Requirements Engineering Challenges in Large-Scale Agile System Development

Rashidah Kasauli, Grischa Liebel, Eric Knauss, Swathi Gopakumar, Benjamin Kanagwa
Chalmers | University of Gothenburg
Gothenburg, Sweden
{rashida,grischa}@chalmers.se, {eric.knauss}@cse.gu.se, bkanagwa@gmail.com

***Abstract*—Motivated by their success in software development, companies implement agile methods and their practices increasingly for software-intense, large products, such as cars, telecommunication infrastructure, and embedded systems. Such systems are usually subject to safety and regulative concerns as well as different development cycles of hardware and software. Consequently, requirements engineering involves upfront and detailed analysis, which can be at odds with agile (software) development. In this paper, we present results from a multiple case study with two car manufacturers, a telecommunications company, and a technology company that are on the journey to introduce organization wide continuous integration and continuous delivery to customers. Based on 20 qualitative interviews, 5 focus groups, and 2 cross-company workshops, we discuss possible scopes of agile methods within system development, the consequences this has on the role of requirements, and the challenges that arise from the interplay of requirements engineering and agile methods in large-scale system development. These relate in particular to communicating and managing knowledge about a) customer value and b) the system under development. We conclude that better alignment of a holistic requirements model with agile development practices promises rich gains in development speed, flexibility, and overall quality of software and systems.**

***Index Terms*—requirements engineering, large-scale agile, system engineering**

I. Introduction

Despite wide critic, agile approaches have significantly contributed to the way software is developed [23]. While initially focused on small teams [23], [17], [1], [24], success stories have led to their application at large scale [6], [20], [29] and in system development [7], [2], [20], an environment that is characterized by long lead times [2] and stable, sequential engineering practices [26]. In these environments, new challenges arise, especially with respect to managing requirements [30] and companies struggle to implement efficient requirements engineering (RE) [19], [32], [5]. Existing works on agile RE (e.g., [27], [13], [4]), mostly focus on proposing new approaches, practices, and artifacts [12]. There is however a lack of empirical studies that investigate the phenomenon of RE in relation to agile methods in the domain of large-scale system development [13], [12], [15].

In this paper we report *RE related challenges of large-scale agile system development* i.e. the development of a product consisting of software, hardware and potentially mechatronic components that includes more than 6 development teams [6] and is aligned with agile principles [23]. Through a multiple case study of four large-scale system development cases, based on 5 focus groups, 2 cross-company workshops and 20 semi-structured interviews, we answer the following three research questions from an RE perspective:

RQ1: *What are possible scopes of applying agile methods in large-scale system development?* We aim to better understand the general context of potential challenges in industry.

RQ2: *How is the role of requirements characterized in large-scale agile system development?* With companies turning towards agile software development methods or even scaling up agile practices such as continuous integration to system level, the role of requirements is far from clear. Thus, we examine different situations in which requirements are used in agile software development.

RQ3: *Which requirements related challenges exist in large-scale agile system development?* Building on RQ1 and RQ2 results, we then investigate challenges and implications to RE.

The contribution of this paper is a report of real-world RE challenges related to applying agile development in large-scale systems. These challenges are effectively hindering a faster and more sustainable development of software. We find challenges with respect to the setup of agility in large-scale system development and with communicating knowledge between different parts of organizations. In particular, there is a challenge of distributing and breaking down knowledge about customer value and about building and maintaining system understanding. We are also highlighting the need for systematic approaches to engineering requirements. Thus, we hope that our work helps to establish RE practices that better support agility within large-scale system development.

II. Background and Related Work

Agile methods like Scrum and XP are being adopted in large-scale system development companies [29], even though they were originally intended for use on a small scale [17], [1], [24]. Existing work on this topic shows that companies successfully adopt agile methods, but that several challenges remain. In a survey with 13 organizations in 8 European countries and 35 individual projects on the adoption of XP and Scrum, Salo and Abrahamsson [29] report successful adoption of these methods and appreciation among practitioners. Lindvall et al. [22] study the potential of adopting agile methods with ABB, DaimlerChrysler, Motorola, and Nokia. The authors' conclusion is that, overall, agile methods could

2332-6441/17 $31.00 © 2017 IEEE
DOI 10.1109/RE.2017.60

suit the needs of large organizations, in particular for small and collocated teams. However, integrating agile into the company environment could be challenging. Lagerberg et al. [20] report based on a survey at Ericsson that applying agile on a large scale facilitated knowledge sharing and effective coordination. In a systematic literature review on the adoption of agile methods at scale, Dikert et al. [6] identify 35 challenges, e.g., coordination in a multi-team environment with hierarchical management and organizational boundaries. In a position paper by Eklund et al. [7], research challenges of scaling agile in embedded organizations are presented. These challenges include, e.g., coordination of work between agile teams or taking into account existing ways of working for systems engineering. Similarly, Berger and Eklund [2] present, based on a survey with 46 participants, expected benefits and challenges of scaling agile in mechatronic organizations, including efficiently structuring the organization, understanding of agile along the value chain, and adaptation to frequent releases.

