Empir Software Eng (2018) 23:2550–2596
https://doi.org/10.1007/s10664-017-9555-8

Large-scale agile transformation at Ericsson: a case study

Maria Paasivaara[1] · Benjamin Behm[1] · Casper Lassenius[1] · Minna Hallikainen[2]

Published online: 11 January 2018

Abstract Many large organizations are adopting agile software development as part of their continuous push towards higher flexibility and shorter lead times, yet few reports on large-scale agile transformations are available in the literature. In this paper we report how Ericsson introduced agile in a new R&D product development program developing a XaaS platform and a related set of services, while simultaneously scaling it up aggressively. The overarching goal for the R&D organization, distributed to five sites at two continents, was to achieve continuous feature delivery. This single case study is based on 45 semi-structured interviews during visits at four sites, and five observation sessions at three sites. We describe how the organization experimented with different set-ups for their tens of agile teams aiming for rapid end-to-end development: from component-based virtual teams to totally cross-functional, cross-component, cross-site teams. Moreover, we discuss the challenges the organization faced and how they mitigated them on their journey towards continuous and rapid software engineering. We present four lessons learned for large-scale agile transformations: 1) consider using an experimental approach to transformation, 2) consider implementing the transformation step-wise in complex large-scale settings, 3) team inter-changeability can be limited in a complex large-scale product — specialization might

Communicated by: Hakan Erdogmus

✉ Maria Paasivaara
Maria.Paasivaara@aalto.fi

Benjamin Behm
Benjamin.Behm@aalto.fi

Casper Lassenius
casper.lassenius@aalto.fi

Minna Hallikainen
minna.hallikainen@ericsson.com

[1] Department of Computer Science, Aalto University, P.O. BOX 19210, FI-00076 Aalto, Finland

[2] Oy LM Ericsson Ab, Hirsalantie 11, FI-02420 Kirkkonummi, Finland

be needed, and 4) not using a common agile framework for the whole organization, in combination with insufficient common trainings and coaching may lead to a lack of common direction in the agile implementation. Further in-depth case studies on large-scale agile transformations, on customizing agile to large-scale settings, as well as on the use of scaling frameworks are needed.

Keywords Agile software development · Large-scale agile · Adopting agile · Enterprise agile · Scaling agile

1 Introduction

Increasing pressure to reduce cycle time, improve quality, and swiftly react to changes in customer needs are driving companies, large and small, to adopt agile software development (VersionOne 2016). Agile development can improve efficiency and quality (Livermore 2008a), and enable shorter lead times and a stronger focus on customer needs (Petersen and Wohlin 2010).

Even though agile software development methods were originally designed for single, small teams, during recent years, large organizations have increasingly adopted them (Hossain et al. 2009; Larman and Vodde 2010; Leffingwell 2007). A recent systematic literature review (Dikert et al. 2016) revealed the lack of systematically conducted studies on large software development organizations adopting agile methods. The review identified only six scientific studies on large scale agile transformations, as almost 90% of the included papers were experience reports written by practitioners. According to the State of Agile Survey (VersionOne 2016), 43% of the self-selected respondents worked in development organizations having more than 50% of teams using agile, while only 4% of respondents stated that none of their teams were agile, and 62% of almost 4000 respondents came from an organization with over a hundred people in software development. While the survey is non-scientific, and problematic from a methodological point of view (Stavru 2014), it is the largest reoccurring survey on agile adoption, and it indicates that a significant number of big organizations use agile. Moreover, practitioners at the XP conference in 2010 listed the topic "Agile and large projects" as the number one top burning research question (Freudenberg and Sharp 2010). In recent workshops on large-scale agile development, the introduction of agile methods was one of the highlighted themes needing more research (Dingsøyr and Moe 2013; 2014).

Large organizations often have big projects executed by large and distributed development organizations, requiring agile methods to be scaled. According to (Leffingwell 2007), scaling involves many challenges, including coordination between several agile teams, lack of up-front architecture, lack of requirements analysis, as well as all the challenges of distributed projects, as many large organizations are distributed. Despite these challenges, many large companies have chosen to adopt agile methods, even though research on how to scale agile methods to large-scale projects (Hossain et al. 2009), and on successfully conducting agile transformations in large organizations is largely missing (Dikert et al. 2016).

The purpose of this paper is to start filling the gap in the literature on large-scale agile transformations. We investigate how one large-scale R&D product development program within Ericsson adopted agile methods at scale. We present the motivation for the transformation, the steps taken, the challenges encountered, as well as the mitigating actions taken to tackle the challenges.

The case organization was a new R&D product development program at Ericsson developing a XaaS[1] platform and a set of services. Ericsson's customers, telecom operators, can provide a number of services to their customers using the platform.

The development organization wanted to develop the capacity for continuous delivery (Rodríguez et al. 2016). As a step towards that goal, the organization adopted agile methods (Schwaber and Beedle 2002). The planning of the agile adoption started in late 2012 and the full-scale roll-out took place during 2013. By spring 2014, the development organization had grown from two team at the end of 2011 to 15 development teams, distributed to five global sites. Thus, this can be viewed as a large-scale agile adoption according to the definition used in (Dikert et al. 2016), which states that large-scale agile is *software development organizations with 50 or more people or at least six teams*.

In our previous work, we presented the initial results of the transformation (Paasivaara et al. 2014a) and how the case organization had used Value Workshops as to facilitate organizational alignment during the transformation (Paasivaara et al. 2014b). This paper elaborates on and extends the previous papers by presenting an in-depth analysis of the case, including an additional research question (RQ1), a more detailed description of the research method, with an additional validation interview, a significantly extended results section going deeper into the results, and a completely new discussion section.

The paper is structured as follows: Section 2 provides an overview of the previous literature, Section 3 describes the case background, research goals and methods, Section 4 presents our results, Section 5 discusses the results, and finally, Section 6 concludes the paper.

2 Related Work

In this section we present relevant previous work. First, we explain what we mean by large-scale agile software development, and why it is important to study. Second, we discuss why large organizations are interested in large-scale agile, as well as challenges and success factors of the transformations.

2.1 Large-Scale Agile Development

Agile methods were originally developed for small organizations, and despite success stories, large-scale application has proved challenging (Dybå and Dingsøyr 2008). Challenges in large-scale agile adoptions relate partly to organizational size bringing inertia, which slows down the change process (Livermore 2008b). Another challenge is the need to interface with and integrate existing processes and organizational structures (Boehm and Turner 2005).

Agile methods focus largely on intra-team practices, which work well in small organizations. A challenge in large organizations is that it is necessary to coordinate and communicate between several development teams, and also between different organizational units. Agile methods provide little guidance on how agile teams should interact with the environment at large. Because of this, large organizations must tailor the methods to fit

[1] XaaS: "anything as a service" or "everything as a service" The acronym refers to an increasing number of services that are delivered over the Internet rather than provided locally or on-site. (Banerjee et al. 2011)

their specific needs. As a consequence, practices requiring additional formal communication may need to be put in place, which might reduce agility (Lindvall et al. 2004).

Large organizations are often globally distributed, which brings the need to apply agile in distributed projects. During recent years agile practices have gained a foothold in global software engineering projects, and there is evidence of benefits of agile use (Hanssen et al. 2011). However, agile methods are largely based on frequent internal and external collaboration and communication (Highsmith and Cockburn 2001), and such close collaboration is inherently challenging in global work, which complicates the use of Agile in global software engineering (Hanssen et al. 2011). On the other hand, Agile has qualities that brings remedy to the challenges caused by distance in global work. Suitable agile practices may bring distributed sites closer each other by improving coordination and communication (Holmstrom et al. 2006).

During recent years frameworks for scaling agile software development have been suggested by several consultants, e.g., the Scaled Agile Framework (SAFe) (Leffingwell 2015), Large-Scale Scrum (LeSS) (Larman and Vodde 2015), and Disciplined Agile Delivery (DAD) (Ambler 2012). However, documented experiences on the usage of these frameworks is still scarce, e.g., how they are used, to what kind of circumstances they are best suited, and what the challenges and success factors of their usage are. The State of Agile Survey (VersionOne 2016) shows that a large number of companies seem to be using some framework, as 27% of the respondents reported using SAFe, 6% LeSS and 4% DAD. In addition, most respondents (72%) stated using Scrum or Scrum-of-Scrums to help to scale (respondents could make multiple selections).

A recent systematic literature review on large-scale agile transformations (Dikert et al. 2016) reported that only six, or 12% of existing 52 reports were scientific. Most of the selected papers were experience reports published in the XP and Agile conferences, showing practitioner interest in the topic, and that academic research is lagging behind.

None of the scientific studies included in the systematic literature review by (Dikert et al. 2016) focused directly on the transformation process, even though they briefly described it. Two of the papers (Abdelnour-Nocera and Sharp 2007; 2008) reporting on the same case concentrated on the effects of the agile transformation. A study about Ericsson R&D Finland (Rodríguez et al. 2013)[2] focused on how Lean thinking is implemented, however the focus was mostly on the current state instead of the transformation process. A paper from Nokia Siemens Networks (Korhonen 2012) studied whether the visibility, reaction speed, quality, or motivation had changed, comparing the situation before and after the transformation. (Murphy and Donnellan 2009) studied the good and bad aspects of communication during an agile transformation and (Vlaanderen et al. 2012) in their multiple-case study of two cases analyzed the Scrum introduction paths. While evaluating the relevance of each of the research papers regarding how well they provide information on large-scale agile transformations on scale a 1-5 (1: some sentences revealing factors relating to transformation, 5: the entire paper focuses on describing how the transformation proceeded). Dikert et al. (2016) gave two of the papers (Abdelnour-Nocera and Sharp 2007; 2008) (reporting on the same case) the rating 3, while the rest received only either 1 (one paper) or 2 (3 papers). This reveals how the current research on large-scale agile transformations is lagging behind the state-of-the practice.

[2]A different case project from ours

2.2 Motivation to Initiate an Agile Transformation

According to the literature one of the top reasons for a large software development organization to start an agile transformation was to reduce the time to market (Gat 2006; Goos and Melisse 2008; McDowell and Dourambeis 2007; Prokhorenko 2012; Silva and Doss 2007), as the competition and market situation was changing towards speedier deliveries (Greening 2013). Companies want to improve their competitiveness, or even fear that they are losing competitiveness. Thus, their response is to improve delivery speed and responsiveness to change.

Another significant motivator was software project management related reasons. Many companies had experienced problems related to project management (Long and Starr 2008), people management, and managing schedules (Chung and Drummond 2009) that they were hoping to correct.

The old process and the whole way of working was considered problematic due to overhead, seen in extra bureaucracy causing needless costs in the form of unproductive meetings (O'Connor 2011), process gates (Chung and Drummond 2009), change management overhead (Vlaanderen et al. 2012) and excess documentation (Hansen and Baggesen 2009; Murphy and Donnellan 2009). Slow processes with long cycle times led to late feedback (Beavers 2007; Ranganath 2011).

2.3 Challenges and Success Factors of Large-Scale Agile Transformations

Any organizational transformation that involves numerous individuals will face challenges. A systematic literature review (Dikert et al. 2016) identified 29 success factors for large-scale agile transformations grouped into 11 categories and 35 challenges in 9 categories. The review identified the following main challenges for large-scale agile transformations: other functions unwilling to change (mentioned by 31% of the reported cases), lack of guidance from the literature (21%), reverting to the old way of working (19%) and misunderstanding agile concepts (19%). The top challenge categories mentioned were agile difficult to implement, integrating non-development functions, change resistance and requirements engineering challenges.

The most salient success factors identified were: coaching teams as they learn by doing (29%), ensuring management support (29%) and customizing the agile approach carefully (26%). The top success factor categories listed are: choosing and customizing the agile approach, management support, mindset and alignment, and training and coaching.

The State of Agile Survey (VersionOne 2016) reports the following tips for large-scale agile transformations: consistent process and practices (mentioned by 43% of respondents), implementation of a common tool across teams (40%), agile consultants or trainers (40%), executive sponsorship (37%), and internal agile support team (35%).

3 Methodology

3.1 Background

This paper uses a single case study methodology (Yin 2009) in a software development organization at Ericsson developing a XaaS platform and a related set of services. We subsequently refer to this whole as the "product".

The product provides a set of services to business customers, who use it to provide services to their clients. Originally, the platform was designed for a single customer. At the

time of our interviews, the product was in its early life-cycle with tens of customers, the number of which was expected to grow rapidly, and the product was considered to have a vast market potential.

Architecturally, the product consisted of modules, subsequently referred to as components. Some components were developed by third-parties and some by Ericsson. The development of the components required highly specialized expertise due to their complexity.

Ericsson acquired the product in 2011. Before the acquisition, approximately 30–35 people, including external contractors and consultants, developed the whole platform. As part of the acquisition, Ericsson hired around ten domain experts from the previous development organization and took over the further product development. Directly after the acquisition, the newly built organization had to focus on knowledge transfer from the external consultants to Ericsson's employees and to the newly hired consultants.

The development organization at Ericsson grew rapidly: from two teams at the end of 2011 to 10 teams in spring 2013 and 15 teams by the spring of 2014. New developers and teams were added to the organization gradually. The biggest increases happened in late 2012 and during 2013. In the fall of 2012 an external consultancy provided personnel to the project (at site E), and both internal and external recruiting was done. During the summer and fall 2013, five agile teams were added to Site A, part of which were reassigned from another project at Ericsson.

During this time, the size of organization increased from a few dozens to around 200. In spring 2014 the development organization consisted of agile teams (typically consisting of 7-9 persons), Product Owners, architects, agile coaches, line managers, product managers and other managers. In addition, the organization included sales personnel, and customer support and operations.

At the time of our data collection (Fall 2013 - Fall 2014) the development organization was distributed to five sites in three countries as illustrated in Fig. 1. Four of the sites were in Europe (sites A, B, C, D) and one (site E) in Asia. Site E was a subcontracted site, not Ericsson's own. In addition, customer support and operations were located at a sixth site (site F), which was not considered as part of the development organization, and was therefore not included in this study.

