne (2017), the most popular reasons for working agile are to accelerate software delivery, manage changing priorities, and increase productivity.

2.1.5 The agile requirements engineering process

Requirements engineering, whether in agile or non-agile development processes, puts focus on customer involvement. However, while the waterfall model keeps the most of the customer interaction to the beginning of the development, agile requirements engineering keeps the customer involved throughout the whole process (Paetsch et al., 2003).

Advocates of agile development processes have toned down the need for immutable requirements specifications (Aurum and Wohlin, 2005). The consensus in the agile community is that while requirements specifications can be useful, the resources spent on managing them should be limited to an absolute minimum. This is because they will most likely change anyway, and any form of documentation that does not directly deliver value to the customer is considered waste. Even requirements that the customer explicitly specifies may be considered “waste”, if they do not provide a benefit to the business of the customer. (Aurum and Wohlin, 2005)

2.1.6 Agile development in a large scale

Large-scale agile can be categorised as 2 - 9 collaborating teams (Dingsøyr et al., 2014). Based on a number of studies discussing large-scale agile software development and their interpretations, Dikert et al. (2016) defined large scale to denote software development organisations with 50 or more people or at least six teams.

The most prominent challenges for large-scale agile transformations is agile methods being difficult to implement, non-development function being difficult to integrate, change resistance, and requirements engineering challenges (Dikert et al., 2016).

Developing in a large scale often increases the level of complexity. Meyer (2014) states that this is something that often makes the agile notion of iterative development, which is to deliver customer value every sprint, problematic. According to him, this approach works only in systems where the complexity is additional, that is, the logic of every additional feature can just be added without too much refactoring and rebuilding. The alternative is multiplicative complexity, something that is much more common in large-scale software systems.

Meyer (2014) also calls the belief that one can incrementally program features in more complex systems one of the principal limitations of the agile approach.

2.1.6.1 Scaled Agile Framework

The Scaled Agile Framework (SAFe) has become one of the most commonly used frameworks for working with agile software development in large organisations VersionOne (2017). SAFe groups agile teams together in a so-called Agile Release

Train (ART) consisting of 50-125 people of various competences and roles. The teams in this cross-functional group are supposed to deliver working solutions every few weeks. The schedule and velocity of the team is fixed for any given program increment. (Scaled Agile, Inc., 2018)

Program increments and iterations are both timeboxes during which incremental value is delivered. During the program increments, typically 8-12 weeks long, the ARTs build and validate a full system increment and demonstrate value. The timeline in program increments is split into iterations with a recommended duration of 2 weeks. Iterations are to agile teams what program increments are to ARTs (Scaled Agile, Inc., 2018).

The SAFe framework provides the practitioner with an “implementation roadmap”, which is a visual model and a set of written articles with instructions on how to implement the framework Scaled Agile, Inc. (2018).

In the context of requirements engineering, SAFe differs from traditional approaches like the waterfall model and the V-model. While a waterfall software requirements specification is often mainly filled with functional requirements, nonfunctional requirements, and use cases, SAFe:s “Requirements Model” describes the system with epics, capabilities, features, stories, and nonfunctional requirements (Scaled Agile, Inc., 2018).

As described in SAFe, a feature is a service that fulfils a stakeholder need. In the configuration of SAFe used in the FOTA project, features will be the highest tier of requirements and they are broken down into stories. Features are requirements on ART level and are sized to fit within one program increment. Stories are requirements written in the user’s language and describe a small piece of functionality. Stories are requirements on team level and are sized to fit within one iteration (Scaled Agile, Inc., 2018).

2.1.7 Safety-critical systems

Safety-critical systems are described by Ge et al. (2010) as those where any failure is likely to result in the loss of human life or the damage to the environment. To create safety-critical systems, heavyweight processes such as the V-model have traditionally been used, characterised by emphasis on up-front design (Ge et al., 2010).

The standards and guidelines that exist for the development of safety-critical systems have at least two main uses: to provide the acquirer of a system proof that the system will to the greatest extent be free of loss inducing defects, and to help developers to create products that perform according to the expectations of the stakeholders (Ge et al., 2010).

2.1.7.1 ISO 26262

Also titled “Road vehicles – Functional safety”, ISO 26262 was defined by International Organisation for Standardisation (ISO) in 2011 and is an international standard for functional safety of electrical and/or electronic systems in production automobiles. ISO 26262 covers the entire automotive product development process and provides guidance on automotive safety life cycle activities.

Regarding requirements engineering, for example, ISO 26262 states in part 8, paragraph 6.4.3.2 (International Organization for Standardization, 2011):

> **6.4.3.2** Safety requirements shall be traceable with a reference being made to:
>
> a) each source of a safety requirement at the upper hierarchical level,
>
> b) each derived safety requirement at a lower hierarchical level, or to its realisation in the design, and
>
> c) the specification of verification in accordance with 9.4.2

The development process shall be adopted and all the work-products shall be provided when developing an item in compliance with ISO 26262 (International Organization for Standardization, 2011). A safety case supposed to progressively compile the work products that are generated during the processes should be provided. Concerning documentation management the documentation process shall be planned in order to make documentation available. The documents shall be:

a) precise and concise,

b) structured in a clear manner,

c) easy to understand by the intended users, and

d) maintainable

The safety case shall be organised to facilitate the search for relevant information. Flexibility is allowed if properly introduced, i.e., via the application of tailoring rules (International Organization for Standardization, 2011). ISO 26262 requires a safety case to be created and is described by Stålhane and Myklebust (2016) as:

> “Safety case is an efficient method for helping the developing company to focus on the simple but important question ‘How do you know that your system is safe enough?’ The idea of a safety case is not to provide a mathematical or statistical proof, but to argue as one would in a court of law – thus the name safety case.”

2.1.8 Safety-Critical with Agile

In a comparative study of agile methods vs. plan-driven methods in developing software by Boehm and Turner (2004) it was suggested that agile methods might not be well suited for safety-critical systems. These systems require stable and inflexible requirements.

One of the main possible incompatibilities between safety-critical software and agile development methods is the need for thorough documentation and tracing by several standards, while agile prioritises “working software over comprehensive documentation” (Beck et al., 2001).

Ge et al. (2010) describes a difficulty with developing safety-critical systems in an agile way is how to develop both a software system and a safety case iteratively. He describes a way to allow for compatibility between agile processes and safety-critical software development, where it is sufficient to make a hazard analysis in the up-front design process, along with a modular safety argument that is aligned with the functional development (Ge et al., 2010).

In contradiction to the earlier stated incompatibilities between safety-critical software and agile Stålhane and Myklebust (2016) states that “it is much more efficient to build the safety case by inserting information when it becomes available during project development – an agile approach also resulting in increased safety awareness and understanding”. Developing companies believe that they need complete knowledge about the system to write a safety case. Because of this, all too often developing companies leave the creation of a safety case to the end of the project which has turned out to be a costly solution (Stålhane and Myklebust, 2016).

2.1.8.1 SafeScrum

SafeScrum is an adaptation of the Scrum framework developed in collaboration with the Norwegian software industry. According to SINTEF / NTNU (2018), it is motivated by the need for a possibility to use methods that are flexible with respect to planning, documentation and specification, while still being acceptable to the IEC 61508 standard. The standard ISO 26262 earlier described is an adaptation of the Functional Safety standard IEC 61508.

In a case study conducted at Autronica Fire & Security (Hanssen et al., 2016), the authors examined how SafeScrum could be used in the organisation. Autronica deliver safety-critical software, which means they need to follow the regulations of IEC 61508.

A conclusion from the case study was that even though an agile methodology was used, the need for complying with safety standards still required some work dedicated to planning and quality assurance.

Not only was there a need for a robust requirement management tool since the standard demand traceability down to the level of individual requirements, test cases, and code pieces, but also for manual verification work by the quality assessor.

This need for extra attention and documentation could be seen as an opposition to the agile philosophy. The authors, however, argue that the tool chain used and the iterative nature of Scrum help with assession quality without too much extra work.

2.1.9 Tracing and documenting

Requirements traceability is defined by Gotel and Finkelstein (1994) as the ability to describe and follow the life of a requirement, in both forward and backward direction, ideally through the whole system life cycle.

Center of excellence for software traceability (2018) defines software traceability as an extension of the above definition "to encompass multi-directional traceability centred on diverse artefacts. Traceability supports safety analysis, change impact analysis, project intelligence and much more across evolving, adapting software systems"

There are several commercial tools for supporting traceability, but most of them were made with traditional development processes in mind (Espinoza and Garbajosa, 2011). Three elements that should be user-definable in traceability models to allow for traceability while working agile, are traceability links types, roles and linkage-rules (Espinoza and Garbajosa, 2011). However, most definitions of those elements in the literature describe them as predefined, which leads to complex models that do not fit real needs (Espinoza and Garbajosa, 2011).

The reason described for why these elements have to be user-definable is because artefacts are not represented the same way in waterfall development as in some agile methodologies. For example, in TDD (test-driven development) and SDD (software design and development), requirements are formulated as test cases, which makes them closely and bidirectionally connected with the tests. Those links need to be modelled in such a way that test cases are stated both as requirements and tests for the code. (Espinoza and Garbajosa, 2011)

The team at Bosch are not going to use TDD or SDD, and they still want to represent their development artefacts in the traditional way. This means that commercial tools tailored towards traditional development processes might still work well.

2.1.10 Merging agile with waterfall

Agile development promises great business benefits, but there is doubt whether organisations are really implementing this philosophy in full (West et al., 2011). A majority of all enterprises using "agile" in fact diverge from their methodology from time to time, sometimes because they feel that the agile methodology slows down their work, sometimes because they feel that it simply does not help them. Many corporate governance rules still require plans and requirements to be established before the development work can start. (West et al., 2011)

In the case with Bosch, many common difficulties with implementing agile is combined with safety standards demanding traceability between system artefacts. It is likely that the development practises at Bosch will have to conform to some combination between agile and V-model in the future, rather than implementing a 100% agile methodology.

According to West et al. (2011), one of the most important practises in a agile-waterfall hybrid framework is to embrace the agile value of limiting the resources spent on upfront requirements specification. The primary role of documents should be to enable high-level planning and a description of the problem area. This fits well with the FOTA group at Bosch, because one of the above stated reasons for them to switch to agile is that they have to work with uncertain and changing requirements.

Parts of Scrum are often used when an organisation uses a hybrid framework with influences from both agile and older development methods (West et al., 2011). West et al. (2011) list some important principles to follow when using Scrum in this hybrid context, for example to build cross-functional teams, include testing within the sprint, continue to engage with the business after the requirements phase, and accept changes and deviations from the initial requirement specification. These principles will be desirable for the project group to strive for during the agile transition.

2.1.11 Agile contracting

A common challenge faced by agile practitioners is that many customers demand a fixed contract with fixed time, cost and scope. The result is a contradiction between the agile value of “responding to change” and the customer need for certainty.

To better understand the challenges and benefits of agile contracting, some advice from two studies regarding this topic is presented below.

2.1.11.1 Case study at Info Tech

A case study by Franklin (2008) carried out at Info Tech, Inc. shows this contradiction and how an implementation of contracts between agile and non-agile units in an organisation can be done. Info Tech, Inc. is a company focusing on statistical and economic consulting services and at the time of the study, one of their customers was the Attorney General of Florida.

The non-agile side would only accept agreements stating that full functionality should be implemented and delivered before any payment was received.

On the agile side, many sprints were used to simply get things started. One conclusion drawn from this study is the importance that the customer recognises the value of every sprint in itself, not only completely implemented features. The best way to realise this is thorough documentation and sufficient communication.

An ideal way to develop agile software is with a T&M (Time & Materials)-model that allows flexibility in the scope, but this does not fit every customer (Franklin, 2008). Another lesson is the importance of small milestones between the bigger deliverables.

2.1.11.2 Study on organisations in India

Another study on seven organisations in India, performed by Hoda et al. (2009) gives valuable advice on how to deal with contract negotiation when trying to work agile in a less agile environment. The advice is as follows:

Change the mindset of the customer: To convince customers to start using agile contracts instead of fixed ones, they need to understand the value of only paying for the important deliverables. Most features are never used anyway and agile contracts prioritise the important ones.

Present alternatives: One possible approach is to offer a limited number of iterations, promise to show working software every so often and letting the customer opt out whenever they like. This limits their risk as they can only lose one sprint.

Hide the agile way of working: Listed as a last resort in the study. If the differences between the agile methods and what the customer has in mind are too big, there is a risk of losing the deal. In this case, it can be better to hide the agile way of working for the customer, or at least to a great extent conform the deliveries for them.

2.1.12 Working agile in non-agile organisations

In a study by Boehm and Turner (2005), based on a series of workshops, the researchers have drawn conclusions about challenges in large-scale agile development in traditional organisations and possible solutions to these.

One challenge emerges when agile and non-agile teams are developing software for the same product. In this situation, the artefacts might look completely different and be difficult to put together. There might be a need to define interfaces between agile and non-agile business units.

In this situation, there might also be a problem with product life cycles of different length. Agile teams want to deliver in short iterations while non-agile teams work more long-term.

The requirements are often written in different ways. Agile requirements are more functional and, to a reasonable extent, less formal.

Several possible solutions to these challenges were found. For example, it is important to plan in advance which parts of the system that will be developed in an agile way. The most appropriate ones are small, GUI-intensive applications. One should adjust the architecture and look out for these parts.

2.1.13 Water-scrum-fall

In a literature study, Theocharis et al. (2015) tried to find out how organisations actually implement agile methods and how they manage these transformations.

Some notable conclusions arise from this study: it is vital that project managers are given stability to perform their tasks, plan and control. They also have to fit this together with external organisational units, like HR and sales. Because agile methods mostly deal with systems and development, agile interfaces towards the other parts of the organisation are often lacking.

2.2 Case studies in similar contexts

We will here provide the reader with a couple of studies that investigate commonly occurring challenges and solutions in environments similar to ours.

These studies examine large-scale organisations, where agile development is affecting the whole organisation or at least a business unit within it. They provide valuable experience about dealing with some of the obstacles expected to be seen in an organisation like Bosch.

2.2.1 Hospital Case Study

This case study performed by Vaidya (2014) looked at Cambia, a non-profit health care company headquartered in the United States. The study analyses a transition from traditional waterfall practices to agile in a large-scale environment. Because the IT department provides software and integration services to the rest of the company, which in turn offers health care solutions, the case can also be seen as safety-critical.

Some of the most commonly recognised frameworks for scaling agile are LeSS, SAFe and DAD. Cambia, however, do not use any of these as they have developed their own framework which better suits their needs, with some of the practices mentioned below:

- **Synchronised enterprise sprint schedule**: The agile teams at Cambia were formed around components rather than features. This created dependencies between different teams, something that was addressed through creating a synchronised sprint schedule, with the same sprint length across the enterprise. All the sprint planning meetings were held on the same days, enabling the product owners to coordiate dependencies.
- **Scrum of scrum meetings**: This practice can also be found in the SAFe framework.
- **Hardening sprints**: It was not possible to deliver working software every single sprint. At Cambia, some sprints right before the quarterly releases were used for testing and planning and there called "hardening sprints".
- **Planning meetings**: These meetings were shared across multiple teams and projects, and were held each quarter of a year.
- **Communities of practice**: These teams consisted of members who dedicated 10-20% of their working time to organising activities with for example knowledge sharing, research and training.

The IT department of Cambia seem to have experienced several benefits of introducing their agile-scaling framework. For example, most teams have reported a high commit versus accepted ratio for their user stories. Also, most projects manage to keep their defect backlogs to under double digits, a significant improvement from before the agile transition.

One challenge is that it is difficult to have a full-time product owner on every agile team, since those people need to fulfil other management responsibilities. Since the work and budget planning is still organised in a traditional way on the enterprise level, the capacities are unevenly spread across the different teams.

The teams are, as mentioned above, organised around components rather than features. This sometimes leads to a lack of synchronisation and coordination. This is to some extent solved by dedicating members to teams, but this team organisation approach goes against what is advocated by both SAFe and LeSS.

Most initial resources related to the agile transition were spent on the forming of agile teams, but the adaptation of agile practices such as continuous integration and continuous delivery has been slower than desirable.

2.2.2 Gap Inc. Direct agile transformation

Gap Inc. Direct had traditionally followed a waterfall delivery methodology. After 7 years of traditional IT, they decided to adopt agile (Goodman and Elbaz, 2008). The need for changing way of working was seen when projects slipped release dates and quality standard had to be lowered for the final deliveries.

In the report they mention three success factors. The first one was “Ambitious Pilot Project”, where they chose a large project to have more visibility even though it meant a more significant risk. Also, because of the size of the project they had to make significant investments into their IT processes and infrastructure, which was positive.

The second factor they mention is “Massive Investment in Continuous Integration”. They moved all development teams to one code branch and made investments in the automation of the build process and the streamlining of the delivery pipeline.