With respect to agile RE or RE in combination with the use of agile methods, there is less existing work. Based on a mapping study with 28 analyzed articles, Heikkilä et al. [12] conclude that the definition of agile RE is weak. Furthermore, they report several problematic areas such as the use of customer representatives, prioritization of requirements or growing technical debt. In a case study by the same authors at Ericsson, the flow of requirements in large-scale agile is studied [13]. Perceived benefits include increased flexibility, increased planning efficiency, and improved communication effectiveness. However, the authors also report problems such as overcommitment, organizing system-level work, and growing technical debt. Similarly, Bjarnason et al. [4] investigate the use of agile RE in a case study with nine practitioners at one large-scale company transitioning to agile. The authors report that agile methods can address some classical RE challenges, such as communication gaps, but cause new challenges, such as ensuring sufficient competence in cross-functional teams. In a case study with 16 US-based companies, Ramesh et al. [27] identify risks with the use of agile RE. These are, e.g., the neglection of non-functional requirements or customer inability. A systematic literature review on agile RE practices and challenges reports eight challenges posed by the use of agile RE [15], such as customer availability or minimal documentation. However, the authors also report 17 challenges from traditional RE that are overcome by the use of agile RE. The authors conclude that there is more empirical research needed on the topic of agile RE. Other studies have addressed the use of traditional RE practices and agile RE. Paetsch [25] provide a comparison between traditional RE approaches and agile software development while identifying possible ways in which agile software development can benefit from RE methods. The authors conclude that agile methods and RE are pursuing similar goals in key areas like stakeholder involvement. The major difference is the emphasis on the amount of documentation needed in an effective project. Meyer, in contrast, regards the relationship between RE and agile more critical, describing the discouragement of upfront

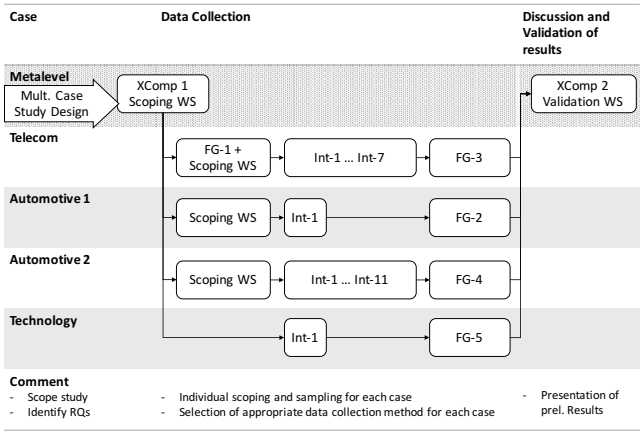

Fig. 1. Overview of multiple case study research design

analysis and the focus on scenario based artifacts (i.e. user stories) as harmful [23], however not based on empirical data.

In summary, there is substantial existing work on the adoption of large-scale agile in system development, including empirical studies. However, existing work either focuses on identifying and evaluating agile RE practices [12], [15], or at presenting the current state of practice at single companies [13] and without explicitly targeting system development [4]. Hence, additional empirical work is needed to understand the complex phenomenon of agile methods and RE in the domain of large-scale system development. Our study contributes with a cross-case analysis of large-scale agile development and the role and scope of RE in this context. Specifically, we aim to compare how agile methods are adopted with different scopes in multiple companies (RQ1). Additionally, we aim to characterize the role of requirements (RQ2) and requirements-related challenges (RQ3) within this scope.

III. Research Methodology

In our multiple case study [28], we investigate one telecommunications company, two automotive companies, and one company developing software-intensive embedded systems (referred to as the Technology Company in this paper). All four cases represent large companies developing products and systems that include a significant amount of software, hardware, and (with exception of the Telecom Company) also mechanical components. All case companies have experience with agile software teams and have the goal to further speed up the development of their software intense systems.

Sampling and Data Collection: Figure 1 gives an overview of our research design. Starting from a common case study design and common research questions, we conducted a cross-company scoping workshop (XComp 1 Scoping WS) to secure commitment from participating companies, align the goals of the study and finalize the research design. We then scheduled individual scoping workshops (Scoping WS) with each company, except for Technology Company, which, despite genuine interest in the study, could not free

TABLE I
DATA SOURCES

Type	Company	Role(s)	Label
Focus Group	Telecom	2xTest Architect, System Manager	FG-1
Focus Group	Automotive 1	Process Manager, Specialist Platform Software	FG-2
Focus Group	Telecom	2xTest Architect, System Manager	FG-3
Focus Group	Automotive 2	System Responsible, 2x Function Owner, System Quality Engineer	FG-4
Focus Group	Technology	RE Change Agent, Chief Engineer	FG-5
WS	Automotive 1	Verification Manager, Specialist Platform Software	XComp 1
	Telecom	Test Architect, System Manager	
WS	Automotive 1	Verification Manager, Specialist Platform Software	XComp 2
	Telecom	Test Architect, System Manager	
	Automotive 2	Test Architect, System Manager	
	Technology	Chief Engineer Software	
Int	Telecom	Test Architect	T-TA-1
Int	Telecom	System Manager	T-SysM-1
Int	Telecom	System Manager	T-SysM-2
Int	Telecom	Developer	T-Dev-1
Int	Telecom	Scrum Master	T-ScrM-1
Int	Telecom	Area Product Owner	T-APO-1
Int	Telecom	Operational Product Owner	T-OPO-1
Int	Automotive 1	Technology Specialist (Safety)	A1-TS-1
Int	Automotive 2	Component Design Engineer	A2-CDE-1
Int	Automotive 2	System Design Engineer	A2-SDE-1
Int	Automotive 2	Function Owner	A2-FO-1
Int	Automotive 2	Function Owner	A2-FO-2
Int	Automotive 2	Software Developer	A2-SD-1
Int	Automotive 2	Software Developer	A2-SD-2
Int	Automotive 2	Product Owner	A2-PO-1
Int	Automotive 2	Scrum Master	A2-SM-1
Int	Automotive 2	System Tester	A2-ST-1
Int	Automotive 2	Functional Tester	A2-FT-1
Int	Automotive 2	Software Quality Expert	A2-SQE-1
Int	Technology	Requirements responsible	Tec-SRR-1

up resources for this study at that time. During these scoping workshops, we selected with help of our company contacts the most appropriate case in terms of availability and available experience on the topic, e.g., a specific product or component (partially) developed with the use of agile methods. These cases were selected to accommodate two aspects: *variation* to allow better generalization of results and *convenience*, since there was an interest to investigate the research questions in each particular case. This allowed us to cover a variety of perspectives during data collection, i.e., system overview, customer experience, development, integration, and testing. Our generic data collection instrument can be found online[1]. Data collection was adjusted according to each individual case based on resource availability and commitment. For instance, the Telecom case relates to a large product development by many Scrum teams and we relied on a focus group followed by interviews[2] (denoted *Int* in the figure) with a variety of roles (see Table I). In contrast, the Automotive 1 Case relates to one Scrum team and we choose a focus group with the entire team, complemented with an interview of a safety expert. Interviews lasted approximately one hour and followed a similar interview instrument for all companies with domain specific adjustments for each company. For focus groups and cross-company workshops we scheduled three hours.

