Investigating the Adoption and Application of Large-Scale Scrum at a German Automobile Manufacturer

Ömer Uludağ*, Martin Kleehaus*, Niklas Dreymann*, Christian Kabelin†, Florian Matthes*
*Chair for Informatics 19, Technische Universität München (TUM), D-85748, Garching
Email:{oemer.uludag,martin.kleehaus,niklas.dreymann,matthes}@tum.de
†Ventum Consulting, D-80797, München
Email:{christian.kabelin}@ventum.de

***Abstract*—Over the last two decades, agile methods have been adopted by an increasing number of organizations to improve their software development processes. In contrast to traditional methods, agile methods place more emphasis on flexible processes than on detailed upfront plans and heavy documentation. Since agile methods have proved to be successful at the team level, large organizations are now aiming to scale agile methods to the enterprise level by adopting and applying so-called scaling agile frameworks such as Large-Scale Scrum (LeSS) or Scaled Agile Framework (SAFe). Although there is a growing body of literature on large-scale agile development, literature documenting actual experiences related to scaling agile frameworks is still scarce. This paper aims to fill this gap by providing a case study on the adoption and application of LeSS in four different products of a German automobile manufacturer. Based on seven interviews, we present how the organization adopted and applied LeSS, and discuss related challenges and success factors. The comparison of the products indicates that transparency, training courses and workshops, and change management are crucial for a successful adoption.**

***Index Terms*—large-scale agile development, scaling agile frameworks, large-scale scrum, case study**

I. Introduction

Emerging in the 1990s, agile software development methods such as Extreme Programming [1] and Scrum [2] have transformed and brought unprecedented advancements to software development practice by emphasizing change tolerance, team collaboration, and customer involvement [3], [4]. With these methods, small, co-located, self-organizing teams work closely with the business customer on a single-project context, maximizing customer value and software product quality through rapid iterations and frequent feedback loops [3]. Since agile methods have proved to be successful at the team level, large organizations are now aiming to scale agile methods to the enterprise level [5]. Version One's 12th survey on the state of agile [6] also reflects this industry trend towards adopting agile methods in-the-large. The survey shows that 52% of the 1492 respondents work in companies where the majority of teams are agile. However, the adoption entails new challenges, such as inter-team coordination, dependencies to other existing environments or general resistances to changes [7], [8]. Mainly promoted by consultants, scaling agile frameworks such as the Scaled Agile Framework (SAFe), Disciplined Agile Delivery (DAD), and Large-Scale Scrum (LeSS) [4], [5] pledge to resolve the aforementioned issues. Although the number of organizations using scaling agile frameworks is increasing, scientific literature providing in-depth case studies on the adoption and application is still scarce [6], [7]. We aim to fill this gap by presenting a case study on the adoption and application of LeSS in four different products of a German automobile manufacturer. Based on these objectives, our two research questions are:

- *Research Question 1 (RQ 1): How has LeSS been adopted in different products?*
- *Research Question 2 (RQ 2): How is LeSS applied in different products?*

The remainder of this paper is structured as follows. In Section II, we present the background of our paper and provide an overview of related works. In Section III, we present the research approach of this paper. We briefly describe the case organization and present the results of our case study in Section IV. We discuss the main findings in Section V before concluding the paper with a summary of our results and remarks on future research in Section VI.

II. Background and Related Work

A. Large-Scale Scrum

The LeSS framework (see Figure I) was released in 2008 based on the experiences of Craig Larman and Bas Vodde [9]. It extends Scrum with scaling rules and guidelines without losing sight of Scrum's original goals. Unlike traditional Scrum, LeSS specifies organizational changes. Furthermore, it aims to facilitate coordination between multiple Scrum teams by having a product owner (PO) responsible for a central backlog and several teams. Coordination between teams is done similarly to Scrum where they perform a sprint planning and sprint review. For smaller products, all product members join the same sprint planning and review. For bigger products, a team representative should be sent to the meetings. Although LeSS aims to solely work on principles, it still comprises the following four components [10]:

- **Rules:** Rules define the foundation of LeSS. Similar to Scrum, the focus lies on the structure of teams, roles