The third factor was “Rethinking our Assets”. Before, they had code as their main focus but afterwards changed to a holistic view where tests would be given higher value.

There is a key difference worth noting between the case described in the study of Gap Inc.Direct and the FOTA project and that is the safety regulations. Gap Inc. Direct did not have any safety regulations to comply with.

2.2.3 ISO 26262-compliant and the agile documentation management

In a study by Gallina and Nyberg (2015) at Scania, findings and solutions are reported on the challenge of combining agile and ISO 26262-compliant documentation management styles.

The solutions are categorised as soft and hard pieces of solution. Their soft piece of solution is aimed at triggering the employee’s sense of responsibility.

During the study ISO 26262 was under revision and a consensus concerning its interpretation was still to be achieved. They argue that these circumstances were favourable and may have constituted an opportunity to align employees towards a common strategy. Such a strategy would be to achieve a company-wide interpretation of ISO 26262, jointly with safety assessors, in order to comply with the standard in an agile manner.

Next soft advice was to train team leaders with respect to ISO 26262 and that teams, working at different abstraction levels, should take the responsibility to choose appropriate ways to comply with the standard according to their current way of working.

The last soft advice found in the paper is to provide a number of work products just enough to maximise agility. Gallina and Nyberg (2015) states that ISO 26262-compliant tailoring should be applied under the supervision of safety assessors in order to identify allowed minima.

The first hard advice is to make the tool-chain interoperable using the OSLC standard, a standard for creating specifications for integrating tools. These tools should support all phases of the product life cycle.

The second hard advice is to use a tool for safety case generation. Following thoroughly specified generative rules the tool should from an input of interrelated work products generate a safety case representation.

2.2.4 Requirements Engineering Challenges in Large-Scale Agile System Development

In a paper presenting the results of multiple case studies, Kasauli et al. (2017) discuss "the challenges that arise from the interplay of requirements engineering and agile methods in large-scale system development". Out of the four case companies in the study, there are two car manufacturers which we consider to be under similar safety regulations as FOTA. The challenges found are divided into two areas of challenges: Shared Understanding of Value (1) and Build and Maintain System Understanding (2).

Under the area of challenges that is Shared Understanding of Value (1), three challenges are described.

1a) Customer Value to Team

This challenge regards the organisational levels between the customer and the developer teams. Each transaction of information leads to a loss of content. It was found that all four companies had dedicated roles to channel information from stakeholders to the developer teams. The agile methods used implied breaking down large features into sub-features to be finished during an iteration. It was found that during these breakdowns of features it was difficult to retain customer value in the sub-features.

1b) How to Write Meaningful User Stories

It was found that user stories had two purposes: to describe why and what the end customer wants from the system, but also work descriptions for the team. Even though user stories are useful they may still be troublesome. They are hard to write and breaking down features too much could deteriorate requirements quality if not maintained properly. When broken down, they can be hard to trace to requirements.

1c) Feedback and Requirements Clarification

If feedback is slow there is a risk that the developer teams are already working on other parts of the product and do not remember what the feedback is about. Reasons for long feedback cycles found were mechanical or hardware development being slow or non-agile customers taking long time to try out features. Customers often see little value in giving quick feedback. This challenge was found to be more common if the system under development is to be integrated into a larger system.

Even if feedback cycles are short there is a need to manage the knowledge gained. Teams might not realise that they have knowledge of value for other parts of the system.

Under the area of challenges that is Build and Maintain System Understanding (2), five challenges are described.

2a) Inform and Synchronise Between Teams

During development there is often a need to exchange information between the teams, a process which is both difficult and time consuming. Whether developer teams receive already broken down requirements or receive features to break down there are issues regarding synchronisation between teams. Awareness about interaction and dependencies of the features is necessary.

2b) Creating and Maintaining Traces

User stories are derived from features and not always systematically derived from existing requirements, meaning tracing from a user story to a requirement is not always possible. In agile teams tracing user stories to requirements was considered documentation and should not be part of the agile process. Even though tracing is valuable and sometimes required, there is not enough incentive for the developers to maintain traceability. There is a need for a way to simplify the creation of traces.

2c) Bridging the Gap Between Plan-Driven and Agile Development

Parts of the development in the cases were done following a plan-driven approach, while other parts were agile. When tests were developed against the plan-driven requirements, problems would arise if these were not updated properly. As backlog items are easy to understand it is common that many stakeholders get involved when working agile, which can be seen as a positive. However, this may overexpose the function owner making it more difficult to keep requirements updated, meaning tests risk being invalid.

2d) Tests and User Stories Not Sufficient

Using tests both as tests and requirements is common within agile. This is seen as a way to reduce the documentation effort, however, there are issues.

Even if the full system behaviour can be described by user stories it is too difficult to derive a full picture. Another problem is to express a single requirement through tests only. In order to do this, a large number of tests would be necessary and it would be hard to reconstruct this requirement in later stages only from the tests.

2e) Establishing an Agile Tool Chain

It was found that traditional tools were used for requirements management while agile tools were used for development. These kinds of tools are too separated and to be able to perform requirements engineering in a more agile way more situated tools are needed. A simple issue tracking tool is not likely to be sufficient.

2.3 Case context: Bosch FOTA project

This study was carried out within a project at Bosch where the agile transition was just about to be put into practice. The FOTA group at Bosch are looking for a systematic way to find out what benefits and challenges this transition could imply to their requirements engineering practices, and the study aims to provide Bosch with guidance on how to implement requirements engineering to supplement agile.

2.3.1 FOTA project

The abbreviation FOTA stands for “Firmware Over-the-Air”, and consequently suggests the purpose of the project, to create a product capable of delivering firmware updates remotely to vehicles. The product will include software for electronic control units (ECU) inside the vehicle, a fleet management system and a cloud service responsible for delivering the updates.

The FOTA project is responsible for creating a platform from which adaptions and extensions will be created to be delivered to the customers. It is expected that because of this, requirements will be frequently changing and new requirements added.

For the FOTA project the input of requirements will come partly from inside the team as features originating from prediction of the market. Also, they will come as real customer requirements from other departments in Bosch, either in the form of features or functional/nonfunctional requirements depending on the Bosch internal agreements. The development will be done partly by the development teams in FOTA but also by non-agile teams from other departments at Bosch. For these non-agile teams there will have to be some sort of interface between these different development processes.

The study is limited to the project FOTA and the findings are not representative for the entirety of Bosch. During the time of the study, it was planned to be about 100 people in the FOTA group.

2.3.2 The development process

Before transitioning into agile development with the FOTA project, the waterfall model was used to structure the development process. However, the ISO standard that Bosch need to follow for functional safety is tailored towards the V-model development process. More specifically, the process has to be documented in the structure of the V-model.

Even though the V-model is central in the structuring of artefacts in FOTA, it was never the development process used. While developing agile, FOTA will still have to structure the artefacts according to the V-model. This is required by the safety regulations.

2.3.2.1 Agile practices to be implemented in FOTA

In the FOTA project only features and stories will be used, as epics and capabilities are too large and general. As described in SAFe, a capability is a higher-level solution behaviour that typically spans over multiple ARTs, but the FOTA project only consists of a single ART, which limits the benefit of using capabilities. Epics become unusable in this context as well, since they belong to the tier above capabilities in SAFe:s “Requirements Model”. This selective use of parts from the Requirements Model is supported by SAFe, which allows for using only what is necessary for scaling agile development in the current context (Scaled Agile, Inc., 2018).

The five most commonly used agile practices according to the State of Agile report by VersionOne (2017) are “Daily standup”, “Sprint/iteration planning”, “Retrospectives”, “Sprint/iteration review”, and “Short iterations”. Among the ten most popular agile practices, the FOTA group are using or are planning to use most of these. For a full list of their planned agile practices, see appendix A.1. Since members of FOTA have different previous experience with agile there is a possibility that the practices used within certain teams will differ slightly.

2.3.3 Interaction with customers and suppliers

Customer contact is handled by other departments of Bosch and the FOTA project receives requirements from these. Because of this, when this study writes about “customers”, it generally means other units at Bosch to which the FOTA unit delivers. These customer units deliver, in turn, to other companies in the automotive market.

With “suppliers” it is meant other Bosch units that deliver software to the FOTA unit. Parts of the development for FOTA will be developed by teams that are not part of FOTA. These teams have their own development process, some being agile while others not. Some of these agile teams have been shaped to be agile internally and non-agile externally. This means that both some of the agile teams and the non-agile teams require fixed and complete requirements to start development, something FOTA will struggle with as they will work with changing requirements.

2.3.4 ALM

To simplify the documentation and traceability, a company-specific configuration of IBM’s Rational Collaborative Lifecycle Management (CLM) (IBM, 2018) suite of tools will be used, called Application Lifecycle Management (ALM). ALM is usually an umbrella term that covers several different disciplines but in FOTA ALM is the name of the tool suite. CLM is a suite of integrated tools providing requirements management, quality management, change and configuration management and project planning and tracking. To do this CLM combines IBM Rational Team Concert, IBM Rational DOORS Next Generation and IBM Rational Quality Manager. ALM will be configured to fit the agile development process at FOTA. To further increase requirements and components reuse, concepts and terminology introduced in version 6.0 of the IBM Jazz applications will be applied, including baselines, steams, configurations, global configurations, and components.

ALM will provide the possibility for other departments to access and deliver directly into ALM. However, it is expected that not every department will do so, if any at all.

3

Method

3.1 Research method

The study was conducted as a case study. A case study methodology provides a deeper understanding of the phenomena under study (Runeson et al., 2012). According to Yin (2003), a case study is "an empirical enquiry that investigates a contemporary phenomenon within its real-life context" which relates back to the purpose of this study. Here, the phenomenon being studied is the transition from waterfall to agile, which was observed during its implementation.

3.2 Data collection

Data in this study was collected qualitatively by observations, semi-structured interviews, and a workshop.

3.2.1 Data triangulation

Data triangulation is vital when trying to draw conclusions from qualitative data, as it is less precise than quantitative data. This study uses several forms of data triangulation, listed by Runeson et al. (2012):

- **Source triangulation** is using multiple data sources to collect the same data, for example when different interviewees are asked about what challenges the project group have encountered.
- **Observer triangulation** is using more than one observer in the study. In this case, there are two observers.
- **Methodological triangulation** is using different types of data collection methods, in this case observations, interviews and workshops.

3.2.2 Observations

During the first two months, data was collected by observations, followed by semi-structured interviews. Observations were used because such a data collection methodology provides a deep understanding of the phenomenon that is studied (Runeson et al., 2012). They were carried out by attending meetings and workshops, cooperating with the team in the initial requirements engineering work, and gathering knowledge by asking questions and discussing. The discussions were primarily held with the company supervisor, who is also responsible for the tool chain within the project.

At the beginning of the study observations were planned to be part of the data collection as a way to answer the research questions. However, it was early discovered that since the FOTA project was still in an early phase of the transition, it was not feasible to let the observations make up data points by themselves.

FOTA was already in the phase of the transition where the previous development method was no longer being used and the new agile method was just being planned. The observations were therefore often done during workshops, which meant that challenges and solutions were usually not observed in practice, but rather just mentioned by the people involved in the project. Because of this observations were not considered sufficient in answering the research questions. It was decided that although observations were useful for understanding the context and designing further data collection methods, interviews was a better way to get more detailed data for answering the research questions.

Because of this the main purpose of the observations was to give the researchers a better understanding of the domain of the study rather than serving as data points to be part of the result. Thus, no formal or structured method for performing observations was created.

The observations played an indirect but important role towards the results of the study. Interview questions were formed based on the observations, specifically those about challenges and solutions we expected after making observations for the first couple of weeks.

3.2.3 Interviews

A significant part of the data in this study was gained from semi-structured interviews. The reason to choose semi-structured interviews is that they allow for improvisation and exploration of the issues (Runeson et al., 2012). Semi-structured interviews have the benefit that they do not limit the scope of the answers as much as structured interviews, although being more time consuming to analyse (Runeson et al., 2012). The practitioners were interviewed about experienced and expected challenges and how effective they perceived currently practised and potential solutions. More detailed descriptions of the different rounds of interviews during the study can be found in 3.2.3.1 and 3.2.3.2. For a detailed view of the interview templates, see A.2.

The first round of interviews was carried out during the first two months. These interviews were conducted to collect data about challenges associated with the agile transition and solutions to these challenges. This data was then used to answer the research questions RQ2 and RQ3. The second round of interview was carried out to collect data about the reasons for the agile transition in order to answer RQ1. The structure of the interviews was based on the data collected from the observations already made, in order to find out which direction the further research was going to take.

The interviews were held with a group of people with different roles in the FOTA project. Runeson et al. (2012) recommends selecting interviewees based on differences due to the qualitative nature of a case study. The interviewees were therefore chosen to represent as many different roles as possible in the FOTA project at Bosch. A more detailed description of their roles and experiences with agile and requirements engineering can be found in Table 4.1.

3.2.3.1 First round of interviews

The same interview template was used for all interviews in the first round. This template can be found in appendix A.2.1. The interviews were scheduled for 30 minutes with each participant, a time interval that showed to be sufficient in most of the cases.

To begin with, the participants were asked about their role in the project, as well as their experience with agile development and requirements engineering.

The interviewees were first asked to share their thoughts about expected challenges related to the agile transition, primarily focusing on requirements engineering. No suggestions or leading questions were asked during this part of the interview, in order to give the interviewee a chance to share an unbiased answer.

Next, the interviewees were asked to give their thoughts about the challenges that could be expected according to the literature and our observations. Among these challenges were difficulty to maintain enough tracing between artefacts to comply with safety regulations, difficulty to understand how to work with SAFe, customer contact challenges, and interaction with internal departments.

During the rest of the interview, the participants were asked about solutions they believed could solve these challenges. The same pattern as with the challenges, with an open question at first and then a more structured discussion, was repeated. This pattern, with more open questions at first and more leading towards the end, was used to capture a variety of thoughts and ideas but also confirm or reject the expectations for this particular case. The solutions explicitly asked about were the tool suite ALM, SAFe, training sessions, and agile contracting.

Because the duration of the interviews was limited, only the challenges and solutions thought to be the most prominent were explicitly asked about.

3.2.3.2 Second round of interviews

As in the first interview round, the same template was used for every interviewee. This template can be found in appendix A.2.2. The time scheduled for every interview was 30 minutes.

The first couple of interview questions aimed to give an understanding about the thoughts of the participants about the reasons for the agile transition. A similar approach as in the first interview round was used, by first asking more openly if the interviewee could think of requirements engineering related reasons for the agile transition, and then about reasons expected from the literature and the observations.

The second half of the interviews was spent on asking follow-up questions from the first interview round. Some of our findings about the challenges and solutions (research questions RQ2 and RQ3) had been challenging to interpret and needed some clarification. Thanks to this part, we could be sure to give a more accurate representation of the opinions of the participants in the report.

3.3 Data analysis

As described by Runeson et al. (2012), qualitative data analysis was used. In short, this can be described as identifying generalisations in terms of patterns, sequences, relationships, and so on. In this study, the most prominent patterns were similarities in what the interviewees described as important challenges, possible solutions, and reasons for the transition.

When Runeson et al. (2012) describes the steps in the analysis, the data is in the form of interview transcripts. Due to confidentiality regulations, we were not allowed to record the interviews, but detailed notes were taken throughout every interview and read through directly afterwards.

The initial step of the data analysis in this study was to read through all the interview notes several times, also directly after the interviews, in an attempt to find common patterns.

Then, the coding phase was started by finding a set of commonly occurring themes. The codes as well as the completed coding for both interview rounds can be seen in appendix A.3 and A.4. Not all reasons and challenges can be found in the interview coding as some were found during the workshop and not coded. As briefly mentioned above, the most appropriate way to group quotes and thoughts together was considered to be by their relation to the different challenges the interviewees expected, the solutions they believed to work or simply had opinions about, and the reasons for the transition into agile. This opened up for a possibility to link challenges and solutions together in a clear way during the analysis.

This data coding was done iteratively, for the possibility to add new themes, challenges and solutions during the grouping and linking of notes and quotes. All of the coding was done and discussed together, something that is also suggested by Runeson et al. (2012).

3.4 Workshop

After analysing, coding and discussing the material from the interviews, a workshop session was held towards the end of the thesis work. The workshop lasted for two hours and took place at the same office as the observations and the interviews. The participants of the workshop were five of the six interviewees, two of them via Skype, as well as the two researchers of this study.

First we told the participants the goals of the workshop. The goals were: member checking (giving participants a chance to confirm that we had an accurate understanding of their answers), possibly gaining new insights, synchronising the thoughts about the transition, and presenting our conclusions so far.