Not all researchers participated in all interviews and focus groups, but for each case we had one dedicated researcher who was present in all data collection events of that particular case. We recorded interviews and focus groups where possible and had at least two researchers take notes otherwise.

Data Analysis: For data analysis, we relied on a thematic coding approach [10]. For each case, at least two researchers familiarized themselves with the data and highlighted noteworthy statements and assigned a label or code to each. Based on a card sorting approach, we then in the entire group of researchers discussed and iteratively combined codes into 30 candidate themes, from which we derived four high-level clusters containing 3-5 themes each. To validate the clusters, we discussed the outcome of our analysis in a reporting workshop with all participating companies.

Threats to Validity: By design, the external validity of case studies is low. Hence, generalization of our findings might not be possible to different companies or domains. In particular, we cannot reason about challenges for small-scale or pure software development. We believe that while some challenges might be visible there as well, they can likely be managed ad hoc or within the scope of agile practices. We designed our study to identify common challenges across participating companies. Thus, our research method does not support any deep argument about differences between companies, domains, and market positions. However, given that we found similar themes in all four cases, we expect that these apply similarly to other companies or projects in large-scale systems engineering.

To increase internal validity, we used data triangulation between interviews and between case companies. Further, the results of our analysis were discussed in a cross-company workshop (XComp Validation WS) with the companies. The workshop included key roles from each company that were already involved in the study. We also used the workshop to discuss underlying root causes and challenges that are shared by all companies (see Section V). To avoid a too restricted view on smaller parts of a project or a product, we selected interviewees from different parts of the development, including at least one team and several system level roles in each case. We relied on a convenience sample and companies provided us with access to dedicated experts in the research field (agile transformation, RE) with a genuine but diverse interest in the field. While we hope that this improved internal validity, it might have introduced a selection bias, which we tried to

[1] http://grischaliebel.de/data/research/KGLKK_re_agile.pdf

[2] We refer to participants of interviews and focus groups as interviewees

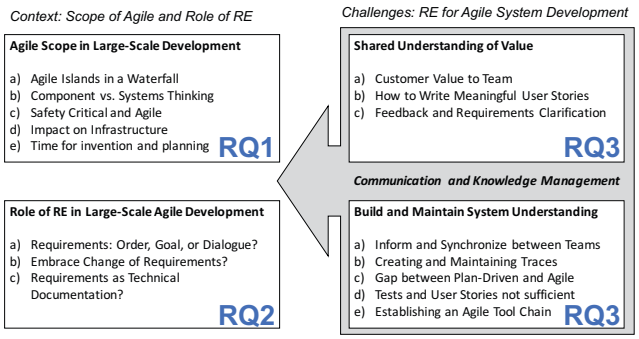

Fig. 2. Themes with respect to research questions.

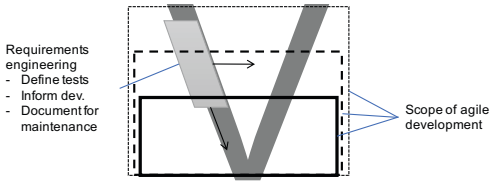

Fig. 3. Different scopes of agile development within system development. (Light grey box denotes RE, arrows indicate how requirements are used for informing developers about what to implement, testers about what to test, and for documenting the system for maintenance. Three black boxes show the different agile scopes discussed in this case study.)

mitigate by encouraging participation of both proponents and opponents of agile/RE.

We limited threats to conclusion validity by improving the interview instruments in multiple iterations and by conducting interviews in pairs of two researchers. With all four case companies, we have a prolonged involvement and therefore a mutual trust among the parties exists. The data analysis was discussed and refined among the authors in several iterations.

IV. Findings

This section presents our findings in relation to our research questions. Fig. 2 presents an overview of our themes.

A. What are possible scopes of applying agile methods in large-scale system development? (RQ 1)

1) Context of Case Companies: Figure 3 gives an overview of different scopes we identified in our cases. The next Section IV-A2 elaborates on common themes across these scopes.

a) Telecom Company: The Telecom Case relates to the development of one major product. More than 30 Scrum teams develop in parallel based on a scaled agile approach (adopted from SAFe [21]). Scrum sprints are based on a backlog and a hierarchy of product owners breaks down product requirements and customer visible features to backlog items. While these product owners represent the customer requirements towards the product development, system managers represent a system requirements perspective. The overall effect is a continuous development stream and feature flow, which is supported by a powerful infrastructure that enables continuous integration and testing. Pre-development generates knowledge about new features, which enables effective planning for continuous delivery.