After the acquisition, when moving the development to Ericsson, experts on specific components were hired both internally and externally to several sites. As each component

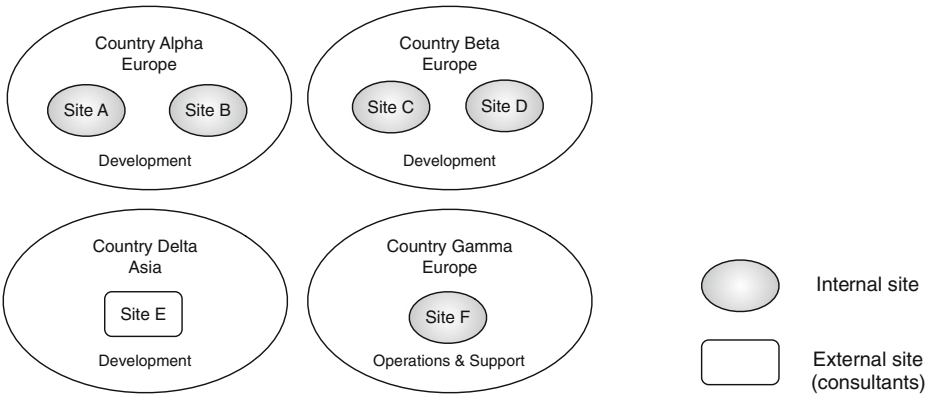

Fig. 1 Project sites and distribution at the time of our data collection (Fall 2013 - Fall 2014)

required deep expertise, learning new components takes a lot of time. The competences for each component were in many cases not located at a single site, but distributed to several locations. Moreover, a single feature could span several components, requiring different expertise to develop, see Fig. 2. Thus, matching features spanning several components to the component-based competences located at different sites provided significant challenges for rapid end-to-end development. This feature-component structure remained the same during the transformation, even though the organization structure around was changed.

Ericsson has traditionally used a plan-driven software process. However, during recent years the company started a global adoption of agile software development. The studied organization started its transformation, or "the agile journey", as they call it, in late 2012 and the first agile pilot team was formed in early 2013. This transition has been particularly challenging, as the organizational growth has been significant and rapid during the transformation, which is still ongoing.

3.2 Research Goals and Questions

Our research goal was to investigate how this large, globally distributed organization reorganized its development and processes by taking agile methods into use.

We purposefully selected this information-rich case (Patton 1990), as we had the possibility to gain access based upon participation in a joint research program, and we had previously studied another agile transformation in the same company (see (Paasivaara et al. 2013)), thus we knew that this case would provide us rich data on the studied phenomenon. We selected a revelatory case (Yin 2009), which enabled us to study a yet unstudied phenomenon. This case enabled us to study, over a longer period of time, how a large, globally distributed organization developing a complex product takes agile into use, the steps of the transformation, as well as challenges and mitigating actions. As discussed earlier, this is a topic that has not been studied scientifically almost at all, thus we saw this as a unique research opportunity. The case setting provided us with access to an industrial real case setting in a rarely studied empirical context and allowed us to follow the transformation over a period of time, thus proving us a unique dataset.

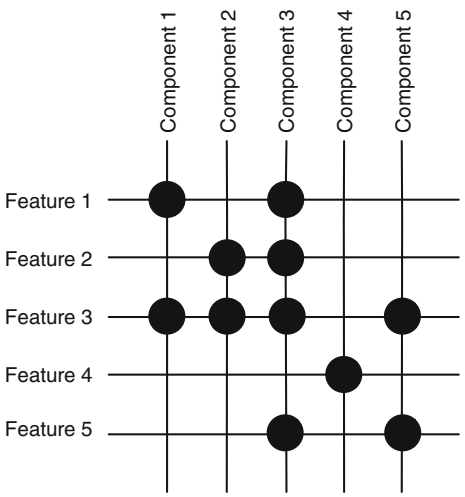

Fig. 2 Features vs. components

We posed the following research questions:

RQ1: Why did the organization initiate an agile transformation?
RQ2: How did the transformation proceed?
RQ3: What challenges did the organization encounter?
RQ4: How did the organization mitigate the challenges?

3.3 Data Collection

The data collection took place between September 2013 and September 2014. We used three sources of data: 1) interviews, 2) observations, and 3) company internal documents.

The first three authors collected the data together. The fourth author, a representative of the organization, was our main contact person during the study, as well as one of the key informants. She helped us select the interviewees and arranged access to the events we observed. She also validated the findings of this paper by reading and commenting on the paper draft. Figure 3 shows the data collection timeline.

3.3.1 Interviews

We conducted a total of 45 semi-structured interviews in three rounds: 1) 31 interviews on the transformation journey, 2) twelve interviews on value workshops (which were one of the major steps during the transformation journey described later on), and 3) two validation interviews after analyzing the data.

The goal of the transformation interviews was to study the large-scale agile adoption. During that first interview round we conducted 31 interviews of altogether 34 persons at four sites.

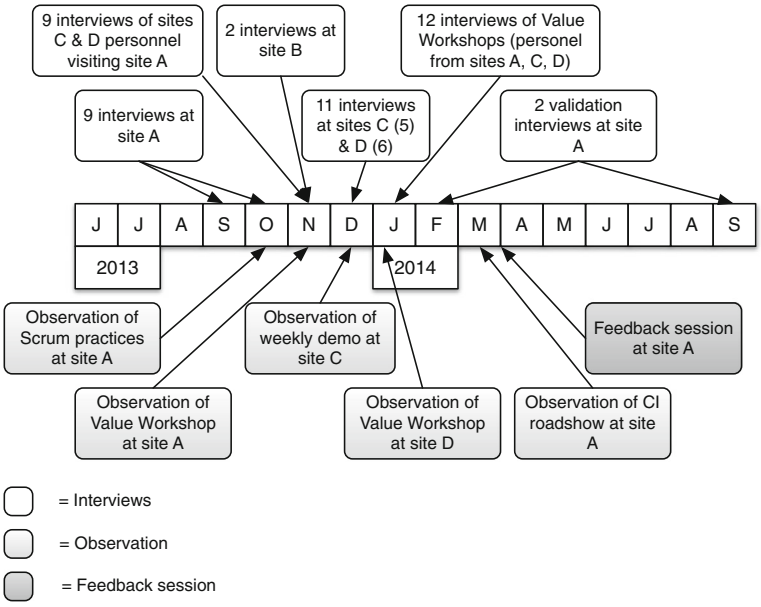

Fig. 3 Timeline of the data collection

The roles of the interviewees included development team members (i.e., members of agile teams such as developers and testers), Product Owners, coaches and managers. We aimed to interview a broad representation of the organization, talking to informants in different roles, with various backgrounds and representing different organizational levels in order to gain as complete a view of the situation as possible. We mainly selected persons with long experience with the organization to be able to reflect the whole transformation journey, but also a few persons joining later on to give us another perspective. Many of the interviewees had a long background at Ericsson. About half had joined Ericsson over ten years ago. 2/3 of the interviewees had joined the studied case project over a year before our interviews and only 1/3 had less than a year of experience from the case project. A bit more than half of the interviewed persons had a background in agile methods before joining the project, and around half of them had transferred to the project from the first agile project at site A, reported in (Paasivaara et al. 2013). All interviewees were selected with the help of case organization representatives. The interviews typically lasted one hour, but the length ranged from half an hour to two hours. Especially for the few first interviews, we reserved more time, as we asked more background questions in order to understand the history of the organization and the starting point of the transformation. These early interviewees were managers and coaches, who had a broader overview of the organization. The subsequent interviews were focused on the transformation and were somewhat shorter. During the first interview round, two researchers participated in all interviews, one being the main interviewer (Author 1, in a few interviews Author 3) and the other one taking detailed notes, as well as asking additional questions (Author 2).

During the first interview round we learned that the organization had started to define common values, and would be working further with these values in workshops. The common values and the related value workshops were an important step during the transformation journey, which was the reason why we decided to study them further. We had a possibility to participate as observers in both workshops and, after the second workshop, interview 12 participants from three different sites. This formed the second interview round. These interviews were short, ranging from 15 to 30 minutes each. The interviewees ranged from team members to managers. These interviews were conducted by a single researcher (Author 1), who selected the interviewed persons amongst the workshop participants.

During the third interview round two interviews were done to validate our results after we had analyzed the data. The first interview took place after we had analyzed the data from the first interview round and the second one after analyzing the second round interviews. The main purpose of these interviews was to deepen our understanding about topics that emerged when analyzing the data, as well as validate that we had understood particular issues correctly, e.g, the product structure. In the first of these interviews two researchers and two interviewees were present and in the last one, one researcher and one interviewee. These interviewees were selected as they had a broad view of the whole organization and were actively involved in the transformation in the whole organization in their roles as line manager, project steering committee member and organizational coach.

The number of interviews and interviewees differ, as in two interviews we had two interviewees and in another three. The multi-person interviews during the first interview round were due to interviewee time limitations. For example, we could have interviewed only one of the three coaches (who had been doing exactly the same work) and only one of the consultants (also working tightly together), but the interviewees suggested group interviews to which we agreed, as we thought it would give us a broader picture than conducting single person interviews. In addition, in the last validation interview we had two interviewees, with whom we checked our results and asked clarifying questions.

During the two first interview rounds we used an interview guide approach with predetermined topics as suggested by Patton (1990). The main topics were the same for each interviewee but the questions were adjusted based on their position and background. Table 1 shows the roles interviewed and the interview guides can be found in Appendix A and B.

All interviews were conducted face to face in the organization's facilities and recorded. The recordings were transcribed by a professional transcription company.

As part of this study, we visited all the European sites (A, B, C, D). Unfortunately, due to budgetary restrictions, we were not able to visit the Asian subcontracted site (E), but were able to interview one representative of that site who temporarily was located at site D. Except for that one interview, the data we have on site E are the descriptions given by people at sites A, C, and D who closely collaborated with that site. However, our results focus on the main internal sites (A, B, C and D), as this was where the large-scale agile transformation took place. Site E as an external site, was not actively included in the transformation, and the plan was to drop the site in the near future.

3.3.2 Observations

To support the interviews, we conducted five observation sessions of altogether 31,5 hours during seven days. The events observed were selected carefully to support our study: 1) to see in practice how the basic Scrum practices were implemented in the case organization we observed how a Scrum team performed the activities related to a sprint change: sprint review, retrospective and sprint planning. 2) To understand how the major coordination events worked in practice, we observed the weekly Product Owner meeting, as well as two common bi-weekly demos, where a team or teams who have finished something that might interest others demonstrated their work. 3) To follow major transformation events, we observed both 2-day value workshops (arranged at sites A and D) and one continuous integration (CI) roadshows (arranged at site A). As explained later on, common values and the related value workshops were one of the major transformation steps. They were

Table 1 Interviewees and their roles[a]

Role	Site A	Site B	Site C	Site D	**Total**[b]
Development team members	3 + 2	1	1 + 5	3[c]	**15**
Product owners	2	1	1		**4**
Architects			1	1 + 2	**4**
Coaches	3 + 1 + 1	1	3	3	**12**
Subsystem Responsibles	1			3	**4**
Line managers	3 + 2 + 2		1		**8**
Other managers			6	1 + 1	**8**
Total	**20**	**3**	**18**	**14**	**55**

[a]Transformation interviews + value workshop interviews + validation interviews

[b]The sum exceeds the total number of interviews, as a few persons had several roles (e.g. being both a subsystem responsible and a coach or a development team member), and a few persons participated in multiple interviews (e.g., persons participating in the validation interviews).

[c]One of these persons was from site E, but was working at site D at the time of the interviews.

Bold indicates that the columns and rows represent sums

organized to unite the globally distributed organization. Building the CI system and spreading the CI knowledge and CI mindset in the organization was another major transformation step. During the CI roadshow sessions the persons who had participated in building the first CI system presented the current situation of CI and the goals of CI to the other teams, as well as discussed current challenges in the area. Similar CI roadshow events were organized at three other major sites.

The first and second author conducted the observations as non-participants. During the breaks they discussed with the participants. The observers took detailed notes during the observation sessions on what happened, what was presented and discussed, who were present, and how the participants behaved. For confidentiality reasons, the observation sessions were not recorded, as during those sessions details of new product features were discussed. Such details were, naturally, highly confidential, and as a result we were not allowed to record the sessions. The information gathered during the observations was used to support and complement the interviews. Table 2 shows the details of the observations.

3.3.3 Documents

We received a number of documents from our interviewees, e.g., slides discussing the process, working practices, product and organization structure, as well as a fictional story called the "Showcase", created by the agile coaches together with the management team to describe how this organization would look like in two years. These documents were used to triangulate and complement the information received in the interviews.

3.4 Data Analysis

We analyzed the data qualitatively, using the Atlas.ti software package. We coded the data in six main categories: four main themes according to our research questions, and two context categories, i.e. organization structure and case background. The research question-based themes were motivation for the transformation, phases of the transformation, transformation challenges, and mitigations and success factors. We then proceeded with detailed coding, resulting in 605 codes, such as *Business flow definition*, *Daily Scrum*, and *Domain owner meeting participants*. Following this, we grouped the detailed codes into a total of 58 code families, such as *Development Practices*, *Coaching Community of Practice*, and *Cross-site teams*. The qualitative coding of the transcriptions of the first interview round was done by one researcher (Author 2), while two researchers (Authors 1 and 3) instructed and closely followed the process discussing together daily. The second round interviews were coded by one researcher (Author 1).

Table 2 Observations

Observed sessions[a]	Site	Duration	Observer
Scrum rituals, PO meeting, bi-weekly demo	A	7 h	Author 2
Value workshop 1	A	6 h + 3 h	Authors 1 & 2
Bi-weekly demo	D	0.5 h	Author 2
Value workshop 2	D	6 h + 3 h	Author 1
Continuous integration roadshow	A	6 h	Author 1

[a]In chronological order

3.5 Limitations and Validity

We discuss the validity of our research from four viewpoints: internal validity, construct validity, external validity and reliability (Yin 2009). The fourth type of validity, statistical conclusion validity, is not relevant to this study.

Internal validity concerns the validity of the causal relationships observed in the case (Yin 2009). As this is a descriptive case-study, we refrain from theory building, and the reported causal relationships represent the views of our subjects. The threat that this might not perfectly represent reality remains.

In case study research, construct validity concerns how well the description of the case represents reality. We interviewed people who were actively involved in the ongoing transformation. Therefore, it is likely that their views and recollections reflect reality as the events discussed were contemporary. However, there are always risks related to respondents' bias due to personal opinions or social pressure. The construct validity of a case study can be increased by the triangulation of data sources, investigators, theories and methods (Jick 1979; Yin 2009). We used several types of triangulation: we collected data by several methods, from several subjects and by several researchers. First, as it is not recommended to conduct a case study by relying on a single data source (Yin 2009), we collected data by three different methods: interviews, observation, and document analysis. Second, we interviewed a large number of subjects in different roles, with varying backgrounds, from different sites, and with differing length of experience in the organization to get as broad representation as possible. Third, the data was collected by three researchers, who all conducted interviews and two participated in the observation sessions. 32 of 45 interviews were conducted by two interviewers and one observation session, a two-day value workshop was observed by two researchers. All three researchers participated in data analysis and writing. Three different investigators collected and analyzed the data. We employed three data collection methods: observations, interviews and document collection. Our data sources included observation notes, interview transcripts, and company documents.