The findings from the observations and the two interview rounds were presented. For every finding, a discussion was held about whether it was correctly understood or if anything should be changed, removed or added. Also, the mapping between solutions and challenges was discussed.

4

Results

In this section we will present the results, which come primarily from the two rounds of interviews and from the workshop at Bosch. A summary of the interviewees, interviewed during both of the interview rounds about all three research questions, can be seen in Table 4.1. How the found data was analysed to get to these results is described in Section 3.3.

4.1 Knowledge from observations

Even though the observations did not serve as a provider of data points on their own (see 3.2.2) they brought several challenges to light that the FOTA group would need to address for a successful agile transition, as well as potential solutions to some of these challenges.

From the observations came knowledge about subjects that would be valuable for further examination during the study: the tool suite ALM, the practical impact of safety regulations, the position of the FOTA project relative to other departments in Bosch, selection of interviewees, the planned use of SAFe and agile practices, and the development process used before the transition into agile.

During the first observations, carried out during a three day long workshop, the configuration of the tool suite ALM was discussed. This resulted in knowledge about the planned agile development process as well as FOTA's planned interactions with outside FOTA development teams and other internal departments.

We had the chance to participate in the creating of features, as described by SAFe, from a set of traditional requirements. These were created in an attempt by the FOTA group to start requirements engineering in an agile way following the SAFe framework. However, these features were difficult to write in the way described by SAFe, even for the experienced requirements engineer in the FOTA group. This indicated that agile practices might be difficult to adopt.

Table 4.1: Table of interviewees

ID	Role	Experience with agile	Experience with RE
I1	Project manager	Some experience working in partly agile teams	Prioritising and breaking down requirements
I2	System architect	Been Scrum Master, has worked in Scrum teams, not so fond of Scrum, experience with kanban	Inexperienced, only little knowledge about tracing, no experience writing requirements
I3	Section manager	Participated as product owner, tried out some agile measurements, not so deep knowledge	Requirements engineer for one year
I4	Test leader	Many years at other company	Not written requirements, have had interaction with requirements while writing tests
I5	Requirements engineer	Worked close to agile teams, no experience of personally working agile, have a general knowledge about agile	Requirements and system-design for 10 years
I6	Product owner for tool chain and method support	Previously lead transition from waterfall to agile, Certified SAFe Agilist	As a project leader, experience with producing requirements for hardware

The usage of the first prototype configuration of ALM was also observed. This did not give an answer of whether ALM would be successful but it still provided knowledge about the tool.

4.2 RQ1: Reasons for agile transition

There were three main reasons mentioned by the interviewees for the agile transition, seen in Table 4.2. The following sections present the meaning of each reason as described by the interviewees as well as their thoughts about these reasons.

Table 4.2: Table of reasons

Reason ID	Reason
R1	Handling changing requirements
R2	Motivating the employees
R3	Shorten development cycles

4.2.1 Handling changing requirements (R1)

One of three main reasons for the FOTA unit at Bosch to carry out the agile transition is a need for handling change in requirements. This was first seen during the observations and later confirmed in interviews.

The FOTA project deals with several different customers, some more flexible than others in their requirements. One interviewee mentioned that customers want to know the plan from the beginning to the end but at the same time want to have room for integrated change.

A common rationale was to have the ability to postpone decisions as far as possible. Every customer project would look a bit different, and the technical solutions would not be the same. One interviewee stated that since the FOTA group is in some way building a platform to be used by many different stakeholders, it was natural to build it incrementally. Most of them thought that it is rare for a fixed waterfall plan to be kept throughout the implementation process.

One interviewee wanted to make it clear that it was not only requirements from a certain customer that would change, but often also the importance of the stakeholder. Who is in the main focus could change during the project and this would obviously cause change in the requirements specification and/or the plan for its implementation.

However, the opinion that the requirements were likely to change was not completely shared by all interviewees. One of them said he was not sure that it was so common with changing requirements in the automotive industry as in other parts of the software industry.

4.2.2 Motivating the employees (R2)

The other main reason for the agile transition was that people preferred to work agile, or that agile development is better for motivating people in different ways.

One aspect of this relates to the above-mentioned desire to let decisions be made where they are most relevant. Not only does an agile way of decision-making let the teams postpone uncertain decisions until they are necessary, it also makes the decisive power more distributed. Letting the developer teams make decisions relevant to them was believed to be something that could increase their motivation and involvement.

Agile development would let the employees have more control over the development, as opposed to the stereotypical waterfall approach of taking decisions at the management level and commanding the developers to implement it. It was said that agile development would also give the developers a bigger picture of the whole project by giving them more responsibility.

Interviewees believed agile to be beneficial for the teamwork, increase creativity, and making the developers enjoy their work more. Several interviewees stressed the importance of creating a working environment similar to how the developers wanted it to be.

4.2.3 Shorten development cycles (R3)

The third reason for the agile transition, according to the participants in the study, was to shorten development cycles. There is an increasing pace on the automotive market, making a short time to market important in order to have a competitive advantage.

4.3 RQ2: Challenges associated with agile transition

There were several challenges mentioned by the interviewees regarding the transition towards agile. In Table 4.3 the challenges are presented. The following sections present the meaning of each challenge as described by the interviewees and the impressions from the interviewees about the challenge.

4.3.1 Customer agreements (C1)

The "customers" in this section are other departments at Bosch with which FOTA has a customer relationship. Also see Section 2.3.3.

Several interviewees mentioned the difficulty of compiling contracts together with customers while working agile. The general view was that most companies, especially in the automotive industry, practise a way of working and managing customers far from the agile ideals described by Beck et al. (2001). While these ideals advocate that

Table 4.3: Table of challenges

Challenge ID	Challenge
C1	Customer agreements
C2	Supplier agreements
C3	Functional safety difficult with isolated functionality
C4	Difficult to begin
C5	Keeping requirements updated and traceable
C6	Understanding the SAFe practices
C7	Resistance from management

"contract negotiation" should be kept to a minimum in favour of "working software", the automotive industry usually demands complete requirements specifications before the development can start.

It was said that a big challenge was to map the set of requirements they obtained from their customers to their own agile working processes. Requirements would have to be translated into features as used in SAFe. Sometimes one requirement would have to be split into many features or one feature would consist of many requirements or a mix of both. Finding the correct way to do this was considered a significant challenge.

Another concern regarding the customer agreements was how to estimate costs. The other units which FOTA delivers to need to know the cost of the work done in FOTA to be able to charge their customers the right amount. Previously, the scope was clear and fixed, but while working agile the requirements would continuously change. With fixed requirements specifications the FOTA group used to be able to charge the customer for later changes, but there was confusion about how this would be handled in an agile environment.

> "[The customers] want management and control, deadlines and approving deliveries. Especially since we're in the automotive industry. " –I3

> "Maybe agile is better, the customer has more opportunity to change things, for better or worse. It could also be misused." –I4

There was a concern that even though the FOTA unit would implement the deliverables in an agile way, they would be forced to deliver a bunch of agile packages because the customer is not working agile. This would significantly delay feedback.

4.3.2 Supplier agreements (C2)

The transition from waterfall to agile that this study treats is not yet a company-wide one, but still limited to a few business units around the FOTA group. This means that much of the business being done across the borders of internal business units has to be adapted to different levels of agility, since the FOTA unit will be more agile than most other units within Bosch. Also, as West et al. (2011) states, the nature of management in most larger organisations does not promote agile development and deliverance.

All the interviewees had opinions about the role of contracts with other departments delivering to FOTA. Several of them mentioned that the business environment wanted agreed upon contracts, like requirements specifications, before the suppliers could start most of their development work.

There were slightly different opinions of whether SAFe would be able to help with addressing this challenge, but the general view still was that SAFe helps in integrating agile and non-agile teams. With the help of program increments, they can come together in the end.

> “The complications come in interactions with other business units, parts of the company that have been working non-agile for a long time. If a team gets a completed specification to implement, we can’t take it bit by bit.” –I2

> “We have suppliers too, some of the teams are in other units at Bosch. There are units saying they won’t start working until they have an entire requirements specification and system description, both architecture and requirements.” –I6

4.3.3 Functional safety difficult with isolated functionality (C3)

Mentioned as a challenge was that specifying isolated functionality would be troublesome as some functionality would be cross-cutting over the whole system. Functionality would need to be specified in enough detail according to regulations, but to do so knowledge about the surrounding system was necessary. Because agile methods focus on small iterations, meaning smaller parts are specified at a time, and less documentation than a waterfall process there is an increased risk that details about the surrounding system would not be available.

4.3.4 Difficult to begin (C4)

A common concern among the interviewees was that it was hard to start working with agile methods before having something up and running. A well-known characteristic of agile methods, especially Scrum, is that business value should be delivered every sprint. This could be complicated in practice, since most features in a system are dependent on several different modules that need to be implemented before the features are ready for delivery.

This challenge therefore consists of two issues related to the beginning of the agile development work: firstly, the need for a concrete approach for implementing the agile way of working; secondly, the difficulty of delivering business value to the customer from the very first iteration.

The general opinion was that developing and delivering isolated features would require a larger amount of work to prepare for the first deliverables, compared to increments later in the process.

> "Agile assumes that you make small improvements to what you already have, but we have nothing yet, so how do we get started? The first step becomes much more than you imagine at first, later it's getting more agile, but at first it will be waterfall, otherwise we'd never get anything done." –I5

We did to some extent expect that the safety regulations would increase this challenge even more, but the interviewees did not fully agree with that statement. One of them said that the difficulties in getting started with the agile transition were more related to factors within the organisation. One of these factors could be a large number of people involved in the project, situated at different locations, making the project highly distributed. Another reason was that the agile transition could not be started in time due to old projects demanding attention from those involved. It might be true that the safety regulations would affect the difficulty of getting started with the agile transition, but the interviewee offering this explanation said he had not yet seen it directly.

Another interviewee said that safety regulations do not add an large enough amount of complexity to the project for it to be a huge challenge in itself. However, there would be people saying that waterfall is the only way to work with safety-critical projects, and having to convince those people would add another thing on the to-do-list.

A third interviewee wished to separate the agile transition and the safety regulations. In his opinion agile was a method of planning while safety regulations were more related to the end product, for example by providing clear requirements for it.

There were some opinions about the SAFe implementation roadmap (see Section 2.1.6.1). Several interviewees believed that the roadmap could offer the management a tangible suggestion for the first steps of the plan, making it easier to create an implementation plan than if one were to start out with nothing at all.

At the same time, it was clear that the implementation roadmap alone would not provide enough guidance for the FOTA group to immediately launch the agile transition. Some things, important in the context, were thought to be missing, for example what tools to choose or how to deal with non-agile requirements coming from outside.

> "Greatly simplified, that roadmap. (...) Not 100% sure that it will solve all the questions when you follow it exactly by the book, you have to add stuff." –I1

4.3.5 Keeping requirements updated and traceable (C5)

As described earlier, thorough documentation and traceability was required in the FOTA project to comply with standards and regulations. However, we expected the transition into agile to increase the difficulty of maintaining this thorough documentation and traceability. This expectation was based on the fact that agile methods put lower focus on documentation while also allowing for more frequent changes of requirements.

Some interviewees agreed with the statement that working agile would increase these difficulties. One interviewee, who did not agree, said that traceability is already difficult by nature and will not be more difficult because of agile, although traceability would be handled differently, which would initially be a challenge. Three reasons as to why working agile potentially would increase the difficulty of keeping traceability were mentioned.

The first reason was that the people at Bosch had already seen an agile transition increase the above-mentioned difficulties. These experiences came from another project at Bosch that had been carried out before.

The second reason was the concern of increased difficulty of keeping tests updated and synchronised with frequently changing requirements as a result of working agile. Many of the tests were automatic and would have to be changed when the requirements changed, which would lead to tests always lagging behind relative to requirements. One interviewee mentioned that while agile development has many benefits, it is sometimes easier to receive a larger collection of requirements and write tests afterwards, as in the waterfall procedure.

A third reason was that it is easy to add traceability to a complete system, as would be done while working according to waterfall. Working with parts, as is done in agile, is more difficult.

While some interviewees did not agree with the statement that working agile makes traceability more difficult they still saw a challenge. They argued that keeping traceability is difficult both while working agile and waterfall and that the way to keep traceability is different in both cases. A new way of working with traceability would have to be implemented, which the interviewees believed to be difficult. This would mean that once up and running, the work of keeping traceability would be no more difficult while working agile than waterfall, but during the transition it would still be a challenge.

The argument that agile allows for frequent changes and would increase the difficulty of traceability was also resented by some interviewees. They argued that when working according to the waterfall model changes are also common, just not planned for, which makes traceability as difficult as it would be while working agile.

Mentioned as a reason for why traceability is difficult was that too much focus is on code. Code is only a part of the end product and developers and managers should be aware of the value of all artefacts.

4.3.6 Understanding the SAFe practices (C6)

When faced with the question whether a lack of agile understanding within the teams could be a problem, the most common answer from the interviewees was that the people involved in the FOTA project generally had enough previous experience with agile in general. Many of them had former experience from other agile environments, and one interviewee stated that they were more used to agile than to waterfall development.

The interviewee responsible for the development methods used in the FOTA project, however, had concerns about the agile understanding. It was mentioned that people actually did not work agile, even if they believed so. Also, writing user stories seemed difficult for many people.

A problem seen by the interviewees was that if not understood correctly, agile practices would not yield as good result as expected. Another problem mentioned was that a common reaction to uncertainty is to fall back to waterfall development.

It should be mentioned that most interviewees wanted to distinguish between agile methods in general and the SAFe framework they intended to implement. They believed that some general SAFe training was missing among the teams and needed to be done as part of the transition.

4.3.7 Resistance from management (C7)

A challenge that was brought forth was that of change resistance from people in high decision-making positions in the Bosch organisation. In the current context, this means a resistance against the agile transition happening across the whole company.

A factor related to this was that transitioning into agile initially would lead to more visible costs. In the beginning of an agile transition there would be a substantial cost for planning the implementation of SAFe. These events, while essential for agile to work, would lead to costs that would be questioned by managers on higher organisational levels, increasing the risk that these important events would not take place.

The interviewees believe the current way of working to be more expensive in the long run compared to agile, but with many costs hidden or inevitable. It was said to be much easier to get costs approved if they had a direct visible impact, for example, investing more in an already delayed project to try and catch up with the deadline.

4.4 RQ3: Solutions to the challenges in RQ2

There were several solutions mentioned by the interviewees to the challenges of RQ2. In Table 4.4 the solutions are presented, along with the challenges the solutions are meant to address. The following sections present the meaning of each solution as described by the interviewees and the impressions from the interviewees about the solution. The solutions sometimes lead to concerns among interviewees, worrying that they might lead to new challenges.

Table 4.4: Table of solutions

Solution ID	Solution	Challenge(s) addressed
S1	ALM and other tools	C5
S2	Agile contracting	C1, C2
S3	Agile prototype with incrementally added complexity	C4, C6
S4	SAFe training and tutorials	C6
S5	Waterfall in the beginning	C4

4.4.1 ALM and other tools (S1)

It was already clear from the observations that the tool suite ALM (see Section 2.3.4) would be used and the interviews aimed to find out whether or not the practitioners thought it would work as planned. Even though ALM is one of the largest tool suites that will be used in FOTA it was still believed that the entire tool-chain is what will be the solution.

The opinions about the usability of ALM were generally positive. Despite the difficulties with traceability, the interviewees believed that using ALM would solve the challenge of Keeping requirements updated and traceable (C5).

It was evident that the interviewees considered a tool suite like ALM or tools like the ones ALM consists of to be an absolute necessity for the FOTA project to comply with the existing standards and regulations. The goal with using ALM was that the traceability would be automated to the extent that the developers would not have to think about it while doing their regular work. Most of the interviewees explicitly stated that ALM facilitated the linking between artefacts.

However, some concerns were manifested. For example, the product owner of the tool chain mentioned that ALM offered too many options in relation to their current needs. He imagined that the teams would need to spend more time on training in the future, to become more familiar with using it.

Another concern that was raised was that although ALM would facilitate the linking of artefacts, it could be hard to know if those links have been updated according to the latest version of the test suite and the requirements.

4.4.2 Agile contracting (S2)

In this report, the solution of agile contracting refers to a set of practices including changing the mindset of the customer, presenting alternatives and hiding the agile way of working from the customer (see Section 2.1.11). The interviewees either explicitly used the term agile contracting or mentioned some of practices above. The challenges agile contracting could aim to solve were said to be Customer agreements (C1) and Supplier agreements (C2).

According to one of the interviewees, the approach towards less agile customers could be described as being careful of communicating to them in an agile way, similar to what was described in the case study by Hoda et al. (2009). Internal buffers of deliverables were created in order to fulfil the traditional agreements.

For solving challenges related to agile contracting with non-agile business units, a solution considered by some interviewees was that the other business units changed their way of working. However, they did not consider this reasonable in a close period of time. It was mentioned that the key to make other departments interested in aligning their processes towards agile would be to show the benefits of this way of working to the managers.

