Requirements Engineering (2019) 24:315–340
https://doi.org/10.1007/s00766-019-00319-8

RE 2018

The impact of requirements on systems development speed: a multiple-case study in automotive

S. Magnus Ågren[1] · Eric Knauss[1] · Rogardt Heldal[2] · Patrizio Pelliccione[1,3] · Gösta Malmqvist[4] · Jonas Bodén[4]

Received: 3 December 2018 / Accepted: 4 July 2019 / Published online: 16 July 2019
© The Author(s) 2019

Abstract

Automotive manufacturers have historically adopted rigid requirements engineering processes. This allowed them to meet safety-critical requirements when producing a highly complex and differentiated product out of the integration of thousands of physical and software components. Nowadays, few software-related domains are as rapidly changing as the automotive industry. In particular, the needs of improving development speed are increasingly pushing companies in this domain toward new ways of developing software. In this paper, we investigate how the goal to increase development speed impacts how requirements are managed in the automotive domain. We start from a manager perspective, which we then complement with a more general perspective. We used a qualitative multiple-case study, organized in two steps. In the first step, we had 20 semi-structured interviews, at two automotive manufacturers. Our sampling strategy focuses on manager roles, complemented with technical specialists. In the second step, we validated our results with 12 more interviews, covering nine additional respondents and three recurring from the first step. In addition to validating our qualitative model, the second step of interviews broadens our perspective with technical experts and change managers. Our respondents indicate and rank six aspects of the current requirements engineering approach that impact development speed. These aspects include the negative impact of a requirements style dominated by safety concerns as well as decomposition of requirements over many levels of abstraction. Furthermore, the use of requirements as part of legal contracts with suppliers is seen as hindering fast collaboration. Six additional suggestions for potential improvements include domain-specific tooling, model-based requirements, test automation, and a combination of lightweight upfront requirements engineering preceding development with precise specifications post-development. Out of these 12 aspects, seven can likely be addressed as part of an ongoing agile transformation. We offer an empirical account of expectations and needs for new requirements engineering approaches in the automotive domain, necessary to coordinate hundreds of collaborating organizations developing software-intensive and potentially safety-critical systems.

Keywords Requirements engineering · Continuous software engineering · Automotive systems engineering

✉ Eric Knauss
eric.knauss@cse.gu.se

S. Magnus Ågren
magnus.agren@chalmers.se

Rogardt Heldal
Rogardt.Heldal@hvl.no

Patrizio Pelliccione
patrizio.pelliccione@gu.se

[1] Department of Computer Science and Engineering, Chalmers | University of Gothenburg, 412 96 Gothenburg, Sweden

[2] Western Norway University of Applied Sciences, Bergen, Norway

[3] University of L'Aquila, L'Aquila, Italy

[4] Knowit AB, Gothenburg, Sweden

1 Introduction

The automotive industry is currently going through rapid change, driven by new technology (electric/hybrid cars, autonomous driving, and connected cars) as well as new competitors (e.g., through companies such as Google,[1] Apple,[2] but also Tesla and Uber). This change manifests for example in Original Equipment Manufacturers (OEMs) increasingly turning into software companies. Where previously, electronics and software were introduced in cars to optimize the control of the engine, they now drive 80% to 90% of the innovation in the automotive industry[3]. In order to keep a competitive edge in this context, many OEMs have the need to increase development speed, (i.e., fast and early feedback on the product level), thereby improving time to market, flexibility (i.e., the ability to rapidly react to change), and the overall product quality.

We use the words of a software manager working in one of the OEMs we involved in this study to explain this need for speed and its relation to quality:

> "Perhaps we should start with why we should be faster. Autonomous drive, for example, is an area where there's incredible research and development activity going on simultaneously. So long lead times work poorly for us, that's one driving force. /.../ If you're gonna be fast, you need good quality. To get good quality, you need to ensure that with every change, everything that worked yesterday still works today. That's another driving force." – R11

There are various ways to improve (software) development speed, including agile methods, but also practices of continuous software engineering [30]. Continuous Integration (CI) is extensively used in the software industry to develop and release software more rapidly, as well as to reduce risk [32]. CI and its extension to continuous delivery and deployment have been reported to accelerate time to market as well as to improve product quality [45, 72]. It has been suggested that, indeed, software-intensive companies must move toward these practices to stay competitive in today's increasingly competitive markets [74].

Nowadays, OEMs are changing toward embracing CI [55], continuous experimentation [27], and large-scale agile methods [59, 60]. While initially focused on small teams [6, 49, 68, 75], success stories have led to the application of agile methods at large scale [23, 58, 82] and in system development [7, 24, 58], an environment that is characterized by long lead times [7] and stable, sequential engineering practices [77].

In these environments, new challenges arise, especially with respect to managing requirements [51, 84] and companies struggle to implement efficient requirements engineering (RE) [16, 57, 95]. Existing works on agile RE (e.g., [10, 39, 79]), mostly focus on proposing new approaches, practices, and artifacts [38]. While there is a trend toward analyzing the phenomenon of RE in relation to methods to improve (software) development speed, on the scope of whole systems [52] or organizations [39], these studies usually take the view of stakeholders involved in requirements engineering and development [38, 47].

In this paper, we investigate the management perspective, in order to complement previous research. Knowing that management in many automotive OEMs aims to increase development speed through agile methods or practices of continuous software engineering, we want to understand how managers envision automotive companies organizing themselves, with respect to managing and engineering requirements. We aim to provide a high-level visionary perspective, more concerned with creating long-term competitive advantage than getting short-term tasks done. In this context, we investigate the following research questions:

RQ1: Which aspects of the current way of working with requirements impact development speed?
RQ2: Which new aspects should be considered when defining a new way of working with requirements to increase development speed?
RQ3: To what extent will either aspects be addressed through the ongoing agile transformation?

This paper is an extension of a study published in the 26th IEEE International Requirements Engineering Conference (RE'18) [2]. In the initial study, we performed 20 qualitative interviews with managers and technical leaders of two automotive OEMs. From these interviews, we derived a first model that relates aspects of requirements engineering to development speed. In this paper, we extend the study by adding 12 additional interviews that provide validation of our qualitative model as well as add additional insights, e.g., with respect to the role of traceability, quality assurance of requirements and handling risk, and a new mindset when doing requirements that facilitates incremental work and focuses on interactions. Moreover, we added an evaluation of how the transition toward agility that is in place in the contacted companies is solving or expected to solve identified challenges and limitations.

The rest of this paper is structured as follows. Section 2 describes the context in which the research has been performed. Section 3 describes our research method. Section 4, gives an overview of the themes that emerged and their

[1] https://www.google.com/selfdrivingcar/.

[2] https://en.wikipedia.org/wiki/Apple_electric_car_project.

[3] According to industry experts: https://tinyurl.com/y9jnoupd.

effect on development speed. With respect to each research question, Sections 5, 6, and 7 then present the themes in detail and discusses them in relation to related work. Section 8 concludes the paper with a discussion of our contributions and their potential to guide future research.

2 Context of cases

Both case companies are automotive OEMs: one car manufacturer and one heavy vehicles manufacturer. Both companies are large, with many organizations in several countries, they produce several different models of vehicles, and they have had a long history of many different owners. Embedded systems play a key role, however, service-oriented systems have become increasingly important.

The goal of both case companies is to increase development speed as well as flexibility to react to changing market needs—a response to increasingly fast and disruptive changes in the automotive domain in recent years. As one mechanism to achieve this, both companies have transformation initiatives ongoing, with the goal of adopting the Scaled Agile Framework (SAFe, [60]) for their development organizations. Consequently, both case companies are currently concerned with discussing development speed in relation to ways of working and it appears to be the hope that transition to large-scale agile will provide an organization that can support fast deployment of new functions, especially if they are mainly software based.

At the outset of the transformation, development is done in teams usually consisting of six to eleven persons. Teams in different departments work according to different processes, with most teams working according to a waterfall process, while in some departments agile teams have emerged. Some scaling of agile methods has begun. Most projects have a budget spanning from four to nine digits in USD. Project scopes vary, from minor adjustments of a product for a specific market to entire new vehicles. A project delivery typically consists of developments in mechanics, hardware, and software technologies. The release processes are set up to serve major market introductions every few years. The company cultures are finance- and commitment-oriented, with a strong focus on a phase-gate process.

SAFe offers different configurations depending on scale and our case companies range on the higher end with respect to scale. Within the full configuration of SAFe, practices are categorized in four abstraction levels: portfolio, large solutions, program, and team level. Requirements engineering practices can be seen across all these levels, and since we are especially interested in the impact on software, we cannot even ignore the team level, which in our case is covered through technical experts among our respondents. The reasons for transitioning toward a large-scale agile framework include the realization that previous attempts to increase speed and flexibility of individual software teams are limited in their effectiveness without comprehensive support from the complete organization.

While there are differences between both case companies in certain aspects (e.g., scale of organization or number of variants to be covered by development), the similarities outweigh them. Both companies need to manage inter-dependencies between hardware and software development cycles as well as large and complex supplier networks. Traditionally, OEMs are organized in a number of departments, relating to the key building blocks of vehicles (incl. for example Powertrain, Body, Infotainment). The general aim is to map each of these departments to an agile release train and both companies were defining the practicalities of this setup with respect to development cycles and suppliers at the time of our interviews.