Particular to the Telecom case, hardware development is largely decoupled from the software development. New hardware becomes available with a regular, but low frequency. Thus, the software development sets the pace of system development, which can be seen as continuous and agile, in that it embraces agile values as much as possible. In Figure 3 this is shown by the largest, dotted box, which implies that the whole scope of a traditional V model is covered.

b) Automotive Company 1: In Automotive 1, agile methods have been successfully applied to in-house development of software components. In the light of growing competition from software-centric companies, e.g., on autonomous driving, there is a desire to scale up these fast-paced approaches from developing software components to developing complete functions, thus including agile development of hardware and mechatronic. The selected case is a pilot project that re-implements a whole customer function in an agile way. Integration of this function into a real vehicle requires additional verification with respect to safety and overall system behavior. Thus, we would characterize this situation with the second largest box in Figure 3, where a function owner takes responsibility for one particular function and implements it with an agile team.

c) Automotive Company 2: With Automotive 2, we selected a case responsible for safety critical functionality developed in house. As with Automotive 1, agile teams develop software. For this, software requirements are transformed into backlog items. In order to speed up development of this differentiating functionality, different measures have been taken to speed up the overall system development. While this development still corresponds to the V model, the introduction of a shared information model that supports storing requirements, design elements, tests, and implementation models helps to shorten the development time significantly. In FG-4, participants referred to this approach as *narrow V model* (comparable to *agile loop* in [7]). In Figure 3, we describe this as the smallest box, not to refer to overall development speed, but to the fact that hand-over between plan-driven and agile development happens on a low level of abstraction.

d) Technology Company: The Technology Company develops mechanical products, both for consumer markets and for industrial development and manufacturing. Their system development includes several system elements. Software development is mostly confined to two of these elements, both of which are characterized by agile methods and practices such as Scrum and Continuous Integration. As with Automotive 2, we refer to this situation with the smallest box in Figure 3.

2) Agile Scope in Large-Scale System Development: Summarizing the four cases, we recognize that some case companies have come a long way towards continuous software engineering and enterprise-wide adoption of agile [31]. Others are currently moving in that direction. Our research aims for common themes, regardless of the scope of agile adoption (for which we control with RQ1) or agile maturity (which we

did not explicitly investigate in this study). In the analysis of interview data, we uncovered the following themes that relate to the scope of agile methods.

a) Agile Islands in a Waterfall: From a product perspective, a plan-driven or stage-gate approach is important. Release of a new product needs to be planned and longer development cycles for hardware and mechanical components need to be scheduled. All of our case companies have agile software development teams that operate within the context of a larger system engineering process, which one interviewee described as agile islands:

> *"It feels like agile islands in a waterfall."* — FG 2

The challenge we found here regardless of agile scope in the specific case is continuous information exchange between plan-driven and agile parts of an organization. Incubation of new innovative ideas, facilitating quick feedback loops, and quick learning on potential business value are important assets to remain competitive, yet they are hard to integrate into the overall system development approach in all our cases.

b) Component vs. Systems Thinking: The scope of agile development also relates to the relationship between components and the system that is build based on it.

> *"We need to balance project autonomy, freedom, empowerment with platform reuse, consolidation of data, and unambiguous requirements. The teams have a lot of tacit knowledge, which is not available beyond their scope. But how much ceremony should we force on teams?"* — FG 3

Even beyond designing a single system, knowledge about customers and their needs should be maintained for future projects. Without a good knowledge management approach, this can collide with the desire to allow empowered component teams to make fast, local decisions.

c) Safety Critical and Agile: Most of our case companies are subject to safety standards and regulations. Several participants in FG2 and FG4 expressed concerns that the development of safety critical software together with corresponding standards could impede agile development. As examples, the participants expressed the need for documentation and tracing that is required by several standards, such as ISO26262 [16]. However, an expert for functional safety in Automotive 1 stated that the need for documentation and tracing is related more to the size of the company and the system rather than regulations.

> *"Many see that as a problem. Many say that it's safety problem, it is a 26262 problem. But we say* [..] *we need to document anyway since then half a year later it is a different team* [*working on the same software*]*"* — A1-TS-1

According to our interviewees, standard conformance could be combined with agile development if only this was planned in a systematic fashion, e.g., by sandboxing safety critical parts.

Still, the need to guarantee safety relates to both component and system level. Depending on the scope of agile methods, this introduces interfaces between plan-driven parts of the organization and agile teams.

d) Impact of new features on infrastructure: In system development, integration testing often depends on a strong laboratory setup that allows testing hardware, software, and potentially mechanics together. While a new feature might mainly depend on changes of software and can be provided in an incremental, fast-paced way, it could require an update of the test environment, which may include sophisticated hardware and environment models. However, changing the test environment might take as long as finishing the software components, thus introducing delays, if not started in due time. Similar concerns relate to other infrastructure for continuous integration, delivery, and deployment.

This theme shows that independent of the scope in Figure 3, there is a need to maintain a system-level perspective beyond self-organized teams and to allow requirements related information to escalate to this level as early as possible.

e) Time for invention and planning: Study participants in Automotive 1 reported that an exploration of solution space is difficult within agile sprints, as it would be impossible to commit to a fixed schedule without deep knowledge about new features. Pre-development is required to better understand the impact of new features. If this is done by a dedicated group, this would imply documentation and hand-over of results and slow down the process. As a remedy, specific exploration sprints were brought up. Another solution could be to transfer engineers between pre-development and agile system development, so that they can also share their knowledge with team members. Either way, this highlights the need for good scoping of agility within system development.

To answer RQ1, the four cases show that agile development is indeed applied on very different organizational levels. Irrespective of this scope, common issues with RE occur. Handovers between the boundary of agile and plan-driven parts of the company even occur in the Telecom case.

B. How is the role of requirements characterized in large-scale agile system development? (RQ 2)

a) Requirements: Order, Goal, or Dialogue?: In all our cases, software development teams are enthusiastic about agile methods. As a consequence however, many see requirements as something that comes into the way of being agile. A requirement is seen as an order, interfering with the empowerment of teams. Instead, developers tend to prefer user stories, that set goals, but do not limit the agile team's autonomy in deciding how to reach them, i.e., to claim ownership.