The external validity of research concerns the domain to which the research results are generalizable (Yin 2009). To help the reader to understand the contextual factors of the case organization, we have described the context in detail.

Reliability concerns whether different researchers had produced the same results if they had studied the same project (Yin 2009). The main threat to reliability in this case is the variability in data collection. We minimized this threat by involving several researchers in the interviews, and having the analysis results checked by both other researchers and company employees. This triangulation makes our results robust against threats to reliability (Jick 1979; Yin 2009). Most data collected converged between the investigators, methods and data sources and revealed no notable threats to the construct validity or reliability of our results.

After analyzing the data, we arranged a feedback session in March 2014 to validate our results. The feedback session took place in the site A team area with a videoconference connection to the other European sites: B, C, and D. The whole organization was invited to the session and around thirty people participated actively in the session. We received positive feedback: the organization had already started implementing some of the suggested improvements and would take into account our findings when planning the next improvement steps. No corrections to our findings were presented. Feedback we gave to the organization did not affect our results, as the session was organized after our main interview rounds and only one validation interview took place after the feedback session. Finally, the

fourth author of this paper, a representative of the case organization commented on the final draft of this paper.

4 Results

4.1 Motivation

In this section we answer our first research question, RQ1: *Why did the organization initiate an agile transformation?*

According to our informants, there were three main motivators for the transformation in the case organization: 1) Agile software development was becoming an important part of Ericsson's corporate strategy, 2) a dissatisfaction with the current way of working, and 3) a need to enable rapid end-to-end flow of features and continuous deployment.

4.1.1 Agile as Part of the Corporate Strategy

At the corporate level, Ericsson had identified the need to be more agile, and had made the adoption of agile methods a strategic priority. Several successful agile transformations had already taken place in various units withing the company. However, each unit inside Ericsson was given the freedom to choose whether and how to adopt and apply agile. At site A, the biggest site of our case organization, a previous, still ongoing project, had started the transformation earlier (see (Paasivaara et al. 2013)). A large group of people from that project were gradually transferred to the case organization, and to them, agile was already a natural way of working. Thus, given their exprience with agile, it became natural also for the case organization to start thinking about adopting agile.

4.1.2 Dissatisfaction with the Current Way of Working

After the product was acquired, the case organization started to implement Ericsson's traditional, waterfall process framework, even though the first development teams did not use any well-defined development process. The early development teams were simply assigned new features with preassigned deadlines.The teams then implemented the features as they saw fit. Our interviewees reported that development was slightly chaotic at this time, but features were finished on time. The lack of a defined process was not considered a major problem, because there were only a couple of small teams working on the product, in addition to a group of external consultants.

However, as the organization started to grow in 2012, it became necessary to implement at least somewhat orderly process. The first step, in 2012, was to implement a component-team based model, which seemed natural, as the product was composed of several components, each requiring specialized technical knowledge. The component teams had members with deep expertise on their individual components. When developing features spanning several components, virtual feature teams were used. In these, specialists from different component teams collaborated on a specific feature. There were several issues with the component based structure: it was challenging to plan and coordinate work, as features depended on several components, and experts were not always available when needed; the work was not considered efficient; development lead-times were long; teams had difficulties in finishing promised features on time; and team members felt that this way of working was stressful and somewhat chaotic. The rapid organizational growth from around twenty

persons to over one hundred exacerbated the situation. Thus, they felt that change was needed.

4.1.3 The Need to Enable Rapid End-to-end Flow and Continuous Deployment

At the time of our study, the product was released every eight weeks, the same rhythm as when it was acquired by Ericsson. However, this was considered too slow, and the goal of the organization was to transition towards continuous deployment. The idea was that a new feature could be taken as part of the product instantly when ready.

> My dream is that we shouldn't have releases at all, but that a feature goes to production right away when it is ready. It means that what we do here should include coding and verification in the team, as well as continuous integration and automatic regression tests so that we can trust that when they [the team] say it's ready, we can just push it to the system. — Manager

Thus, the hope was that going agile would enable them to implement each feature in a cross-functional team as efficiently as possible, from requirement until delivered as part of the product. Moreover, by using agile practices, Ericsson aimed to optimize the whole end-to-end flow:

> Our goal is that we can make this whole end-to-end chain work in a new way, to remove all waste, all unnecessary handovers, and [...] to optimize the whole flow, from customer requirement until deployment. — Manager

Optimized end-to-end development would help the organization to respond quickly to changing customer requirements, as well as to provide customers constant visibility on what is coming out next.

To achieve these goals, a wide spectrum of organizational improvement actions were undertaken, as described next.

4.2 Phases of the Transformation

In this section we answer our second research question, RQ2: *How did the transformation proceed?*

The overall approach to the transformation was experimental — based on their previous experience in transitioning another product program to agile, the managers had learned that it is impossible to plan the transition in detail and execute it with a "waterfall mindset". Instead, the managers and coaches took an experimental approach, purposefully focusing on a single key change or improvement target at a time. This way, the main transformation steps were not planned beforehand, but were decided one at a time on a need basis. Thus, the phases we report below are the researchers' construct that we present as a way of structuring the discussion rather than as a prescription for conducting agile transformations.

We discuss the transformation organized by three main phases: 1) introducing agile, 2) finding common ground through value workshops, and 3) towards continuous integration and deployment. In addition, we describe the situation before the transformation as Phase 0: knowledge transfer and component-based teams. The main phases, as well as some major events are presented in Fig. 4. As illustrated in the Fig. 4, the phases were somewhat parallel and most did not have clear ending dates. Next, we describe each phase in detail.

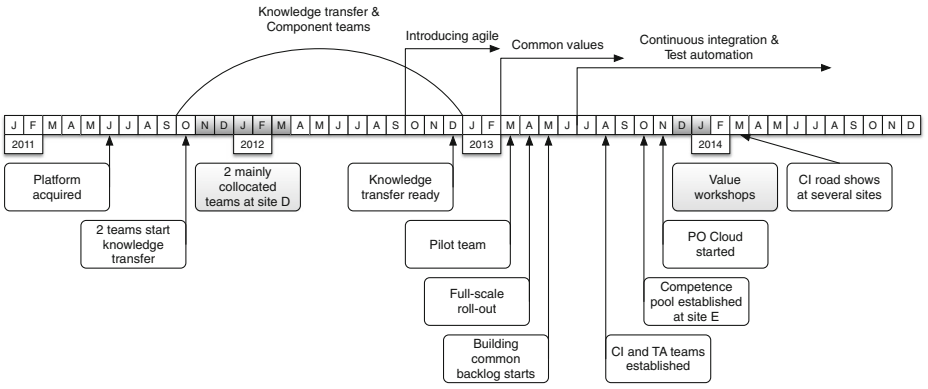

Fig. 4 Timeline

4.2.1 Phase 0: Knowledge Transfer and Component-Based Teams

Knowledge transfer from the original development organization, including external consultants, started soon after the acquisition. The first two teams were built in fall 2011. During winter 2011-2012, these teams worked partially collocated at site D and partially as distributed teams, as part of the team members came from sites A and B. However, during the most intensive knowledge transfer period, most members worked collocated at site D for longer periods. These first teams were working without any specific process. Instead, team members collaborated informally aided by "agile seating", i.e. they shared a single large table:

> As I see it, we had no process in the beginning that we would have been following... So no Agile processes, nor [any] traditional waterfall model. — Team Member

When growing the organization from the initial two teams, the idea was to hire experts with knowledge of specific system components. In particular, they intended to use internal recruiting as far as possible. As a result, the experts were located at different Ericsson sites. In addition, a consultancy company offering experts with specific domain knowledge was hired at site E. Even though the organization had started talking about agile already in late 2011, they decided to go for a component-based team structure in early 2012. The main reason for this was that each component required highly specialized knowledge and it was time-consuming to learn even a single component.

> You cannot really ask people to learn more than one component in two years. — Product Owner

Furthermore, Ericsson had a long history in using a waterfall type process. Thus, this initial organization structure was based on component teams and a sequential, waterfall type, process. Typically, a single component team comprised of 10–20 people, was distributed to multiple sites, and communicated through weekly or daily teleconferences.

Experts from these component teams were selected to virtual feature teams, as illustrated in Fig. 5, whenever the development of a new feature would start. Virtual feature teams were loosely structured—team members performed their own feature-related tasks for their component, and then passed the work further. Usually, a new virtual team was established for every feature.

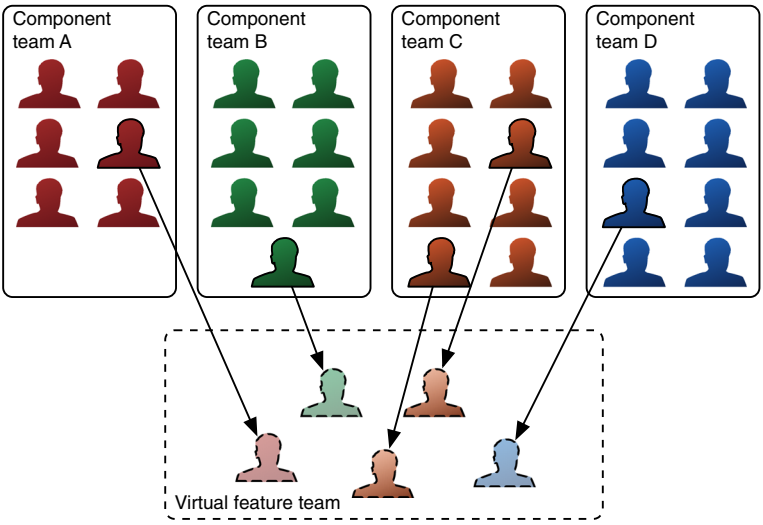

Fig. 5 Virtual feature team

This component-based organization structure had several challenges, e.g., suitable resources for a new feature were not always available, and virtual feature team members simply performed their own tasks individually, and did not actually work together as a team.

> Setting up the virtual teams was challenging because we had the feature and then we found three guys [with competence A] but we don't have [competence B] because they're all busy with other features. So here we have the resources [with competence A] available but then we cannot wait for three weeks, so the guys start with something else. So it's like a puzzle all the time. — Manager

Furthermore, the team members considered it challenging to work in virtual teams. The people you were supposed to collaborate with changed constantly and it took time to make the acquaintance of new people, hindering the development of trust and slowing down team building. The interviewed team members reported that at that time they identified themselves more with the component teams, rather than with the constantly changing virtual feature teams.

As a whole, the organization structure based on component teams and virtual feature teams created on top of them was seen as too rigid and not being able to answer market requirements fast enough. It was not efficient nor predictable enough.

4.2.2 Phase 1: Introducing Agile

When the organization decided to move to Agile software development, the idea of creating cross-functional, cross-component teams was born. Here, we focus on the organization and team structure while moving to agile, as it turned out to be both important and challenging. The structure was tested and modified several times.

At the team level agile, teams were given the freedom to themselves decide the practical agile implementation, guided by the coaches. Thus, no common agile framework was prescribed or used.

The organization structure evolved into the current agile team structure through four phases:

1. Building a pilot cross-functional agile team
2. Full-scale roll-out of cross-component, cross-functional agile teams
3. Creating a competence pool providing team members to cross-component teams according to the needs of each feature
4. Cross-component, cross-functional teams specializing on specific business flows

Pilot Team The first pilot team was created in early spring 2013 to evaluate the new concept. This team was formed of volunteers from two sites, who had an avid interest in adapting agile ways of working. According to our interviewees this team both collaborated remarkably well, using the agile practices and achieved good development results. However, one problem was that some of these volunteers had a central role in their previous component team, and their absence affected the work of those teams. Therefore, management decided to dismantle the pilot team after a few weeks and start a full-scale agile rollout with cross-component, cross-functional teams.

Full-scale Roll-out: All European sites were involved in forming the teams. Line management set the frames for the new teams, and the coaches worked on developing guidelines. Team formation was discussed in several videoconference sessions involving the future team members. Based on these discussions, the teams were formed so that in Country Alpha the teams were either site-specific (in site A) or distributed within the country to be able to allocate experts on one specific component located at one of the sites (site B) to different teams. The other set of teams were created between Countries Beta and Delta to mix in highly experienced product architects and technical coaches from sites C and D (usually two persons from sites C and D per team) with experts on third party components from an external consultancy company at site E (around ten persons from site E per team). Altogether 10 teams were created.

Competence Pool: However, this setup between Countries Beta and Delta had to be slightly adjusted as the optimal mixture of knowledge on different components depended highly on the specific feature to be developed. All features did not involve all components, thus how much knowledge on each component was needed in a team depended on the feature. Moreover, consultants had quite narrow focus areas, and the case organization did not see having them broaden their knowledge on other components as cost-efficient due to high attrition rate at the consultant company. Thus, the five quite large teams between Countries Beta and Delta were rearranged into four smaller teams of 7–9 core team members, while the rest of the consultants at site E formed a competence pool, from which suitable resources were chosen to teams according to the needs of the next feature, as illustrated in Fig. 6. Teams at the other sites (sites A and B) remained the same.