In order to understand the role of requirements engineering in this situation, the managers of each department were our primary respondents. In a second step, we extended our interviews to architects (responsible for the overall as well as release train specific architecture) and to experts tasked a) with driving the agile transformation, b) with ensuring quality, and c) with managing cross-cutting concerns such as base technologies. Technical experts are distinguished engineers tasked with managing (and increasing) knowledge in their specific domains, such as architecture, processes, methods, or software engineering. They are the first contact points for any question in those domains and either consult or drive any improvement initiatives relating to their expertise. Related and previous work has looked into these aspects from a developer point of view which we complement in this study.

3 Method

We used a multiple-case study design organized in two steps, with two automotive OEMs as cases (as described in Sect. 2). To acquire first-hand perspectives from our respondents beyond individual projects, we collected qualitative data through interviews. These data in a multiple-case study allow us to investigate our research questions in a real-world context [81].

3.1 Respondent selection

In the first step, we have focused our selection of respondents on high- and mid-level managers, complemented with technical experts with a high-level view on processes and architecture. Where previous work has focused on developers and requirement engineers, we complement these works with the perspective of managers, who rely heavily on requirements

Table 1 Selection of Respondents

	Role	Automotive experience
Case company one—first round of interviews		
R1	Technical expert, architecture	> 30 years
R2	Manager SW dept.	> 25 years
R3	Manager SW dept.	23 years
R4	Manager SW dept.	N/A
R5	Manager mechanical dept.	> 20 years
R6	Manager SW dept.	15 years
R7	Technical expert, architecture	18 years
R11	Manager SW dept.	1 year
R15	Manager mechanical dept.	> 20 years
R16	Technical expert, process	> 10 years
R17	Manager SW dept.	> 5 years
R18	Manager SW dept.	> 20 years
R19	Manager SW group	N/A
Case company one—second round of interviews		
R7	Technical expert, architecture	19 years
R15	Manager mechanical dept.	> 20 years
R21	Technical expert, SW quality assurance	6 years
R22	Agile transformation leader	7 years
R23	Technical leader, SE and Mgmt.	18 years
R24	Change leader	N/A
R25	Technical expert, architecture	> 20 years
R26	Technical expert, architecture	12 years
R27	Technical leader, agile SW development	> 10 years
Case company two—first round of interviews		
R8	Technical expert, process	20 years
R9	Manager SW dept.	10 years
R10	Manager SW group	> 5 years
R12	Manager SW dept.	> 25 years
R13	Manager SW dept.	21 years
R14	Manager SW tool dept.	12 years
R20	Manager system dept.	19 years
Case company two—second round of interviews		
R10	Manager SW group	> 5 years
R28	Manager, technical solution	2 years
R29	Agile transformation coordinator	> 18 years

in their roles of dividing and leading work. Targeting comparable software development intensive system areas at both case companies, we interviewed the managers of these areas and the managers of all immediate subordinate departments. The exact subdivision differs between the companies, yielding more respondents from case company one. The sampling strategy was thus to exhaustively cover corresponding parts of both company management structures.

In the second step, we validated our results with additional respondents, which we partially recruited from the initial respondents to validate our qualitative model, and partially among technical experts and change managers to broaden our perspective.

Table 1 gives an overview of our respondents, their roles and years of experience. In the first step, a total of 20 respondents were interviewed, over the two case companies, with each interview lasting approximately one hour. In the second step, we conducted 12 interviews, interviewing nine additional respondents and three recurring, over the two companies. For confidentiality, respondents are kept anonymous and referred to with running ids R1–R29.

This research was not conducted under national regulations that demand prior approval from an ethics board.

Nevertheless, prior to each interview, we acquired consent from the respondent to use their responses in anonymized form.

3.2 First step of the study

For the first round of data collection, we employed a qualitative approach, using semi-structured interviews. Semi-structured interviews employ an interview guide with questions, but allow the order of questions to vary to fit the natural flow of the conversation. Our interview questions took an exploratory approach. We asked respondents about their current situation and development speed, intended changes to ways of working, hindrances to these changes, and how to overcome these hindrances. The data have a broader scope than requirements engineering; in this paper, we report on requirement-related aspects within this broader scope. Except for the fact that this paper is an extension of a previous publication [2], none of these data have been used in prior publications.

All interviews[4] in the first round were conducted by the first author together with one or more of the coauthors and were recorded and transcribed. The resulting transcripts span 340 pages (more than 165.000 words). We relied on thematic coding [34] in which we assigned one or more codes to relevant statements in the transcripts. For this task, the authors split into two teams that worked in parallel. One team focused on a priori coding based on our research questions, while the second team performed complementary emerging coding, starting from any mention of requirements in the transcripts. We then merged the coding schemes from both teams in a workshop to ensure that we do cover the full data in our synthesis of findings. As an example of the coding process, the following statements were first coded as *contracts* and *procurements*, respectively. They were then grouped with the theme *requirements-based contracts hinder fast collaboration.*

> "As soon as a change occurs, it has to be stated in the contract and re-negotiated. And then there will be new requirements specifications and such." – R7

> "Another impact is how much [Case company 1] specifics you push out in your requirement specifications. If there's a lot of [Case company 1] specifics, yeah then it'll take a very long time and be very expensive to do this stuff. And as soon as we're to change something we need to go out to our suppliers and [negotiate] these changes." – R7

Quotes have been translated from the respondents' native language to English and edited for readability. Colloquialisms have been kept, in order to convey the tone of the conversation and to reflect the informal nature of the interview setting.

3.3 Second step of the study

The second round of interviews was based on a slightly adjusted process. Since this round was not as exploratory in nature, we designed an interview instrument with both closed and open questions.[5] We focused on Likert scale questions, based on the themes that emerged from the first round of interviews. The interview format also allowed respondents to ask for clarification of the questions, if necessary. In addition, we asked respondents about the reasoning behind their answers and whether any aspects were missing.

Interviews in the second round differed in length; interviews with initial respondents went rather quick, while interviews with new respondents took up to 90min and more. The average is around 37min. We conducted the interviews in parallel and recorded them (with the exception of one, where we took extensive notes). The recording thus covered the reasoning about the Likert scale answers and the discussion of potentially missing aspects. Each author made themselves familiar with the recordings and one author carefully went through all recordings to extract the key issues with respect to our research questions.

3.4 Threats to validity

To classify potential threats to validity, and reason about our corresponding mitigation strategies, we follow the scheme proposed by Runeson and Höst [81].

Internal In terms of threats to internal validity, we followed a systematic approach in setting up the study and best practice guidelines for both data collection and analysis [81]. Moreover, the interview questions might have influenced the respondents to consider factors that would increase development speed, at the expense of what currently provides speed. To mitigate this risk, we spent time discussing how to phrase the questions and types of questions to avoid. Still, despite making an effort to ask questions in a neutral way, respondents might have considered mainly negative aspects of the current requirements engineering practices with respect to development speed.

External Generalizability is inherently limited for case studies. All interviews were done within one country with automotive companies developing large-scale software for

[4] The interview guide used for the first round of interviews is available online at https://doi.org/10.5281/zenodo.1299206.

[5] The instrument used for the second round of interviews is available online at https://doi.org/10.5281/zenodo.1888011.

embedded systems, and this software is expected to have a long lifetime. Therefore, our findings may not apply to smaller companies, other countries, or for software with a shorter life expectancy. Global presence of both case companies may make our findings more general, but cultural aspects may persist and could have an impact on how practitioners reason about requirements engineering in relation to development speed.

Moreover, we focused the study on one system area at each company, selected for being the most software development intensive ones at the respective companies. Although we complemented with additional respondents from other system areas, an in-depth study of another area could uncover further detail, possibly contradicting our findings.

Construct All authors have prior experience with the automotive domain, which we leverage to ensure construct validity. The academic coauthors have longstanding collaborations with both case companies, whereas the industrial coauthors were, at the time the study data were collected, working at case companies one and two, respectively. Thus, the interview situation was informal and characterized by mutual trust. Furthermore, the interview guide was refined through multiple iterations, with input from senior industry experts.

Reliability To ensure reliability, we used observer triangulation during the interviews. In the first round of interviews, the first author conducted the interviews, joined by one or more of the industrial coauthors, who observed and asked follow-up questions for additional clarifications. Three of the authors did the coding independently of each other and discussed the results afterward. The translation of quotes was done by the main author and checked by the coauthors for correction of any translation error.

The second round of interviews was conducted by several authors in parallel. The interviews were recorded (with one exception, where we took extensive notes). One of the authors listened through all recordings, others listened in to particularly controversial or interesting passages. Thus, we made sure that at least two authors were familiar with each interview. We then discussed new aspects that should be added to our findings with respect to a research question, new aspects relating to any theme in our results, and – based on the Likert scale questions—any findings about the relation to the agile transformation of each company.

Comparing the different approaches in both interview rounds, it is noteworthy to discuss the cost and value of creating interview transcriptions. Our first interviews were much more exploratory in nature. Through several iterations, we revisited the data and attempted to categorize our codes in a meaningful way. This forced us to revisit the interview data several times and would not have been feasible without transcripts that allowed for full-text search as well as comprehensive tracing from themes to codes and to interviews. In contrast, the second round aimed to collect additional views on the themes we already had. In addition, we explicitly asked for any new aspects to add. Thus, it was straight forward to extend our existing model with the new data and the transcriptions did not appear to offer a positive return on the time we would have to invest.