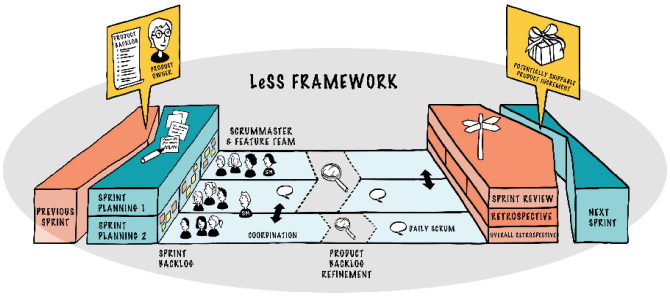

Fig. I: Overview of the LeSS framework [10].

within the team, definition of the requirements of the product, and the development process.

- **Principles:** Principles provide answers on how to apply LeSS in specific enterprise contexts.
- **Guides:** Guides support the adaptation of the rules and a subset of the experiments by providing tips and best practices.
- **Experiments:** LeSS encourages teams to experiment, fail, and learn new concepts.

When organizations aim to scale over eight teams, *LeSS Huge* should be used. LeSS Huge introduces additional elements that are necessary to manage hundreds of people, such as the concept of requirements areas (RAs). RAs are organized around customer-centric requirements. All RAs follow the same sprint cadence and aim for continuous integration across the entire product. An area PO (APO) focuses on one RA and is responsible for an area product backlog (APB). The APO acts essentially the same way as the PO would in the smaller LeSS framework.

B. Related Work

According to Version One's 12th survey on the state of agile, 29% of 1,492 respondents reported using SAFe and 5% LeSS. Although the number of organizations using scaling agile frameworks is increasing, there exists only a handful of papers documenting reported usages [7]. However, some literature reviews have been conducted by academics for studying scaling agile frameworks scientifically. For instance, Alqudah and Razalo [5] provide a literature review on scaling agile frameworks. Therein, they compare seven identified scaling agile frameworks such as DAD, LeSS, Nexus, SAFe, and Spotify based on five criteria: team size, training and certification, adopted methods and practices, required technical practices, and organization type. Based on a structured literature review, Uludağ et al. [11] give a primary analysis of 20 identified scaling agile frameworks such as DAD, Enterprise Scrum, LeSS, Nexus, SAFe, Scrum at Scale, and Spotify. Last but not least, Putta et al. [12] provide a multivocal literature review, which includes both peer-reviewed and non-peer reviewed case studies and experience reports on organizations that have adopted SAFe. Besides these secondary studies, some researchers also present primary studies in the form of case studies on the adoption of scaling agile frameworks. For example, Pries-Heje and Krohn [13] describe a case study from the financial software company SimCorp on how they adopted SAFe. Further, Paasivaara [14] provides a case study highlighting success factors and challenges of the SAFe adoption in the globally distributed software development company Comptel. Moreover, Paasivaara and Lassenius [15], [16] describe a case-study on scaling Scrum in a large globally distributed software development project at the global telecommunications company Nokia. Therein, the authors describe significant challenges the project faced when applying LeSS. Last, Uludağ et al. [17] present a case study from a large insurance company on how they combined domain-driven design with LeSS to support a large-scale agile development program with three agile teams. Although there are some primary and secondary studies, they merely mention LeSS in passing. To the best of our knowledge, literature documenting in-depth case studies on the adoption and application of LeSS in a real-life context is still scarce.

III. Case Study Design

A case study is a suitable research methodology for this paper as we aim to study a contemporary unexplored phenomenon, namely the adoption and application of LeSS, in a complex, real life context [18]. We followed the guidelines described by Runeson and Höst [18] for the case study research process.

Case study design: The main objective of this paper is to study the adoption and application of LeSS in a large IT organization. Based on stated objective, we defined two research questions (see Section I). Our single-case study is of exploratory nature as we analyze an unexplored phenomenon [18]. The case organization was purposefully selected as it provides a unique opportunity to compare different occurrences of LeSS inside a single company and as it represents an information-rich case [19]. Our units of analysis are four products at the car manufacturer's IT department which use LeSS to develop complex software.