4.4.3 Agile prototype with incrementally added complexity (S3)

A solution suggested was to develop a prototype systems design where some dimensions of complexity is ignored, later adding these dimensions to this prototype until arriving at a full systems design. This was said to serve as a way to solve the challenges of Difficult to begin (C4) and Understanding the SAFe practices (C6).

Their way of working would be independent of their surroundings, leading to agility being achieved more easily. Using this prototype as a starting point for the real systems design, the reduced uncertainty would increase agility during the later work, as the team already would have experience with agile and already finished some of the systems design.

4.4.4 SAFe training and tutorials (S4)

Training sessions had already been observed as a solution that would be used. The project leaders were taking courses in SAFe and the training was put in place to increase the Understanding the SAFe practices (C6).

When asked about their opinions about the planned training sessions, the interviewees in general said that the development teams needed to at least have seen how agile works. Also mentioned was that product owners and scrum masters should get training to avoid pitfalls. It was mentioned that the FOTA group follow the SAFe implementation roadmap which includes training of the executives, managers and leaders.

However, the interviewee responsible for the development methods used in the FOTA project added that the implementation of SAFe had in practice been limited because of incoming customer projects. As written in Section 4.4.1 he also thought that more time should be spent on training, in this case to make the teams more familiar with SAFe.

A reason mentioned as for why training is important for everyone involved was that there will often be new findings and new best practices. Even people that have a lot of experience with agile can benefit from agile training.

Also mentioned was the importance of everyone using the same terminology. Training sessions would serve to synchronise the terminology used among the people involved, regardless of their level of experience.

One interviewee also stated the need for simple and clearly written tutorials for how the agile work should be carried out at the department.

4.4.5 Waterfall in the beginning (S5)

It was suggested that the development should follow the waterfall model for the first couple of iterations, at least for the initial requirements engineering work. The first iterations of development would have to be larger than later in the project, otherwise it would be tricky to get started. Once a product was in place, the development work could be done in small iterations. This was said to be a solution to Difficult to begin (C4).

One interviewee explained that it is not possible to work agile with the fixed set of requirements that were strongly related to safety and not negotiable.

Overall, the interviewees were of the opinion that some parts of the development work could never become fully agile in the safety-critical, automotive setting that they acted in. This did not seem like something they considered a problem, but rather something that was obvious and implicit.

> "There will come some things you can't work agile with. (...) The other parts you can do as agile as you want. But for the safety you can't change the fix-stable. Then it's really hard to have an agile comprehensive view. We are locked in by the safety parts. In order to follow the regulations, you can't iterate over new requirements." –I4

> "I think there will always be some purpose with waterfall in the beginning." –I2

5
Discussion

This discussion chapter will be structured around the research questions, starting with the reasons for FOTA to conduct the agile transition and moving forward towards the challenges and solutions related to this transition.

5.1 RQ1: Reasons for agile transition

- **RQ1:** *What are requirements engineering related reasons to expect for a transition from waterfall development into agile in a project like FOTA?*

After the observations and interviews, we had found two main reasons for the agile transition at FOTA: Handling changing requirements (R1) and Motivating the employees (R2). After having discussed these with the participants at the workshop session, we decided that another distinct reason for the transition was to Shorten development cycles (R3). After discussing the reasons for the transition we will evaluate if the transition should have taken place or not (see section 5.5).

5.1.1 Reasons compared to related work

As mentioned in section 1.2, not having clearly defined reasons for turning agile could mean that much of the transition is wasted effort. The FOTA group seem to agree on a few defined reasons for their transition.

In this section, we will compare the reasons for turning agile at FOTA with the most popular reasons for agile development as found in the annual State of Agile report (VersionOne, 2017). We will also explain the mappings in more detail. (See also section 2.1.4.)

Table 5.1: Reasons compared to existing work

Reason	Reasons in existing work
Handling changing requirements (R1)	Enhance ability to manage changing priorities (VersionOne, 2017)
Motivating the employees (R2)	Increase productivity (VersionOne, 2017) Improve team morale (VersionOne, 2017)
Shorten development cycles (R3)	Accelerate software delivery (VersionOne, 2017)

5.1.1.1 Handling changing requirements (R1)

We believe that reason R1 is very similar to the reason "Enhance ability to manage changing priorities". In the State of Agile report, it is the second most popular reason why companies use agile development (64%). In the projects of the FOTA group, priorities will change with both changing requirements and changing stakeholder focus.

5.1.1.2 Motivating the employees (R2)

This reason could be compared with "increase productivity", the third most popular reason (55%). If the employees at Bosch feel motivated and trusted to take decisions closely related to their part of the development work, this could indeed help increasing the productivity. It is also likely that they take better decisions, having more direct knowledge about their part than anyone else.

The similarity between R2 and "improve team morale" is obvious. This reason is number 9 on the list from the State of Agile report (28%). It is interesting that it is found so low on the list, since R2 seemed like a very important reason at Bosch, at least according to some of the interviewees.

5.1.1.3 Shorten development cycles (R3)

Shorten development cycles (R3) is very similar to "accelerate software delivery" (see also section 2.1.4), the most popular reason in the State of Agile report. This is a reason for 75% of the responding organisations in their study, and the number has increased over time.

The importance of having a short time to market has increased for every company in the software industry. This was discussed in the FOTA group at the workshop and it was very clear that Bosch is no exception.

5.1.2 Other common reason for agile transition not mentioned

Among the reasons for turning agile listed in the State of Agile report, we have identified clear similarities between the reasons for the FOTA group and the three most popular ones in the software industry. We will take a look at the fourth and fifth most popular reasons listed in the same report, which we did not find in the interview data, and discuss to which extent they could apply to the transition at FOTA.

The fourth most popular reason in the report is “Better business/IT alignment” (49%). Not much in our findings indicates that the FOTA group has significant challenges or problems with their alignment between business and IT, at least not anything that could benefit from turning to agile. Since so many parts of the business structure are following traditional development patterns like waterfall, the decision to turn to something else is probably not driven by a wish to align business and IT, at least not in the short perspective. It will be a different situation if the agile transition becomes successful and implemented through the whole company in the future.

The fifth most popular reason in the report is “Increased software quality” (46%). While not believing that the FOTA group would not want to increase the quality of their software if possible, we did not find that this was one of the biggest reasons for their agile transition. This could be because the FOTA group already believe that the software holds sufficient quality and would not benefit so much from agile development. It could also be that the main focus when it comes to “quality” is that the systems being developed comply with the safety regulations. Because of this, the discussion becomes more about whether it is possible to transition to agile while still complying to the regulations than whether agile development could lead to better software quality.

5.1.3 Relation of reasons to incremental requirements engineering

One of the reasons for the agile transition was to be able to **handle changing requirements (R1)**. Incremental requirements engineering, as imagined by the FOTA group after the agile transition, will fit the reality of changing customer requirements better. Working with fixed requirements specifications is insufficient for handling a varying set of customer requirements.

Another reason was that the new way of working would help with **motivating the employees (R2)**, partly by letting the employees take decisions more relevant to them instead of having the decisions pushed on them from a management level.

In a traditional requirements specification for a system, the requirements could be divided into different levels, for example goal-level requirements or design-level requirements (Lauesen, 2002). We believe that it is generally a good idea to let decisions about certain requirements be taken by the people responsible for implementing them. When developers are faced with a higher responsibility, they will need to get a better overall picture of the business goals of the system. One aspect of this is understanding the relation and linking between requirements on the lower levels and requirements on the goal level.

According to Amabile (1993), unmotivated employees are likely to expend little effort in their jobs, avoid the workplace as much as possible, exit the organisation if given the opportunity, and produce low quality work. If their motivation is increased, this would lead to better work in the organisation and better requirements engineering work as a result.

The third reason for the agile transition was to **shorten development cycles (R3)**. There are several reasons why the requirements engineering work would benefit from shortened development cycles. Shortened cycles leads to more instant feedback which increases the chance that the requirements are correctly understood, validated and implemented. This makes the product better aligned with the real customer needs.

Working with customer requirements in an incremental way facilitates their prioritisation and the decision of which of them to implement first. When considering "customer value" in an agile manner, one can make it easier to implement the most important requirements first and deliver something valuable in less time. This also brings the benefits of more frequent feedback as mentioned above.

5.2 RQ2: Challenges associated with agile transition

- **RQ2:** *Focusing on incremental requirements engineering, which challenges need to be taken into account on the way to a successful transition into agile in a project like FOTA?*

As expected, there is a multitude of challenges associated with a transition from waterfall to agile in a safety-critical context.

Several of the findings were related to agreements, contracts and interfaces to other departments and customers. Another prominent challenge is where to begin, with iterations and short sprints, when these contracts demand the foundations of the system being more or less completely built before implementing the rest.

In this section we will discuss the challenges we found and their relation to commonly found challenges in the software industry found by other studies. In parallel, we will discuss other hypothetical situations where these challenges would not exist or would not be a problem. This can hopefully lead to a broader understanding of how to avoid or mitigate some of the challenges for Bosch and other organisations in a similar situation.

5.2.1 Challenges compared to existing work

In this section, the challenges found in FOTA are compared with common challenges found in large-scale organisations transitioning to agile. The main sources when comparing challenges are from a large literature study conducted by Dikert et al. (2016), as well as a case study by Kasauli et al. (2017).

The challenges will be mapped together in Table 5.2 and a more detailed description is provided further down in this section.

Table 5.2: Challenges compared to existing work

Challenge	Challenge in existing work
Customer agreements (C1)	Feedback and Requirements Clarification (Kasauli et al., 2017)
	The development process shall provide the customer the ability to give feedback (Hohl et al., 2018)
	Challenges in adjusting to incremental delivery pace (Dikert et al., 2016)
Supplier agreements (C2)	Inform and Synchronise Between Teams (Kasauli et al., 2017)
	Tests and User Stories Not Sufficient (Kasauli et al., 2017)
	Interfacing between teams difficult (Dikert et al., 2016)
	Other functions unwilling to change (Dikert et al., 2016)

Continued on next page

Table 5.2 – continued from previous page

Challenge	Challenge in existing work
Functional safety difficult with isolated functionality (C3)	Maintaining the compliance to standards is seen as highly challenging (Hohl et al., 2018)
Difficult to begin (C4)	Lack of guidance from literature (Dikert et al., 2016)
Keeping requirements updated and traceable (C5)	Creating and Maintaining Traces (Kasauli et al., 2017)
	Maintaining the compliance to standards is seen as highly challenging (Hohl et al., 2018)
Understanding the SAFe practices (C6)	How to Write Meaningful User Stories (Kasauli et al., 2017)
	Lack of training (Dikert et al., 2016)
	Misunderstanding agile concepts (Dikert et al., 2016)
	Interpretation of agile differs between teams (Dikert et al., 2016)
	Requirement refinement challenging (Dikert et al., 2016)
	Creating and estimating user stories hard (Dikert et al., 2016)
Resistance from management (C7)	Management unwilling to change (Dikert et al., 2016)
	Management in waterfall mode (Dikert et al., 2016)

5.2.1.1 Customer agreements (C1)

Kasauli et al. (2017) lists “Feedback and requirements clarification” as a common challenge. Several aspects of this challenge have been seen at Bosch, for example that it could be caused by a need to deal with non-agile customers, creating long feedback loops, and a large number of stakeholders causing a complex variety of requirements, something also seen at Bosch. Hohl et al. (2018) recommend that the customer shall be integrated into the development process. This means that FOTA would benefit, in terms of feedback loops, from keeping a closer relationship with their customers if possible.

5.2.1.2 Supplier agreements (C2)

Even though the challenge of how to "Inform and synchronize between teams" listed by Kasauli et al. (2017) seems to deal more with teams in the same business unit, it could be applied to FOTA and other units at Bosch in the sense that it is a challenge to correctly understand dependencies and break features down to be developed by different teams. Kasauli et al. (2017) states that synchronising development and channelling the right information between teams is time-consuming and something that limits their agility.

Dikert et al. (2016) mentions the challenge "Interfacing between teams difficult". This challenge is for example caused by the notion that agile development is flexible on team level, but not always between different teams working together in a larger organisation. This resembles the situation at Bosch, where contracts between the different units are written in a waterfall manner in order to not having to deal with different levels of agility between the units, for example in terms of the nature of their development iterations.

Kasauli et al. (2017) also mentions the challenge of "Tests and user stories not sufficient". One example of this is that the agile idea of using test cases as requirements documentation does not give enough information for the development. It is difficult to get a complete overview of the system from test cases alone. We believe that this would be even more of a challenge at Bosch, where most of the suppliers are not working agile. They do not accept a collection of test cases as a full requirements documentation.

The challenge "Other functions unwilling to change" found by Dikert et al. (2016) is also highly relevant here. The long-term goal being discussed at the FOTA unit of convincing other business units in working agile is being hindered by the general unwillingness to change the way of working.

5.2.1.3 Functional safety difficult with isolated functionality (C3)

One of the common challenges listed by Hohl et al. (2018) is that "Maintaining the compliance to standards is seen as highly challenging", indicating that Bosch is not the only organisation where complying to safety regulations is a non-trivial challenge.

5.2.1.4 Difficult to begin (C4)

Dikert et al. (2016) describes how there is a “Lack of guidance from literature” when it comes to how to implement a large-scale agile solution, and the team environment in reality was considered messier than the ideal cases described in literature. A number of different approaches were also described, making it difficult to choose one of them. This was undeniably seen in the FOTA group at Bosch, where the project manager was struggling to find any good advice from literature on how to implement the SAFe solution in practise. We too had trouble finding any concrete advice in the literature that could work well in the reality at Bosch. This leads us to the conclusion that more literature with concrete advice, grounded in empirical research, would be a welcome addition to the research field.

5.2.1.5 Keeping requirements updated and traceable (C5)

Tracing directly between requirements and user stories is often not possible, as Kasauli et al. (2017) observes with the challenge of “Creating and maintaining traces”. This is seen at Bosch, where a significant amount of work with mapping requirements to user stories needs to be done. The above-mentioned observation made by Hohl et al. (2018) that “Maintaining the compliance to standards is seen as highly challenging” is relevant here as well.

5.2.1.6 Understanding the SAFe practices (C6)

An observation at Bosch was that many team members had difficulty writing user stories in a correct way. Kasauli et al. (2017) gives support to the statement that there is often significant difficulty in “How to write meaningful user stories”. One reason for this is that stories providing direct value to the user might be too large to implement within one iteration. When they are further broken down, the quality of the requirements is sometimes compromised. The challenges of “Requirement refinement” and “Creating and estimating user stories” have also been observed at Bosch.

Dikert et al. (2016) have observed a number of other examples of team members “Misunderstanding agile concepts”. While not having examined all of these in detail, after the interview and workshop sessions we were left with the impression that mistakes like these are sometimes committed in the FOTA group.

Most interviewees in the FOTA group agreed that the lack of SAFe training was a significant challenge at the moment. Lack of agile understanding usually needs to be addressed with training, a statement supported by Dikert et al. (2016) who have found that “Lack of training” is an evident problem causing difficulties in the transformation. SAFe training at Bosch will also be a significant way of synchronising the “Interpretation of agile between teams”, something that often differs according to Dikert et al. (2016).

5.2.1.7 Resistance from management (C7)

The resistance from management could be a significant obstacle for the agile transition at Bosch. According to Dikert et al. (2016), having a “Management unwilling to change” could undermine a whole transition and hinder the agile way of working from spreading beyond the development teams.

Here is also a risk of still having the “Management in waterfall mode” even after the agile transition. Dikert et al. (2016) write that they have seen cases of this in their study.

5.2.2 Will expected challenges prove to be challenges?

Having looked at the found challenges and analysed their connection to common challenges from some larger studies, we can discuss a relevant topic: whether they really should be considered challenge in the context that the FOTA group is working in. At first glance the challenges found in this study seemed reasonable. However, when they were coded, analysed and checked a second time with the interviewees, it became apparent some of them were not necessarily challenges.

This section does not mention Customer agreements (C1), Supplier agreements (C2) or Resistance from management (C7). We have no doubt that these will be actual challenges at Bosch as these challenges had already been experienced in FOTA. Therefore the decision has been taken to not discuss them here.

5.2.2.1 Functional safety difficult with isolated functionality (C3)

An example of a more uncertain challenge is Functional safety difficult with isolated functionality (C3). The challenge was brought up by one of the interviewees, who believed that specifying functionality enough to comply with the safety regulations would be a major challenge.

In section 2.1.8 the implications of developing safety-critical software with agile methods are described. Boehm and Turner (2004) state that agile methods might not be well suited for safety-critical systems, which increases the likelihood of C3 being a challenge. However, Stålhane and Myklebust (2016) states that building a safety case in an agile approach is not only well suited, but also more efficient.