It is important to note that we did not aim for saturation [83]. In the first round of interviews, we instead exhaustively interviewed all potential respondents (all managers at the targeted level). In the second round, we actively aimed for broadening the diversity of our sampling, by including key stakeholders of many different roles. Thus, we were not surprised to learn new aspects and we cannot guarantee that further interviews would not yield even more. Clearly, more work is needed to establish a solid theory about the effects of RE on development speed in scaled agile contexts. This paper is a step in that direction.

4 Findings overview

Figure 1 gives an overview of our findings in relation to our research questions. The figure connects key characteristics of current and future automotive RE that emerged from our interviews with the overall goal of our case companies to increase development speed.

The left-hand side in Fig. 1 shows the themes related to RQ1 (impact of current way of RE on Development speed). For example, the figure shows that a *rigid requirements process* forces early decisions, and by that, it impacts developments speed negatively. Similarly, on the right-hand side, we list themes that emerged in relation to RQ2 (aspects of future ways of RE and their relation to development speed). As an example, *post-development specification* positively impacts development speed, because it reduces the workload. For RQ3, we indicate with a dashed border those themes which our respondents considered addressed by the ongoing agile transformations.

We discuss the findings for each research question individually in the following sections.

5 Which aspects of the current way of working with requirements impact development speed (RQ1)?

With respect to RQ1, we found the following themes; they indicate that our respondents judge the current way of working as not supporting the desired development speed.

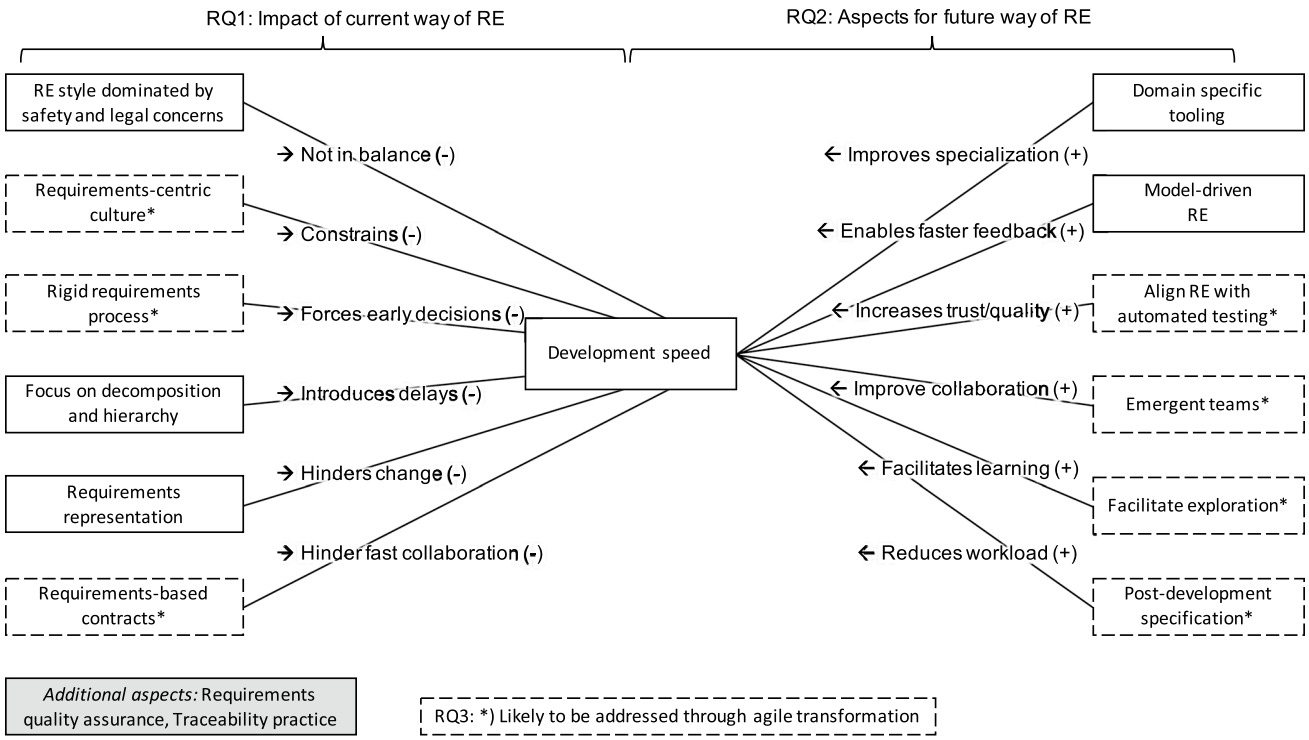

Fig. 1 Causal relations between concepts. Dashed line indicates which aspects will likely be addressed through the agile transformation (RQ3), gray box lists additional concepts from the second round of interviews

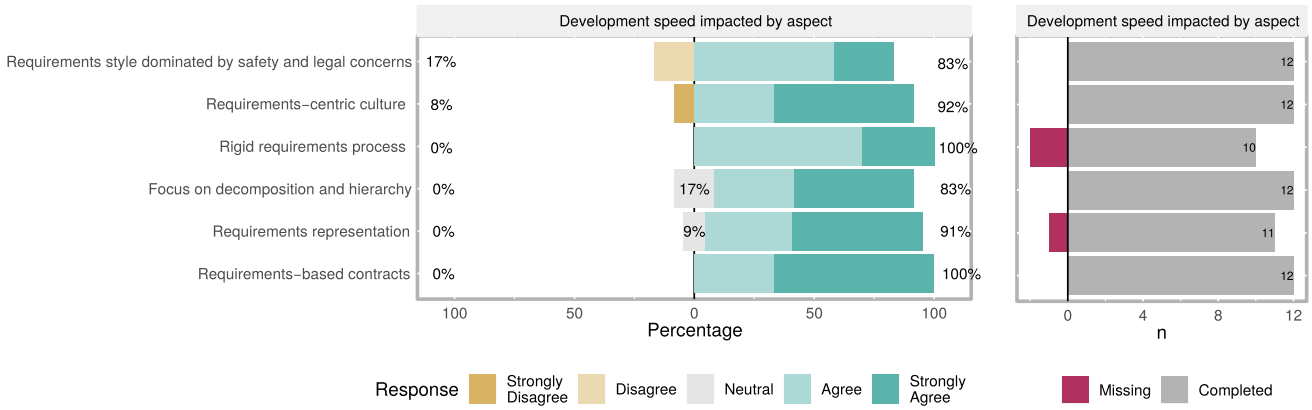

Fig. 2 Level of agreement with the impact of aspects of the current way of working on development speed

Summary: Impact of current RE on dev. speed (RQ1)

(1) Requirements style dominated by *safety* and legal concerns neglects development speed
(2) Requirements-centric *culture* constrains development speed
(3) Rigid requirements *process* forces early decisions
(4) Focus on *decomposition* and *hierarchy* introduces delays
(5) Requirements *representation* hinders change
(6) Requirements-based *contracts* hinder fast collaboration
(*) Additional aspects raised in second round of interviews: current *traceability practice* and *quality assurance of requirements* (imbalance between risk vs. lead-time)

Note that Themes 1–6 emerged from our first round of interviews. Figure 2 shows the level of agreement from respondents in the second round that those aspects indeed impact development speed. We note a few disagreements, concerned with requirements style and requirements-centric culture, which we will discuss below. In addition, two more aspects emerged from our discussions with respondents. Note also that the findings in the remainder of the section come from both rounds of interviews.

For each theme, we give details based on a narrative supported by representative quotes in the remainder of this section.

5.1 RE style dominated by safety and legal concerns

Automotive systems are inherently safety-critical, not least because of how they are perceived by customers and users:

> "That's something that can be perceived as very frightening for the customers and also be dangerous if you just out of the blue suddenly brake the car." – R6

Consequently, safety-related and legal issues must be handled systematically, e.g., by tracing development and verification results to requirements.

> "We have product liability, legal requirements, documentation obligations. If something happens—if someone crashes and the airbag doesn't deploy—in accordance with which requirements have we developed, in accordance with which requirements have we tested and verified and so on for our product liability." – R3

Our respondents confirm the fundamental impact this has on requirements work.

> "We have safety classed components. So it affects a lot how we think about requirements decomposition, traceability, test and so on. It sets the bar" – R19

Thus, safety and legal requirements are requirements in the traditional sense of the word, quite different from agile user stories that represent an often negotiable user goal. Depending on standards and frameworks, this can imply certain ways of working, including certain approaches toward decomposition or an implied bias toward upfront analysis.

> "We're very much driven by safety requirements, ISO 26262, and that process is very waterfall. /.../ One wants to see the decomposition from high-level requirements all the way down to component requirements." – R8

Such decomposition is necessary to manage safety and legal concerns, but the usual way to do it upfront before development can generate unnecessary delays. In addition, the implied reviews and certifications also challenge continuous integration.

> "So we still have a bottleneck with certifications and government stamps [authority approvals]. CI is difficult for us at Powertrain because a lot of what we have affects legal requirements and emissions, quite simply. We surely have some areas where we could run CI, but it's perhaps not as obvious for us as it is for Infotainment or Navigation." – R15

Thus, in order to have a competitive edge, and to increase development speed, the question is whether equally safe systems can be built with an approach that allows for more flexibility (to the extent necessary for implementing, e.g., CI).

Summarizing, ten out of the second-round respondents agreed or strongly agreed that a RE style dominated by safety and legal concerns slows down the development speed. However, we highlight that all respondents pointed out that requirements related to safety and legal concerns have to be treated seriously. It can then happen that some requirements not related to safety and legal concerns get treated with that same rigor. Finally, some respondents thought that safety and legal issues were not the main reason slowing down the development process, but that they could be used as an excuse for being slow.