According to our interviewees, agile methods imply a bidirectional flow of information, since they advocate to explore the best way of satisfying a customer need through incremental and iterative work. Agile teams are used to uncover new aspects, details, or even new requirements. With respect to RE, the main advantages brought up by interviewees on all levels are related to ownership and include consequent orientation towards user and customer value as well as the fact that developers have the mandate and competence to update user stories within the scope of agile development. Thus, user stories and implementation are usually extremely consistent. This fast paced agile learning however needs to be fed back to system-level requirements models, a task that is currently challenging in all our cases.

b) Embrace Change of Requirements?: Sufficient facilities for updating system requirements based on agile learning are currently missing. Thus, such updates are manual work, leading to inconsistencies, which are expensive to remove and can be considered waste in the overall development process. In addition, developers have little intrinsic motivation to update requirements models based on updates to user stories, as they are not part of their delivery (usually code and tests). If however requirements updates were not propagated, the system requirements view would become quickly obsolete and detached from the real system. Consequently, roles responsible for customer and high-level system requirements (product owners, function owners, system managers), fear a loss of important knowledge for later maintenance of the systems. A more systematic approach to manage requirements updates received from agile teams would make their jobs much easier.

Another key challenge highlighted by our interviewees relates to the timeliness of required information. Agile development is for example able to digest changes and information that only incrementally becomes available, but it struggles with information provided in bulk.

c) Requirements as Technical Documentation?: With respect to a wider adoption of agile methods and their rejection of requirements, we recognized across all companies a tendency to encounter more inconsistencies between requirements and implementation. This is partly due to the aforementioned lack of propagating changes of user stories to the system requirements model. While this relates to classic requirements challenges, the effect of introducing agile methods has shifted the problem somewhat: agility helps investigating assumptions as well as resolving ambiguities and inconsistencies early. Knowledge gained in this way has high value as a reference for maintaining the system, i.e., as technical documentation. But as long as this knowledge is not efficiently shared in the whole development organization, the challenge remains effective.

For technical documentation, some developers would prefer to use automated regression tests, since this would put less overhead on their work. Most of our interviewees however, especially on higher levels of abstraction in system development, indicated that higher-level requirements are of critical importance here, since reconstructing the intended behaviour of a component from a set of a dozen test cases and hundreds of user stories renders them useless for technical documentation. We refer to the challenge of tests and user stories being insufficient for specification in the next section.

C. Which requirements related challenges exist in large-scale agile system development? (RQ 3)

With respect to RQ3, we see two communication related areas of challenges: *Shared Understanding of Value* and *Build and Maintain System Understanding* (see Fig. 2).

1) Shared Understanding of Value:

a) Customer value to team: Agile development has been adopted for all our case companies as presented in Section IV-A1. Study participants indicate a large *distance between customers and developers*:

> *"We have so many different levels between the real customer, the customer units and [the teams], with each transaction you lose so much content"* — T-TA-1

In all our cases we found dedicated roles that channel information from multiple stakeholders down to the teams. It is not trivial to bridge that gap, direct interaction of teams and stakeholders can lead to chaos when established plans are circumvented, on-site customers are not an option, and product-owners hard to implement.

However, one participant exclaimed that the focus on agile practices occupied the teams so much that this caused a neglect of product value.

> *"The idea of producing value for the customer has been discarded. Teams just want to be agile (i.e., independent)"* — XComp 1

However, value creation is not solely the teams' responsibility as the requirements breakdown starts from the customer units, in the Telecom case, or from the function management units, in the Automotive cases.

To ensure frequent delivery, application of agile methods implies breaking down large features into smaller sub-features and tasks that can be finished during a typical Sprint. This is the practice in all our case companies even though the used methods differ.

> *"[..] usually, a feature is what you can sell to customers but teams talk of sub-features [..] "* — T-ScrM-1

One interviewee (T-APO-1) pointed out that it is hard to break down the requirements such that they carry user value, a challenge also recognized in other cases (FG-2 and FG-5).

b) How to write meaningful user stories: User stories provide a fast means to share knowledge both on a high and a low level in an agile system development. In the Telecom as well as in the two Automotive cases, user stories are used for two purposes as one interviewee says

> *"... so there are user stories that of course take the view from the end customer and describe what the end customer wants from our system and why. But then there are other user stories that are more like work descriptions of what the team should achieve and those could be like internal things that need to be developed in order to keep the architecture constrained."* — T-SysM-2

A Function Owner in FG-4 (Automotive 2) specifically expressed that high-level user stories could help to communicate value early. However, as discussed in Theme *a) customer value to team*, it is particularly difficult to write user stories that have direct value for the user. Such user stories would typically be too large to be completed and demonstrated in one sprint. Yet, breaking it into more user stories could deteriorate requirements quality since not enough effort goes into maintaining the requirements. It is therefore hard to understand which user story can be traced to requirements. In summary, user stories are hard to write at the scale and complexity of the cases in our study, yet they offer a unique opportunity to bridge distances between customer and developer.

c) Feedback and requirements clarification: In several companies, study participants raised the issue of long or complicated feedback cycles. At Automotive 1, one study participant named *slow mechanical or hardware development* as one of the main reasons for long feedback cycles. If software

has to be tested together with actual hardware, feedback on software functionality is postponed until the hardware is ready.

One study participant stated a second reason — often *customers are not agile* and take a long time to try out and approve new features. By the time feedback then reaches the agile teams, they are already working on another part of the product and do not remember exactly what the feedback is about. That is, for the teams the feedback comes too late, while customers do not see value in giving quick and frequent feedback, even on smaller increments. This challenge is especially encountered if the system under development is supposed to be integrated into a larger system at the customer site, as for example in the Telecom and Technology cases.