The permanent cross-component teams were complemented with component-based Communities of Practices (CoPs). CoPs are *groups of experts who share a common interest or topic and collectively want to deepen their knowledge* (Wenger et al. 2000; 2002). In the case organization, the CoPs were open to anybody interested in the topic. The CoP culture was also dynamic. New CoPs were founded when an active individual took the initiative. When a CoP was not needed anymore, or had difficulty attracting participants, it ceased to exist. Most CoPs met on a regular basis, as well as had discussion forums, wiki pages etc. for communication. The usage of CoPs at Ericsson is described in more detail

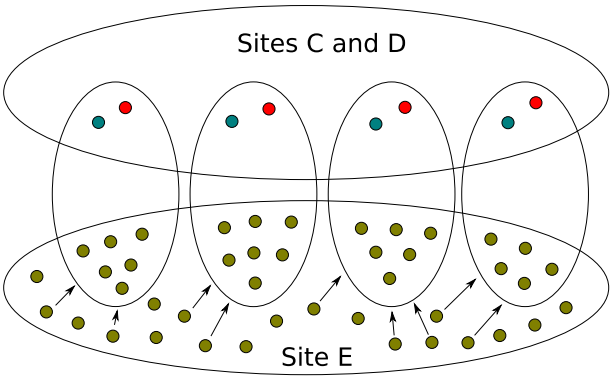

Fig. 6 Cross-component teams (between countries Beta and Delta) with a competence pool. People at site E who are not allocated to teams form the competence pool (19 persons)

in (Paasivaara and Lassenius 2014). In the Component CoPs, the experts for different components collaborated across teams inside each component. Forming the CoPs was easy, as they consisted mainly of members from former component teams. Thus, most members had previously collaborated closely. The daily or weekly component meetings were replaced by weekly Component CoP virtual meetings. Most CoPs started to function well with the help of the coaches. The biggest problem was how to transfer the component-specific improvement items, e.g., refactoring, agreed in CoP meetings, to the team backlogs. In addition to Component CoPs, other CoPs on specific topics were formed, e.g., a CI CoP and a Coaching CoP.

While forming the permanent cross-component teams during the spring and summer of 2013, the organization was both hiring new team members externally and adding whole teams by moving them from another, still on-going project that had been using Agile for several years at site A. Thus, the number of teams grew quickly during this phase: from 10 teams in spring 2013 to 15 in fall 2013.

Specialization in Business Flows: In the beginning of the transformation, the goal had been to create teams that would be both cross-component and cross-functional, and that any team would be able to implement any feature that happens to be at the top of the backlog. However, the organization soon learned that this would never work in practice.

The product included a large number of components, many of them developed using different technologies, and each component required deep technical knowledge. To solve this problem, the case organization created teams specializing in use cases spanning several components, or business flows as they called them, with a few teams working in each business flow. This would not require the members to have deep knowledge on all the components of the product. Within the business flows each team could implement end-to-end functionality, from requirement to deployment. The most important of these were Service Exposure, SIM[3] and subscription management, Billing and Rating Services and Connectivity Services. This was the structure when our study ended. The features developed within

[3]subscriber identity module

these business flows were mainly done by one team each, however regarding big features several teams could collaborate. The size of the features varied from small ones that one team could implement in a week to bigger ones that could take half a year to develop. Before being called business flows, some managers referred to them as domains:

> We have broken down the system into domains [business flows] now, different areas. The idea is to have a PO [Product Owner] for the domain, and this is also the product manager for the domain...and maybe the backlog should be for that domain only...It is five functional domains, and one cross-functional one. — Manager

4.2.3 Phase 2: Finding Common Ground Through Value Workshops

The development organization grew quickly from 10 (spring 2013) to 15 teams (fall 2013), while introducing agile development at the team level. Even though the goal had been to form predominantly site-specific teams, due to the knowledge differences between the sites, approximately half of the teams ended up as cross-site teams. There were lots of people who had never met. In addition, people at one site did not necessarily know what was happening at the other sites regarding the development and the transformation. There were clear borders between the sites:

> I see site politics as one of the problems. It's difficult to communicate between the sites. So we build up some kind of, us vs. them feelings. That hinders our way of working. We don't have a perfect flow in the system. Because we don't really trust each other. And that's a problem. — Coach

Moreover, management noticed that the organization lacked a common direction, regarding both the future direction of the product, as well as the way of working, and there were site-based and history-based opposing views. Thus, management and coaches decided that the next step in the transformation journey would be to define a common direction and build a "we spirit" to help people identify themselves with the single product organization rather than with their competing sites.

> Why we have started with values, [...] is that we would have a common baseline to continue further, [...] a baseline on which we build this common understanding and common direction. That we have something common to discuss together. I have seen as a problem in this whole project that different sites and different people have taken a bit different direction. — Manager

The work on the common organizational values started in early 2013. The first step was the *Futurospective*, a workshop where the agile coaches and managers created a vision for the organization a couple of years ahead. Based on the results of the Futurospective, the coaches wrote a *Showcase*, a fictional story of how the organization would look like and how it would work in two years time, after tight collaboration and joint creation of a success story. The idea of the values was born during a workshop on how to make the organization "more agile". Thus, the values were based on the one hand on the ideas and principles of agile, and on the other hand on the three core values of whole Ericsson: professionalism, respect, and perseverance. The five core values were created in collaboration between the coaches, the management team and a few developers, and are: *One organization*, *Step-by-step*, *Customer collaboration*, *Passion to win*, and *Fun*.

To share the values with the whole organization, a series of *Value Workshops* were organized during winter 2013–2014.

The goal of the value workshops was twofold: 1) to create a common vision for the whole organization in the form of common values, and 2) to create contacts and collaboration, as well as building a "we" spirit across the sites by having people meet face-to-face.

The value workshops were held as two 2-day workshops at the biggest development sites, A and D, with around 20 people traveling from three other European development sites. The whole management team, all coaches, as well as a few team members traveled. The only site that did not have workshop participants was site E, the consultant firm, with the exception of a few consultants who were working on the sites where the workshops were arranged. The aim was that all team members from sites A, B, C and D would participate in one of the workshops, as well as meet all managers and coaches face-to-face. The results from the first value workshop organized at site A was shared with the other sites by having a videoconference call during the result presentations between the sites A, C, and D (site B participants were at site A already).

Besides meeting face-to-face, the goal of the value workshops was to jointly discuss and elaborate the values. Purposefully, the values were not defined beforehand, but the managers and coaches presented the values in both workshops using examples. What each value really means were discussed in small groups. In Table 3, we have collected some examples of what these values could mean based on the Showcase, examples provided and the value workshop discussions.

The workshops included different kind of group activities and exercises: within the whole group, within individual cross-functional teams, as well as in highly mixed teams with people from different roles and from different sites. For example, in one exercise, the teams considered what the values would mean in practice in that specific team, and what kind of concrete behaviors they would lead to. The coaches from different sites planned and

Table 3 Values

Value	Examples
One organization:	We identify ourselves as belonging to the [name of the product] family. We are [name of the product] people more than we are sales persons, designers, testers, systems or operations. We prioritize [name of the product] level goals over local site or personal goals when needed. We engage with and trust appropriate competences and colleagues regardless of organization or location. We act as ambassadors for the whole [name of the product].
Step-by-step:	We continuously, step-by-step, improve our way of working. We apply continuous learning, we make changes, trials and errors. We make decisions on a team level. We promote and perform prototyping. We split our work into smaller pieces and work on them one step at a time, delivering value constantly to our customers.
Customer collaboration:	We form partnerships with our customers. Development teams collaborate with customers on a regular basis aiming to exceed the customer's original expectations. We propose customer participation in demos and encourage regular customer interactions. We support team members in contacting customers or to get customer views.
Passion to win:	We take risks, we are not afraid of mistakes. We learn from our mistakes. We don't give up. We challenge each other's ideas and start discussion. We share successes — small as well as big ones. We step back and see the big picture and remind others.
Fun:	We are empowered and proud people, proud of being part of bringing a complex product into the market with many happy customers. We share good news, feelings and work. We show trust, empowerment and fun. We support increased transparency and cooperation. We have great fun working together.

facilitated these workshops as a collective effort. For more detailed description of the activities during the value workshops see (Paasivaara et al. 2014b).

The first impression of the value workshops was highly positive. In particular, participants felt that the organization took a huge step closer to the goal of being a single organization building a common product. Especially, meeting with people from other sites and talking face-to-face was a benefit that all interviewed participants mentioned.

> The value this event brings, that I see, is that we are no longer just names and faces behind the screen. You see real people and talk to real people. — Team Member

Regarding the values, most workshop participants seemed to feel that the chosen values were good:

> I completely agree with these values. [...] [the values are] not so easy as before to forget, or ignore in the daily work, I think that's the main benefit of the workshop. — Team Member

Several interviewees agreed that they would personally act differently in the future and that the events had clarified the values and made them meaningful.

> I will probably do a lot of things differently. [...] I'm gonna try to collaborate more, between the teams. Because I think that's one of the biggest flaws we have right now. — Team Member

Some participants worried that the values would be forgotten after the events, expressing that good intentions formed during the workshops are not enough to implement the values in the normal working environment. The plan to tackle this was to have the coaches help the teams work towards the common values and exhibit the behaviors they had planned. Many of our interviewees also suggested some kind of a common follow up for these events after half a year or so.

> I would say a follow-up in maybe six months or something like that, just to have a recap of what has changed, what has happened, what I have done. Just a kind of retrospective, just to see what is happening and what kind of next steps we can take. [...] All sites should be involved with that follow-up, [...] because we should fight for this one [name of the product]. — Team Member

Even though the values were considered good and the workshops beneficial by all of the interviewed participants, some were still hoping to have an even more concrete vision than what the values and the showcase provided. Especially, a concrete product vision or roadmap was asked for. However, that was not a goal of these workshops this time.

4.2.4 Phase 3: Towards Continuous Integration and Deployment

The lack of continuous integration (CI) and test automation were major challenges on the way towards continuous deployment, as the integration and testing phase took several weeks before each release. The goal was to get rid of the integration and testing phase, and having the teams integrate and verify the system functionality immediately.

> I think [that] the goal is that we should be able to...when something is ready...it should...pass through and be deployed directly into production. If we can deploy

> something...maybe the first user (story), I mean not a complete thing and deploy it and test it with a key customer. — Manager

The work towards this goal started in fall 2013 by creating three new teams concentrating on implementing CI and test automation. Most team members came from another product developed by the same organization, and in which agile methods had been in use for several years. In that product CI and test automation had been a major and extremely successful effort. Thus, the teams had ample relevant knowledge and experience.

A future goal was to spread the CI knowledge, goals and mindset to the whole organization: from teams up to the management by *Continuous Integration Road Shows* arranged during spring 2014. These consisted of information events and trainings for the whole personnel, e.g., on the selected test framework.

One of the Lean principles (Poppendieck and Cusumano 2012), optimizing the whole, is behind the goal of end-to-end development. In this case, end-to-end development meant developing system functionality from a customer requirement to new functionality being part of the product and used by the customer. One way to shorten the lead time of end-to-end development is to develop each functionality in a single team. This removes extra handovers and non-value adding waiting. CI and automated testing aims to optimize the last part of the end-to-end flow before the release.

Another action the case organization took to optimize the flow was to involve the teams in the early phases, i.e., in planning and design. The idea was that the teams would themselves conduct initial studies on new features: Feature Investigations (FI) and Feature Concept Studies (FCS). The purpose of these studies was to quickly investigate whether a feature is doable, how much effort it might require and how it could be implemented. Previously, experts such as architects had conducted the studies during the planning phase of the waterfall model. However, now the aim was to perform less profound studies quickly whenever new feature requirements appeared. The expected benefits were threefold. First, as the studies are not that profound, quick feedback can be received. Second, when teams are involved they learn more about the features, thus speeding up implementation since no extra handovers or documentation are needed. Third, as the number of experts doing these studies was limited, reducing their work with FIs and FCSs would free them up to focus on more profound issues. At the time of our research, the studies were already assigned to the teams. However, in-team experts, e.g., architects or subsystem responsibles, normally took the main responsibility for conducting the studies.

At the end of our study period, the organization had six releases per year but the goal was that the teams would be able to deploy new features into production immediately when they are finalized.

4.3 Challenges and Mitigations

In this section we answer the last two research questions, RQ3: *What challenges did the organization encounter?* and RQ4: *How did the organization mitigate the challenges?*

Overall, our interviewees considered this agile transformation very successful: they had taken major steps towards their target—a unified agile organization having the capacity to deliver value continuously. However, the journey had not been without problems. Next, we discuss the major challenges encountered (see Table 4), as well as how the organization attempted to solve them. All challenges were not yet resolved by the end of our study period, however, the transformation journey continues as the organization continuously attempts to solve new challenges as they emerge and continuously improve their way of working, following their experimental approach to the transformation.

Table 4 Challenges and Mitigations

Challenge	Mitigation used by the organization
Change resistance	Building a leadership team with an Agile mindset
Technical debt	Establishing a common backlog, investing in system improvements, introducing supportive roles: subsystem responsibles and architects
Lack of a common agile framework	Setting up a Coaching Community of Practice to support similar coaching across teams and sites
Lack of coaching and coaches	–
Lack of agile training	Coaching as a substitute for training
Cross-site teams	Cross-site visits, visiting engineers, providing high quality videoconference equipment, cross-site value workshops
Working as "a real team"	Pair work to broaden the knowledge of the team members to other components, letting teams specialize in specific business flows
Any team cannot implement any feature	Teams specialize in business flows
Lack of continuous integration and test automation	Building CI teams and appropriate test environments, creating automated tests for legacy code, training personnel on continuous integration, e.g., by arranging "CI Roadshows"
Agile teams in a waterfall organization	Adding end-to-end responsibility to cross-functional teams: building continuous integration and test automation systems and involving teams in early planning
Challenges in defining the Product Owner role	Setting up a Product Owner Team, the "PO Cloud"
Challenges in breaking down the requirements	–
Backlog challenges	Building a common backlog
Constant change	–

4.3.1 Change Resistance

The first initial attempts to start the transformation were in 2012, but they did not lead anywhere as the issue polarized the organization. Some did not want to change the way of working at all, and those willing to change had different views on how the transformation should be conducted. Initially, the leadership team was not willing to sacrifice deliveries in order to support the transition. Several leadership team members found it more important to deliver new features than to focus on a major organizational change.

> The top operative management was located in [site C], and they hadn't adopted the Agile philosophy. There was so much resistance that it was absolutely impossible to drive the change from bottom up. — Team member

During summer and fall 2013, the leadership team was reorganized, and new members having previous experience in agile transitions and strongly supporting the transformation were added. After this change, the transformation was rolled out full-scale, with less resistance.

However, at the time of the interviews, there were still groups of people in the organization, who had not yet adopted agile thinking. For example, the product management had still a quite plan-driven mindset, as illustrated by the following quote:

> In product management there's still some belief that a plan is the truth and trying to fulfill that is a good thing. — Coach

The evidence related to change resistance was strong in both the sense that it was mentioned and discussed in depth by many respondents, as illustrated in the quotes above, as well as in the fact that it was the explicit reason for changing the membership of the leadership team.