5.1.1 Safety and legal concerns in context of related work

Our findings are in line with an increasing body of work that reports on efforts to balance safety and regulation with agility [31, 37] or continuous software engineering [30].

To our knowledge, such approaches have not been rolled out on the scale of a complete OEM and its software-related supply chain. However, experience on the level of individual teams exist [7, 53], but often describe struggling with respect to conflicts between safety-dominated culture on system level and agile-dominated culture on team level [52].

5.2 Requirements-centric culture constrains development speed

With OEMs focusing on integrating components from hundreds of suppliers into highly differentiated products, they have developed requirements engineering as a core competency and integral part of their culture. Although the importance of understanding requirements and constraints cannot be discarded, there is an indication of systematic over-specification or even over-emphasis on requirements specification. Contrariwise, new pressure for increased development speed demands more focus on implementation.

> "Requirement specifications, it's in our spine, also we in the electronics and software community, us oldies. It doesn't work, and we saw that quite clearly when we built the [system] architecture we have today." – R1

A requirements-centric culture puts much focus on making requirement specification. There is a long history of writing large requirements specifications particular to suppliers. This strong focus on documentation seems to not always pay off. The value that requirements could have come in far too

late, so they are often disregarded, but still a lot of effort is spent on them. However, we want to highlight that none of our respondents indicated that requirements were unnecessary, which can be best illustrated by:

> "There is an assumption that we can develop the same thing without requirements, and that is not true." – R22

The level of details creates expectations with the developers and they become requirements bound. The number of requirement authors per developer is also regarded as too high.

> "We have a tradition of focusing a lot on specification. We specify our way to the solution, which isn't quite what we want to do now. It means that we're staffed with highly skilled specification writers, but very few developers. The developers are also used to someone telling them exactly what to do." – R10

There is also room for improving how requirements are specified. The problem is not necessarily the strong focus on requirements specification and the amount of requirements. The problem is in a lack of common template or way of dealing with them. This would have many benefits: (1) cut down time, (2) increase the quality, and (3) facilitate understandability of developers and testers.

Also, forecasting is the typical approach.

> "I think we have created a structure where we have tried to specify and plan our way out of a very complex reality." – R11

Transition to more agile, continuous, and product focused development will affect the role of requirements in the overall development process.

> "[Requirements] can be a lot fewer; if you become product oriented and look more at agile and such, then you can scrap lots of requirements. But it's an effect of a faster and more efficient flow." – R16

Thus, current requirements-centric culture constrains development speed. Our respondents suggest that in order to increase development speed, the role of requirements should be adjusted. This is also shown in the next theme, related to the requirements engineering processes.

Summarizing, eleven out of the second-round respondents agreed or strongly agreed that requirements-centric culture slows down development speed. There is a large focus on thinking first and on getting things right from the beginning. Indeed, some of the requirements need a lot of upfront modeling and experimentation. However, the general feeling is that sometimes this goes too far; better then to try out a bit first, learn from it, change it if necessary, and then document it more thoroughly afterward. One respondent also said that there should be an effort toward creating a culture of failing fast and learning from it.

The disagreement is based on the opinion that a requirements-centric culture is a consequence of managing complex supplier networks. The disagreeing respondent is in their role concerned with managing the quality of supplier contributions. We interpret the otherwise strong agreement to this aspect as the hope that new ways of defining contracts not based on concrete requirements will mitigate such quality problems more efficiently.

5.2.1 Requirements-centric culture versus speed in context of related work

Recently, an increasing number of papers concern requirements engineering in relation to agile methods [4, 47, 52]. We believe that our findings support these previous works in that the mindset about the role of requirements must indeed change both in research and practice to enable faster development of increasingly complex systems. Related work warns that a push to faster development, e.g., based on agile methods, can lead to neglecting quality requirements [4, 79] and also demands for increasing the ability of being an agile customer [79]. Related work also suggests that agility on an organizational level introduces challenges with respect to prioritization and growing technical debt [38].

5.3 Rigid requirements process forces early decisions

In addition to the requirements style and requirements-centric culture, the current requirements process impacts also the development speed.

> "I don't know really if I think it's crazy wrong how we view requirements, but how we've made a process that throws them [around] is quite devastating. /.../ It's more how we choose to have the development process, I think. I don't have anything against requirements." – R16

A recurring theme from our respondents is the early freeze of requirements that the processes prescribe.

> "The whole project setup is built on planning really the entire duration. You have plans you lock, freeze and everything. Freeze of requirements and freeze of everything, and you do that early." – R8

This requirements freeze is a consequence of a cultural assumption that an accurate, useful upfront specification is indeed possible.

> "I'm sure requirements work is equally difficult at other companies. It's difficult and complicated here as well, perhaps mainly because the forms of it today are built around the idea that you are able to state exactly what you want, very early." – R12

Our respondents express doubts that this assumption holds within all aspects of an increasingly complex product in a rapidly changing market.

> "Have a smarter way of working with content and backlog, instead of saying that '2020 we'll have this itty-bitty function' and we start specifying and discovery it was super difficult. 'But now we've said, now we've promised.' and so it goes." – R2

Also, the time between specifying detailed requirements and getting feedback through integration and acceptance testing, which, in turn, often leads to requirement changes, is very long.

> "Some time passes and then there's integration, and then there's system test at the supplier, and then something is sent back [to us] 6 or 8 or 12 weeks later, from when you released a specification." – R17

> "In case we have a node where we send the specification [off for development], and it then takes half a year. Then people have moved on" – R18

Summarizing, out of the ten second-round respondents that answered, all agreed or strongly agreed that rigid requirements processes forcing early decisions have a negative impact on development speed (two respondents choose not to express an opinion about this aspect). Despite such strong agreement, it seems that becoming more agile than today is not simple. One important aspect is to promote a culture in which requirements are not put into stone too early, but rather updated as one learns more. The respondents that did not offer an opinion could relate to the problems that our study brought up but proposed that strictly following improved processes and improved training would be required to avoid problems in supplier relationships.

5.3.1 Process and early decisions in context of related work

The difficulties of early decisions have been discussed for a long time in software engineering, prominently for example in relation to the cone of uncertainty [11]. While some researchers highlight the merit of upfront requirements analysis and specification to avoid unnecessary work and identify problems early [68], others have reported difficulties when forcing decisions in phases when not enough information is available [25]. Our findings add to the empirical evidence of the need for future research with respect to (1) differentiating between different levels (e.g., team, product, portfolio [60], and (2) supporting end-to-end responsibility of (product) teams [38, 39], and (3) moderating local decision making [24, 28].

5.4 Focus on decomposition and hierarchy introduces delays

Because of the complexity of automotive systems, development is generally organized in a number of abstraction levels. This also results in a corresponding hierarchy, many roles, and a large number of handovers, all of which are considered to negatively affect development speed.

> "I think we have too many roles /.../ We've tried to establish a logic for how we decompose requirements. It's very V-model influenced. So we think that for each new level we need a new role, and that's very many handovers. We start with some property making requirements, and we have a function level making requirements, we have a function realization level distributing and formulating requirements, we have a subsystem level, and then we land on component level, and hardware and software components and so on. Just the fact that we have so many steps hampers us, I think." – R6

This can be seen as a vicious circle: the process emphasizes decomposition of requirements and, in turn, this leads to the creation of new roles. These roles can then become a source of new requirements. Consequently, it is difficult to make decisions on requirements, since too many different roles throughout the development value-chain must potentially be involved.

> "The result is that very, very many [engineers] come with requirements on things needing to be done, without there existing a structure [wherein] to prioritize what's most important. It's a somewhat impossible equation for the developer, who is to realize this function. There can be 80, 100, or 150 requirements from different sources without any real sorting." – R11

However, some hierarchy seems to be necessary and cannot be avoided:

> "Organizations are hierarchies by nature. If there's an organization where the parts that are building something, normally at the bottom of the hierarchy, if they cannot talk directly to each other, if they need to follow the hierarchy, then it becomes very slow and everything is bad." – R7

From an organizational point of view, it is thus good if talk can happen across the hierarchy, not merely along the hierarchy. Conversely, from a technical point of view,

if communication in a software system bypasses intended structures, this can turn the system into the proverbial spaghetti. It thus seems that organizational hierarchy need not by itself be an impediment to development speed. However, there seems to be a limit to how many layers that are useful to have. In our cases, the companies use up to seven layers to break down the requirements, which seems far too much.

> "There are aspects that require being broken down, but we overuse this too much." – R25

> "I think we can get rid of one or two layers" – R21

Also, all these layers create both dependencies and isolation:

> "All these layers create a lot of dependencies." – R24

> "All these layers and decomposition are super negative, create a lot of isolated groups that do not talk together." – R27

However, one respondent pointed out that legal requirements can make decomposition necessary in order to get speed further downstream.

Prioritization between requirements is thus eventually left to each developer.

> "So, all the different requirements from the different projects, and the different lead times, are anyway going to the same, it's the same developer sitting at the bottom." – R13

Not being able to decide about requirements as well as a lack of ready access to business representatives delay development within cross-functional teams.

> "One needs contact with our business side /.../ and have this difficult dialog about what's most important to do, and in which order. Instead of each developer meeting and discussing all conflicting requirements." – R11

Summarizing, ten out of the second-round respondents agreed or strongly agreed that focus on decomposition and hierarchy introduce delays and slow down the development process. However, some hierarchy seems to be necessary and cannot be avoided, since it is part of the nature of an organization. Legal requirements can make decomposition necessary, in order to get speed further downstream. However, having too many layers seems also to create both dependencies and isolation.