A third reason for complicated feedback cycles is that there is a *large number of stakeholders*, both external and internal. Due to the complex nature and the scale of the products developed by our case companies, there is rarely a single customer. Instead, requirements inflow occurs from many different sources, e.g., customers, authorities, managing subcontractors and sourcing, or standardization organizations. In many cases, requirements need to be discussed with and communicated to other stakeholders within or outside the organization, delaying feedback.

Even if feedback and requirements clarification can be facilitated despite the challenges raised here, gained knowledge must be effectively managed as participants in FG-5 (Technology Case) pointed out. Two aspects of this challenge were raised: First, it is unclear where knowledge about a specific customer can be managed beyond the team and current project. Secondly, in continuous product development, teams might not realize that they have valuable knowledge for other parts of the system development, while those other parts do not know that valuable knowledge is available.

2) Build and Maintain System Understanding:

a) Inform and Synchronize Between Teams: Teams receive requirements from the product managers through several organizational levels in the Telecom case. Furthermore, they often need to exchange information with other teams to synchronize the development. This process of channeling the 'right' information towards and between teams is difficult and time-consuming. Hence, it limits agility and speed of teams.

A similar issue was raised at Automotive 1, where participants discussed which organizational scope agile methods should have with respect to *time for planning and invention* (Section IV-A2e). FG-2 participants wondered whether agile should be limited to the development only, or should start from a feature request. In the former case, developers would receive feature requests in the form of already broken down requirements for implementation. In the latter case, developers would have to do the breakdown of a feature request into smaller units themselves. While both cases seem to be feasible, the question is how teams can be synchronized in any of these cases. If requirements are broken down by an external role or team, possibly in a plan-driven way, they need to be handed to different agile teams and their work needs to be synchronized. If they are broken down and implemented within one team, multiple agile teams only need to synchronize when there is interaction with or dependencies to features developed by other teams. Awareness about such dependencies is a pre-requisite.

b) Creating and Maintaining Traces: Several interviewees in the Telecom case stated that their system requirements work as a documentation of what the system is doing, rather than a plan of what shall be implemented. In their view, system requirements are used as an input for specifying test cases and to allow analysis for new features, not as a plan for implementation.

> *"You can't really afford to have this kind of static requirements work upfront which will be a waste anyway when you implement stuff. The way we handle requirements now is more like a system description."* — T-TA-1

Since user stories relate directly to feature implementation and are not always systematically derived from existing requirements, direct tracing is not always possible.

A similar situation occurs in Automotive 2, where product owners write user stories based on plan-driven requirements they receive as an input. These user stories can in fact be rather local development tasks and backlog items that do not require tracing to system requirements. Thus, traces are not systematically managed, which can lead to additional work in cases where such backlog items become relevant for tracing to system requirements. The fact that often only the product owner is aware of which user stories originate from which requirements can slow down collaboration between agile teams and plan-driven RE teams. Interviewees in the agile teams considered tracing user stories to requirements to be documentation, which should not be part of the agile process. Instead they preferred to spend their time on implementation:

> *"I don't think traceability is not required or something like that. It's just that my focus hasn't been on documenting the function. I just focus on doing implementation and developing the function."* — A2-PO-1

This view was also shared in Automotive 1: While participants stated that tracing is valuable, or even required by standards, they felt that right now there is not enough incentive for agile developers to create traces. They wished for an incentive or directly visible benefit for the developers as well as for simplifying trace creation.

c) Bridging the Gap Between Plan-Driven and Agile Development: In the Telecom case, we found that system managers feel disconnected from the agile teams. Their role is to be experts on a certain part of the system and support teams with their knowledge of the system requirements. However, as one interviewee stated they currently cannot be in contact with all teams and might therefore not get a notification if something has been changed with respect to existing requirements.

> *"If [..] a team updates a past requirement, perhaps I should get like a notification on that so I can ask them 'Have you forgotten X?'."* — T-SysM-1

Similar challenges exist with the other companies, e.g. in the Automotive 2 case where agile teams can add new backlog items or change existing ones in collaboration with the product owner. However, since agile teams do not interact directly with system requirements (see *b) creating and maintaining traces*), they do not consider knowledge about them to be

of importance. Further, backlog items are easy to understand, even for stakeholders not directly involved, and allow them to share their opinion. While this is generally perceived positively by the interviewees, it was also brought up that this can cause the function owner to be overexposed to change requests. One function owner expressed this as follows.

> *"The more people look into requirements, the more they read them, the more iterations it will become.* [..] *there is going to be more opinions, comments and also more work."* — A2-FO-1

As this can lead to inconsistencies between changed and new backlog items and the system requirements, e.g. in the case where a system requirement related to a new user story already existed, increased gatekeeping becomes necessary. This generates effort for backlog grooming by the (agile) product owner, and managing of system requirements by the (plan-driven) function owner. The current separation between both worlds does not seem to be ideal, since product and function owner can easily become bottlenecks, and late resolution of inconsistencies can create additional effort. If the actual implementation deviates from the original requirement or when some requirements are not implemented, this will surface as problems during system integration and testing.Tests are developed against the plan-driven requirements and are therefore in need of an up-to-date version.

> *"If I have a requirement saying this thing should happen, when I test it, I find out that what is supposed to happen doesn't happen.* [..] *And then I find out the requirement wasn't updated. So actually the implementation was correct but the requirement isn't matching the implementation."* — A2-ST-1

Further, if the system has to be evolved or maintained in the future, outdated requirements can cause misunderstandings.

d) Tests and User Stories not sufficient: The idea of using test cases both as actual test artifacts and as requirements documentation is wide-spread in the agile community [23] and was also discussed by several participants. While this was seen as a potential way to reduce documentation effort, several issues with this approach were brought up. According to several study participants, user stories and test cases do not carry enough information to serve as a means of documentation:

> *"Tests are written in a pragmatic way. They do not capture the 'why'."* — Tec-SRR-1

Other interviewees throughout the companies added that one would need a number of tests to document any significant requirement, which will then be hard to reconstruct from just reading the tests during maintenance.