Preparation for data collection: We focused on first degree data collection techniques according to Lethbridge et al. [20]. Hence, we collected data by conducting seven semi-structured interviews with one feature team (FT) member, one PO, one agile coach (AC), one scrum master (SM), and three different architects (EA - enterprise architect, SA - solution architect and IA - IT enterprise architect) in four different products of which all adopted and applied the LeSS framework (see Table I). All interviews lasted between 45 minutes to one hour each. The interviews followed a semi-structured questionnaire and were rather conversational in nature to maintain adaptability to the roles and individual experiences of the interviewees. We interviewed seven different roles from four different products to enable triangulation of data sources [18].

Analysis of collected data: All interviews were transcribed and coded using open coding as suggested by Miles et al. [21]. After creating preliminary codes, we refined and consolidated our codes by merging related ones and removing duplicates. Based on that, we looked at groups of code phrases and merged

TABLE I: Overview of interviewed roles and their assigned products.

Role	MPN	OTD	PPM	PFS
PO	-	-	-	1
FT	-	-	-	1
AC / SM	1	1	-	-
EA / SA / IA	-	-	3	-
Total	**1**	**1**	**3**	**2**

them into concepts, which were later related to our formulated research questions.

IV. Results

A. Case Description

The case under investigation concerns a German multinational company that currently manufactures luxury cars and motorcycles and employs more than 120,000 people within the whole organization and around 4500 people in its IT department. In the past, the IT department focused mainly on standardization and cost optimization of the running IT. Driven by digitalization, the IT management saw the increase in agility of the IT department as an essential improvement for its performance and flexibility. At the end of 2016, the IT management decided to transform the IT department into an agile product organization over the next two to three years following the slogan *"100 % agile"*. It committed itself to the goal of transforming all current IT projects to agile and aligning the IT organization completely with agile principles by the end of 2019. Contemporaneously, the autonomous driving department of the case organization chose LeSS as its new working model with the aim of achieving easier communication and coordination, more transparency and shorter decision-making paths throughout the entire department. Success stories of the autonomous driving department with LeSS reached the IT department. In parallel the IT management allowed individual IT projects to choose an appropriate scaling agile framework for themselves, which is why many IT projects decided to adopt LeSS at the beginning of 2017. Still, a large part of software development activities is outsourced to external partners working partly plan-driven.
The four investigated IT projects (from now on products): "Material Part Number" (MPN), "Order-to-Delivery" (OTD), "Product & Price Master Data" (PPM), and "Price Finding Service" (PFS) also decided to adopt LeSS for their product development (see Figure II). MPN aimed to replace the case organization's old legacy system for managing and storing its bills of materials. OTD developed, maintained, and improved IT systems for the case organization's intra-plant logistics. PPM was responsible for the sustainable design of the case organization's master data application. PFS was responsible for the development of a new pricing software.

B. LeSS Adoption

We used exploratory questions for the following categories: (1) timing and duration, (2) reasons and to-be-addressed problems, (3) combined frameworks, (4) training, (5) adaptations, (6) challenges, and (7) lessons learned to evaluate the adoption of the framework.
All four products had recently adopted LeSS for no longer than two years. However, the duration of the adoptions varied between three months to one year because of different complexities. LeSS was introduced to handle inter-team coordination and to balance the limitations of Scrum in large-scale projects. Additional reasons were its moderate degree of complexity while maintaining sufficient guidance for the coordination of multiple agile teams working on the same product. Before the adoption, the products had problems with synchronizing teams, managing their dependencies, and information loss between teams. LeSS was also selected to enable the transition from traditional project thinking to product and customer orientation. During the adoption, LeSS was combined with preexisting lean and agile methods such as Kanban and DevOps. For instance, MPN and PPM adopted LeSS in combination with Kanban to manage epics and features and to track the progress of tasks using Kanban boards. In all products, the adoption was accompanied by a comprehensive training of all employees which range from one single presentation by the SM / AM to a two-days individually adjusted workshop with external experts since employees possess varying levels of prior knowledge in agile software development. The adoption of LeSS in each product led to individual adjustments, e.g., all products extended LeSS by a domain level with a superordinate portfolio layer to coordinate all products within the IT department and align them with the organization's strategic objectives. A so-called "one-calendar" approach, centrally managed by the sub-units of the IT department, eased the adoption by enabling common sprint cadences across products. Although this approach was perceived as contradicting agile values, especially in terms of self-organization, one interviewee stressed its importance:

"*Actually, it's about self-organization in agile. But with a few hundred teams, it's difficult if one has a meeting there or the other one has the daily at other time. Somehow they have to synchronize. Then this one-calendar was our approach to bring structure into it.*"

— SM, OTD

We identified four common problems during the adoption of LeSS. First, although LeSS is minimalist and tries to define roles, artifacts, or processes that are at least needed for large-scale agile development [10], the products had concerns regarding numerous coordination meetings. The SM of OTD described the problem as follows:

"*Many people are simply arrested in thousands of other meetings and if the PO is missing at a meeting such as sprint planning, this is quite sub-optimal.*"

TABLE II: Overview of interviewed products and general information.

	MPN	OTD	PPM	PFS
Start of Adoption	April 2017	Early 2017	End 2017	March 2018
Number of involved Employees	∼600	∼70	∼90	∼40
Sites	Germany	Germany, South Africa	Germany, Poland, Portugal, Russia	Germany, Portugal, Russia
Number of Feature Teams	15	5	6	3
Sprint Lengths (in weeks)	3	3	3	3
LeSS Adoption	LeSS	LeSS	LeSS Huge	LeSS

— SM, OTD

Having too many meetings can reduce the productivity of the employees:

"*The PO attends so many meetings that he doesn't have time to write user stories himself. . . . And if you haven't written them yourself, it's incredibly difficult to accept them.*"

— PO, PFS

Second, due to the currently organizational structure, each product had two POs, one from the specialist department and one from the IT department, which led to the so-called "dual leadership" (see Section IV-C). Third, the employees feared losing status and power. This problem affected primarily middle management employees that were partly retrained to POs during the adoption. Fourth, the adoption was hampered by the lack of agile mindset and understanding about LeSS.
Last but not least, we asked all interviewees about lessons learned. Ex post, clear communication of the big picture of the adoption can increase transparency and sharpen awareness of involved stakeholders. In addition, one FT member mentioned that practical exercises in training courses can ease the adoption:

"*At first glance, it makes the adoption of LeSS easier when practicing and understanding the theory in small projects, preferably in a playful way.*"

— FT, PFS

A summarizing comparison of the four LeSS adoptions can be found in Table III.

C. LeSS Application

We asked the interviewees how their products applied: roles, artifacts, and processes.

Roles: All three standard roles, namely FT, PO, and SM are included in the respective products (see Table IV). In contrast to Scrum, the number of FTs (development teams in Scrum) is two to eight within LeSS. In LeSS Huge, this number can be scaled even higher. Although MPN has 15 FTs, it decided to use multiple LeSS implementations because the subordinate products do not have an overarching product character. FTs of OTD, PPM, and PFS were spread across the world. According to the SM in OTD, one problem was that FTs were still too heavily controlled by external factors and therefore were not entirely self-organized. For PPM, it appeared to be less of a problem. PFS' FTs were already 90 – 95% self-organized. For this reason, there is great potential for improvement by allowing FTs to act as self-organized units. As already indicated in Section IV-B, the role of the PO was shared by several persons, i.e., one PO was responsible for the business side and one for the IT side. This arose from the fact that one person alone did not have the required knowledge to manage the product properly, so the products decided to split the responsibilities. The "dual leadership" created difficulties in coordination for external and internal teams.

"*. . . there still exists one PO on the business and one PO one the IT side but then in the form of a dual leader. If you look at LeSS or Scrum then I have to say having a dual leader is the worst thing to do.*"

— EA, PPM

The suggested improvement was to remove dual leadership by bundling responsibilities into one PO. The EA of PPM suggested that the IT PO should work more with FTs and act as a substituting PO where he could collect valuable insights. In addition, PPM had six product owners who were responsible for subareas, representing APOs as suggested by LeSS Huge [10]. Last but not least, all products introduced the role of the Domain PO (DPO) with more traditional project management functions. Their responsibility included synchronization of the FTs and planning the budget and capacities. They also took overall responsibility for the products and coordinated at portfolio level with each other.