A reason for why this would be a challenge is that in order to specify functionality in enough detail to comply with the safety regulations, knowledge about the surrounding systems is necessary. When developing agile there is a risk that this knowledge is not available when a certain piece of functionality is specified.

This reasoning is fair since when developing agile parts of the system will not yet be known in detail when other parts are being specified. However, it assumes that the functionality has to be fully specified from the beginning, which is false. As described in section 2.1.8, detail can be added over time as more knowledge is gained, which could mean that C3 does not have to be a challenge in the current context.

On the other hand, Ge et al. (2010) describes it as difficult to develop both a software system and a safety case iteratively. Ge et al. (2010) suggest a solution to this challenge, as briefly explained in section 2.1.8.

C3 could still be considered a challenge but through a different reasoning than when it was first found. If the safety case is built iteratively, the notion that agile hinders enough specification of functionality is not something that would make C3 a challenge. However, building a safety case this way is more challenging. This would mean that C3 still is to be considered a challenge.

5.2.2.2 Difficult to begin (C4)

This challenge, as mentioned before, relates to both the difficulty of delivering customer value for every increment, but also the somewhat broader challenge of carrying out the agile transition in the given context. The challenge in the latter situation is the need for concrete plans for how to begin with restructuring the organisation when transitioning into agile.

When it comes to delivering customer value for every increment, it seems like those responsible for the agile transition in the FOTA group have thought of a plan: use the first couple of increments for building the technical foundation, and then plan for more value-driven increments where they can deliver visible value in an agile way.

This is not a procedure that would be considered 100% agile, and the interviewees seem to be fully aware of this. Thus, when discussing whether delivering customer value from the first sprint is a challenge, one must first decide whether that is actually the goal from the beginning. That would be the case in a fully agile setting, but as previously mentioned, the FOTA group is not fully agile and is likely never going to be.

Most likely, working in the automotive industry with safety regulations makes such a purely agile procedure impossible, which is why the FOTA group imagine something more like the “Water-Scrum-Fall” approach mentioned in section 2.1.13. Even though the “Water-Scrum-Fall” approach could partially address the challenge of delivering value for every iteration, the agility will still be compromised by this approach. From an agile point of view, C4 is still a challenge.

5.2.2.3 Keeping requirements updated and traceable (C5)

Whether or not Keeping requirements updated and traceable (C5) should be considered a challenge is also debatable. It is clear from the result that this will be a challenge during the transition. Since the interviewees agreed that the traceability had to be handled in a new way and that traceability is difficult by nature means that this new way to handle traceability will be a challenge during the transition.

However, because it was not clear whether working agile increased the difficulty of traceability or not it can still be debated if the combination of agile and traceability is a challenge in the long run. Since the agile manifesto (Beck et al., 2001) values "working software over comprehensive documentation" there is still a reason to believe traceability, which is part of documentation, will be a challenge in an agile setting.

A possible reason as for why some interviewees did not see a challenge in keeping traceability while working agile may be because they already have a solution planned, ALM and other tools (S1). In summary, C5 will be a challenge with agile, and the FOTA group seem to view ALM and other tools as solutions to this challenge.

5.2.2.4 Understanding the SAFe practices (C6)

Even though the majority of the interviewees did not regard Understanding the SAFe practices (C6) as a major challenge, it is still possible to be. This is because interviewee I6, who is both the most knowledgeable regarding agile judging from Table 4.1 as well as being responsible for method support, believed it to be a challenge.

Even though interviewee I6 could spot some lack of agile understanding among the teams does not necessarily mean that there is insufficient knowledge about SAFe. It could simply be that it is easier to spot flaws than correct use. If this is the case, the observed lack of SAFe understanding does not have to be a challenge.

On the other hand, if the other interviewees have an insufficient knowledge about SAFe they would be less likely to spot lack of SAFe understanding in the teams. There could be a greater lack of understanding than the majority can see, but if they could see it, they might also consider it a challenge.

5.2.3 Relation of challenges to incremental requirements engineering

The challenge with **Customer agreements (C1)** makes the requirements engineering work quite complicated. As long as the FOTA group are working agile and the customer departments are not, the FOTA group need to maintain two different ways of requirements engineering. Firstly, incremental requirements engineering according to the agile framework SAFe. Secondly, fulfilling the non-agile agreements with external departments in the traditional way. Delivering according to non-agile requirements specifications, even though developed according to agile internal processes, will reduce the benefits of incremental requirements engineering, for example getting frequent feedback from the customer unit as mentioned in Section 5.2.1.1.

The challenge with **Supplier agreements (C2)** shows that the current situation with supplier departments is not optimal. If the FOTA group need to give the suppliers a full requirements specification in order to have anything developed for them, it means that this specification needs to be sufficiently good from the beginning, with all the forethought and upfront planning it means. This procedure denies the benefits of incremental requirements engineering.

Another problem is similar to what was mentioned about Customer agreements (C1). When some of the needed development work is done incrementally within the FOTA group, and the rest by non-agile external departments, there is a need to maintain two different ways of requirements engineering.

Functional safety difficult with isolated functionality (C3) means that specifying functionality is difficult because knowledge about related requirements is not always available while doing agile requirements engineering, even though it is necessary from a safety perspective. It's a collision between needing to know everything upfront and discovering requirements gradually. An important part of the ISO 26262 standard is that the requirements should specify the functionality in enough detail. The challenge is how to do this incrementally.

The challenge **Difficult to begin (C4)** means that it will be difficult to work fully agile with the requirements. This is because the agile philosophy is that customer value should be delivered from the very first iteration while all the requirements engineering work is done incrementally throughout the whole development process. The reality at Bosch is that creating something of customer value requires substantial effort and is not easily done within the first couple of iterations. A solution to this could be to compromise some of the agility and dedicate the first iterations to upfront requirements engineering work. The requirements engineering would resemble waterfall rather than agile, and the approach risks leading to further problems. We elaborate on this in Section 5.3.3.

The challenge of **Keeping requirements updated and traceable (C5)** originates in incremental requirements engineering in combination with safety-criticality. While working with traditional requirements engineering traceability is not as much of a challenge because requirements change less frequently. This challenge makes it necessary for the requirements engineering process to use a solution such as ALM and other tools (S1).

Understanding the SAFe practices (C6) from an incremental requirements engineering perspective means not understanding or having difficult in producing requirements in the way SAFe advocates. For example, the interviews revealed that many employees have trouble writing good user stories. Another problem is how to translate customer requirements into the corresponding concepts defined by SAFe.

If there is **Resistance from management (C7)** regarding the agile transition in general then incremental requirements engineering will also face resistance. This could mean less resources spent on the solutions found in RQ3, which in turn affects requirements engineering.

In the interviews, resistance was said to arise from more visible and direct costs. Working with requirements in an agile way may result in less predictable contracts because of the solution of Agile contracting (S2), which could possibly increase resistance because management would have less control. Management has less control because there are no upfront specifications, while developers have more control. Meaning, it is possible that the resistance is directly related to requirements.

5.3 RQ3: Solutions to the challenges in RQ2

- **RQ3:** *What are solutions to the challenges mentioned in RQ2 seen by the practitioners in FOTA? To what extent are they expected to be useful in solving challenges related to requirements engineering?*

When comparing the solutions found with existing work it became apparent that the implementation of the tool suite ALM is a necessity. Besides this it was believed that agile contracting, having an agile prototype with incrementally added complexity, SAFe training and tutorials, and starting out with waterfall in the beginning would to some degree decrease the challenges found in RQ2.

5.3.1 Solutions compared to existing work

In this section, the solutions found in FOTA are compared with solutions, success factors and recommendations found in existing work regarding large-scale organisations transitioning to agile.

The solutions are mapped together in Table 5.3, and a more detailed description is provided further down in this section.

Table 5.3: Solutions compared to existing work

Solution	Solution in existing work
ALM and other tools (S1)	Massive Investment in Continuous Integration (Goodman and Elbaz, 2008)
	Interoperable tool-chain (Gallina and Nyberg, 2015)
	Tool for safety case generation (Gallina and Nyberg, 2015)
	Need for a robust requirement management tool (Hanssen et al., 2016)
Agile contracting (S2)	Agile contracting (Hoda et al., 2009)
Agile prototype with incrementally added complexity (S3)	Start with a pilot to gain acceptance (Dikert et al., 2016)
	Gather insights from a pilot (Dikert et al., 2016)
SAFe training and tutorials (S4)	Provide training on agile methods (Dikert et al., 2016)
	Sending a large set of people to professional training (Fry and Greene, 2007)
Waterfall in the beginning (S5)	Water-Scrum-Fall (West et al., 2011)

5.3.1.1 ALM and other tools (S1)

As seen in Table 5.3, there were several recommendations to have some solution similar to S1. While the other solutions say that there is a need for a solution like S1, "Interoperable tool-chain" gives more detail as to how the implementation should be done. Focus is on the interoperability of the tools and this is found in ALM. ALM is in itself a set of different integrated tools, as described in section 2.3.4.

In the SafeScrum case study conducted at Autronica Fire & Security (Hanssen et al., 2016), the authors found that there was a "Need for a robust requirement management tool", which is included in S1.

5.3.1.2 Agile contracting (S2)

According to the study by Hoda et al. (2009), a way to introduce agile agreements towards customers is to present the economic benefits of agile, for example only having to pay for features that are actually implemented. The same study states that even though it should be considered a last resort, one way is to actually hide the agile way of working for the customers. This could be done when one wants to avoid losing a deal with a non-agile customer. According to the findings from the interviews, this seems to be something like what the FOTA group at Bosch are planning to do in their agile contracting with other business units at Bosch, at least in the initial phase.

A possibly better way presented in the same study is to change the mindset of the other party. This seems to be something that most of our interviewees have in mind. Statements like "they will probably have to change" and "only possibility is to show to the management the benefits of working agile" show that people responsible for carrying out the agile transition believe that there is a chance to convince the rest of the organisation that another way of working is possible.

5.3.1.3 Agile prototype with incrementally added complexity (S3)

Both "Start with a pilot to gain acceptance" (Dikert et al., 2016) and "Gather insights from a pilot" (Dikert et al., 2016) give reason that solution S3 should be carried out, however no more detail as to how a pilot should be implemented is given.

Out of the two, only "Gather insights from a pilot" have a similar goal as found in FOTA, which is to increase experience. "Start with a pilot to gain acceptance" could easily be seen as a solution to the challenge Resistance from management (C7), however this was not found as a reason for this solution in FOTA. This might be because the prototype approach is seen only as a tiny part of a long-term plan to change mindsets in the Bosch organisation, and therefore was not immediately brought up.

5.3.1.4 SAFe training and tutorials (S4)

In a report describing a successful agile transformation by Fry and Greene (2007) it was said that one of the key contributors to the success was "sending a large set of people to professional training and hiring external, experienced consultants to assist team members, Scrum Masters, Product Owners and functional managers with the process".

It was considered that Scrum Masters and Product Owners in FOTA should be given extra training, something also considered a success factor as stated by Fry and Greene (2007).

When Dikert et al. (2016) present the success factor that is to “Provide training on agile methods” it is stated that an agile transition could even fail completely without training.

5.3.1.5 Waterfall in the beginning (S5)

The findings from the interviews, that the people responsible for the agile transition see waterfall in the beginning as something inevitable and implicit in the current context, fits quite well together with the findings of West et al. (2011) in their report about “Water-Scrum-Fall”. On top of the common reasons from their report is the safety-criticality and the automotive business traditions for Bosch, which further adds difficulty to any attempt of managing requirements in a 100% agile way.

5.3.2 Will the solutions solve the challenges addressed

Each of the solutions found had one or two challenges which it aimed to solve, which can be seen in Table 4.4. In this section we will discuss for each solution to what degree is solves the challenge(s) it targets. In section 5.4, we will discuss the degree to which all the challenges are solved.

5.3.2.1 ALM and other tools (S1)

One of the goals with the implementation of ALM is to facilitate traceability. The question is then whether or not it will be successful. In existing work it is clear that some sort of set of tools like ALM will be necessary. This does not mean ALM will solve the challenge of traceability, but at least supports that without a tool suite like ALM the challenge cannot be solved in a reasonable manner.

Increasing the chances of the solution being successful, ALM allows for a lot of configuration including traceability link types, roles and linkage-rules being user definable as required to allow for traceability while working agile (Espinoza and Garbajosa, 2011).

Given the existing work, this solution is necessary and if carried out as planned there is nothing found that would suggest it would not solve Keeping requirements updated and traceable (C5). However, Kasauli et al. (2017) found that establishing an agile tool chain is a challenge in that the implementation might not be correct from the beginning. Adaptation can be made with experience and in the long run C5 should be fully solved.

5.3.2.2 Agile contracting (S2)

As described in 4.4.2, the people at FOTA did not believe that changing the mindset of the other party was likely to happen in a close period of time. In the meantime, the process could at least be accelerated by the use of an Agile prototype with incrementally added complexity (S3), demonstrating the benefits of an agile way of working, as described in the section below. This could be done in parallel with the approach of presenting alternatives, that is offering a limited number of iterations in order to minimise the risk for the customer as described in section 2.1.11.

The solution Agile contracting (S2) is supposed to solve both Customer agreements (C1) and Supplier agreements (C2). As described in section 2.1.11, this solution is composed of three strategies, but it is only the most difficult strategy of changing the mindset of the other party that can fully solve the challenges targeted. Only using the strategy of hiding the agile way of working would only solve a small part of the challenge. If the other party is avoiding collaboration with agile organisations, hiding the agile way of working can be a solution. But it will not solve the problem of feedback being slow and contract negotiation being difficult.

Because of this, S2 will not fully solve the challenges it addresses until the mindset of the other parties is changed, something the interviewees did not consider reasonable in a close period of time.

5.3.2.3 Agile prototype with incrementally added complexity (S3)

This solution was not explicitly asked about during the interviews because we did not see any indication of this solution during the observations. The reason for this is because this is not a planned solution, but mentioned during the interviews as a possible solution to Difficult to begin (C4) and Understanding the SAFe practices (C6).

A possible flaw with this idea would be the possibility that the systems design produced in the prototype cannot be used in a later stage. When complexity is added to the systems design there might be a lot of work adapting to this new complexity. In the end it might have been less work adding all dimensions of complexity from the start.

According to West et al. (2011), one of the most important practises in an agile-waterfall hybrid framework is to embrace the agile value of limiting the resources spent on upfront requirements specification. It seems like this solution would require upfront requirements specification which is not preferable. However, this will have to be accepted since solving the challenges is higher prioritised than keeping the initial way of working 100% agile.

For solving Difficult to begin (C4) this solution gives a plan on where to start, and at least partially solves C4. Regarding Understanding the SAFe practices (C6), using the practices will yield experience. It is uncertain whether that is enough to solve C6.

5.3.2.4 SAFe training and tutorials (S4)

As mentioned in the result, even though most people involved seem to have a fair knowledge of agile, the training can be very useful for synchronising the teams with a common terminology and updated knowledge in accordance to the latest research on the area. Kasauli et al. (2017) found that channelling the "right" information towards and between teams is difficult and time-consuming which limits agility of teams. If there are misunderstandings in the terminology used the flow of the "right" information is likely to be even more difficult and time-consuming.

This solution was aimed to target the challenge of Understanding the SAFe practices (C6). Judging from the solutions found in existing work there is no doubt that SAFe training and tutorials is important, however the question is whether the FOTA group can reach the sufficient level of training that is necessary to solve the challenge. From the interviews it was found that the members of FOTA have an idea of what they think is enough training. With too little training more can be added and with too much training at least C6 will be solved, although not optimally. Hence, there is no reason to believe that S4 will not be successful.

5.3.2.5 Waterfall in the beginning (S5)

From the interviews, we draw the conclusion that the context makes it impossible for the FOTA group to do anything but keep some of the waterfall influences in the requirements engineering work even after the agile transition.

Similarly to the solution of Agile prototype with incrementally added complexity (S3), this solution at least gives a starting point, which will help to solve Difficult to begin (C4).

5.3.3 Relation of solutions to incremental requirements engineering

ALM and other tools (S1) are necessary for FOTA, as stated in Section 5.3.2.1. A risk with incremental requirements engineering is that if requirements is allowed to change more often than in the traditional setting, it is challenging to manage traceability. Also, according to some of the interviewees, with traditional requirements engineering traceability could in theory be managed through a much simpler traceability tool, but agile requirements engineering makes traceability more important in a safety context. Thus, a more advanced traceability system is needed in agile requirements engineering. Based on this and the findings from the interviews, it is concluded that incremental requirements engineering is not possible without ALM and other tools (S1).