5.4.1 Decomposition and hierarchy in context of related work

Efficiently structuring organizations has been reported as a key challenge for agility in mechatronic organizations [7]. Even without the explicit goal to transition to agile, strict vertical hierarchies and purely plan-driven ways of working alone have been reported to fail for engineering complex automotive systems [25]. This is to a large extent due to too many levels, including (1) the inability to ensure that decisions are made on the most appropriate abstraction [25], (2) the challenges with clearly communicating about the relationship between high-level, vague requirements, and details added to lower abstraction levels later on [62], and (3) the difficulties to coordinate requirements (and their change) across these levels [8].

5.5 Requirements representation hinders change

The current way of specifying assumes upfront knowledge about an optimal decomposition of distributed functionality. Teams then work on requirement specifications for separate components. To increase development speed, our respondents indicate that instead of focusing on detailed specifications of individual components, one should instead provide a high-level view on how a distributed functionality will be provided through the interaction of those components, including an early agreement of the interfaces.

> "What we do is that we specify [component] content, we don't specify the exact interfaces. Which means that when a number of coders, each on their own, develop and we put this together as distributed functions, of course it doesn't work. Then, we run a couple of loops before we've found these interfaces and stabilized them." – R16

> "There are no clear interfaces in the software structure or the function structure today, which is an obstacle. You could say that there's a brake system, an infotainment system, a chassis system and so on. But we haven't built it like that, and it's something that hinders us because we can't create isolated teams who can easily work independently. Because they are so dependent on other teams. The systems depend on each other." – R13

Requirements are typically expressed in prose, which is an obstacle for testing and iterative development. Findings on alternative ways of expressing requirement are given in Sect. 6.2.

> "Our software specification is currently text-based. It's very hard to iterate and test." – R3

At the lowest level the textual specification is practically pseudocode, consequently hard to understand, and of limited value to support speed in development.

> "Some specifications are nothing other than pseudo-C-code to describe a functional behavior. And those

specifications become extremely difficult to understand. " – R13

"Or you write at a very very detailed level, with the consequence that you write almost pseudocode, only in prose." – R11

Our respondents also point out that it is not necessarily bad to have textual requirements. Instead, there is a need for more training in writing good quality requirements.

"I think the text, natural language, is a universal tool that should be used. It should be used brief, succinct, and precise. There are methods for that." – R21

It is very important to have requirements expressed at the right level of abstraction.

"What's difficult, I think, is to discuss the level of requirements: how detailed do you need to make a requirement before it becomes design?" – R19

This would also reduce the number of requirements in general. Furthermore, using the framework AUTOSAR can bring great benefit since one can refer to the standard, and this leads to less requirement specification. In addition, a few organizations have moved away from detailed requirements, such as the infotainment organization, which describes the requirements at a high level, using use cases. Permitting more flexibility, however, requires suppliers understanding the domain very well.

The textual specification is sometimes used also to communicate with suppliers. The situation is made even more complex by possible further decomposition of the requirements and translation to another language.

"When they wrote their specification it was sent to the supplier. But the supplier did not in all parts know English, so then the specification was translated to Japanese. Then it was decomposed to the different sub-areas. After that, that company had different suppliers, and they were outside of Japan. So then it was translated to English again." – R4

Summarizing, out of the eleven second-round respondents that answered, ten agree or strongly agree that the current way of specifying requirements hinders change. Having a textual representation of requirements can be seen as problematic; however, the root of the problem seems to be in the lack of expertise in writing good quality requirements.

5.5.1 Requirements representation in context of related work

Automotive requirements engineering has been reported to suffer from scale and complexity [62, 93]. In order to manage requirements at this scale, textual natural language requirements specifications are the norm in the embedded industry [35, 73, 87, 94]. Textual requirements encourage requirements reuse [41], enable formal exchange formats (such as ReqIF [48]), and support a systematic and often heavy-weight approach to baselining and change management. Braun et al. [13] report three fundamental challenges they observed in the automotive industry. The reported challenges are increasing size and complexity of software-intensive embedded systems, increasing economic relevance of software in the automotive domain, and inappropriate requirements engineering. Yet, in line with our findings, practitioners have been reported to be increasingly dissatisfied with using natural language for requirements specification [87] and the appropriateness of requirements engineering approaches in automotive [13]. We believe that our findings in relation to related work encourage more work on modeling behavioral requirements in local teams [63] with a focus on identifying cross-cutting concerns. This should be complemented with an effort to define and evolve interfaces between such teams [76]. At the moment, however, specification is too often solution based, less often scenario based, and rarely goal based in the automotive domain [44]. In line with our findings, this focus on specifying solutions has been reported to scale poorly for complex systems, especially with respect to managing change.

5.6 Requirements-based contracts hinder fast collaboration

OEMs rely on requirements to define contracts with suppliers and specify what should be delivered by sub-contractors. Giving requirements such legal quality does, however, hinder fast collaboration:

"But to work as we do now, where we specify in detail what [the suppliers] should do, and then wait for them to implement it, and send it back, it's not a fast way to solve problems." – R10

This is especially true when inevitable changes become necessary. In particular, the current way of working does not encourage early feedback from suppliers and can introduce unnecessary costs.

"There's no economic incentive to be part early. Then it drops to change management. We change the specification, it should say 'and' instead of 'or' or something. But change management costs a lot, a lot more than the software change itself." – R3

The payment model of the current change management setup does not foster collaboration.

> "Then we lock ourselves in and bring requirements not possible to realize, and of course [the supplier] wants to be paid for doing this change." – R8

In addition to change management, payment is typically linked to component cost.

> Another important part is finding contracts and agreements making it a win–win to have a good dialog [throughout the development process]. It's not that today. Today, we put a specification on the table and then we negotiate about what the component cost will be." – R3

However, the current way of optimizing for low component cost does mean that closer collaboration with suppliers can be more expensive.

> "Today we specify in detail to our suppliers and then we use them as an implementation resource. I think we can tie them [in] closer and develop more together. Though we have contracts that hinder us today, a delivery can for example have a price, meaning that we keep the number of deliveries low. /.../ It also depends on which component they supply. With some suppliers we have a closer collaboration, but that has cost a lot more." – R13

In the typical setup, however, suppliers are implicitly encouraged to develop software as late as possible, and thus avoid additional cost through changes. This can significantly reduce development speed since feedback and problems surface very late. A key function of contracts is to clarify the distribution of responsibility. Basing contracts on strict requirements mean that any deviation can be seen as a breach of responsibility, with associated consequences. This hinders open-ended collaboration.

> "Trying to stretch reality and reach further by make requirements more incisive, and then trying to handshake those with a supplier, you've got an arduous journey. Because the supplier will regard it as a contract and say 'If I can't meet this requirement I'll be held accountable. What if I know I can't detect this little thingy in that long distance all 24 hours in all weathers.' because it's easy to write such a requirement. You get a discussion about deviation management and spend plenty of time on that, instead of starting developing and see how far you get." – R6

Also, with regard to contracts, the current practice has a built-in slowness.

> "[For] the contracts toward the suppliers, it's obvious you need solid requirements to reason around. But it also steers, if you take agile vs. waterfall, where waterfall says to add requirements, and then someone works on them and cascades them further. /.../ But I personally think it's quite devastating because it builds this loop-time." – R16

Summarizing, all of the second-round respondents agree or strongly agree that requirements-based contracts hinder fast collaboration between OEMs and suppliers. This is particularly true when changes become necessary. To counteract potential problems, in the typical setup, suppliers are implicitly encouraged to postpone the actual development of code as much as possible. That this aspect receives such broad agreement highlights its importance. Many problems with requirements engineering, such as rigid processes, decomposition, and requirements-centric culture, would be easier to solve if a better way of designing contracts was found.

5.6.1 Contracts and collaboration in context of related work

Few works discuss contracts in the context of continuous software engineering. In line with our findings, legal contracts have been reported as an impediment for inter-organizational continuous integration and delivery, but also to facilitate negotiations between organizations [92]. More works have been published on agile contracts [42, 100], suggesting, for example, to keep in mind the agile capabilities of customers, when negotiating a contract that allows agile development [42]. Systematic frameworks for defining agile contracts are currently emerging [100], but to our knowledge, there is not much guidance for defining contracts for agility or continuous software engineering in complex supply chains. Our findings indicate that the role of requirements in such guidelines must be different from today.

5.7 Additional aspects: quality assurance and traceability

From our question about additional aspects to add, two things emerged: Requirements Quality assurance and Traceability. With respect to quality assurance of requirements, respondents mentioned a lack of balance between the risk of requirements-related problems with increased lead-time through extensive reviews.

> "Formal peer reviews etc. of requirement specifications today. /.../ If we have long and too comprehensive requirements we are spending enormous amounts of time doing these kinds of activities." – R23

It is necessary to ensure that requirements are testable and that they conform to shared basic technology requirements, otherwise, ordering components from suppliers will be overly expensive. However, if a supplier's contribution does

not achieve the desired effect, the resulting change requests will be similarly expensive. The hope for the future seems to be flexible contracts. Thus, currently, quality assurance is negatively impacting development speed, especially because changes happen often and this leads to a lot of rework with suppliers. It is important to highlight that this does not mean that quality is secondary. Instead, this is to highlight the importance of finding a balance.