"*They [DPOs] are actually only responsible for the budget and capacities and should not be involved in the actual work.*"

— EA, PPM

Figure II provides an overview of the different PO roles in the respective products. In OTD, the role of the SM did not exist per se, but instead the role of the agile master (AM). The AM of OTD was not only responsible for helping FTs

TABLE III: Comparison of the LeSS adoptions in the four products.

	MPN	OTD	PPM	PFS
Reasons for Adoption	• Simplicity of LeSS • Optimizing product cut • Dealing with high project complexity • Enabling change from project to product	• System optimization goals from LeSS • Success stories of the autonomous driving department • Three-day introduction to LeSS by Craig Larman • Enabling change from project to product	• Dealing with high project complexity • Enabling change from project to product	• Reducing dependencies between agile teams • Minimizing loss of information between agile teams • Handling inter-team coordination
Combined Frameworks	• Combined LeSS with SAFe to have a superordinate portfolio level • Combined LeSS with Kanban to manage epics and features	• Combined LeSS with DevOps so that FTs have the required skill-set and the full responsibility to release software	• Combined LeSS Huge with Kanban to track and monitor the progress of tasks	–
LeSS Training	• 2-day training program for each employee extended with LeSS	• SM coaching • External agile coaches train internal employees who in-turn coach FTs	• 2-day training program for each employee extended with LeSS Huge • External agile coaches train internal employees	• External agile coaches train internal employees
Adaptations	• LeSS extended by a domain level • Common sprint cadences are facilitated by a one-calendar approach	• LeSS extended by a domain level • Common sprint cadences are facilitated by a one-calendar approach	• LeSS Huge extended by a domain level	• LeSS extended by a domain level
Challenges	• Sharing a common vision • Fear of losing power • Communication gaps between products • High complexity of the product • Correct product cuts • Balancing up-front planning vs. emergent design • Dual leadership of the PO role	• Numerous coordination meetings • High complexity due to the number of FTs • Low prior knowledge of agile methods • Traditional line responsibilities of employees complicate their focus on agile working • Dual leadership of the PO role	• Numerous coordination meetings • Splitting large requirements into smaller requirements • APOs keeping own APBs • Synchronizing APBs • Dealing with cultural differences between FTs • Missing transparency regarding roles and responsibilities • Dual leadership of the PO role	• Numerous coordination meetings • Fear of losing power • Specialist departments have low knowledge of LeSS • Establishing agile mindset • Changing roles and responsibilities due to agile working • Dual leadership of the PO role
Lessons Learned	• Communicating the big picture of the adoption to obtain the commitment of all stakeholders • Performing Inspect and Adapt at regular intervals	• Performing Inspect and Adapt at regular intervals	• Reducing responsibilities to a few people to increase decision-making speed	• Practical exercises in training courses deepen the understanding of LeSS

TABLE IV: Identified results for the application of roles within the products.

Role	MPN	OTD	PPM	PFS
FT	✓	✓	✓	✓
PO	✓	✓	✓	✓
SM	✓	✓	✓	✓
Additional Roles	✓	✓	✓	✓

to apply LeSS and self-organize, but also for the change management and people management:

"*The agile master is responsible for the methodology, i.e., method training, shielding the team from stakeholders who want something, eliminating impediments, and organizing events.*"

— SM, OTD

All products involved additional roles which go beyond LeSS. For instance, PFS included the role of the PO support, who was responsible for creating user stories, as the PO had no time for this. PFS also introduced the business analyst (BA) role, who was responsible for dealing with the problems and requirements of FTs. He was also the first point of contact for external parties. Each product was