Several interviewees saw similar problems with user stories, as they would only reflect single scenarios. The overall system behavior would then emerge from the syntheses of all these single scenarios. To derive this full picture from tests or user stories only would, however, be too difficult:

> *"If we don't specify this kind of complete* [*requirements*] *specification, we could try to use all* [..] *user stories* [..]. *But then we must base the understanding on* [..] *lets say* [..] *2000 user stories* [..] *and try to find a good way of describing the complete system."* — T-SysM-2

It is interesting to note that this challenge surfaces early on, i.e., when an incoming customer request is analyzed. Therefore, if agile teams only develop backlog items based on finished requirements that they receive from other parts of the organization, they might not be aware of this challenge and therefore consider test cases to be sufficient.

While in the Telecom case the issue of understanding system behavior from user stories or tests was mainly discussed with respect to new features, participants in Automotive 1 raised it especially for system maintenance. FG-2 participants agreed that user stories or test cases would not be appropriate to understand the behavior. They were unsure what form of documentation should be used instead, which level of detail the requirements should be on, and how they could be different from 'traditional' requirements.

e) Establishing an agile tool chain: Several participants in FG-2 (in Automotive 1) raised the topic of an adequate tool chain for agile RE. They reported that, at the moment, they use traditional tools for requirements management and tools that are aimed at agile development, such as JIRA, for the actual development. These tools are however largely separated and, thus, they felt that in order to be able to perform RE in a more agile way, they would need appropriate tools. This is an interesting issue, as a simple issue tracking tool is not likely to address their needs as they operate in a multi-disciplinary, regulated environment. In particular, there are formal requirements for traceability imposed by safety standards such as ISO26262.

Similarly, in the Telecom Case, current tooling was brought up as a hindrance for speed and agility. Interviewees described the current process of updating system requirements as too slow and cumbersome. They stated that by introducing a more efficient tool solution, engineers could potentially be more motivated to make changes to requirements and by this narrow the gap between agile user stories and requirements.

The need of a tool-chain that better supports agile information flows was confirmed by the other two cases as well.

V. Discussion and Implications

Even though the four cases differ in their context, i.e., domain and scope of agile methods within system development, we found common concerns and challenges with respect to RE. Based on our findings, we draw the following four conclusions:

Conclusion 1: *Challenges of RE for large-scale agile system development relate to communication and knowledge management.* While related work implies that communication challenges are mitigated by agile approaches and less prominent in agile RE [15], [4], all our challenges relate to communication and knowledge management. Both aspects are at the core of Agile and RE, indicating a need for fundamental research in these areas specifically for system development.

Conclusion 2: *Challenges relate to two areas of requirements knowledge: User Value and System Understanding.* We identified two categories of RE challenges in the large-scale agile system development domain; *shared understanding of value* and *building and maintaining system understanding*. While pre-agile RE approaches differ between user and system requirements specifications, we are not aware of related work

that makes this distinction for RE in the scope of agile development. Surprisingly, we found that companies were not very interested in agile RE practices themselves (which is the focus of the majority of related work, as [15] indicates). In contrast, they found it more important to understand how RE can support agile methods in large-scale system development and how agile development can be integrated with existing processes. Our findings suggest that such support cannot be offered sufficiently by traditional, upfront RE, as indicated [23], [13]. Similarly, we did not find any specific roles that emerge in the large-scale agile environment, comparable to the roles presented in [14]. Our results suggest that continuous and agile development methods on a large scale require new concepts.

Conclusion 3: *Challenges relate to the interplay of stakeholders from three domains: customer, development, and integration & testing.* The development domain is generally embracing agility and characterized by a dislike for traditional requirements and bulk updates. The require better *synchronization between teams* and wish for *establishing an agile tool-chain*. In contrast, the customer domain is concerned with breaking down customer-visible features in order to communicate *customer-value to team*. They require better support for *writing meaningful user stories* and for *bridging the gap between plan-driven and agile development*. The integration and testing domain is struggling to *create and maintain traces* and with the fact that *user stories and tests* are not sufficient to build and maintain sufficient system understanding.

Conclusion 4: *In order to yield their full benefits, agile practices and a holistic system requirements model must be better aligned.* Key challenges occur when there is an interaction, or a lack thereof, between the three domains and we believe that industry would benefit from new impulses from research in the following areas.

Bridge distance between customer and developer: We found in all four case companies that the distance between the customers and the development is perceived to be too large. In particular, it was described as difficult to break down a feature request into small packages that both have customer value and can be delivered in small iterations. However, agile values such as individuals and interactions [8] as well as agile practices such as continuous delivery [18] depend on a good notion of value. Yet, we found this particularly hard to establish in large-scale system development, because of unclear customer role and scale. In case of an external customer, any customer visible feature will imply more work than can be done within one sprint or by one team. This makes feature decomposition necessary and it is impossible for a single team to demonstrate customer value at the end of a typical sprint. Related work in this direction has in particular pointed out challenges with the practice of customer representatives [12], [27], [15], but it seems that the notion of value itself might be problematic.

Establish information flow and knowledge management: Similarly, the information flow between the domains currently does not well support agility, in the sense that information exchange and feedback cycles are often too slow. As our results suggest, it is crucial to establish suitable exchange and management of knowledge throughout large-scale agile system development. Agile development works best with a continuous inflow of new requirements and can in turn help to resolve ambiguities and refine requirements just in time, as new knowledge becomes available. However, it is important to support updating system requirements models and to coordinate the information flow between parallel teams. This finding suggests that communication issues continue to be relevant in large-scale agile RE, in contrast to what is suggested by related studies, e.g., [15], [4].