To make full use of the benefits of working agile, **Agile contracting (S2)** should be used to facilitate collaboration between FOTA and other parties. If the mindset of the other parties is changed, with the help of agile contracting, then for incremental requirements engineering this means that requirements would be discussed and prioritised with the other parties each iteration, just like regular agile. With immediate access to customers and their involvement the clarity and understandability of requirements are improved (Balasubramaniam et al., 2010). However, if the more likely event happens that the agile way of working has to be hidden from the customer, then there will be agreements on upfront requirements that are iterated behind the scenes. This means less customer involvement and possibly less accurate requirements.

Developing an **Agile prototype with incrementally added complexity (S3)** means that high level requirements would need to be created to allow for systems design in the prototype. Working with incremental requirements engineering in a prototype, the same way as with systems design described in Section 4.4.3, will give the practitioners experience with incremental requirements engineering.

SAFe training and tutorials (S4) are needed to increase the knowledge of how to work with incremental requirements engineering. As mentioned in Section 2.1.6.1, the SAFe framework defines new concepts to replace the traditional requirements. Training the employees in working with these new concepts is necessary in order to succeed with incremental requirements engineering.

When using **Waterfall in the beginning (S5)** incremental requirements engineering through the whole process is no longer possible. The whole point of this solution is to avoid working incrementally for a period of time in the beginning of the project. West et al. (2011) states several problems with too much upfront requirements engineering, which is a risk with this solution. These problems are: Too many early requirements are too many wrong requirements; Users do not always communicate effectively what they want; The team takes less ownership of the outcome; The development team is less cross-functional. Even though the solution of Waterfall in the beginning (S5) brings benefits, it also delays and prevents incremental requirements engineering.

5.4 Are all challenges solved?

This section will discuss to what degree the challenges are addressed by the solutions. In Table 5.4 it can be seen that challenge C3 and C7 are not targeted by a solution. Challenges C4 and C6 are both targeted by two solutions. In Table 5.5, in the end of this section, the degree to which all challenges are solved can be seen.

Because challenge C1, C2, and C5 are all only targeted by one solution these will not be discussed here. To what degree they are solved are entirely dependent on one solution alone and a discussion about these solutions can instead be read in section 5.3.2.

Table 5.4: Table of challenges with solutions

Challenge	Addressed by solution
Customer agreements (C1)	Agile contracting (S2)
Supplier agreements (C2)	Agile contracting (S2)
Functional safety difficult with isolated functionality (C3)	–
Difficult to begin (C4)	Agile prototype with incrementally added complexity (S3) & Waterfall in the beginning (S5)
Keeping requirements updated and traceable (C5)	ALM and other tools (S1)
Understanding the SAFe practices (C6)	Agile prototype with incrementally added complexity (S3) & SAFe training and tutorials (S4)
Resistance from management (C7)	–

In section 5.2.2 we discussed whether or not Functional safety difficult with isolated functionality (C3) really should be considered a challenge, and argued that it still likely is. The implications of not solving C3 would mean that FOTA would not comply with the regulations, which is not an alternative. Thus, the challenge has to be solved in some way. We believe that the challenge is solvable, and in section 2.1.8 a suggested solution is briefly explained. A possible reason as for why no solution was found for this challenge might have been that even though the interviewees had some knowledge about possible solutions, no concrete plan had been formulated.

Resistance from management (C7) may be partially solved by Agile prototype with incrementally added complexity (S3). Dikert et al. (2016) states that a pilot can be used to gain acceptance by showing success, which in turn would reduce resistance from management. Using this logic, all solutions found will be used to increase success of the agile transition, consequently reducing resistance from management.

If the challenge C7 is not solved, there is a risk that investments in other solutions are delayed or stopped. This would mean less success in the agile transition and more resistance from management, creating a vicious circle.

The challenge Difficult to begin (C4) is mapped to two different solutions, Agile prototype with incrementally added complexity (S3) and Waterfall in the beginning (S5). The question arises whether the two solutions are compatible, and whether the challenge will be solved if the solutions are combined. Using S3 means that a significant amount of systems design would be done, more than in the planned agile iterations. This is similar to what is meant by Waterfall in the beginning (S5), meaning the solutions can be combined. Both solutions solve C4 in the same way, by giving a starting point to work further from. These starting points would probably be equivalent if both solutions are used. This could imply that the solutions together will not solve the challenge better than any of them alone. How well each solution will work is discussed in section 5.3.2.

Understanding the SAFe practices (C6) should to a large degree already be solved by SAFe training and tutorials (S4). Since no incompatibility between S4 and Agile prototype with incrementally added complexity (S3) can be seen, it is concluded that C6 will be fully solved.

Table 5.5: How well challenges are solved

Solved to degree	**Challenges**
Mostly solved	C4, C5, C6
Partially solved	C1, C2, C7
Not solved	C3

5.5 Evaluating the Agile Transition at FOTA

In this section we will discuss whether or not the transition to agile in FOTA was the correct decision. The cost of the solutions found in RQ3 and the cost of unsolved challenges found in RQ2 will be compared to the benefits of the reasons found in RQ1. Since the research questions has been asked and discussed from a requirements engineering perspective, the evaluation will also be from a requirements engineering perspective. Because an agile transition is a long term investment, non-recurring costs during the transition will not be regarded a deciding factor in the evaluation.

If challenges C1, C2 and C7 are not solved, the FOTA group will have to keep acting in a non-agile environment in the future. This could mean that the way of working is forced back to something similar to the old waterfall approach. However, if this is the case, any long-term costs would not be greater than they were before the transition.

Failing to solve C6 could in a similar way inhibit a successful agile transition, leading to a loss of agility but no significant increase in long-term cost.

Out of all the seven challenges found in RQ2 it is only Functional safety difficult with isolated functionality (C3) and Keeping requirements updated and traceable (C5) which pose any significant risk of increasing long term costs. If not solved, both of those challenges will lead to either safety regulations not being fulfilled or significant extra work to fulfil said regulations in the future. Leaving any of the other challenges unsolved would “only” lead to a decrease in agility.

None of the five solutions found in RQ3 should lead to considerable long term costs. Both ALM and other tools (S1) and SAFe training and tutorials (S4) will of course be costly in the initial stages but after the transition the training should be mostly finished and ALM will be up and running. Maintenance costs of the tool-chain could be seen as a long term cost, but traceability tools will be needed whether the teams are working agile or not.

In a study by Korhonen (2013) on evaluating the impact of an agile transformation the success of the transformation was evaluated based on to which degree the goals set for the transformation was fulfilled. In the same way it will be evaluated to which degree FOTA will see the benefits of the reasons found in RQ1. As seen above, the cost of solutions or the cost of unsolved challenges are not the main concerns, and neither of these will be the deciding factor when evaluating the transition. The most important factor in the evaluation will instead be to what extent the organisation can reap the benefits of agile development when the challenges and solutions in large part will affect the possible level of agility.

5.5.1 How much agility, and its benefits, is lost?

One of the fundamental agile values, described by Beck et al. (2001), is “working software over comprehensive documentation”. According to Meyer (2014), the agile movement uses two kinds of criticism to advise against unnecessary documentation work, the *waste criticism* and the *change criticism.*

The waste criticism is that creating requirements that will not be used, or are implemented but later shown to be incorrect or unnecessary, is wasteful. In the FOTA case, however, requirements engineering work that is done upfront for the sake of complying with safety regulations is by no means wasteful.

The change criticism deals with the fact that the requirements of every software project will change, so one should not spend too much time on writing them down. As previously mentioned, the non-agile business culture nevertheless forces the FOTA group to have a detailed plan in order to start the development work. Furthermore, safety regulations force the FOTA group to document all requirements and their changes from start to end of the project. This makes the upfront requirements engineering work necessary.

In conclusion, the benefits of not “wasting time” on documentation told by the agile movement does not apply in FOTA to a large extent. The FOTA group simply need to practise a significant amount of documentation work, and no benefits would be reaped from discarding it.

We believe that the FOTA group are currently, and will probably also end up, using an approach resembling the Water-Scrum-Fall practice explained by (West et al., 2011) for the same common reasons that are stated in their report. Mostly, these reasons are related to the agile teams only being able to implement their way of working in parts of the organisation which they have influence over. The practices in the rest of the organisation, like business analysis and release management, follow non-agile business rules about agreements and deliveries. This means that there is a need for interfaces against the non-agile parts of the organisation, very much like the situation in FOTA.

West et al. (2011) warn practitioners of agile development about compromising Scrum principles like building cross-functional teams and including testing within the sprint. If one does not follow these principles, the implementation of the “scrum” part could become incorrect. The FOTA group at Bosch plan to follow all the principles listed by West et al. (2011), and will not have to worry about this, even if they are in fact following something like Water-Scrum-Fall.

To evaluate the agile transition, one needs to evaluate each reason found in RQ1 and its benefits, and discuss whether those benefits would be present even if FOTA cannot be fully agile.

Handling changing requirements (R1) is specifically impeded by the fact that the suppliers to FOTA are not working agile. If the suppliers are not open to changes in requirements, FOTA cannot forward changing requirements to them as well as intended. However, there will be cases where changing requirements will not affect non-agile suppliers, for example if the FOTA group are developing these parts of the system by themselves instead of having them developed by a supplier. In these cases the benefit of Handling changing requirements (R1) will still be present.

There is a risk that Motivating the employees (R2) will suffer from FOTA not being completely agile because feedback will be slow. Five critical factors regarding motivation in the Job Characteristics Model by Hackman and Oldham (1980) are: Autonomy, Variety, Significance, Feedback, and Ability to Complete Whole Tasks. Part of why working agile increases motivation is because feedback is received faster. As seen in the challenge Customer agreements (C1) part of the problem with customers not working agile was that they were not giving feedback quickly. However, agile increases motivation by increasing all the five critical motivational factors (Tessem and Maurer, 2007), not just feedback, so motivation should increase with the transition into agile.

Shorten development cycles (R3) will somewhat be negated by the solution Waterfall in the beginning (S5). The solution means that for some period of time agile practices to a large degree will not be used, especially short iterations. These short iterations allow for Shortening development cycles (R3) but because waterfall will only be used in the beginning of the project the benefits of Shortening development cycles (R3) will still be present in the long term.

In summary, from a requirements engineering perspective the benefits from the agile transition in the long run seem to outweigh the risks and costs of the challenges and the solutions.

5.6 Suggestions for future research

The difficulties with finding a way to start out with agile development, the challenge C4, was one of the most prominent challenges found in this study. Almost all of the interviewees mentioned it, despite it not being one of the challenges we expected before the interviews and weaved into the interview questions.

As mentioned in section 5.2.1.4, there indeed seems to be a lack of concrete advice from the literature on how to begin an agile transition in general or a SAFe implementation in particular. A suggestion for future research is to examine successful cases of these implementations with the pronounced goal to compile a set of empirically supported practical pieces of advice for on how to implement a large-scale agile transition and for how to deliver customer value from the first iterations.

As pointed out in Section 1.4, there is still a need for studies which focus on the role of RE in large-scale agile development. In a systematic literature review, Dikert et al. (2016) mention a lack of case studies about agile development in a large scale. This study widens the research about requirements engineering in a large-scale agile context, but there is still a need for more case studies in similar contexts.

5.7 Threats to validity

In this section possible threats to the validity of the study are described. Different aspects of validity in the context of a case study are mentioned, as well as the actions taken to mitigate the threats.

5.7.1 Construct Validity

Runeson et al. (2012) describes construct validity as “to what extent the operational measures that are studied really represent what the researcher has in mind and what is investigated according to the research questions”.

The main source of data in this study is interviews. Because of confidentiality regulations, these could not be audio-recorded. To mitigate this threat to validity, detailed notes were taken during the interviews, in an attempt to reflect as accurately as possible what the interviewees were saying.

The interviews in this study were held in Swedish or English, depending on the language preference of the interviewee. This poses a threat to validity, both in the situation when the same set of questions were asked and discussed in different languages, as well as when the interview data were put together for the English report. One term might have a different meaning to different people, and discussing in a foreign language makes it even more difficult to ensure a common framework.

An alternative to this approach would be to conduct every interview in English. Unfortunately, this would pose a similar threat to construct validity since it is not the first language of the researchers and the interviewee. The currently used approach has the benefit that the participants do not have to put effort into overcoming the language barrier during the interview, and can instead focus on generating data that can later be analysed and translated when there is more time.

Another possible threat to the construct validity is the possibility of bias from leading questions during the interviews. There were some challenges or solutions for which the opinions of the interviewees were sought, however, asking about these might have made the interviewees omit other challenges or solutions. To mitigate this threat the interviewees were never asked about specific challenges or solutions until they had the chance to say the ones they knew before.

Part of this threat is also the question of which challenges to ask structured questions about. Due to limited insight into the whole working environment and the project in the beginning, it was never obvious which challenges were to be expected. In retrospect, there could be several challenges that were to be expected, but that we missed to consider from the very beginning. One example is the "difficult to begin" challenge.

When preparing the interviews there were constraints about which participants to choose. The interviewees were selected by the company supervisor, something that limited the possibility to choose interviewees of different roles as desired. However, as mentioned in section 5.7.4, the group of interviewees made up a relatively exhaustive part of the set of possible roles of interest for the study. It is likely that the same group of interviewees had been chosen if there were no constraints on the selection.

The time scheduled for interviews was limited to about 30 minutes per person in each interview round. This was sufficient for giving the interviewee a chance to give well developed answers for all of our questions, but very little time left for follow-up questions. While some follow-up questions generated by the analysis from the first interview round could be asked in the second interview round, this time limit affected the construct validity of the study.

During the workshop (see section 3.4), only five of the six interviewees were present, and two of them via Skype. To make sure the findings had been validated with all six interviewees member checking with the interviewee missing from the workshop could have been done at a later time. However, due to time constraints this was not possible. This will simply have to be accepted and the validity should still be high because most interviewees were present during the workshop.

5.7.2 Internal Validity

Internal validity is a term related to causal relationships. Low internal validity in this study means inaccurate conclusions drawn about the causality between the agile transition and the findings during the observations, interviews and workshop.

Our first on-site involvement with Bosch's FOTA project was a three-day workshop with discussions about the future implementation of a feasible version of the lifecycle management system ALM. This was considered a good way to get familiar with the working environment and the system as early as possible. However, this first impression might have had an impact on our impression of what solutions would be most important in solving the challenges that would later be discovered. There is a possibility that ALM was given a disproportionate amount of significance.

All research questions in this study had a clear focus on requirements engineering and consequently the questions asked during the interviews reflected that. However, some of the findings are only indirectly related to requirements engineering, for example the challenge of Understanding the SAFe practices (C6). We do not see this as a problem because it is entirely possible that the requirements engineering work will be more affected by something indirect than direct.

In some parts of the discussion section the requirements engineering focus is fully visible, while for other parts it is less central. However, all of its parts are discussing the findings, which come from a requirements engineering perspective.

The question arises whether the findings less directly focused on requirements engineering come from the transition in general or from requirements engineering specifically. During the interviews we noticed that the interviewees sometimes forgot the scope of requirements engineering, in which case they were reminded of the focus of the study. Either way, all findings can easily be related to requirements engineering and thus we believe the study is still within its supposed scope.

When it comes to the group of interviewees, a sample size of six might be considered small when it comes to giving an accurate representation of the thoughts and expected challenges within the group. However, the interviewees have been chosen to represent a versatile and exhaustive set of roles from the business unit.

5.7.2.1 Interviewees - role, experience

There are several ways that the roles of the interviewees could have influenced their view of the project and thus their answers. A few are listed below.

- Some of the interviewees lacked experience of agile development and/or requirements engineering. This could increase the risk of missing considering some important challenges or solutions to them. In an ideal setting, all the people participating in the interview would be sufficiently experienced within the subjects under study.
- The project manager, I1, made clear that he believed in the tools that the project was going to use. While giving reasonable arguments for his thoughts, it is doubtful that a project manager would be of a different opinion, or at least mention a different opinion in an interview.

5.7.3 External Validity

This aspect of validity concerns the possibility to generalise the findings, to which extent the findings are of relevance to other cases and other people outside this case (Runeson et al., 2012).

External validity is particularly difficult to achieve in case studies. Cases are often selected because of factors that are uncommon or unique to the particular case. In this study one of the factors that set this case apart from many other cases is the safety regulations, even though this is not so uncommon, especially in the automotive industry.

For a case study the intention is to enable analytical generalisation so that the findings are relevant for cases with similar characteristics (Runeson et al., 2012). To do so, much effort in this study is put into describing the characteristics of this case. Being involved, as a researcher, in the daily requirements engineering process in the project leads to a good understanding of the context and the processes taking place. Since the context of the case and the ambition to switch to a more agile way of working is not uncommon in the software industry, the study can still be of value to other researchers and practitioners.

5.7.4 Reliability

The reliability of a study is connected to what extend its result depends on the specific researcher. In theory, if a study with high reliability were to be conducted again by different researchers, the result would still be the same.