Traceability is identified as an important instrument to keep the quality of requirements and at the same time to align them with other artifacts. Its implementation is not perceived as benefiting speed, however.

> "Some of the traceability requirements are actually slowing us down because it's not stated clearly what should be traceable." – R27

Respondents also point out that current traceability management solutions are not satisfactory and they slow down development. In general terms, effort and benefit of tracing are not distributed in a balanced way, thus leading to bad quality traces and unnecessary overhead, e.g., maintaining useless trace links.

5.7.1 Quality assurance and traceability management solutions in the context of related works

The industry need for quality assurance of requirements is testified in the investigation made by Sikora et al. [86, 88] In a case study of six companies, Bjarnason et al. [8] describe challenges and practices in aligning requirements with verification and validation. Unterkalmsteiner et al. [91] present an assessment tool called REST-bench, which illustrates the coordination in software development projects and identify concrete improvement opportunities. The tool has been defined and validated together with five companies.

For what concerns traceability, the majority of empirical studies on traceability, focuses on validating specific technical approaches [3, 20, 85], or specific aspects of traceability such as assessment [80] and benefits of traceability [66]. Demuth et al. [22] conducted a study on how to use traceability for systems engineering to facilitate change notification and consistency checking of artifacts. Figueiredo and De Souza [29] and Helming et al. [40] describe tools for facilitating collaboration in a distributed environment or notifying users about changes.

Wohlrab et al. [96] conducted a multiple exploratory case study with 24 individuals from 15 industrial projects, with the aim of understanding collaborative aspects of traceability management and how it is situated in existing development contexts. They provide empirical evidence of how culture, processes, and organization impact traceability management and collaboration, and principles to support practitioners with collaborative traceability management.

Cleland-Huang [17] highlights that traceability in projects following agile methodologies is just as important as in non-agile ones. Espinoza and Garbajosa [26] argue that the lack of formal documentation and formal requirements in agile contexts calls for traceability practices that go beyond those of non-agile projects. Gayer et al. [33] give a concrete example of integrating traceability in an agile context.

6 Which new aspects should be considered when defining a new way of working with requirements to increase development speed (RQ2)?

As our summary of the themes in relation to RQ2 below shows, our respondents' suggestions aim to define a way of working that emphasizes requirements to (1) support high-speed development and (2) ensure that the necessary technical documentation and traceability are established without overhead.

Summary: Desired way of working with RE (RQ2)	
(1)	Aim for domain- or context-specific *requirements tooling*
(2)	Leverage *model-based RE* for fast feedback
(3)	Align requirements and *automated testing*
(4)	Emergent teams to improve collaboration
(5)	Facilitate learning through *exploration*
(6)	Complement *lightweight pre-development RE* with consistent/accurate *post-development specification*

In our second round of interviews, we asked respondents whether these new aspects may have a positive effect on development speed. Since many of these aspects have not yet been rolled out and only limited concrete experience existed, we aimed to make things a bit more specific by also asking whether they think it would be a good idea to spend effort on these aspects. Note that Theme 1–6 emerged from our first round of interviews and got confirmation in the second round of interviews. In the second round of interviews, we also found recommendations for focusing on interactions instead of on artifacts, the need for iterative requirements engineering, and for separating legal and safety-critical requirements from other requirements. These aspects are complementary to our results but do fit into a broader view of the six themes above. Figure 3 gives an overview of their answers. Sometimes, respondents disagree that an aspect should be addressed locally. But generally, if an aspect is deemed important, our respondents also suggest to address it with high priority.

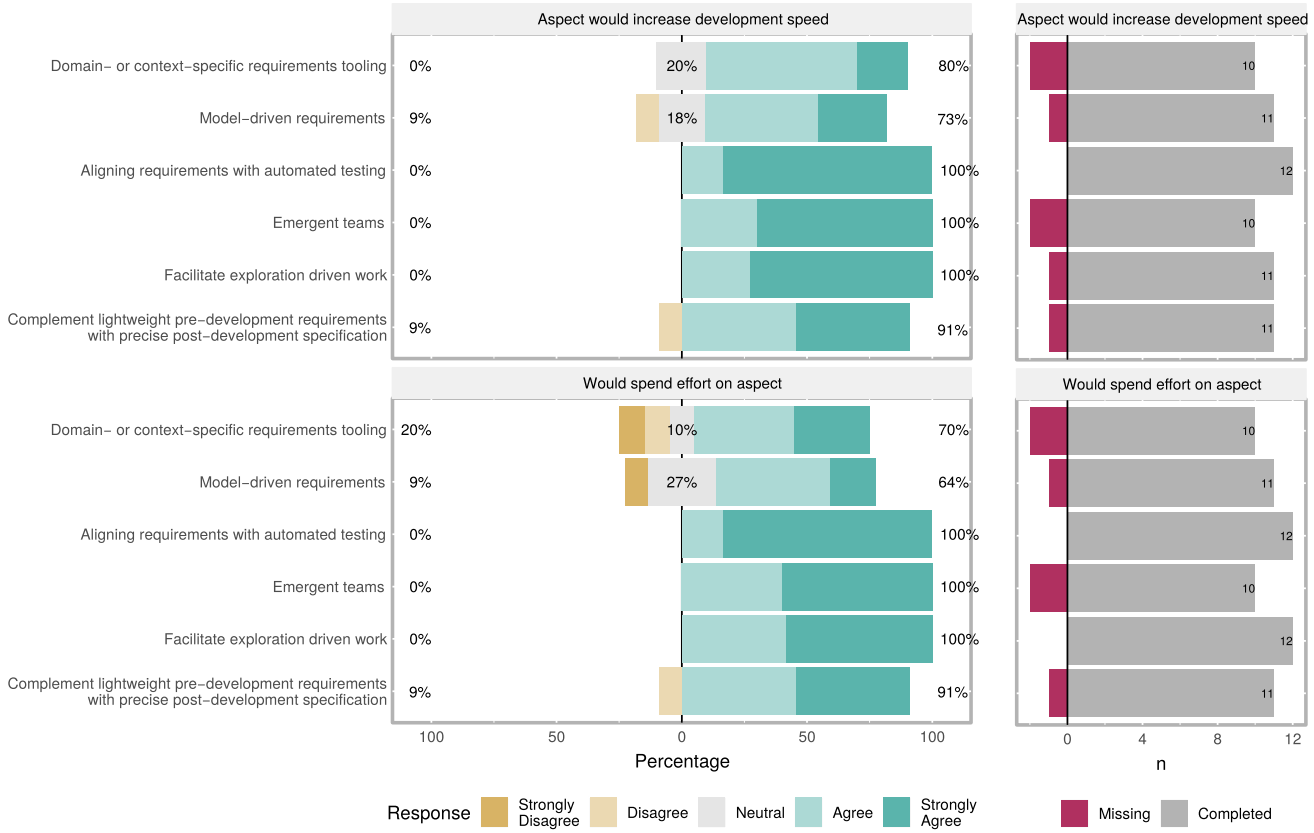

Fig. 3 Agreement of respondents to the themes in relation to RQ2

We present details about each of these themes as narratives, supported by example quotes from interviews in the remainder of this section.

6.1 Aim for domain- or context-specific requirements tooling

Our respondents clearly express that tooling is an important aspect. With respect to requirement tools, however, there was quite some frustration expressed.

> "I've never encountered a requirements management tool where someone said: 'This is so [swearing omitted] good, it makes my life worth living.' Rather everyone is swearing over it, and I think that unfortunately, it's difficult to make a requirements management tool good." – R17

This frustration comes partly from the fact that requirements on different levels are related to different parts of the system. One example of this is the signal database, which defines data to be shared between the different components in a car. Requirements often relate to specific signals; without tool support, it is very hard to avoid inconsistencies and unnecessary rework. Domain-specific system engineering tools can offer such support and should, according to our respondents, be more widely used:

> "[Tool name withheld] is a tool, on system level, which gives an extract to our signal database, where you can configure your control units. That chain is much more exact than a bunch of requirement specifications. /.../ It has plenty of shortcomings, but it's still a sign that a bunch of text-based requirement documents is old fashioned, it doesn't work in the modern world." – R1

One major driver for changing tooling is the trend to develop more software in-house instead of ordering it from suppliers.

> "Historically, [company 1] has sourced all software externally, meaning that in [tool name withheld] you work until you have a specification at a certain level. Then you pass it to a supplier who continues decomposing it. When we started developing our own software we got [tool name withheld] even though our needs are the same as for an external supplier, in terms of managing the requirements, decomposing them,

linking test cases to requirements. There [tool name withheld] hasn't worked good enough." – R19

Reasons relate to the drivers of moving software development in-house, most prominently the goal to increase flexibility and ability to quickly relate to change. Thus, existing tools and their implied workflows introduce undesirable delay and do not, to the desired extent, facilitate communication across levels.

Summarizing, eight out of the second-round respondents that answered agree that tooling is important. However, current solutions are not completely satisfactory, e.g., for what concerns support for avoiding inconsistencies, support for flexibility, and communication. Since substantial work has done on these aspects, not all agree that further effort should be spent. But definitely, tooling can enable development speed if it clearly supports incremental work.

6.1.1 Domain and context-specific tooling in context of related work

Insufficiencies of requirements tooling with respect to specific industry needs are known in the literature [15]. Specifically, it is essential to find a trade-off between diversity and alignment of requirements engineering practices in organizations [56]. This must also be reflected in tools, which must be carefully selected to support the specific needs of a given context [15, 19]. Even though this has proven to be difficult at the scale of automotive system engineering [98], our findings suggest that an investment in this aspect is important and further research is dearly needed.