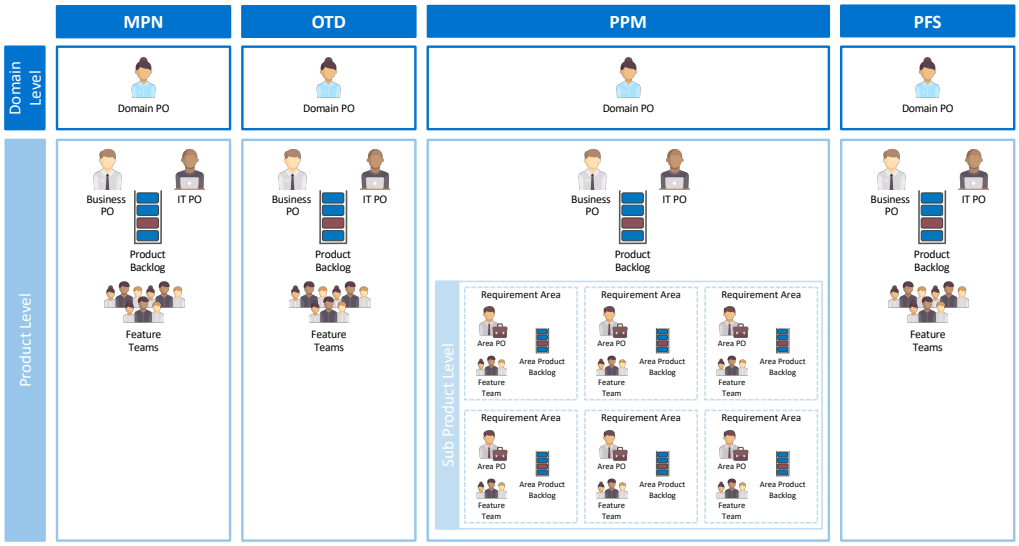

Fig. II: Overview of the different PO roles in the four products.

accompanied by different types of architects, such as IT architects, enterprise architects, and solution architects that mainly acted as external consultants for various architectural topics. Such topics include: showing the positioning of the product in the overall organization context, guiding FTs in the realization of the to-be architecture of the product, and ensuring that FTs adhere to architectural standards. In this context, the EA of PPM also stated that:

"*My personal opinion is that when somebody does not know how to continue they call the architects. They are really just ad-hoc fire extinguishers, who have to come quickly.*"

— EA, PPM

Artifacts: Table V provides an overview of used artifacts. In general, there existed a product backlog for all products. The main difference was that MPN, OTD, and PFS had one product backlog, whereas PPM had one common and six RA-specific product backlogs, so-called area product backlogs. According to one interviewee, this caused the following problem: if there is only one common product backlog on product level for several sub areas, the sole prioritization based on importance becomes difficult, as it cannot generally be said that one sub product is more important than another one. In general, all products had a sprint backlog, sprint goal, and product increments as artifacts. The definition of done (DoD) was implemented by all products and has been discussed at a great deal as they were not fully used. The PO of PFS described the DoD as particularly important, since it defines clear expectations regarding user stories towards external partners. However, PFS was the only product that had a finished DoD. Instead, the interviewee in MPN stated:

"*Basic artifacts as the DoD do exist in every product, however are not used.*"

TABLE V: Identified results for the application of artifacts within the products.

Artifacts	MPN	OTD	PPM	PFS
Product Backlog	✓	✓	✓	✓
Sprint Backlog	✓	✓	✓	✓
Sprint Goal	✓	✓	✓	✓
Product Increment	✗	✓	✓	✗
Definition of Done	✓	✓	✓	✓
Additional Artifacts	✓	✗	✓	✓

— AC, MPN

This was validated by the IA of PPM:

"*Probably it [the DoD] was introduced once in the beginning and now it is in some corner, somewhere in Confluence on some subordinated page, which nobody has accessed for a long time. This sounds very plausible to me.*"

— IA, PPM

The respondents considered the lack of use of the DoD to be problematic, as the product increment was not assessed according to predefined criteria. OTD followed a different approach. An external team developed a general DoD, and then the PO adapted it for his respective features to fit individual requirements. Some interviewees mentioned two additional artifacts. PPM and PFS also used the definition of entry (DoE) which was created by the PO. The DoE was used to describe rough requirements for individual user stories. In addition, MPN, PPM, and PFS used the definition of ready (DoR) as the last step prior to implementation. The DoR was

utilized to describe user stories ready for implementation". However, one interviewee (SA) in PPM stated that the DoE eventually vanished.

TABLE VI: Identifief results for the application of processes within the products.