The coding of data from the first interview round (see section 3.2.3.1) has been based on reasonable similarities in the answers to interview questions about expected challenges and solutions to said challenges. If another researcher were to conduct the same study, there is a possibility that the researcher would think of a different pattern from the same set of interview data, and the coding would thus be different. Because the second round of interviews was constructed based on the coding from the first round, this would in turn lead to the next step of data collection being differently constructed.

As mentioned above, using different methods for collecting data is a form of data triangulation. Letting the results from early observations influence the interview questions might lower the effectiveness of this triangulation, because this approach makes the different data collection methods more dependent on each other.

In an effort to mitigate these threats to reliability, the formulation of the data coding was done by both researchers together, which lowers the risk for bias (Runeson et al., 2012). The coding was also constructed iteratively in an attempt to avoid missing important codes, which would pose a threat to reliability.

Between the first and the second round of interviews, before designing the questions and the structure of the second round, the company supervisor had the chance to see the data collection results and discuss these. The workshop after the interviews provided a similar chance for all the interviewees. This form of member checking will hopefully improve the reliability.

A questions is whether the same study with the same results could have been conducted by non-Swedish speaking researchers. The company language is English and therefore all observations from workshops and in discussions with the supervisor could easily have been acquired even if the researchers would not speak Swedish. Since all employees in the FOTA group are supposed to be able to communicate in English as part of their job, the interviews could have been conducted in this language as well.

6
Conclusion

In this report, we examined the transition from waterfall development to agile development at the FOTA business unit at Bosch. Through observations, interviews and a workshop session we collected information about the transition in order to answer the research questions:

- **RQ1:** *What are requirements engineering related reasons to expect for a transition from waterfall development into agile in a project like FOTA?*
- **RQ2:** *Focusing on incremental requirements engineering, which challenges need to be taken into account on the way to a successful transition into agile in a project like FOTA?*
- **RQ3:** *What are solutions to the challenges mentioned in RQ2 seen by the practitioners in FOTA? To what extent are they expected to be useful in solving challenges related to requirements engineering?*

When carrying out an agile transition like the one in this study, it is important to decide and define the reasons for the transition. It is also important to identify challenges related to the transition and how one plans to solve or address them. After these steps, a decision must be taken whether the benefits from agile development will outweigh the costs and risks of the transition.

Most of the challenges found in this study had solutions that could be mapped to them. For those without clearly mapped solutions, we believe them to be solvable with a combination of practices described in the discussion section.

After weighing together the tangible risks and costs of the challenges and the benefits of the reasons for the agile transition, we believe that Bosch will gain from the agile transition in the long run.

As stated in the introduction, moving agile to a large scale requires compromises between agility and planning, and many of these compromises relate to methods and knowledge about requirements. As pointed out in Section 1.4, there is still a significant need for studies which focus on the role of RE in large-scale agile development. In this study, we have contributed to the field by providing a realistic, in-depth case study of a large-scale agile transition, focusing on RE challenges. Our findings can be useful for both RE and agile researchers and practitioners.

Bibliography

Amabile, T. M.
1993. Motivational synergy: Toward new conceptualizations of intrinsic and extrinsic motivation in the workplace. *Human resource management review*, 3(3):185–201.

Aurum, A. and Wohlin, C.
2005. *Engineering and Managing Software Requirements.* Springer-Verlag Berlin Heidelberg.

Balasubramaniam, R., Lan, C., and Richard, B.
2010. Agile requirements engineering practices and challenges: an empirical study. *Information Systems Journal*, 20(5):449–480.

Beck, K.
2000. *Extreme Programming Explained.* Addison-Wesley.

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

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

Boehm, B. and Turner, R.
2004. *Balancing Agility and Discipline: A Guide for the Perplexed.* Addison-Wesley.

Boehm, B. and Turner, R.
2005. Management challenges to implementing agile processes in traditional development organizations. *IEEE Software*, 22(5):30–39.

Center of excellence for software traceability
2018. `http://www.CoEST.org`. Accessed: 2018-04-10.

Cleland-Huang, J., Gotel, O., and Zisman, A.
2012. *Software and Systems Traceability.* Springer-Verlag London Limited.

Cusumano, M. A. and Smith, S.
1995. *Beyond the waterfall : software development at Microsoft*. Sloan School of Management, Massachusetts Institute of Technology.

Davis, A., Overmeyer, S., Jordan, K., Caruso, J., Dandashi, F., Dinh, A., Kincaid, G., Ledeboer, G., Reynolds, P., Sitaram, P., Ta, A., and Theofanos, M.
1993. Identifying and measuring quality in a software requirements specification. In *Proceedings of the First International Software Metrics Symposium*, Pp. 141 – 152.

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

Dingsøyr, T., Fægri, T. E., and Itkonen, J.
2014. *What Is Large in Large-Scale? A Taxonomy of Scale for Agile Software Development*, Pp. 273–276. Springer International Publishing.

Espinoza, A. and Garbajosa, J.
2011. A study to support agile methods more effectively through traceability. *Innovations in Systems and Software Engineering*, 7(1):53–69.

Franklin, T.
2008. Adventures in agile contracting: Evolving from time and materials to fixed price, fixed scope contracts. In *Proceedings - Agile 2008 Conference*, number Proceedings - Agile 2008 Conference, Pp. 269–273, Info Tech, Inc.

Fry, C. and Greene, S.
2007. Large scale agile transformation in an on-demand world. In *Agile 2007 (AGILE 2007)*, Pp. 136–142.

Gallina, B. and Nyberg, M.
2015. Reconciling the ISO 26262-compliant and the agile documentation management in the Swedish context. In *CARS 2015 - Critical Automotive applications: Robustness & Safety*, Roy, M., ed., Paris, France.

Gandomani, T. J., Zulzalil, H., and Nafchi, M. Z.
2014. Agile transformation: What is it about? In *IEEE SoutheastCon 2015*.

Ge, X., Paige, R. F., and McDermid, J. A.
2010. An iterative approach for development of safety-critical software and safety arguments. In *2010 Agile Conference*, Pp. 35–43.

Goodman, D. and Elbaz, M.
2008. "it's not the pants, it's the people in the pants" learnings from the gap agile transformation what worked, how we did it, and what still puzzles us. In *Agile 2008 Conference*, Pp. 112–115.

Gotel, O. C. Z. and Finkelstein, C. W.
1994. An analysis of the requirements traceability problem. In *Proceedings of IEEE International Conference on Requirements Engineering*, Pp. 94–101.

Hackman, J. R. and Oldham, G. R.
1980. *Work Redesign.* International Standard.

Hanssen, G. K., Haugset, B., Stålhane, T., Myklebust, T., and Kulbrandstad, I.
2016. Quality assurance in scrum applied to safety critical software. In *Agile Processes, in Software Engineering, and Extreme Programming*, Sharp, H. and Hall, T., eds., Pp. 92–103, Cham. Springer International Publishing.

Hoda, R., Noble, J., and Marshall, S.
2009. Negotiating contracts for agile projects: A practical perspective. In *Agile Processes in Software Engineering and Extreme Programming*, Abrahamsson, P., Marchesi, M., and Maurer, F., eds., Pp. 186–191, Berlin, Heidelberg. Springer Berlin Heidelberg.

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

Hohl, P., Stupperich, M., Munh, J., and Schneider, K.
2018. Combining Agile Development and Software Product Lines in Automotive: Challenges and Recommendations. In *Conference: 24th ICE/IEEE International Technology Management Conference.*

IBM
2018. Rational collaborative lifecycle management. `https://www.ibm.com/us-en/marketplace/application-lifecycle-management`. Accessed: 2018-06-10.

IEEE Xplore
1990. IEEE Standard Glossary of Software Engineering Terminology. *IEEE Std 610.12-1990.*

International Organization for Standardization
2011. *International Standard 26262: Road vehicles – Functional safety.* International Standard.

Kasauli, R., Liebel, G., Knauss, E., Gopakumar, S., and Kanagwa, B.
2017. Requirements engineering challenges in large-scale agile system development. *Requirements Engineering Conference Workshops (REW) 2017 IEEE 25th International*, Pp. 427–430.

Korhonen, K.
2013. Evaluating the impact of an agile transformation: a longitudinal case study in a distributed context. *Software Quality Journal*, 21(4):599–624.

Lauesen, S.
2002. *Software Requirements: Styles and Techniques.* Addison-Wesley.

Lawrence Pfleeger, S. and Atlee, J. M.
2009. *Software Engineering: Theory and Practice*, 4th edition. Pearson.

Mathur, S. and Malik, S.
2010. Advancements in the v-model. *International Journal of Computer Applications*, 1(12):29–34.

Meyer, B.
2014. *Agile! The Good, the Hype and the Ugly.* Springer International Publishing.

Nuseibeh, B. and Easterbrook, S.
2000. Requirements engineering: A roadmap. In *Proceedings of the Conference on The Future of Software Engineering*, ICSE '00, Pp. 35–46, New York, NY, USA. ACM.

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

Runeson, P., Höst, M., Rainer, A., and Regnell, B.
2012. *Case Study Research in Software Engineering: Guidelines and Examples.* John Wiley Sons, Inc.

Scaled Agile, Inc.
2018. Safe for lean enterprises. `http://www.scaledagileframework.com/#`. (Retrieved 19 February 2018).

SINTEF / NTNU
2018. Safescrum. `http://safescrum.no/`. (Retrieved 23 March 2018).

Stålhane, T. and Myklebust, T.
2016. The agile safety case. In *Computer Safety, Reliability, and Security*, Skavhaug, A., Guiochet, J., Schoitsch, E., and Bitsch, F., eds., Pp. 5–16, Cham. Springer International Publishing.

Takeuchi, H. and Nonaka, I.
1986. The new new product development game. *Harvard Business Review.*

Tessem, B. and Maurer, F.
2007. Job satisfaction and motivation in a large agile team. In *Agile Processes in Software Engineering and Extreme Programming*, Concas, G., Damiani, E., Scotto, M., and Succi, G., eds., Pp. 54–61, Berlin, Heidelberg. Springer Berlin Heidelberg.

Theocharis, G., Kuhrmann, M., Münch, J., and Diebold, P.
2015. Is water-scrum-fall reality? on the use of agile and traditional development practices. *Product-Focused Software Process Improvement.*

Vaidya, A.
2014. Does DAD Know Best, Is it Better to do LeSS or Just be SAFe? Adapting Scaling Agile Practices into the Enterprise.

Varhol, P.
2012. Agility with traceability: Blending requirements and user stories. In *2012 Quest Conference*.

VersionOne
2017. The 11th annual state of agile report.

West, D. et al.
2011. Water-scrum-fall is the reality of agile for most organizations today. *Forrester Research*.

Yin, R. K.
2003. *Case Study Research: Design and Methods*, 3rd edition. SAGE Publications.

A

Appendix A

A.1 Agile practices used in FOTA

Table A.1: Table of agile practices used in FOTA

Agile practice
Acceptance Testing
Automated Build
Backlog
Backlog Grooming/Backlog Refinement
Burndown Chart
Continuous Integration
Daily Meeting
Definition of Done
Definition of Ready
Estimation
Incremental Development
Iteration
Iterative Development
Kanban Board
Pair Programming
Planning Poker
Points (estimates in)
Scrum of Scrums
Story Mapping
Story Splitting
Task Board
Test Driven Development (TDD)
Team
Timebox
Unit Testing

Continued on next page

Table A.1 – continued from previous page

Agile practice
User Stories
Velocity
Version Control

A.2 Interview templates

A.2.1 Interview one template

Template for interview round one

Name:

Role:

Experience with agile:

Experience with requirements engineering:

(Explanation of the study)

What requirements engineering-related challenges do you think follow with transitioning from waterfall to agile? (more open part)

We expect at some requirements engineering-related challenges will follow with transitioning from waterfall to agile. What are your thoughts about these? Will they be present in FOTA? (more leading part)

Expected problems:

- Difficult to document and have enough tracing to comply with ISO standard
- Difficult to understand agile way of working (focusing on RE, which parts are difficult?)
- Customer contact difficult (specifically requirements engineering)
- Interface to other internal departments

What practices are you using or intend to use to solve these challenges? Thoughts - did they work, will they work?

Practices we have observed are being used or will be used. Thoughts - did they work, will they work?

- Documentation and tracing system
- SAFe
- Wiki, training sessions
- Agile contracting

A.2.2 Interview two template

Template for interview round 2

What are RE reasons for the transition from waterfall into agile? (more open part)

- What do you think are the most important reasons of the agile transition?
- What problems with waterfall do you think will be solved by the transition?

We expect at some reasons for the transition from waterfall to agile. What are your thoughts about these? Will are be present in FOTA? (more leading part)

Expected reasons:

- Changing requirements
- Agile is preferred by the employees

Follow up questions from interview round one:

Is traceability more difficult because of agile?

Is the agile training needed, or mostly a formality?

Are you familiar with SAFe:s implementation roadmap? If yes, do you think it helps with the challenge that it's difficult to know where to start with agile development?

Do you think that the safety regulations make it more difficult than otherwise to get started with agile development?

A.3 First round of interviews coding

Table A.4: Table of interview code "Customer agreements"

Interviewee	Text coded with "Customer agreements"
I1	Yes, it may be a problem to work fully agile. We already see signs that it's avoided to speak agile to customers, you have to have buffers. You don't have that in agile, you work until it's done.
I2	The same complication in terms of deliveries, agile wants to have test cycles that are in sync with the implementation steps but if we deliver to an organisation that think waterfall, they can't get it with the next cycle in two months. In practice, our agile packages are queued up. The customer can be defined in different ways, but when we deliver something to be tested, we are in practice in a customer relationship, example of how it could die there... it depends, but correct, when you have a delivering organisation or customer, then it will be a problem. If you define the content of deliveries "every four weeks", agility disappears.
I3	One of the biggest, in waterfall you clarify all the requirements and freeze them, and if the customer comes with changes you can charge them for changing. Agile you start with a little bit and the customer can make further changes, for example the scope. We don't finish in time sometimes. Payment becomes problematic. You don't make a baseline where you have analysed the requirements first, not broken down but analysed. "This is what I've understood, how would you like it", then freeze. Waterfall, integrate software every ½ year so it fits. In agile sometimes you remove things until later stages. This could have influence on the time line. You don't make a complete plan, hard to discuss with customer sometimes. Also the customers. They want management and control, deadlines and approving deliveries. Especially since we're in the automotive industry.

Continued on next page

Table A.4 – continued from previous page

Interviewee	Text coded with "Customer agreements"
I4	Maybe agile is better, the customer has more opportunity to change things, for better or worse. It could also be misused.
I6	An environment not working agile, the customer comes from outside and their way of working and handling suppliers leads to customer projects beginning to work in that way and it affects the platform. That's something you have to resist, especially when it comes to requirements engineering. We deal too much with the customer and too little with our own infrastructure first, we need to fill it, we're behind schedule, hard to act towards the customer, we can only react. Front loading, having something to show the customer. The market is customer-driven, the supplier develops according to the customer's demand. Automotive is different, you agree with the customer, it's very non-agile to negotiate a requirements specification with payment and everything [before starting the development].

Table A.5: Table of interview code "Supplier agreements"

Interviewee	**Text coded with "Supplier agreements"**
I1	Especially if they ... they work agile, but to specify the task to their agile team they still think waterfall, they must have everything ready before the handover. We are seeing resistance right now to work agile. "Can't start working if no baseline". They will probably have to change, but a problem right now.
I2	The complications come in interactions with other business units, parts of the company that have been working non-agile for a long time. If a team gets a completed specification to implement, we can't take it bit by bit. That difficulty, the difference, I have seen a couple of times. They must base their code on something. I've met people that when they hear "agile methods" it's "no not that shit, just a way to motivate mess." That may happen if the stool is missing legs.
I3	Same problem as before, if they don't understand the working procedure.
I6	We have suppliers too, some of the teams are in other units at Bosch. There are units saying they won't start working until they have an entire requirements specification and system description, both architecture and requirements. Interfaces to other units, both upwards and downwards.

Table A.6: Table of interview code “Functional safety difficult with isolated functionality”

Interviewee	**Text coded with “Functional safety difficult with isolated functionality”**
I2	It’s cross-cutting concerns, to know the solution you have to specify it. Then you need to know much about the system, there’s a tendency to fall back into a waterfall way of working. The biggest problem, to create those small V:s, where you can isolate some given piece of functionality and specify it well enough from a functional safety point of view so it becomes relevant. A certain function may depend on different hardware, if you come from up above and want to develop it separately as an entity, not so easy. If you have to specify everything you’re not agile anymore. I guess that’s where SAFe comes in, if you can create small V:s... still a problem. Same with safety.