Support analysis of incoming requirements: In all case companies, the need for documented requirements was stated because of a need to understand the current system in order to analyse new feature requests or to maintain the system. Also, several interviewees stated explicitly that using test cases as requirements would not be enough to fulfill this need. Therefore, we see the need for more work investigating the use of different notations, techniques or methods to inform early analysis of incoming requirements. While the user story format has been described as insufficient [12], test cases are often named as an alternative [12], [3]. Our findings suggest that using test cases, even in combination with user stories might not be sufficient, in particular with respect to supporting the understanding of a system's current functionality.

Dealing with system qualities and regulations: As our results suggest, companies are facing challenges when trying to address quality requirements, such as safety and security. Such quality requirements often relate to regulations that can directly challenge agile practices, by requesting end-to-end documentation and tracing of safety-related requirements. While these regulations might be one argument to have requirements documented, interviewees in all companies felt a strong need for documentation even without regulations. Especially the need to understand the current system in order to react to new feature requests or system maintenance were raised in all companies. Traditionally, long upfront analysis and planning aimed to address these needs [23]. However, as companies try to speed up their development, research needs to investigate new ways of dealing with documentation of such cross-cutting issues. Ensuring qualities and addressing non-functional requirements has been brought forward as a challenge in agile RE [15], [12], and first works exist to address regulations in agile [9].

In summary, companies need to aim for an open dialogue to balance system development needs with allowing agility. To address requirements updates and regulative issues, one could include explicit updates to requirements as part of each sprint, e.g. to accommodate safety regulations [11], [9]. To address the lack of intrinsic motivation to update requirements or establish tracing, immediate feedback loops should be established that show value to developers [33]. In addition, gamification approaches resonated well in discussions with practitioners. To address the distance to customer and the challenge of none-agile information flow to development, one could investigate how to best use user stories on a high level

of abstraction and early on as well as to facilitate a dialogue between roles in different domains [8].

VI. Conclusion and Outlook

We presented our results from a multiple-case study with four systems engineering companies on the interaction of RE and agile methods in large-scale development. We studied the scope of agile methods, the role of requirements in this context, and requirements-related challenges of large-scale agile systems development. In all case companies, the way how plan-driven and agile development currently co-exist within the systems engineering environment limits the potential development speed. We found that in all companies, there is a need for some kind of traditional RE, especially with respect to documenting a system's behavior for future feature requests or maintenance. The case companies use different scopes for agile development, ranging from agile development on team-level embedded in an overall plan-driven process up to agile development for the entire product development. Despite the different scopes, we observed similar challenges in all companies. These relate to establishing a shared view of value from the customer and other stakeholders down to development, and to building up and maintaining a shared system understanding. In order to mitigate these challenges, we encourage future work to focus on aligning a holistic requirements model with agile practices. Ideally, this will allow large-scale system development efforts to fully benefit from agile methods, while still systematically managing knowledge about customer value and how the system under construction relates to this.

Acknowledgments

We thank all participants in this study for their great support, deep discussions, and clarifications as well as Jennifer Horkoff and Francisco Gomes for invaluable feedback. This work was supported by Software Center Project 27 on RE for Large-Scale Agile System Dev. and the SIDA BRIGHT project.

References

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

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

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

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

[5] T. Chow and D.-B. Cao. A survey study of critical success factors in agile software projects. *Journal of Systems and Software*, 81(6):961–971, 2008.

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

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

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

[9] B. Fitzgerald, K.-J. Stol, R. O'Sullivan, and D. O'Brien. Scaling agile methods to regulated environments: An industry case study. In *Proc. of 35th Int. Conf. on Software Eng. (ICSE)*, pages 863–872, 2013.

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

[11] G. K. Hanssen, B. Haugset, T. Stålhane, T. Myklebust, and I. Kulbrandstad. Quality assurance in scrum applied to safety critical software. In *Int. Conf. on Agile Software Dev.*, pages 92–103. Springer, 2016.

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

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

[14] R. Hoda, J. Noble, and S. Marshall. Self-organizing roles on agile software development teams. *IEEE Transactions on Software Engineering*, 39(3):422–444, 2013.

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

[16] ISO. Road vehicles – Functional safety, 2011.

[17] T. Kahkonen. Agile methods for large organizations-building communities of practice. In *Agile Dev. Conf., 2004*, pages 2–10, 2004.

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

[19] M. Laanti, O. Salo, and P. Abrahamsson. Agile methods rapidly replacing traditional methods at nokia: A survey of opinions on agile transformation. *Information and Softw. Techn.*, 53(3):276–290, 2011.

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

[21] D. Leffingwell et al. Scaled agile framework 3.0, 2014.

[22] M. Lindvall, D. Muthig, A. Dagnino, C. Wallin, M. Stupperich, D. Kiefer, J. May, and T. Kahkonen. Agile software development in large organizations. *Computer*, 37(12):26–34, 2004.

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

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

[25] F. Paetsch, A. Eberlein, and F. Maurer. Requirements engineering and agile software development. In *WETICE*, volume 3, page 308, 2003.

[26] J. Pernstål, A. Magazinius, and T. Gorschek. A study investigating challenges in the interface between product development and manufacturing in the development of software-intensive automotive systems. *International Journal of Software Engineering and Knowledge Engineering*, 22(07):965–1004, 2012.

[27] B. Ramesh, L. Cao, and R. Baskerville. Agile requirements engineering practices and challenges: an empirical study. *Information Systems Journal*, 20(5):449–480, 2010.

[28] P. Runeson, M. Höst, A. Rainer, and B. Regnell. *Case Study Research in Software Engineering*. Wiley, 1 edition, 2012.

[29] O. Salo and P. Abrahamsson. Agile methods in european embedded software development organisations: a survey on the actual use and usefulness of extreme programming and scrum. *IET software*, 2(1):58–64, 2008.

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

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

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

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