4.3.2 Significant Technical Debt

One bottleneck that prevented the transformation was a high degree of technical debt in the system. Technical debt is a metaphor originally referring to "not quite right code which we postpone making it right" (Cunningham 1992) but that since has expanded to include a spectrum of issues from bad coding to architectural issues (Kruchten et al. 2012)

The system was originally designed for a single customer. Additionally, the development in the previous organization had occurred within strict deadlines. Together, these two factors had resulted in a situation where lots of shortcuts had been taken in development, and the system was not stable enough to be scaled up for a larger pool of users. Improvements had to be made before new features could reasonably be implemented. Moreover, in the beginning, when working in component teams, adding new features had the highest priority, while the quality of the underlying system suffered. That happened partly because the overall architecture was not well understood by the new developers. All this led to increasing technical debt.

During 2012, many system improvements took place and a few components were replaced by Ericsson's own components.

Furthermore, when working in component teams, management used to make the feature implementation decisions according to ever changing customer requirements. Feature prioritization could change constantly, causing major challenges in design, coding and testing. However, this was improved after establishing a common backlog and assigning subsystem responsibles and architects to the development teams.

The technical debt was a pervasive issue that despite its importance was raised only by few technical experts. However, the importance of dealing with it was reflected in the urgency of getting a working CI system in place to harness the product, and the fact that it required serious architectural changes to the components.

4.3.3 Lack of a Common Agile Framework

The organization had decided not to use any common agile framework guiding the teams' day-to-day working practices. Instead, each team could itself decide how to work. The only commonalities between the teams were the common bi-weekly demos, coaches, and the use of Jira as a backlog management tool. In the common demo, usually one team demonstrated their achievements. This demo was open for everyone in the organization and it was organized as a video/audio-conference between the sites. The teams having finished something of interest to others would give a demo. Some Scrum trainings were arranged in the beginning, but participation was voluntary, thus not all participated. Some teams and team members had already agile experience from their previous project, some not. Thus, taking agile into use at the team level was not systematically organized.

Many interviewees expressed that starting with a common agile framework that teams could later on tailor to their needs would have been preferable, as that was how it was done

for example in another still on-going agile project at site A, from where many managers, coaches, and team members had been moved to this project. Several interviewees commented that having a common framework, like in that previous project, would have been a better solution to this project as well.

> I think it's good to start with a common [framework], like start with Scrum or anything. That's where you start, and everybody has to go through that or whatever and then you can go from that. But now it's really difficult. [...] We have to really go back to [the basics] so it's really difficult to do coaching or advice because we are, I don't know where we are. I agree it's kind of a problem. — Coach

Many interviewees from teams even commented that their team had some agile practices in use, but not a specific process to follow. For example, most teams did not use sprints and many teams did not have regular retrospectives or planning meetings. At the time of our study, each team had their own ways of working, often combining Scrum and Kanban, e.g., all teams had a Scrum or Kanban board to visualize their workflow.

> I came from a company where we followed Scrum exactly, but here I feel that we are doing things, but we have no process to follow. — Team Member

> I think Scrum is a very good start, and when you know Scrum then you can shift into other stuff. My feeling here is that we are kind of trying to take a short cut and doing other stuff immediately, so some of these ground pieces are actually missing in quite many teams. — Coach

A few interviewees suggested having a common pulse for the organization:

> I think we need a pulse in [name of the project]. We should have, like, every second week we could have a common planning, a common retrospective, and I really miss that. [...] It's actually on our [Coaches'] Kanban board to start up this pulse, this heartbeat. — Coach

> Some sort of timeboxing could help to push us to work harder and to help us to prioritize our work so that it is done in the right order. — Team Member

The common demo every second week was a start for this pulse and our interviewees found it useful. However, they did not consider this sufficient.

A few interviewees explained that one reason for not starting with a common agile framework was, surprisingly, due to that above mentioned still on-going agile project at site A. This other project had started their agile transformation a few years back with strict Scrum that they later modified towards Scrumban and gave the teams a lot of freedom to choose their ways of working. As part of the personnel from that project had moved to the case project, some interviewees suspected that the managers and coaches moving from that project did not consider it necessary to go back and start with a common Scrum framework and Scrum trainings, as they had done that and were "past that phase". They explained that the persons probably assumed that the rest of this project would be as mature in agile as their old one, making it possible to directly apply the same kind of practices and thinking.

> So they probably tried to short-cut this path through Scrum. So they kind of tried to start where the old organization was. — Coach

However, in practice this was not possible, as a large part of the personnel in this new project was new to agile or had little familiarity with agile, and thus needed basic training

and a framework to start with, before they could start modifying it. Interviewed developers that had moved from that other project to the case project, found having a common agile framework with common Scrum trainings a much better way of starting the transformation than giving quite free hands to the teams.

As the new organization was composed of persons from several internal organizations and sites, none of the groups actually wanted to say that "this is the way we should work". Instead, they tried to come to a joint understanding and a way of working. However, achieving such an understanding takes time. Actually, the managers and coaches coming from that other on-going agile project at site A explained to us that they did not want to "push" too much the ways of working in their previous project, as in the beginning they had done that, but the other sites clearly did not like it, but instead always answered in style "but this is a totally different kind of product". Thus, instead of following the good practices from the previous project, they decided to find together a common way of working for this project.

A major step towards this goal was the creation of a cross-site Coaching Community of Practice (CoP). Having a Coaching CoP that meets regularly aims at helping coaches to establish a common way of coaching and to guide the teams to work in similar ways across the sites. This would be helpful also for people changing teams, e.g., the floating resources in the competence pool.

> I think that's [Coaching CoP] really important. And it must be cross-site, so that we can coach in the same direction, at the same time and have the same view on coaching and the ways of working. So, instead of going into control mode we should coach in the same way. And say the same things about what's good and bad. — Coach

The evidence regarding the problems related to the lack of a common agile framework came from a few respondents, who had participated in an earlier agile transformation within Ericsson. While small in the number of respondents, their insights were deep, and they discussed the issue at depth in our interviews. They very strongly recommended that a common framework should be used instead of giving teams too much autonomy too soon.

4.3.4 Lack of Coaching and Coaches

At the time of our interviews, the organization had both team and organization level coaches. Approximately a third of the teams had their own team coaches. The organizational coaches supported the rest of the teams, each having several teams to coach. However, they were also responsible for helping with agile issues at the organizational level and developing the whole organization and its way of working further. Thus, the coaches lacked time to concentrate on helping individual teams. For example, they could not always participate in their teams' daily meetings or retrospectives. The interviewees found coaching they had received extremely useful, but thought that the number of coaches was not sufficient.

> Last week the coach participated in our daily meetings only once. And during the previous week maybe twice. [The coach] wasn't involved in other things.
> — Team Member

> Currently we have so many teams and there's only a few of us, so we are not able to support teams very well. — Coach

During our study period, the organization slightly increased the number of both organizational and team coaches. As the number of teams grew at the same time, the situation improved only slightly.

The data related to the lack of coaching and coaches came from a few coaches and a few team members, and is not as strong as for the previous categories. While coaching is important, the lack of it did not seem to be one of the most important challenges in this case, as coaches were available.

4.3.5 Lack of Agile Training

The agile knowledge was not yet at a sufficient level despite the fact that the organization had experienced people with knowledge on agile methods. The level of agile knowledge varied a lot from person to person, as some had used agile in their previous organizations while some had never used agile. Even though a few agile trainings had been organized, not all employees had participated in these, as participation had been voluntary and they had prioritized other tasks. The knowledge of agile experts was not spreading as well as it could have been.

A few interviewees even mentioned that the basic terms, such as *feature*, *story* or *definition of done* were not known or understood similarly by all, which is a basic requirement for working together in an agile way.

> Sometimes I send out mail or call people and discuss these, what I feel is basics. And then, for example, I talked to somebody about the definition of done. But then, yeah they kind of agreed some of them, but a couple of days later you get back some questions, "What is the definition of done?". And then you realize, okay. We have to really go back to, so it's really difficult to do coaching or advice because I don't know where we are. I agree, it's kind of a problem. — Coach

The organization had plans to arrange trainings on a need basis. Moreover, the collaboration of coaches across sites and unifying the coaching would in time increase the agile knowledge in the teams.

The evidence related to the lack of agile training came from a few respondents, in particular coaches. Many people in the organization had already earlier received agile training, but the coaches noted clear differences in the knowledge, e.g., across site borders. While the data supports the importance of agile training, the lack of it was not one of the main concerns in the organization.

4.3.6 Cross-Site Teams

Even though one of the principles when forming the cross-component agile teams was to build site-specific teams, ca 50% of the teams were distributed between two or three sites at the end of our study period. Many interviewees commented that this was not a good solution for high quality team work. However, due to knowledge asymmetries between the sites this structure was deemed necessary.

> Q: Do you feel that you are really a team?
> A: No, I don't. I am a team player and I like working in teams, but I don't feel that we have a team spirit. And, I guess it's hard, when you have multiple sites. As long as you don't know the people, you can't possibly care for them either. — Team member

> It would be nice if we could work with local teams, if we didn't have any dependencies, for example. [...] But we do have dependencies between each other, so in that case it's better to have distributed teams, even though it is less efficient on the team level. — Coach

This distribution was mitigated by site visits, e.g., single team members located at site B visited the rest of their team at the site A at least once a week. From site E, there were *visiting engineers* constantly working with teams at sites A and D, as the "eyes and ears" of the site E.

Moreover, high quality videoconference equipment was used between sites A, B, C and D for most meetings. The videoconference connection between the distributed team members at sites A and B, was mentioned to be open sometimes all day to enable ad-hoc communication. Unfortunately, site E, a consultancy company, did not have compatible videoconference equipment. Thus, personnel at that site usually participated in using only audioconferencing.

Many interviewees emphasized that the team members should travel even more, and that they should arrange exchange visits and work co-located, at least for short periods.

> We don't talk to each other that much. So you do not trust each other. If you don't know somebody, it's difficult to trust them. [...] My solution is to travel more. To actually see each other. The teams [should travel]. The ones who really should cooperate. [...] Once you have met each other and worked together for a while, then it's much easier. And that sticks for a while. — Coach

One goal of the value workshops was to increase trust, and make people know each other to lower the threshold for contacting. However, most cross-site teams spanned sites E and C and/or D, and as team members from site E, the consultancy company, did not travel to the workshops, they did not help with building team cohesion as much as they could have.

The use of cross-site teams with members from organizations that had not been tightly integrated before arose as one of the major problems in the case project. It was mentioned by most respondents, and also was the reason for arranging the value workshops.

4.3.7 Working as "A Real Team"

All cross-component teams had experts from several components. As each expert had deep knowledge on his or her component only, some of the interviewed team members felt that all teams were not yet working as "real teams". The global distribution of many teams made this even worse.

> Q: You said that you do your tasks mainly all alone so, what do you do with the other team members?
> A: I don't do anything with them. I don't work...
> Q: So you don't have any collaboration?
> A: Sometimes they come with questions, and I try to answer. — Team member

Due to the deep technical knowledge required team members could not really collaborate, e.g., help other team members working with other components when done with their own tasks. Thus, at times some team members had a too high workload, while others might not have work at all. During one of the value workshops, one team member worked on a task, while the rest of the team participated in the workshop. The team members explained that this individual was doing a critical task that no-one else from the team had knowledge of and thus could not help with. They explained that otherwise the whole team would be solving the problem together instead of participating in the workshop.

One of the goals the organization had was to broaden team member's knowledge on other components, e.g., by working as a pair with an expert of another component in the own team. The goal was to add collaboration between the teams, e.g., pairing teams, so that they

could learn from each other. Moreover, an exchange program across the sites was suggested to enable team members and teams from different sites to learn from each other. Exchanges were going on at the time of our study, even though it was not systematic.

Many components were quite complex, requiring significant effort to learn. This learning was not structured or organized, leaving it mainly to individuals to organize their familiarization.

Teams distributed between sites C, D and E had a couple of very experienced members, e.g., subsystem responsibles, sub-system architects or technical coaches from sites C and D, while the rest of the team was located at site E. As site E was located in Asia, and sites C and D in Europe, there were significant cultural differences between the sites. At the time of this study the technical coaches had taken the role of team leaders and the rest of the team performed individual tasks. These teams seemed to have a long way to go before turning into real agile teams.

The problem related to part of the teams not being "real" teams was mentioned in particular by coaches and line managers, working with teams with members on site E. This can be explained by the fact that we did not interview team members at site E (the consultancy company). The significance of the problem was evidenced by the fact that the organization actively planned for how to make the project succeed without site E.

4.3.8 Any Team Cannot Implement any Feature

The initial goal was to have fully cross-functional and cross-component teams that could implement any feature from the top of the backlog. However, the organization realized that this might never work in practice, as the different components required very specialized knowledge that would take long time to learn.

> For at least half a year ago they said that one team should be able to do any feature, end-to-end. That's impossible. [...] I don't think we will ever get one team who can do end-to-end of all features. — Coach

> It's proved that it's almost impossible to have a cross-functional team that could do any features. We don't have a situation where developers would have competences of many components, and that's why it's easier for teams to focus on a specific area. — Product Owner

> We work a lot with third-party products. And I cannot possibly help someone else working on another platform. And the other way around. They can't help me, so there's not really any point in having cross-functional teams in that sense. — Team member

Thus, the initial idea of fully cross-component and cross-functional teams was discarded, as it was not seen reasonable that any single team could be able to implement just any feature from the backlog, not even after some time of working and broadening the knowledge. At the end of our study period teams started to focus on specific business flows, which would not require knowledge on all the components of the product, but only from a few. Within these business flows, teams would still be cross-functional and able to develop end-to-end features.

The fact that the agile ideal of any team being able to implement any feature was impossible to meet in this project turned out to be one of the main problems, which explains many of the organizational changes done. The evidence for this is strong both as it came up in many interviews, and in the visible actions taken trying to deal with it. Indeed, it can be

considered one of the main findings of our study that there can be cases in which trying to meet this agile ideal might not be feasible.

4.3.9 Lack of Continuous Integration and Test Automation

At the time of our first interviews, most of the testing was still manual, as appropriate CI and test automation systems had not been implemented. Therefore, the development teams had only three to four weeks for implementing new features until the code freeze, when the integration and verification team would start testing it. The testing phase took three to four weeks, after which it took approximately three weeks until the system could be released.

> One thing that is in heat, that is due to that we don't have this CI and the setup. We're living in this waterfall mechanism so four weeks before we go into deployment, then we're more or less locking the code, the mainstream. — Manager

As the organization was building CI and test automation, this challenge would be mitigated. However, the initial effort to build the systems had required a huge effort: three teams focusing on it for several months. The next major effort, *Continuous Integration Road Shows* for the whole organization took place at the end of our study period. The aim of this effort was to train the personnel in CI practices, as well as build a *CI mindset* in the whole development organization.