Processes	MPN	OTD	PPM	PFS
Sprint	✓	✓	✓	✓
Product Backlog Refinement	✓	✗	✓	✓
Sprint Planning (1 & 2)	✓	✓	✓	✓
Daily Scrum	✓	✓	✓	✓
Sprint Review	✓	✓	✓	✓
Retrospective (Team & Overall)	✓	✓	✓	✗
Additional Processes	✓	✓	✓	✓

Processes: In general, all LeSS events were widely adopted in all products (see Table VI). According to the SA of PPM, it was difficult to get all relevant people to attend meetings. Meetings were planned in the products at the same time (one-calendar approach), which made cooperation and coordination between people in different meetings very difficult. In addition, retrospectives within PFS were not implemented yet:

"*Well, we haven't been following the retro so closely so far, but we have decided to do more.*"

— FT, PFS

All products organized communities of practices (CoPs) for collaboration and information exchange within technical and business domains across products. One interviewee also added that:

"*Communities of practices are very helpful for the coordination of processes, methods, and tools as well as for comprehensive harmonization and standardization.*"

— SM, OTD

The architecture CoP was the most outstanding CoP within all products as they were implemented by all products. There, architects and other stakeholders discuss architecture-related topics, make decisions, and design architecture guidelines which also affect FTs:

"*There is an architecture community that not only discusses, but can also make decisions. . . . You have to be able to provide input to the teams, but ideally through a community approach and not through strong external roles.*"

— AC, MPN

Additional CoPs were organized for POs and SMs as well as for the testing domain. The former enables DPOs and POs to coordinate and to find a common direction on enterprise level. At regular intervals, MPN and OTD also have organized inspect and adapt events to check where they are in the adoption process, what to improve, and how to adjust their behavior respectively. Last but not least, PPM and PPM organized big events that took place once or twice a year in one of the sites to facilitate team coherence and trust.

V. Discussion

A. Key findings

We now discuss the main outcomes of our findings.

Key findings RQ1: After analyzing different adoptions of LeSS in four products, we identified following five success factors. First, the adoption must be 100% transparent as higher transparency incentivizes FTs to deliver software in higher quality.

Second, comprehensive training courses and workshops ease the adoption since they provide a shared understanding of new practices, roles, and responsibilities. Third, employees should be involved as early as possible in order to minimize change resistance. Fourth, managers should be aware of dissatisfied employees to retain their status and power. Accordingly, managers should raise awareness regarding the value of change to the organization. Fifth, the motivation behind LeSS should be properly promoted, advertised, and communicated.

Key findings RQ2: Based on the four LeSS applications, the following six key findings emerge. First, although the self-organization of FTs were acknowledged within the organization, sprint cadences were centrally organized using a so-called "one-calendar" approach. Second, all four products were extended by a domain level and supported by a DPO as the products were not necessarily independent from each other. Third, due to the present organization structure, the role of the PO was shared by two or three people resulting in the so-called "dual or even triple leadership". Many interviewees complained about this situation since it slowed down processes and hampered what LeSS wants to achieve: fast and agile decisions. Fourth, all products extended LeSS by involving additional roles such as BAs, PO support, shared services, and solution and enterprise architects. Fifth, all four products extended LeSS with additional processes to facilitate the exchange of shared knowledge between the products. These events included various types of communities of practices such as architecture, SM & PO or testing communities. Sixth, the interviewed products organized large team-building events that took place once or twice a year in one of the sites. With those, they aimed at building trust between FTs and overcoming cultural barriers.

B. Threats to validity

We discuss potential threats to validity along with an assessment scheme as suggested by Runeson and Höst [18]. First, we address **construct validity** by interviewing multiple roles across four different products. We also transcribed, coded, and analyzed the interviews. Second, **internal validity** is not relevant, as this research was neither explanatory nor causal

[18]. Third, we address **external validity** by providing a detailed description of the case and focusing on analytical generalization [18]. This paper provides empirical insights that allow for a profound understanding of the adoption and application of LeSS. The presented findings should be viewed as valuable insights for other organizations that aim to adopt and apply LeSS. Last, we ensure the **reliability** of our results by using a case study protocol with detailed procedures for data collection and analysis. We also collected data from different sources by multiple interviewers to allow data and observer triangulation [18].