Table A.7: Table of interview code "Difficult to begin"

Interviewee	Text coded with "Difficult to begin"
I1	Struggling with earlier is how to start. Agile is based on the assumption that there is a priority backlog with a team up and running. When we get a new set of requirements there is no backlog to put them in. How do you get started? Have not found, been googling, books. "Do not start agile projects until priority backlog", but how do you start? If you have the V-model you break down the requirements, but in agile is not so well described. Building a roadmap works, but it requires that you are up and running and all features are estimated in all backlogs. The roadmap says nothing because there is no anchorage, you can update all the time.
I3	[Safety regulations] It's possible I don't have a complete view of agile but I don't see the connection there. For me, agile is a method of planning. Safety gives me clear regulations and requirements for traceability. More on work product side. Additional work topics pop up as features. Don't think there is so much impact from safety regulations. Traceability is important. Defined outcome.
I5	The problem we have here is that agile assumes that you make small improvements to what you already have, but we have nothing yet, so how do we get started? The first step becomes much more than you imagine at first, later it's getting more agile, but at first it will be waterfall, otherwise we'd never get anything done. To get started with anything, having a product to work agile with. The first step is bigger, then we can work with small iterations.

Continued on next page

Table A.7 – continued from previous page

Interviewee	Text coded with "Difficult to begin"
I6	To start somewhere at all. We know that we need related lined artefacts, to finish the product, but to make people keep that in mind...
	Think all things make it harder, factors within the company. More themselves ... that the whole project is highly distributed, so many involved. Hard to reach out to everyone with that intention, especially when not located in the same place. That decisions are not taken on time, you do not get started as you intend to do. We have been delayed due to important old projects. More such factors. It probably did not matter with safety requirements, not seen so far, maybe it will be later. Difficult to communicate and implement.
	Think it's well-reasoned. I miss some things, a little bit on the tool side. Otherwise, it is meaningful. But when you look at it, it is said nothing about ... can imagine that other people perceive their personal area in the same way ... but at least for me, it would help to know how to get started, at what times you should start incrementally work towards an optimal set of tools.

Table A.8: Table of interview code "Keeping requirements updated and traceable"

Interviewee	**Text coded with "Keeping requirements updated and traceable"**
I1	To solve traceability ... system requirements and customer requirements, you have tools to solve traceability, eg Doors, you must have tools and follow the regulations. Agility will not succeed there either. It doesn't have to be like that, you only need to do things another way. Just like we use tools for static checks. We use tools so we don't have to waste time on it. Our hope is that tools can help us with traceability. The tools we have, they have the ability for that. Agile only means that traceability is built up iteratively, but when we're done the result is supposed to be the same. That can be done without tools too.
I2	The whole base for the problem I see lies in the need for traceability, which originates primarily from functional safety and security. Traceability is simple in itself, difficulty lies in tools, but the logic behind is quite simple. The problem is that if you have a complete system, it's easy to add traceability, but if you work with parts, it will be more complex.
I3	The way we make RE should be independent from traceability from my point of view. Testing only for a small piece of functionality. I can make traceability in the same time. Don't see a challenge other than traceability in itself, no matter waterfall or agile. Would like to say no... traceability is problematic by itself. Because it's additional work for the developers. If the tools can support it helps a lot. For agile... I don't see the biggest problem in the traceability. Traceability is a part of the work product, agile is a part of the planning. I've seen the same problems in formal [waterfall] topics. The requirements change no matter what, in the formal way it's more critical with changes, in agile you only have a subset, waterfall deals with more.

Continued on next page

Table A.8 – continued from previous page

Interviewee	Text coded with “Keeping requirements updated and traceable”
I4	The biggest challenge is that the requirements will change all the time, it will have a great impact on testing. From my perspective: you write tests to cover requirements, many of them are automatic, the requirements are very volatile and the test case has to be changed. Vicious circle. You are always behind schedule with the test cases in order to keep up with the requirements. It must be possible to go back to the right point in time and say “at this time, the status was like this”. The code changes all the time, on a daily or weekly basis. Difficult to show exactly. I think it might be excessive, how much will it change in practice? We will make a final delivery anyway. It all depends on when the requirements are done. That probably doesn’t get easier with agile, don’t know if it gets so much more difficult either. Before, you get requirements and connect developers to them.. with modern tools you can connect everything, it’s not easier or harder, not much extra work. Different type of work? More iterative process instead of once. New part, mapping, new part, mapping... You have to redo it in chunks. Even before, there came new requirements and then you had to sit and redo the mapping. Maybe it was a bit easier before, then you received a bunch of requirements and developed them later, at least in theory that was easier.
I5	I think some types of working methods are independent of waterfall or agile. I think that if you work agile, you do not do so many requirements at a time, but you still do system design for it. Had we worked in waterfall we had many requirements at once. So I agree with that [traceability is not more difficult because of agile]. That job needs to be done anyway.

Continued on next page

Table A.8 – continued from previous page

Interviewee	**Text coded with "Keeping requirements updated and traceable"**
I6	What has been neglected and not adapted to the agile way of working is making sure to synchronise the requirements. User stories, customer value, hard to maintain. The requirements tool may feel excessive. But I see that we need "the V", the product consists of the requirements tree as well, being fully developed. That you can understand in what way a certain feature will exist, which customer requirement is behind it. The product consists of 95% of other than code, but people focus on code. To break that focus on all levels, developers and managers should be aware and allocate resources. Agile can be a little aggravating circumstance, but I think if you stick to ... if you do it properly, you take the time to discuss a story and have the definition of done, traceability is built-in. It's a question about mindset and team culture. It is more in that direction. Then it really does not matter. If you do not make sure that you do everything that happens every time you do a particular work product, requirement or something, code to be tested, same problem with both methods. You can not go for what tool you prefer, it must meet certain requirements, it limits the range of tools that can be used, thus affecting how agile you can work. Influences mostly indirectly. One can make traceability on paper too, but not effective, difficult to get metrics.

Table A.9: Table of interview code "Understanding the SAFe practices"

Interviewee	Text coded with "Understanding the SAFe practices"
I1	How we map requirements against user stories.
I2	No, people have a lot of experience working agile here. They are more used to it than to waterfall really. Falling back to waterfall is often a reaction to uncertainty. Before people learn to understand what a burndown is, they want to know when it's ready.
I3	Of course not everybody understands what benefits and changes there are. Many of the guys are coming from former agile environments, Sony etc. For the leadership in Lund we had a SAFe training, also the management in Lund is quite trained.
I6	Thinking in terms of user stories is difficult. Not for me, but I see that many have problems with it. When I see how features are described in words... big user stories... difficult to get that way of thinking into people's minds, everyone thinks they're working agile or that they have, but you realise that's not really the case.

Table A.10: Table of interview code "ALM and other tools"

Interviewee	Text coded with "ALM and other tools"
I1	We have to have tools that you have in waterfall. There is no conflict between agility and traceability, but once done, traceability must be there just like "the code must be there", but the puzzle ... you must be able to show why and when you made the changes, it's just a matter of change . We have several tools, it's a matter of planning and displaying, I believe in the tools we have.
I2	If you can show progress in any tool, you can lower the uncertainty. That's what on the tool side I think there are things to do. Yes, it definitely does, but it's not just about traceability, but about tracing over variations. There are tools for this too, pure variation, for example, Bosch will use it, but it's not incorporated into CLM.
I3	I've seen in the past when we worked with DOORS etc., linking is quite easy. Going to another tool is always a struggle, linking and copying, a lot of work. One reason why most developers have forgot how to do it. With ALM there will hopefully be a more natural way of working. Traceability will hopefully be easier
I4	- ALM can link pretty good but sometimes it's hard to know if you're running the latest version. Now we're putting things directly into git, bitbucket, but of course it would've been better if you could comment every modification in ALM. But their review processes are quite strict, there's no good tool right now. I want my comments checked and marked with date, but haven't found any good tools. Missing in ALM. Maybe we were planning to use something else, don't remember, but I guess we'll find out. If it works it's great. But in which direction you link, from requirements to test cases or the other way around, can be discussed, we have different views on it. But it makes things clear, you can't miss requirements, everything needs to be mapped to get full coverage which makes it easy in theory.

Continued on next page

Table A.10 – continued from previous page

Interviewee	Text coded with "ALM and other tools"
I6	On the tool side, people are... we're trying to get it into the training there is. I feel that more time should be spent on it. It's underestimated. Training is needed. The tools offer way too many options, on the other hand, people have different needs.

Table A.11: Table of interview code "Agile contracting"

Interviewee	Text coded with "Agile contracting"
I1	If you have entered into formal waterfall contracts, it's difficult to get into agile, you have to hide it behind something else, in order not to break an agreement. There are customers (but have not seen them here yet) who want to go towards agile contracting. Until then, we should be careful about completely agile, the customer doesn't understand. The teams must understand the dependence of another team, then you can negotiate the sprints and deliveries. Then you can create a base for working with non-agile teams, should be work with SAFe with incremental plans.
I2	Our agile packages are queued up in practice. "Agile contracts". The customer can be defined in different ways, but when we deliver something to be tested, we are in practice in a customer relationship, example of how it may die... it depends, but correct, when you have a delivering organisation or customer, then it will be a problem. Should look into agile contracts. The better results we have from agile processes, the more we can sell it to organisations around us. Either command from the top, now everyone should work agile, or influence each other. However, if the agile is to adapt to waterfall, pull the legs off the stool. It will be a bit of a war, different parts affect each other.
I3	Only way is... train the other side, hard with customer. Another project we have done we used some mixed procedures. We showed customer a waterfall fall, not completely but that we have planned over the complete time. "We will do these sprints over time." Only possibility here is to show to the management the benefits of working agile. Make them interested, and they will see what they can do. You need to work deeply together with the customers. I heard that someone started to try this with Volvo. Some laws were problematic, dependency problem... I don't know exactly, second hand info.

Table A.12: Table of interview code “Agile prototype with incrementally added complexity”

Interviewee	**Text coded with “Agile prototype with incrementally added complexity”**
I1	Now we have found a way and that is, we must have an initial phase where architects drive it forward.
I2	The more you can work with prototypes, in terms of systems design, the easier you can actually do the development work agile. If you have a small group of people who control the systems design and produces a prototype that meets any part of this, and apply the proper system to this skeleton, then you can get agility. The first bit becomes agile because it’s independent, the other of less uncertainty. Different dimensions are used 1. Reduce number of organisations which may have demands to a small entity. 2. Perform and do a series production, must make a proper systems design with real software components. “From the first we know this, do not need to iterate through all organisations but can clearly specify some things already at the system level.” The people gain experience and learn from the start, and so they can help reduce uncertainty in the next phase.

Table A.13: Table of interview code "SAFe training and tutorials"

Interviewee	Text coded with "SAFe training and tutorials"
I1	The developer teams need to have seen how it works. For the team it's still the same building blocks, what's introduced every fourth, sixth, eighth week or so is increment planning sessions. There you might have to train the team and prepare them. If you look at other parts it's more like training certain individual roles, for example the product owners. Important to put focus there. Mindset changes.
I3	Many of the guys are coming from former agile environments, Sony etc. For the leadership in Lund we had a SAFe training, also the management in Lund is quite trained. Minimum for product owner and scrum master, training is mandatory, otherwise a lot of pitfalls, many problems. If we go further to SAFe the teams are more involved in the planning, they need a greater understanding. For the teams, SAFe training is not mandatory but strongly recommended from my side, for a wider perspective.
I4	I think that's implied. I've taken courses in scrum and kanban etc, but not the developers per se. But that will probably not be a problem, they get their tasks and no major difference. It's that logical, you get into it soon. I think it's super important. Everyone should take an agile course, even those who think they know, there are always some things you don't know, or science has progressed and there has been new findings. I read through the SAFe-books and there were many new things for me as well. You shouldn't believe you know everything. There are things you don't agree with, but you can choose the parts you like, or the parts that the bosses force down your throat.
I5	I think that one day of SAFe training is pretty good, but you should not immerse yourself too much, it will be too abstract. I miss having concrete examples in such courses. I went for a day and a half on SAFe training, I think it's enough. We're going to use ALM ... it's more concrete. If you do not have the concept of backlogs, it's a bit harder, then you need some kind of intro.

Continued on next page

Table A.13 – continued from previous page

Interviewee	Text coded with "SAFe training and tutorials"
I6	It's a good thing we took this 'leading safe' course, we're trying to implement SAFe according to the implementation roadmap. Start with management training etc... limited in practice because customer projects are coming in from all directions. On the tool side, people are... we're trying to get it into the currently existing training. I feel that more time should be spent on it. It's underestimated. Training is needed. Incredibly important. I always notice that I have had a lot of benefit from having it sorted. Linking quite a lot. All levels must have the appropriate knowledge so that the developers and management have basic knowledge and use the same terminology. Otherwise, it means one thing and the other understands something else ... Also create acceptance for certain types of artifacts, backlogs, what they are good for, different phases. If you do not have that understanding you are lost. Learning it takes a long time by yourself, but now you get it at regular doses. Great learning curve in the beginning and early usefulness.

Table A.14: Table of interview code “Waterfall in the beginning”

Interviewee	Text coded with “‘Waterfall in the beginning”
I2	Being able to do the requirements engineering work, I think there will always be some purpose with waterfall in the beginning.
I4	More difficult, agile-waterfall. There will come some things you can’t work agile with. [Draws on the whiteboard] two axes, fix-flexible two-dimensional scale. The bottom left corner is not possible to change. That part is not possible to do in a very agile way. The other parts you can do as agile as you want. But for the safety you can’t change the fix-stable. Then it’s really hard to have an agile comprehensive view. We are locked in by the safety parts. In order to follow the regulations, you can’t iterate over new requirements.
I5	The first step becomes much more than you imagine at first, later it’s getting more agile, but at first it will be waterfall, otherwise we’d never get anything done.

A.4 Second round of interviews coding

Table A.15: Table of interview code "Handling changing requirements"

Interviewee	Text coded with "Handling changing requirements"
I1	The work is done regardless, but a change in how we work with requirements. You do not need a finished plan in detail, because on day two something turns up and everything changes. Now we will not do that.
I2	Managing stakeholders which are constantly changing. From the beginning one customer in focus. Suddenly who's in focus changes.
I3	With waterfall plan you know what is coming in six months, you make frequent re-plans. In agile you concentrate on the first parts and ignore later parts. We will be more precise in formulating what we are doing. More correct. I know some customers that are quite flexible and change requirements. Other customers are more structured in this case. If they develop a vehicle they sync every part, define communication functionality... But could be that it changes a bit all the time. This is where it gets problematic, the customers would like to have integrated change but also know from the beginning. Also the customers. They want management and control, deadlines and approving deliveries. Especially since we're in the automotive industry.

Continued on next page

Table A.15 – continued from previous page

Interviewee	Text coded with “Handling changing requirements”
I5	At Bosch I’ve actually no idea, but based on my previous experience it can lead to more predictability. You might not know exactly what you get but at least you get something. It’s a reason to be agile if the customer has new requirements. But is that really the case in our world, in automotive? Is there so much change in requirements? It’s rare that the waterfall plan can be kept, you believe you have it under control but often... maybe it’s easier in smaller projects, but in bigger projects you believe something from the beginning but then you see that’s not how it went at all.
I6	I also see that there are parts, if you look into the Stacey matrix, some customer projects where we don’t really know what it will look like. Both the customer requirements... it all develops and gets further in and becomes very concrete. I guess it’s both that and also that we’re not really clear about how the technical solution will look in each case. You have a lot of different... every customer solution will look a bit different. To fill a platform, it’s only natural it’s done incrementally. Some things take time, and some things we’ll need later. ...ask for approaches more towards the agile stuff, so we can steer and not plan in full detail what will come later. I’ve worked much with planning before, it’s a bit of a catchword that ’planning is replacing uncertainty with error’. Some of the customers don’t know at all, some have a vague idea.

Table A.16: Table of interview code "Motivating the employees"

Interviewee	Text coded with "Motivating the employees"
I1	Create motivational factors for the employees. If the development team can make a decision, they should do that, it raises motivation for everyone.
I3	The benefit of agile from my point of view is you can easily work in a team. Working together. The developers are really strongly interested in working agile, to improve the teamwork. Agile is moving everything towards the team side.
I4	It's more fun. Developers have more control, maybe not more say but more influence over development. It's not 'code this, end of story!'. You should give them more responsibility, many developers want it, this is important, you see more of the big picture.
I5	If you set it up the right way and do it the way it was theoretically intended, there will be a bigger commitment from everyone in the project. Creativity and the feeling of involvement. What is requirements engineering in agile project, maybe a bit more distributed? More out in the teams, that's the feeling I get. It could be that way. I think that if you're a developer you feel more involved and motivated in an agile project, instead of someone else making the design before and shove it down your throat. 'Start coding!'. Many have experience from other companies of working agile, of that way of working.
I6	It's mostly from down below [developer level], people think it's fun, [that is] my feeling.