The lack of continuous integration and test automation was mentioned in particular by managers and coaches, as well as the active team members. The organization viewed the existence of a strong CI pipeline as a major facilitator of agile, and its importance can also be seen in the resources dedicated to building it, as well as in the actions taken to spread the understanding.

4.3.10 Agile Teams in a Waterfall Organization

A few interviewees commented that the most of the organization was still in a waterfall mode—only the development teams were agile: product management, release management and integration and release testing were seen as working in a waterfall mode.

> We have teams that try to work in agile, but the rest of the organization is not that agile. We have this, release management, I don't see this as an agile setup actually. — Manager

> We still have too much waste. We're still doing waterfall. We have different phases, we have to have PowerPoint slides, we have different checkpoints and meetings, before we can move on. — Team Member

> Product line management is still quite strongly in the waterfall world, that everything should be planned beforehand, and they expect that the feature they ordered will come out as such. — Manager

This was quite true, as the organization had only recently started to involve teams in the early planning activities and the integration and release verification activities took a long time at the end of each release cycle. The organization had recognized this challenge, and actions to remove the rest of the waterfall had already been taken, e.g., building the CI and automated testing systems, and involving teams in the early planning activities: feature investigation and feature concept studies.

The challenges related to product management not being agile, and the need to have teams involved more in the upfront activities were mentioned mainly by managers and a few enlightened team members. Actions to solve these problems were in the early phases, and this did not strike us as one of the main problems in the project. On the other hand, people strongly commented on the need to reduce the release verification time, and as far as possible integrate that activity in the development cycle.

4.3.11 Challenges in Defining the Product Owner Role

Defining the Product Owner (PO) role has not been straightforward. During our first interviews POs were not responsible of the backlog, nor did they participate in backlog prioritization. Instead, the product line organization took care of that.

> We have a separate prioritization meeting where it (backlog) is prioritized. [...] And I'm not sure who are participating in that, but [the] POs aren't there.
>
> — Product Owner

Moreover, a few interviewees complained that the POs did not know enough about the new features to be able to answer the team member's questions. Instead, they were more like messengers.

> It is pretty hard to explain [the role of a PO], but we have had a portfolio manager on one site, who has been sitting on the backlog and is responsible of it. But on the other hand, we have this kind of PO function. And these POs have been more like technical coordinators and messengers of the product management, so they have not been able to do independent prioritization decisions. — Developer

The situation improved during our study period, when a new PO team, called *the PO Cloud* was established. The PO Cloud comprised all POs, a portfolio manager, a test manager, and a user experience lead. The PO Cloud has an end-to-end understanding of the system, making it possible to develop clear functional requirements for the teams based upon a deep understanding of the business requirements from the customers' point of view. Moreover, a workshop was arranged in which the responsibilities of the POs and the product management organization were clarified.

In addition to the PO role, there were many other roles, partly overlapping with the PO role, such as sub-system responsibles and sub-system architects. To our interviewees it was not clear what the responsibilities of these different roles were. Even persons holding these roles complained that the roles and responsibilities were not clear.

> We have way too many overlapping roles, we have architects, POs, portfolio management and others. [...] There's too much discussion that's preventing us to get forward.
>
> — Developer

Even though Scrum does not recognize an architect role, the rapidly growing organization with a complicated product considered it important to have persons responsible for the sub-systems and their architecture. At the time of our interviews, these persons were located in the teams, and some sub-system responsibles and sub-system architects took care of the team coach role, as well. At the end of our study period, there was on-going discussion, on how to clarify the PO and architecture roles, as well as the responsibilities of these roles in the new team structure based on business flows.

The evidence related to the Product Owner role problems came in particular from the Product Owners, sub-system architects and the sub-system responsibles, i.e., the people

who felt that their roles and responsibilities were unclear. The issue was addressed during the study, indicating the importance of having well-defined and understood roles.

4.3.12 Challenges in Breaking Down the Requirements

The organization was still learning how to break down the requirements small enough to implement within one release by one or a couple of teams. At the time of our interviews, most features still took over one release to implement. Starting a big feature, that would require over one release cycle to implement was challenging according to our interviewees, as that team or teams would no contribute to the next release.

> We have been struggling when we have something that is very big and we can see that this is not fitting into our next release. Then it's too big to start and then it's difficult to start. — Manager

The organization had started a discussion on minimum marketable features, but this idea had not yet been implemented in practice.

At the team level the interviewees found it challenging to split the features into user stories small enough to be implemented in a two week sprint.

> The ability to chew it [feature] into smaller subareas, so that we could do something visible in two weeks is still quite bad. — Manager

> Now they [user stories] are huge to implement. I wish they could be smaller, so that they could be implemented during one sprint, preferably even a few stories [per sprint]. — Team Member

Moreover, as different team members still had quite specialized knowledge, the teams often had to start several stories at the same time so that each team member would have work that would fit his or her competences. This indicates that teams worked hard on optimizing resource usage, i.e. minimizing developer downtime rather than optimizing the flow.

While maybe conceptually a serious problem, the fact that the organization was unable to create user stories small enough to be finished in a single sprint did not seem to cause many problems. Interestingly, it seemed rather to be a minor nuisance that would be nice to solve than a major issue. Except for thinking about the concept of a minimum viable product (MVP), no clear actions were taken related to this issue.

4.3.13 Backlog Challenges

A common backlog was considered as one of the most important improvement targets. Thus, it was one of the first improvement actions taken. It was expected to support the new agile way of working, to help streamline the end-to-end flow, to improve visibility and to help to define the lead time of new requirements.

Earlier, several different backlogs had been in use: an electronic backlog management tool was used for issue tracking, and different stakeholders had their own spreadsheets for managing requirements, features and improvements. This led to poor transparency, and made it impossible to define the cycle time of a single requirement and to see the whole end-to-end flow.

Building a common backlog started in early 2013 and was finished in summer 2013. At the time of the interviews, every new feature and improvement was to be added to this single backlog. The common backlog was for high-level features and improvements. Each team had their own backlog where chosen features and improvements were split into user stories.

Our interviewees expressed favorable opinions about the common backlog:

> The good thing is that we have a common backlog. — Manager

However, some challenges were seen, as well. The common backlog was big, i.e., it had a lot of items. In addition, the original idea that any team could pick the next item from the top of the backlog was not seen feasible.

> I don't like this common backlog because it's just a bin of a huge amount of features and improvements. [...] I think they have cut it down to 200 (features) now. So they have to wait a few years to get it out. — Manager

> We actually had problems as someone says that we have a common backlog, but when a team starts going through the first items of it, they couldn't understand anything. They simply don't have the right competences. An item they were able to do was about the twentieth on the list and they were told that they aren't allowed to do that yet. — Product Owner

As mentioned, the reason for this difficulty was the lack of often very specialized knowledge needed to implement a specific backlog item. To help solve this, teams were starting to specialize in specific business flows, and the idea was to have business flow based backlogs, and each business flow having a few teams.

4.3.14 Constant Change

The journey towards agile had been challenging and stressful for everyone as a vast number of changes had been implemented, while working under high pressure from the customers who continuously expected and demanded new features. The product structure, team structures and the process had been changed in parallel with the rapid organizational growth. Moreover, the development organization was globally distributed to five sites.

> From my point of view, we're currently shooting at a moving target, constantly ... It means that, we try to, or someone changes the organization, in order for it to work better, in a lean and agile way. And after a couple of months, they realized that it didn't really work. So they make another change, and so on. — Team member

Moreover, activities not directly contributing to feature development, like the development of the CI and test automation systems, had tied up several teams, leaving fewer resources to work on new features. As the organization aims to constantly improve its way of working, the constant change might never be over, even thought the biggest changes towards agile adoption seemed to be almost done.

The fact that change was constant came up in most interviews, and was thus strongly supported. However, most respondents did not consider it a big issue, and preferred to focus on the fact that the constant change mostly brought improvements to their work.

5 Discussion

In this paper we presented motivation, phases, and challenges and mitigations related to a large-scale agile transformation in a single case organization. As the first in-depth study of an agile transformation, we think that the case study adds significant value both to research and to other organizations with a similarly challenging set-up, needing to customize agile. Next, we discuss our main findings and lessons learned.

5.1 Motivation for Agile Transformations

In this section we discuss the first research question, RQ1: *Why did the organization initiate an agile transformation?*

The organization had three main motivations for the transformation: alignment with the corporate strategy, dissatisfaction with the current way of working, and a need to enable rapid end-to-end deliveries of features and continuous deployment.

The two last motivations have been widely reported in the literature as well. Several cases discuss problems and dysfunctions with their old processes as a motivator for adopting agile, (e.g., O'Connor 2011; Chung and Drummond 2009; Vlaanderen et al. 2012; Hansen and Baggesen 2009; Murphy and Donnellan 2009). The need for rapid deliveries and continuous deployment were related to a need of getting new features to the market faster, i.e. reduce the time-to-market, reported previously (e.g., Gat 2006; Goos and Melisse 2008; Greening 2013; McDowell and Dourambeis 2007; Prokhorenko 2012; Silva and Doss, 2007).

The corporate strategy to adopt agile, naturally had an impact on the program, and helped make the decision to adopt agile. While we think this motivation is becoming increasingly common, in particular with the popularization of "Enterprise Agile", we did not find cases in the literature that explicitly reported this.

5.2 Large-Scale Agile Transformation Phases

In this section we discuss our second research question, RQ2: *How did the transformation proceed?*

The transformation started with a pilot phase with volunteers working in an agile team. This worked remarkably well, which was not unexpected, as the importance and benefits of piloting in agile transformations has been reported by several authors (e.g., Berczuk and Lv 2010; Chung and Drummond 2009; Fecarotta 2008; O'Connor 2011). The pilot team was dismantled and a full-scale transformation was started after only a short time for two reasons: the team quickly was able to show that agile could work well in the organization, and the other parts of the organization started to suffer. The volunteers for the pilot were highly skilled and active developers, and their absence from their component teams was dearly felt, as they now became an "all-star" team, which was not optimal from the organization's point of view. This pilot related problem has been discussed in (O'Connor 2011).

Due to the distributed nature of the organization, different sites had diverging views regarding both the future of the product and the way of working. Thus, there was a need to align the various parts of the organization to make agile work, and to create a strong product and organizational identity. To this end, the organization arranged "value workshops", in which common organizational values were discussed, and people from the various sites were able to meet face-to-face. These value workshops were considered critical from the point of view of transformation success. We are not aware of other reports discussing how to align various parts of a large, distributed organization in relation to, or as part of, an agile transformation.

Continuous integration is a cornerstone of agile, and particularly important for scaling (Gat 2006; Moore and Spens 2008). Implementing CI became the main focus after the value building work. The organization had three dedicated teams working on implementing CI, a known good practice reported in other large-scale agile transformations (Beavers 2007; Fry and Greene 2007; Gat 2006). This amounted to a significant investment in getting CI working, also reported in (Gat 2006; Moore and Spens 2008; Rodríguez et al. 2013). As implementing CI requires not only suitable tools, but also a change in the mindset of

developers, e.g., to start implementing automated tests for new code, a series of CI roadshows were organized. We are not aware of other reports on how to get developer buy in and change the mindset in a full-scale CI rollout.

5.3 Challenges and Mitigations in Large-Scale Agile Transformations

In this section we discuss the third and fourth research questions, RQ3: *What challenges did the organization encounter?* and RQ4: *How did the organization mitigate the challenges?*

During the transformation, the organization encountered several challenges that they tried to mitigate. Next, we discuss the challenges and mitigations according to the challenge classification by (Dikert et al. 2016, Table 11, p. 95) to facilitate easy comparison with the literature.

The first category, "change resistance", a common problem in large-scale agile transformations (Dikert et al. 2016), was also visible in our case. The transformation had challenges in the beginning, as all members of the leadership team did not fully support going agile, and wanted to focus on deliveries rather than transforming the organization. The situation was resolved by reorganizing the leadership team to involve more people with agile experience.

The category "lack of investment", contains the items lack of training, lack of coaching, too high workload, old commitments kept and challenges in rearranging physical spaces. In the current case, we identified lack of training, lack of coaches and coaching, as well as trying keep existing commitments, all of which have been previously reported by other organizations. However, the workspaces had already been completely renovated to support agile development, which explains why we saw no problems related to that.

The category "Agile difficult to implement", contains the items misunderstanding agile concepts, lack of guidance from literature, agile customized poorly, reverting to the old way of working, and excessive enthusiasm. In our case, the organization mentioned the lack of guidance from the literature. Similarly to (Benefield 2008), the organization struggled with finding a good balance between control and autonomy, giving teams too much freedom, as a common agile framework was not emphasized.

According to the literature, the category "Coordination challenges in a multi-team environment" contains items like interfacing between teams difficult, autonomous team model challenging, and global distribution challenges. Cross-site teams posed problems in our case, as half of the teams were distributed between two or even three sites. Surprisingly the case did not experience significant problems with coordination between the teams. This might be partly explained by the existence of the PO team, in which the product owners closely collaborated, well-functioning communities of practice, and the team specialization into business flows.

A particularly prevalent category of issues in our case was "Different approaches emerge in a multi-team environment". The two main issues in this category, according to (Dikert et al. 2016) are that the interpretation of agile differs between teams, and problems in using both old and new approaches side-by-side. The lack of a common framework led to a situation in which teams had different practices, making it difficult to switch teams. Moreover, the surrounding organization was still in a "waterfall mode". For example, product management and release engineering was done mostly in a traditional way. For product management, this was much of a mindset issue. In release engineering the lack of agility could partly be explained by the lack of a working CI system in the early phases of our study. However, the situation improved during our study period when investing in building functioning CI and automated tests. Furthermore, there were challenges in defining the Product Owner role, as product management was unwilling to give the POs the power to

actually prioritize features. This made the POs feel like messengers between product management and the development teams rather than real POs. This situation improved with the implementation of the PO team, and maturation in the understanding of agile in product management.

We were, perhaps surprisingly given the organization's age and business, not able to identify any clear issues related to the category "Hierarchical management and organizational boundaries". Items in this category include the role unclarity for middle managers in agile, management remaining in waterfall mode, keeping the old bureaucracy, and keeping internal silos. Many of these issues were encountered and solved at the biggest site (site A) already during an earlier large-scale transformation.