VI. Conclusion and Future Work

The success of agile methods for small teams inspired large organization to apply them at large-scale by using scaling agile frameworks [4], [5]. Although the number of organizations using these frameworks is increasing, scientific literature providing in-depth analysis is still scarce [6], [7]. We aimed to fill this gap by providing a case study regarding the adoption and application of LeSS in four different products of a German automobile manufacturer. Our findings indicate that a transparent adoption incentivizes teams to deliver high-quality software and that comprehensive training courses and workshops ease the adoption. We also found out that the case organization extended LeSS with additional roles and processes to facilitate the exchange of shared knowledge between products, to build trust between teams, and to adapt it to the current organizational structure.

While we are confident that our findings will contribute to the existing body of knowledge on actual experiences related to LeSS, we encourage other researchers to conduct further in-depth case studies on LeSS and other scaling agile frameworks. For instance, it would be interesting to study to what extent companies have to adapt their organizational structures and processes in order to use LeSS. In future studies, researchers should also conduct cross-case analyses with the goal to compare the adoption and application of LeSS in different types of organizations, e.g., digital natives vs. traditional companies.

References

[1] K. Beck, *Extreme programming explained: embrace change*. Addison-Wesley, 2000.

[2] K. Schwaber and J. Sutherland, "The scrum guide," Scrum.org, Tech. Rep., 2017.

[3] P. Kettunen, "Extending software project agility with new product development enterprise agility," *Software Process: Improvement and Practice*, vol. 12, no. 6, pp. 541–548, 2007.

[4] T. Dingsøyr and N. B. Moe, "Towards principles of large-scale agile development," in *Agile Methods. Large-Scale Development, Refactoring, Testing, and Estimation*, T. Dingsøyr, N. B. Moe, R. Tonelli, S. Counsell, C. Gencel, and K. Petersen, Eds. Springer, 2014, pp. 1–8.

[5] M. Alqudah and R. Razali, "A review of scaling agile methods in large software development," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 6, no. 6, pp. 828–837, 2016.

[6] VersionOne, "12th annual state of agile report," VersionOne, Tech. Rep., 2018.

[7] K. Dikert, M. Paasivaara, and C. Lassenius, "Challenges and success factors for large-scale agile transformations: A systematic literature review," *Journal of Systems and Software*, vol. 119, pp. 87–108, 2016.

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

[9] C. Larman and B. Vodde, *Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum*. Addison-Wesley, 2008.

[10] ——, *Large-Scale Scrum: More with LeSS*. Addison-Wesley, 2016.

[11] Ö. Uludağ, M. Kleehaus, X. Xu, and F. Matthes, "Investigating the role of architects in scaling agile frameworks," in *21st International Conference on Enterprise Distributed Object Computing Conference (EDOC)*. IEEE, 2017, pp. 123–132.

[12] A. Putta, M. Paasivaara, and C. Lassenius, "Benefits and challenges of adopting the scaled agile framework (safe): Preliminary results from a multivocal literature review," in *International Conference on Product-Focused Software Process Improvement*. Springer, 2018, pp. 334–351.

[13] J. Pries-Heje and M. M. Krohn, "The safe way to the agile organization," in *Proceedings of the XP2017 Scientific Workshops*. ACM, 2017, p. 18.

[14] M. Paasivaara, "Adopting safe to scale agile in a globally distributed organization," in *IEEE 12th International Conference on Global Software Engineering*. IEEE, 2017, pp. 36–40.

[15] M. Paasivaara and C. Lassenius, "Scaling scrum in a large distributed project," in *International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 2011, pp. 363–367.

[16] ——, "Scaling scrum in a large globally distributed organization: A case study," in *11th International Conference on Global Software Engineering (ICGSE)*. IEEE, 2016, pp. 74–83.

[17] Ö. Uludağ, M. Hauder, M. Kleehaus, C. Schimpfle, and F. Matthes, "Supporting large-scale agile development with domain-driven design," in *International Conference on Agile Software Development*. Springer, 2018, pp. 232–247.

[18] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, p. 131, 2009.

[19] M. Q. Patton, *Qualitative evaluation and research methods*. SAGE Publications, 1990.

[20] T. C. Lethbridge, S. E. Sim, and J. Singer, "Studying software engineers: Data collection techniques for software field studies," *Empirical Software Engineering*, vol. 10, no. 3, pp. 311–341, 2005.

[21] M. B. Miles, A. M. Huberman, and J. Saldana, *Qualitative Data Analysis: A Methods Sourcebook*. SAGE Publications, 2014.