6.2 Leverage model-based RE for fast feedback

Independent from tooling, our respondents also emphasized the potential of model-based requirements engineering. They expect that models will scale better than textual requirements, thus helping to better manage complexity.

"A wall of text of 1200 pages. No supplier in the world cares about it. And even if they do, they will interpret it entirely different than what the writers intended. So you need a much more exact way of describing what you want. Model basis, with complementary simple text, that's number one." – R1

This partly relates to the requirements representation, as one respondent points out:

"In principle, I'd like to get away from as much text-based requirements as possible, for two reasons. First: it's damn difficult to understand. Second: there are always errors when there's much text mass, and there's interpretation." – R2

Our respondents, as highlighted in Sect. 5.5, agree that the way requirements are done today, mostly text-based, slows down the development process. At the same time, replacing textual requirements with models is not always an option.

"If you look at system security, you cannot simply hand in a model, saying 'This is my thinking'. It must be combined with some kind of argumentation as well." – R2

One respondent makes also an example of problems that might arise when using executable models that suppliers get. These models were supposed to give a high-level overview and suggestion on how the code can be implemented, instead of telling them precisely how to do the job. The suppliers were not too happy to hack bad software code generated from the models; basically, the software code gets implemented twice.

Models have, however, proven to allow for early feedback.

"You can do a model beforehand, so you debug as soon as possible. /.../ We've tried to work quite model-based to get through problems with our specification writing." – R18

Thus, relying on model-based requirements to a larger scale is one of the top wishes for future ways of working with requirements.

Summarizing, eight out of second-round respondents that answered agree that model-based requirements engineering would bring opportunity toward development speed. The expectation is that models will scale better and will enable early feedback. However, models cannot completely substitute textual requirements and are not the only way to manage complexity.

6.2.1 Model-based RE in context of related work

Generally, Model-Based Engineering (MBE) promises reduction in defects as well as productivity improvements [5, 70], but suffers from insufficient tool support [5, 69, 70] and is difficult to use in combination with legacy software [46, 70]. While such challenges are certainly relevant for model-driven RE, benefits such as cost savings [54], productivity increases [1], or increases in reusability [61] would be very valuable for managing requirements in automotive system development as well. However, few model-driven approaches explicitly include RE [65]. Several proposed modeling frameworks prescribe or encourage the use of models for RE [13, 78], fewer have been evaluated with practitioners, e.g., [12, 14]. However, the industrial uptake seems to be limited, also because important practitioners' needs are not addressed [36]. Yet, in line with our findings, OEMs are considering adopting model-based RE [36].

6.3 Align requirements and automated testing

Respondents suggest that one way of aligning requirements among them is to establish a forum in which all authors of requirements for an area can meet.

> "You also need transparency, because when so many nodes, or subsystems, are to function together, you need a meeting place where all these specifications or models come together and can be checked against each other." – R1

Another way of aligning multiple and contradicting requirements is letting an end-user-oriented product owner prioritize.

> "The product owner needs to have an understanding of the business and the customers' needs, but also to have an arena where these requirements are prioritized. Then one need contact with our business side and the vehicle project leader who should receive all deliveries. And then one needs to have this difficult dialog about what is most important to do in which order, instead of having every single developer meet and discuss all contradicting requirements." – R11

Approaches such as continuous integration promise to increase development speed but rely heavily on automated testing. Acquiring the ability to quickly derive automated tests for new requirements will require a change in mindset:

> "If you want to build a CI-machine that keeps the product in very high quality over time you need to focus more on provoking errors [rather than testing against a requirements specification]. Finding corner cases." – R11

However, our respondents do not agree with the sentiment that tests could replace requirements. Instead, they emphasize how automation highlights the need for quality of test cases:

> "Test automation in itself is of no value, no, it's devising a good test case that's important. You have to start by conceiving a test case that catches problems and reveals many things. /.../ It's still about having the ingenuity to see through what can go wrong." – R18

A shift toward continuous deployment, i.e., the continuous delivery of software changes to customers, will require further changes in the mindset, introducing strict requirements not only on the product but also toward the deployment infrastructure and specifically for the quality of automated acceptance tests.

> "Continuous deployment is a difficult area for us because we have legal requirements there, we need to certify the cars. But to at all get to continuous deployment you need to have trust that the automatic test covers everything, and there we need to replace much of the manual tests." – R19

There is a need for establishing trust in the automated environment. A way of working with requirements will be most beneficial to speed in development if it is well aligned with these efforts toward automated testing.

Summarizing, all of the second-round respondents highlight the need for aligning requirements with automated testing. Suggested ways to achieve that are (1) establishing a forum to discuss, (2) enabling prioritization of requirements based on end-users, and (3) reliable and effective automatic derivation of test cases from requirements. This aspect stands out through its very strong agreement, both with respect to that this can enable development speed, and that effort should be spent improving the alignment of requirements and tests.

6.3.1 Align requirements and automated testing in context of related work

The relationship between agile methods in testing [18] and RE [8, 47], as well as their alignment [8, 90] has recently received increased attention in research. Both challenges [8] and practices [47] of aligning RE and software testing have been found to be applicable to large-scale system development and system testing [21]. The need for such alignment is also emphasized in large-scale agile frameworks, such as SAFe [60] and LESS [59], e.g., through the practice of specification by example [59]. Our findings suggest that this area of research needs further work to balance quality concerns with the wish to increase development speed.

6.4 Emergent teams to improve collaboration

The complexity of automotive systems has led to many ways of dividing work. This is with good intention since it is well known that it can be efficient to divide complex tasks into smaller parts and then combine them. However, in some cases this seems to do more harm than good:

> "It's so much more efficient than half the bunch sitting and thinking each on their side, writing a spec, sending it, someone implements and you send it out and people try it, and they reply 'it doesn't work, it doesn't work" – R17

> "According to [the old process] each silo is responsible for time, technology, and cost, which leads you to sub-optimize for what's [within your responsibility]. No-one is tasked with checking that the entirety

is optimal. Such sub-optimizations inevitably lead to these shortcuts we touched on, which in turn slows down overall speed." – R7

The problems of working in silos are amplified when working with suppliers, but there are ways forward here as well, and new ideas are tried for improving collaborations.

"Even if we haven't been sitting together [with the supplier] it's been a very tight collaboration. Although we have had requirements specifications at the bottom, in the end it's been plenty of common team activities to find the solutions." – R6

Overcoming silos will thus increase both the development speed and the ability to respond to change. It is also important to leverage the existing capabilities throughout the automotive value-chain.

"I think we sometimes underestimate that you can work with suppliers in a more efficient way. For the next generation of procurements, we're looking at requiring continuous deliveries from the suppliers during the development projects. That would help our CI a lot." – R11

In the second round of interviews, respondents brought up a new aspect that fits into this theme, related to new ways of working with requirements that focus on interaction instead of on artifacts and handovers. In such an approach, high-level requirements would be given to teams, including clear guidelines on what to do with them as well as forms that could be used to provide data. In particular, those step-by-step guidelines and a clear plan to follow-up show an ambition to create a dialog.

Summarizing, all of the second-round respondents that answered highlight the need for emergent teams to improve collaboration. Two respondents did not express an opinion, stating that this theme appears rather unclear. Emergent collaboration is deemed important, in order to bridge silos and to solve dependencies. But those respondents did not agree that teams need to be formed. Instead, emerging, cross-cutting collaboration should be facilitated through supportive roles. The complexity of autonomous systems is naturally requiring division of work in smaller tasks, however, this is often creating silos.

6.4.1 Improve collaboration in context of related work

Geographical distance, but also organizational, cognitive, and psychological distance in software development, have a significant impact on efficiency [9]. Agile methods have some potential to help overcome such distances [64], especially with respect to knowledge sharing and coordination [58], but scaling them beyond team level is challenging [64]. Especially at scale, social network analysis of requirements-centric collaboration is a promising facilitator for collaboration [67] and has been successfully applied within an OEM to coordinate requirements-related work [71]. Since transparency and improved collaboration beyond the scope of an individual organization becomes increasingly important [92], we encourage future research of similar facilitation in software value-chains.

6.5 Facilitate learning through exploration

Our respondents express a desire to work more exploration-driven.

"If you're starting with a new idea, that you hardly know what to call, and start by specifying requirements on it, you will never really get going. It's better to describe what it's supposed to do. There, we sometimes end up in catch-22.[6] 'I can't do this construction if I don't have the requirements ready.' 'OK, what do you want it to do then'?" – R5

"A large part of what we develop we don't quite know how it'll look when finished. More accurately, no-one can write down a complete set of requirements." – R12

"We landed in a notion we called blue bucket. We tried to sort the requirements. Some were green, they were met, no discussion. Some were red, they will never be met. But then we put some in the blue bucket as well. 'OK, we agree that we'll try to get as far as possible, but we don't know if we will reach all the way right now.' So instead, I think, we spent an entire year discussing these requirements. In hindsight, we should perhaps have spent that time developing and then reached the solution a bit earlier. So I think you need to lose this requirements hysteria, and we are doing that." – R6

From interviews in the second round emerges also the need to complement a static view on current requirements with support deltas through baselining. The ability to focus on the trajectory of development, history of changes, and generally a dynamic view on requirements might allow suppliers to provide tests or even target values, to co-evolve tracing from requirements to test and design, to visualize (growing) supplier commitment and compliance, and to allow control for cost.