The category "Requirements engineering challenges" was, not unexpectedly quite visible in the case. The literature study mentions problems with high-level requirements management, challenges of refining requirements, difficulty of defining and estimating user stories, as well as a gap between long and short term planning. In our case, most salient was the challenge of breaking down large features into suitably sized epics and user stories. In the beginning, the organization did not have a common backlog, but several largely independent ones. This issue was resolved in the early phases of the transformation by rolling out a common tool and implementing a single backlog for the whole organization. From the requirements engineering point of view, a large technical debt created problems, as work on it had to be prioritized against development of new features. The common backlog helped to alleviate this problem, as well.

Regarding "Quality assurance challenges", which refers to items accommodating non-functional testing, lack of automated testing, and requirements ambiguity affects QA, our organization faced significant challenges. In the beginning, the lack of test automation and CI created problems, as a huge amount of manual testing had to be done, which reduced the time available for actual development in each iteration. The mitigating actions included serious investment in building a working CI system, as well as CI roadshows to raise awareness about CI and help instill the right mindset in the teams.

In the category "Integrating non-development functions", we have the items other functions unwilling to change, challenges in adjusting to an incremental delivery pace, challenges in adjusting product launch activities, and rewarding model not being teamwork centric. Our organization did not report big issues related to this category. One reason might by that the quite frequent delivery cycle remained the same during the whole transformation.

One particular problem that we identified that we did not find reported in the literature was the fact that the organization was unable to attain the agile ideal of any team being able to work on any item. In addition, teams in the case organization found it difficult to work as "real teams" as they were formed of experts of different components, and learning to work on new components would take a long time. They were gradually mitigating this problem by broadening the knowledge of team members to other components by pair work. Moreover, both of these issues were mitigated by letting teams specialize in specific business flows.

5.4 Lessons Learned

In this section we have collected four lessons that we can learn from this case organization.

5.4.1 Lesson 1: Experimental Transformation Approach

The case organization espoused an "agile mindset" in their experimental transformation approach. The reason for this was that it was difficult to determine up front how to perform

the transformation, despite the fact that part of the organization had previous experiences in transforming a large product development program from waterfall to agile.

In this case, the situation was different from the previous experience: The R&D product development program was not developing a traditional telecom product for operators, but a XaaS platform and services that the customers would not buy, but use as a service. The system consisted of several components, and the organization developing it was new, rapidly growing and highly distributed. In comparison, one previous transformation in the same case company (Paasivaara et al. 2013; Hallikainen 2011) was for a traditional, over ten years old telecom product developed at two sites with a stable organization size. Thus, the set up was quite different. For this reason, the organization felt they could not follow the steps of any previous transformation.

In practice, the experimental approach meant that the organization open-mindedly tried solutions to see if they would work in their setting. If something did not work, it was quickly changed. For example, the organization tried different team set-ups and made changes quickly when problems occurred.

The organization had a mindset that *it is good to try and ok to fail*, since that is the only way to learn. This mindset was pervasive and what we consider *an agile way of implementing agile*. We believe that this is an important aspect of succeeding in large-scale transformation that is widely transferrable to different contexts. The literature review supported this approach (Dikert et al. 2016) as several companies reported that customization of the agile approach is an evolutionary process, see e.g. (Rodríguez et al. 2013; Ryan and Scudiere 2008).

While the literature contains little empirically based guidelines for large-scale agile transformations, the literature on lean and lean software development contain some prescriptive guidelines, e.g. in (Poppendieck 2007; Hibbs et al. 2009; Womack and Jones 2010). Looking at Ericsson's approach, it seems to embody several of the principles suggested by this literature. In particular, the focus on mindset and experimentation is well in line with the suggestions of the proponents of lean.

Mirroring the discussion about this, while this lesson here was learned in the context of large-scale adoption, as evidenced, e.g., by the discussion above about the lean principles, it is likely to be valid also in other, i.e., small organization contexts.

Lesson 1: Consider using an agile mindset and taking an experimental approach to the transformation.

5.4.2 Lesson 2: Stepwise Transformation

As a large-scale lean and agile adoption is a big undertaking, our case organization performed it step by step. Instead of trying to change everything at once, they focused on one change at a time. First, they focused on teams: they experimented to find out a working team set-up and get all agile teams to work well. Second, they aimed to unify the highly distributed organization and to find a common direction by creating and working with common values. Third, they invested in building CI and test automation systems, as well as training the whole organization to be able to contribute to this new way of working.

A big bang transformation approach would probably not have worked well in this case, as the organization had to continue delivering releases to the customers at the same pace as previously during the transformation. Thus, the gradual adoption of practices and structures was considered as a necessity. In this case the step-by-step approach was clearly a successful choice, as it facilitated adoption as everything could not be planned beforehand. The stepwise approach is closely related to the experimental approach, as when starting

the transformation they did not have a plan of the future steps. Instead, they reacted and experimented when changes were needed.

The literature contains reports of both big bang and step-by-step transformation approaches, with step-wise being more commonly used. Often, step-by-step transformations started by piloting, which was reported as one of the success factors (Dikert et al. 2016). Piloting was the first concrete action in our case project, even though it was used for a shorter duration than expected, due to the problems experienced in the rest of the organization, as the component teams lost too many central persons to the pilot team. Still, it was seen as a necessary step in the transformation.

We observed clear high-level transformation steps in our case organization. However, in the literature review, besides piloting, we did not identify clear steps that would be common to all transformations. Thus, this could be an interesting topic for future research.

It could reasonably be argued that stepwise organizational transformation is a widely useful strategy for all kinds of transformations and process improvements initiatives, and indeed, even the venerable CMM(I) takes such an approach. Thus, it is likely that this idea of stepwise improvement is widely applicable, and this case clearly seems to validate such a statement through the success seen here.

Lesson 2: Using a stepwise transformation approach is good in complex large-scale settings, where the transformation takes place during an ongoing development effort. Concentrating on one major topic at a time keeps the attention on the most important change topics.

5.4.3 Lesson 3: Limited Team Interchangeability

In the beginning of the transformation the case organization had a goal that any agile, cross-functional and cross-component team would be able to implement any feature from the top of the common backlog. However, they soon noticed that this was infeasible, due to the product complexity and the asymmetry of competences. The product was composed of several components, each requiring deep technical knowledge. Component specialists were not distributed evenly to different sites. Moreover, each feature would not touch all the components, but only a limited set. Of course, agile teams and individuals can and should broaden their knowledge, but there are limits to what is reasonable and doable.

We believe that also other large organizations implementing agile might find it useful to take into account that the goal of "any agile team being able to implement any feature on top of the backlog" might not be fully feasible.

The solution our case organization identified was to have teams specializing in use-cases spanning several components, or business flows as they called them, with a few teams working in each business flow. Within the business flows, each team could implement end-to-end functionality, from requirement to deployment. This was seen as very important for achieving a fast product development flow, and the end-to-end flow was not compromised. The teams would not need to have deep knowledge on all the components, as the features in a business flow did not touch all components. This solution seemed to work well. Unfortunately, the change took place close to the end of our study period, thus we could not observe whether any further improvements to the setup were needed.

Practitioners have suggested the division of large products into *product areas*, in which teams can specialize (Larman and Vodde 2010). Our previous research has confirmed this (Paasivaara et al. 2013). In this case, however, the difference is that the components in a way formed logical product areas, whereas the use-cases could better be specified into

business flows, crossing several product areas. One can think of this as a matrix structure, as illustrated in Fig. 2.

Lesson 3: In a large-scale complex product any cross-functional team might not be able to work on any item from the product backlog, instead team specialization might be needed.

5.4.4 Lesson 4: Lack of a Common Agile Framework

The organization gave the teams a lot of leeway in how they implemented agile—perhaps too much. The organization started by opting for Scrum, and arranged trainings that, however, everybody did not participate in. The agile background of the project members varied both between and inside distributed sites, as some had participated in trainings and agile projects before, while others had not.

Despite the fact that Scrum was chosen as the basic framework for all teams, only a few actually implemented the majority of the Scrum practices. For example, most teams were not using Sprints, but many did have daily Scrums. As there was a lack of coaches, nobody "forced" the teams to think about the best way for them to work in an agile way. This led to a situation in which some teams, having strong "agilists", conformed quite well to Scrum, but others with less agile experience did not, focusing more on implementation tasks and more or less ignoring the process. Moreover, as people from different sites had not worked together before, and came from different cultures, they did not feel comfortable suggesting: "let's work our way".

Part of the interviewees had experience from another, still ongoing, project at one of the sites (reported in (Paasivaara et al. 2013)). There, the agile transformation had started a few years earlier with heavy support from an external consulting company with common trainings for all, as this was the first agile project at that site. A common Scrum framework was used by all in the beginning, and later on modified towards Scrumban. After the common start the teams got more freedom and took responsibility also in customizing their own way of working. Thus, persons coming from this background to our case project, like many of the coaches and managers, knew that pure Scrum needs to be customized. Moreover, as part of the team members had some agile knowledge already, the teams were given quite free hands to customize, which however did not lead to perfect results.

Close to the end of our study period the coaches were planning to implement similar ways of coaching between the teams and sites, as that would, e.g., make collaboration and changing team members between the teams easier. Thus, they aimed to encourage creating similarities of ways of working in different teams.

Literature emphasizes team autonomy in the way team implements agile practices. *Allowing teams to self-organize* was one of the success factors of large-scale agile transformations (Dikert et al. 2016). On the other hand, *Conforming to a single approach* was mentioned as a success factor as well (Dikert et al. 2016). Finally, *common trainings* and *open events* were suggested for delivering the same message to everybody to ensure that agile understanding across the organization was consistent (Dikert et al. 2016).

When comparing the literature findings to our case, we may hypothesize that the lack of common trainings across the sites, the lack of sufficient and unified coaching and the lack of a clear common approach led to a lack of unified agile mindset and understanding. Thus, giving teams autonomy without enough coaching led to a suboptimal agile implementation in the teams. Interviewees with background from the other transformation within Ericsson, asked for a common framework, as they thought that model had worked well in the previous transformation. Such a framework with common and similar trainings across the sites could

have supported the organization in finding a common direction in agile, and thus providing a common ground for teams to customize the practices later on.

In retrospect, starting with a common agile framework and common trainings seems rather self-evident, and indeed that seems to be the presumption behind any agile implementation, for both large and small organizations. However, the fact that our case organization did not do this, and subsequently run into problems seems to validate this idea, which is increasingly important as the organization grows, as inter-team coordination otherwise becomes very difficult or impossible.

Lesson 4: A lack of common agile framework to start with, a lack of common trainings across sites, and a lack of sufficient and unified coaching may lead to a lack of common direction in the agile implementation.

6 Conclusions

In this paper, we described the large-scale agile transformation of an Ericsson product development program developing a XaaS platform and a set of services towards their future goal of continuous feature delivery. We presented the steps taken, the challenges faced, and the mitigating actions taken, as well as four lessons learned that we think could be applicable to other organizations.

As noted in (Dikert et al. 2016), large-scale agile transformations are seldom easy, and literature provides little advice on how to successfully proceed. Thus, case studies like this one can help provide a basis for a deeper understanding of agile transformations in various contexts that can be used for synthesizing, theory building research.

There is little systematically conducted research on large-scale agile adoption (Dikert et al. 2016). Practitioner literature suggests several scaling frameworks that are actively promoted by their developers. However, independently documented experiences on the usage, customization and benefits of these frameworks is still lacking. Thus, finding validated solutions on what the end result of a transformation should look like or what steps to take is difficult.

As the current agile approaches do not provide good blueprints for what a scaled agile organization should look like, and the recent scaling frameworks are largely unvalidated, there seems to be a need for the organization to tailor its agile approach to fit its own organizational, business and product context. In our case study, for example the various approaches to team organization and the introduction of business flows can be viewed as successful customizations.

This need to customize the agile approach has been reported by other organizations adopting agile in-the-large — successful customization of the agile approach was mentioned as one of the top success factors in an SLR on agile transformations (Dikert et al. 2016).

While all organizations might feel the need to customize their agile approach the issues related to the surrounding organization, the complexity of a large product and the need for specific competences seem to increase the need for method customization. Thus, we think this need is specifically salient in large-scale agile contexts.

For future research, we suggest to conduct additional case studies on large-scale agile transformations, as research in this area is scarce. As the literature review showed (Dikert et al. 2016), only a few case studies exist, even though the topic seems to be highly relevant to large software development organizations moving to agile. Especially, tailoring and customizing of an agile approach to suit different kinds of large-scale organizations would be interesting. In addition, the usage of agile scaling frameworks, such as SAFe, LeSS and

DAD, suggested by consultants, interest companies. However, almost no scientific studies on their usage or suitability to different environments exists.

Acknowledgements The authors would like to thank Ericsson and in particular the interviewees for participating in the study. This work was supported by TEKES as part of the Need for Speed (N4S) SHOK program.

Appendix

A Interview Guide - Round 1 - Transformation Journey

A.1 General Questions

1. Interviewee background (e.g., role and tasks in the organization, history in the organization)
2. Overview of the transformation (e.g., reasons for transformation, goals of the transformation, starting the transformation, agile trainings and coaching, transformation steps)
3. Agile methods and practices (e.g., agile principles followed, agile methods used, agile practices used, your personal opinion about agile)
4. Communication and collaboration (e.g. division of work, inter-team communication and collaboration, collaboration with other sites, interaction with other people in the company, knowledge sharing, Scrum-of-Scrums, communities of practice, successes and challenges in communication and collaboration)
5. Testing and continuous integration (e.g. testing practices, testing levels, test environment, CI goals and practices, releases, release practices, challenges and successes in testing/CI)
6. Challenges and solutions (e.g., biggest challenges of the transformation, solutions implemented/tried, challenges remaining at the moment, solution suggestions)
7. Successes and drawbacks (e.g., successes achieved, benefits of the transformation, benefits of agile, possible drawbacks of agile)
8. Plans for the future (e.g., plans for the next steps, your opinions on what should be done, possible stumbling blocks)
9. Final comments (e.g., anything you would like to comment or add)

A.2 Role Specific Topics and Questions

A.2.1 Managers

1. Overview of the organization (e.g., history of the organization, organization structure earlier, current organization structure)
2. Planning the transformation (e.g., how the transformation was planned, how did you participate in planning / executing the transformation, roadmap for transformation)

A.2.2 Product Owners

1. The role of a Product Owner (e.g., tasks and duties, collaboration with the teams, your role as a Product Owner for remote teams, interaction with other people in the company)

2. Feature handling (e.g., the flow of requirements, interaction with customers or users, interaction with product line, working with backlog, prioritization)

A.2.3 Coaches

1. The role of an agile coach (e.g., how do you work with teams / the rest of the organization, how much time do you spend with teams, how much help teams ask from coaches, how do you promote learning, innovation, and self-organizing, how do you motivate teams)
2. Agile teams (e.g. team formation)
3. Agile methods and practices (e.g., Are teams using text-book Scrum or have you modified Scrum practices to fit better to teams' needs? Are teams allowed to select frameworks they use (e.g., between Scrum and Kanban)?)
4. Coaching Product Owners (e.g., how do you support Product Owners as a coach, how much time do you spend with Product Owners, how much help Product Owners ask from you)
5. Organizational coaching (e.g. how do you participate in organizational coaching, what do you personally do to build a uniform agile the organization, what are the challenges in adopting organization wide agile)
6. Collaboration with other coaches (e.g., what, why, how, how often)

A.2.4 Architects

1. The role of an architect (e.g. tasks and duties, collaboration with development and testing, collaboration with the rest of the organization)
2. The role of architecture (e.g., how is architecture seen in your organization, how is architecture created in practice, how much effort is used to it)

A.2.5 Product Managers

1. Backlog and release (e.g., requirements handling, who makes the decisions of the content of backlogs, backlog prioritization, how do you decide what to include in a release)
2. Relationship with Product Owners (e.g., the division of responsibilities between product manager and Product Owners, collaboration with Product Owners, challenges, good practices, improvement suggestions)
3. Releasing (e.g., release management process in the organization, release frequency, release practices, release team, challenges and successes, improvement goals, improvement suggestions)

A.2.6 Developers

1. Transition to agile (e.g., biggest changes to you as a developer)
2. Agile team (e.g., your teams' tasks/responsibilities, describe your team, team structure, is you team self-organizing, how is your team taking responsibility, team collaboration, team space, trust among team members)
3. Coaching (e.g., help/coaching received, do you / your team get enough support from the coaches, improvement suggestions)
4. Meetings (e.g., meetings that you have / meetings that you or your team members participate, usefulness of the meetings, improvement suggestions)

5. Inter-team coordination (e.g. how it is done, who, when, how often, visibility to what other teams are doing, challenges, successes, improvement suggestions)

B Interview Guide - Round 2 - Value Workshops

1. Interviewee background (e.g., role and tasks in the organization, history in the organization)
2. Beforehand knowledge about the values (e.g., What did you know about values before the value workshop? Do you know why the value workshops were arranged? Do you know where the values come from?)
3. Value workshops (e.g., What do you think about this event? The contents, the way it was arranged, what was good / not good, what could be improved / done differently, benefits of the value workshops for you))
4. Values (How do you feel about the values? Good / bad)
5. After the value workshops (Are you going to do something differently? What should be done after the workshops?)

References

Abdelnour-Nocera J, Sharp H (2007) Adopting agile in a large organization: balancing the old with the new. Tech. rep. The Open University. Faculty of Mathematics and Computing, Department of Computing

Abdelnour-Nocera J, Sharp H (2008) Adopting agile in a large organisation. In: Agile processes in software engineering and extreme programming, proceedings, LNBIP, vol 9, pp 42–52

Ambler SW (2012) Disciplined Agile Delivery: a practitioner's guide to agile software delivery in the enterprise. IBM Press

Banerjee P, Friedrich R, Bash C, Goldsack P, Huberman B, Manley J, Patel C, Ranganathan P, Veitch A (2011) Everything as a service: powering the new information economy. Computer 44(3):36–43. https://doi.org/10.1109/MC.2011.67

Beavers P (2007) Managing a large "agile" software engineering organization. In: Agile Conference (AGILE), 2007 pp 296–303

Benefield G (2008) Rolling out agile in a large enterprise. In: Hawaii International Conference on System Sciences, Proceedings of the 41st Annual, p 461

Berczuk S, Lv Y (2010) We're all in this together. Software, IEEE 27(6):12–15

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

Chung MW, Drummond B (2009) Agile at yahoo! from the trenches. In: Agile Conference, 2009. AGILE '09., pp 113 –118

Cunningham W (1992) The wycash portolio management system. In: Proceedings of OOPSLA

Dikert K, Paasivaara M, Lassenius C (2016) Challenges and success factors in large-scale agile transformations: A systematic literature review. J Sys Softw

Dingsøyr T, Moe N (2013) Research challenges in large-scale agile software development. SIGSOFT Softw Eng Notes 38(5):38–39

Dingsøyr T, Moe N (2014) Towards principles of large-scale agile development. In: Dingsøyr T, Moe N, Tonelli R, Counsell S, Gencel C, Petersen K (eds) Agile Methods, Large-Scale Development, Refactoring, Testing, and Estimation, Lecture Notes in Business Information Processing, vol 199, Springer International Publishing, pp 1–8, https://doi.org/10.1007/978-3-319-14358-3_1

Dybå T, Dingsøyr T (2008) Empirical studies of agile software development: a systematic review. Inf Softw Technol 50(9-10):833–859
Fecarotta J (2008) Myboeingfleet and agile software development. In: Agile, 2008. AGILE '08. Conference, pp 135 –139
Freudenberg S, Sharp H (2010) The top 10 burning research questions from practitioners. Software, IEEE 27(5):8–9
Fry C, Greene S (2007) Large scale agile transformation in an on-demand world. In: Agile Conference (AGILE), 2007, pp 136 –142
Gat I (2006) How bmc is scaling agile development. In: Agile Conference, 2006, pp 6–320
Goos J, Melisse A (2008) An ericsson example of enterprise class agility. In: Agile, 2008. AGILE '08. Conference, pp 154–159
Greening D (2013) Release duration and enterprise agility. In: System Sciences (HICSS), 2013 46th Hawaii International Conference on, pp 4835–4841
Hallikainen M (2011) Experiences on agile seating, facilities and solutions: multisite environment. In: Global Software Engineering (ICGSE), 2011 6th IEEE International Conference on, pp 119–123
Hansen M, Baggesen H (2009) From cmmi and isolation to scrum, agile, lean and collaboration. In: Agile Conference, 2009. AGILE '09., pp 283–288
Hanssen G, Smite D, Moe N (2011) Signs of agile trends in global software engineering research: a tertiary study. In: Global Software Engineering Workshop (ICGSEW), 2011 Sixth IEEE International Conference on, pp 17–23, https://doi.org/10.1109/ICGSE-W.2011.12
Hibbs C, Jewett S, Sullivan M (2009) The art of lean software development: a practical and incremental approach. Theory Prac, O'Reilly Media, https://books.google.fi/books?id=sBy4OrfZyYsC
Highsmith J, Cockburn A (2001) Agile software development: the business of innovation. Computer 34(9):120–127
Holmstrom H, Fitzgerald B, Agerfalk PJ, Conchuir EO (2006) Agile practices reduce distance in global software development. Inf Syst Manag 23(3):7–18. https://doi.org/10.1201/1078.10580530/46108.23.3.20060601/93703.2
Hossain E, Babar MA, Paik Hy (2009) Using scrum in global software development: a systematic literature review. In: Proceedings of the 2009 Fourth IEEE International Conference on Global Software Engineering, IEEE Computer Society, Washington, DC, USA, ICGSE '09, pp 175–184
Jick TD (1979) Mixing qualitative and quantitative methods: triangulation in action. Adm Sci Q 24(4):602–611
Korhonen K (2012) Evaluating the impact of an agile transformation: a longitudinal case study in a distributed context. Softw Qual J 21(4):599–624
Kruchten P, Nord RL, Ozkaya I (2012) Technical debt: from metaphor to theory and practice. IEEE Software 29(6)
Larman C, Vodde B (2010) Practices for scaling lean & agile development: large, multisite, and offshore product development with large-scale scrum. Addison-Wesley Professional Boston. MA, USA
Larman C, Vodde B (2015) Less framework. http://less.works/
Leffingwell D (2007) Scaling software agility: best practices for large enterprises. Addison-Wesley Professional
Leffingwell D (2015) Scaled agile framework. http://scaledagileframework.com/
Lindvall M, Muthig D, Dagnino A, Wallin C, Stupperich M, Kiefer D, May J, Kahkonen T (2004) Agile software development in large organizations. Computer 37(12):26–34
Livermore JA (2008a) Factors that significantly impact the implementation of an agile software development methodology. J Softw 3(4):31–36
Livermore JA (2008b) Factors that significantly impact the implementation of an agile software development methodology. J Softw 3(4):31–36
Long K, Starr D (2008) Agile supports improved culture and quality for healthwise. In: Agile, 2008. AGILE '08. Conference, pp 160 –165
McDowell S, Dourambeis N (2007) British telecom experience report: Agile intervention - bt's joining the dots events for organizational change Agile processes in software engineering and extreme programming, proceedings, LNCS, vol 4536, pp 17-23
Moore E, Spens J (2008) Scaling agile: Finding your agile tribe
Murphy P, Donnellan B (2009) Lesson learnt from an agile implementation project Agile processes in software engineering and extreme programming, LNBIP, vol 31, pp 136-141
O'Connor C (2011) Anatomy and physiology of an agile transition. In: Agile Conference (AGILE), 2011, pp 302 –306

Paasivaara M, Lassenius C (2014) Communities of practice in a large distributed agile software development organization – case ericsson. Inf Softw Tech 56(12):1556–1577. https://doi.org/10.1016/j.infsof.2014.06.008, http://www.sciencedirect.com/science/article/pii/S0950584914001475, special issue: Human Factors in Software Development

Paasivaara M, Lassenius C, Heikkila V, Dikert K, Engblom C (2013) Integrating global sites into the lean and agile transformation at ericsson. In: Global Software Engineering (ICGSE), 2013 IEEE 8th International Conference on, pp 134–143, https://doi.org/10.1109/ICGSE.2013.25

Paasivaara M, Behm B, Lassenius C, Hallikainen M (2014a) Towards rapid releases in large-scale xaas development at ericsson: A case study. In: Global Software Engineering (ICGSE), 2014 IEEE 9th International Conference on, pp 16–25, https://doi.org/10.1109/ICGSE.2014.22

Paasivaara M, Väättänen O, Hallikainen M, Lassenius C (2014b) Supporting a large-scale lean and agile transformation by defining common values. In: Proceedings of the Workshop on Principles of Large-Scale Agile Development (in press)

Patton MQ (1990) Qualitative evaluation and research methods, 2nd edn. Sage Publications, Newbury Park, Calif

Petersen K, Wohlin C (2010) The effect of moving from a plan-driven to an incremental software development approach with agile practices. Empir Softw Eng 15(6):654–693

Poppendieck M (2007) Poppendieck, T. From concept to cash. Pearson Education, Implementing lean software development

Poppendieck M, Cusumano M (2012) Lean software development: A tutorial. Software, IEEE 29(5):26–32. https://doi.org/10.1109/MS.2012.107

Prokhorenko S (2012) Skiing and boxing: coaching product and enterprise teams. In: Agile Conference (AGILE), 2012, pp 191–196

Ranganath P (2011) Elevating teams from 'doing' agile to 'being' and 'living' agile. In: Agile Conference (AGILE), 2011, pp 187–194

Rodríguez P, Mikkonen K, Kuvaja P, Oivo M, Garbajosa J (2013) Building lean thinking in a telecom software development organization: strengths and challenges. In: Proceedings of the 2013 International Conference on Software and System Process, ICSSP'13, pp 98–107

Rodríguez P, Haghighatkhah A, Lwakatare LE, Teppola S, Suomalainen T, Eskeli J, Karvonen T, Kuvaja P, Verner JM, Oivo M (2016) Continuous deployment of software intensive products and services: a systematic mapping study. J Syst Softw

Ryan J, Scudiere R (2008) The price of agile is eternal vigilance. In: Agile, 2008. AGILE '08. Conference, pp 125–128

Schwaber K, Beedle M (2002) Agile software development with scrum. Series in agile software development, Prentice Hall

Silva K, Doss C (2007) The growth of an agile coach community at a fortune 200 company. In: Agile Conference (AGILE), 2007, pp 225–228

Stavru S (2014) A critical examination of recent industrial surveys on agile method usage. J Syst Softw 94:87–97

VersionOne Inc (2016) 10th annual "state of agile development" survey. https://versionone.com/pdf/VersionOne-10th-Annual-State-of-Agile-Report.pdf

Vlaanderen K, Van Stijn P, Brinkkemper S, Van De Weerd I (2012) Growing into agility: process implementation paths for scrum. In: Product-focused software process improvement, proceedings, LNCS, vol 7343, pp 116-130

Wenger E, McDermott R, Snyder WM (2000) Communities of practice: the organizational frontier. Harvard Business Review (Jan-Feb):139–145

Wenger E, McDermott R, Snyder WM (2002) Cultivating communities of practice harvard business review press. MA, Cambridge

Womack JP, Jones DT (2010) Lean thinking: banish waste and create wealth in your corporation. Simon and Schuster

Yin RK (2009) Case study research: design and methods, 4th edn. SAGE Publications, Thousand Oaks, CA, USA

Maria Paasivaara is an Associate Professor at the IT University of Copenhagen and an Adjunct Professor at Aalto University. Her research interests include software engineering processes and practices, continuous software engineering, agile software development, large-scale agile, DevOps, software projectmanagement, global software engineering and software engineering educational research. She performs empirical research in close collaboration with industrial and academic partners and aims at solving real-world problems that are important to the software industry. She has a D.Sc. degree from Helsinki University of Technology.

Benjamin Behm is a software developer at Agilefant Inc. He did his Master's thesis on large-scale agile software development, and now builds tools to support large and small organizations to leverage agile in their daily operations.

Casper Lassenius is an Associate Professor at Aalto University. His research is in the field of empirical software engineering, with recent interests including agile software development in the small and large, continuous software engineering, DevOps, quality assurance, and global software engineering. He has a D.Sc. degree from Helsinki University of Technology.

Minna Hallikainen is a line manager and a change driver at Ericsson R&D Finland whose passion is to work with people. Minna practically walks the talk, as she also is a trainer and a coach. Minna is a trainer in the Ericsson Product Development Leadership program worldwide training and coaching the leaders. Minna writes publications and is a blogger in the Ericsson Careers blog. Minna is also wellness mentor, gym coach and personal trainer, who loves boating on her own boat and dancing and playing music.