Exploration-driven work does differ fundamentally from rigid requirements processes.

[6] https://en.wikipedia.org/wiki/Catch-22_(logic).

> "[The software] has to be ready two years before it goes to production, which is quite silly because we miss out on two years of development time. But here we haven't managed to agree with the rest of the organization that this is a silly requirement." – R13

Summarizing, all of the second-round respondents that answered agree or strongly agree that facilitation of exploration-driven work can increase development speed. Their arguments include the need for shifting from rigid requirements processes toward exploration-driven processes.

6.5.1 Exploration-driven work in context of related work

The need to facilitate learning through exploration is one big driver to look into the applicability of agile methods in automotive system engineering [89]. Typical large-scale agile frameworks, such as SAFe [60] and LESS [59], promise to support such exploration through practices such as enabler stories, specification by example, communities of practice, variable solution intents, and set-based design. Yet, adoption of agile frameworks in automotive system engineering is an ongoing effort [43], and our findings suggest the RE can play a critical role in this process if an appropriate role of requirements can be defined.

6.6 Complement lightweight pre-development RE with precise post-development specification

Especially with respect to safety and legal requirements, there is a certain level of documentation that must be provided. This is an important aspect of requirements engineering that our respondents regarded as orthogonal to other aspects, e.g., related to collaboration above. While requirements for supporting collaboration should be specified with development speed and ability to support change in mind, legally required documentation must be comprehensive as well as an accurate depiction of what is implemented. Our respondents aligned on a specific strategy to navigate this trade-off, as visible in the following quote:

> "In the end you have to document what you came up with, but you don't need to do it in advance, no rather afterward in some sense. So you still have documentation describing the construction. /.../ Of course that must be in place when we run into field problems and so on. We have to be able to troubleshoot our systems. /.../ So it can't be set free entirely, but, I think we have to start constructing more and specifying less in any case." – R6

When asking about additional aspects in the second round of interviews, one respondent related to the difficulties of separating safety and legal requirements from the rest:

> "One identifies the places where safety and legal concerns shall be taken care of but it spills over to the treatment of other types of requirements as well." – R26

There is some hope that this difficulty can be mitigated by finding a constructive approach, i.e., where teams start from high-level requirements and develop a specification of legal and safety-related requirements together with the system under construction. The confinement and separation of concerns, however, must then be provided by a suitable architecture. By pushing the creation of comprehensive requirements documentation into later phases, documenting the requirements that have been (most recently) implemented, OEMs may gain the flexibility to apply more lightweight approaches earlier on.

> "In the agile world we're actually saying that we want non-functional requirements because you always have to have that /.../ we who drive for agility want to remove functional requirements and replace them with our epics, capabilities, features, and stories /.../ we can agree to having requirements in at the top level, and those we trace in design and out to test cases, that we want to do. But we actually want to put very little emphasis on functional requirements, that's our starting point." – R8

The hope is to find a new way of working that combines the best of two worlds: (1) a lightweight and flexible way to manage requirements in order to support high-speed development, and (2) a thorough and accurate documentation of the finished implementation, as required to satisfy safety and legal concerns.

Summarizing, all of the second-round respondents that answered agree that combining lightweight pre-development RE and precise post-development specification can increase development speed. This enables using lightweight development processes earlier, without caring too much about the documentation that might be required, for instance, for safety-critical and legal requirements. Then, the needed documentation can be produced post-development, when it is clear how the system has been implemented. This will improve development speed since no comprehensive documentation needs to be maintained through times of frequent change.

6.6.1 Post-development specification in the context of related work

Agile methods as well as large-scale agile frameworks, such as SAFe [60] and LESS [59], tend to focus on customer value and user requirements, but neglect system requirements [52]. While they agree with reports on the importance of this system perspective, specifically to support long-term maintenance, evolution, and change impact analysis [52], our respondents indicate that it can be beneficial to create such specifications

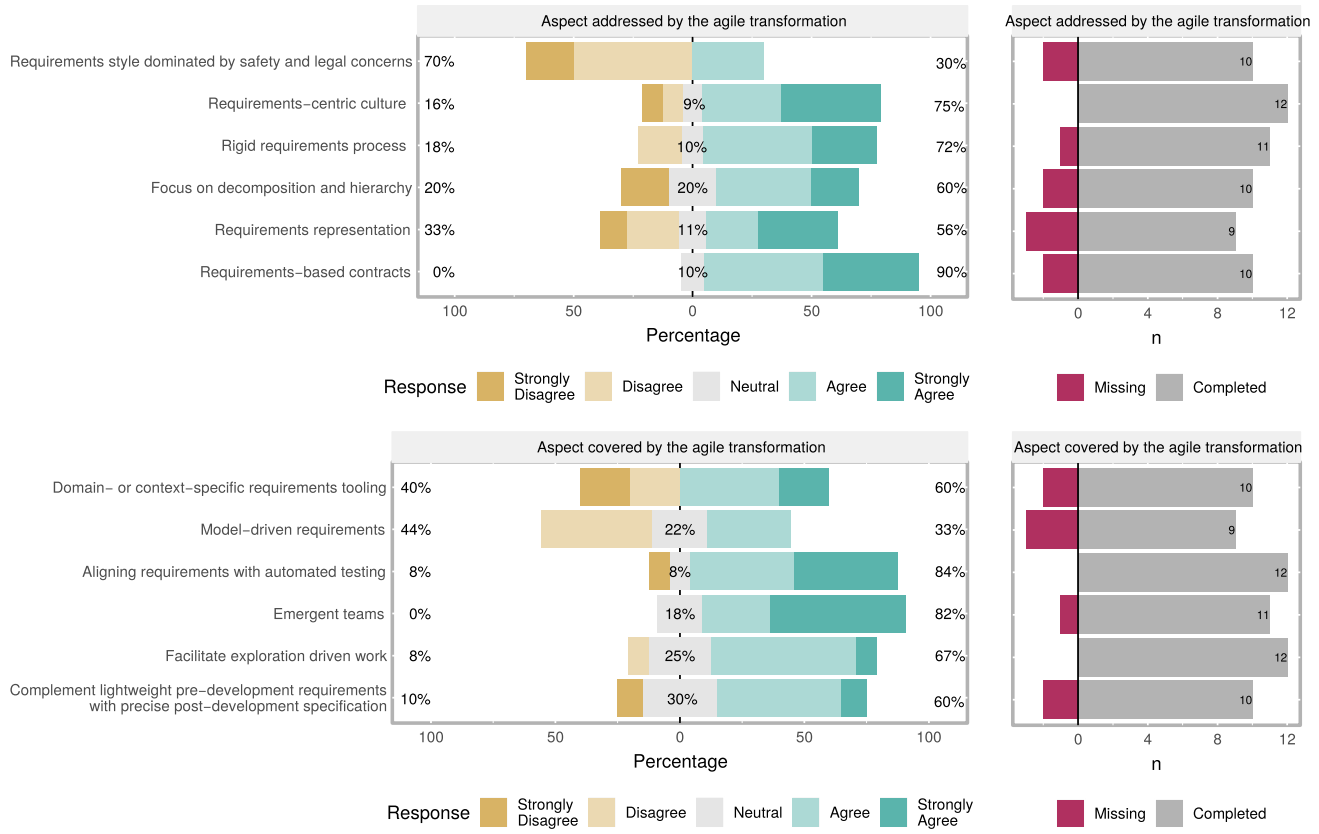

Fig. 4 Respondents' opinion about whether these aspects will be addressed through the agile transformation

after development. This suggestion relates to safe and regulated scrum variants [31, 37] that also ask teams to update requirements late, as part of a sprint.

7 To what extent will either aspects be addressed through the ongoing agile transformation (RQ3)?

We find different views on this research question based on the different perspectives of our respondents. While a requirements style dominated by safety and legal concerns can be problematic for development speed, respondents do not strongly consider this an aspect addressed by the agile transformations.

> "Not today, no. I have not seen that this problem has been dealt with." – R26

Furthermore, this aspect was thought to be connected to requirements-centric culture. For example, when discussing with a technical expert for software development, we struggled to get to a clear answer about the latter. The respondent was referring to "some strongholds" related to safety and legal aspects that will have to keep a rather requirements-centric culture, while at large, he did agree that agile transformation will have a positive influence on how this requirements-centric culture impacts development speed. In Fig. 4, we interpret this as a strong agree to our question, but we note that this will not hold for all parts of an organization at the scale of an automotive OEM.

Figure 4 shows the agreement of second-round respondents to RQ3, i.e., that the current agile transformation will address the aspects uncovered by the previous research questions and their impact on development speed.

Regarding whether the agile transformations address the aspects rigid requirements process, focus on decomposition and hierarchy, and requirements representation, our respondents' opinions are quite varied. In part, this is explained by differing views on the initiatives that are part of the transformations (e.g., requirements representation can be seen as independent of the agile transformation, but also closely related, e.g., with respect to the requirements information model suggested in SAFe). Beyond that, a contributing reason can also be the difficulty of getting an overview during any ongoing transformation effort.

The challenge of requirements-based contracts, however, clearly emerges as an impediment the case companies seek to address through their agile transformations.

> "During the whole [agile transformation] there's been a tough discussion with procurements that we need to get to a more agile situation. It's challenging work but I strongly agree that we aim to get there." – R7

In contrast, domain-specific tooling and model-driven requirements engineering receive the highest level of disagreement. The distribution between agreement and disagreement is fairly even, however. In the case of model-driven requirements, our respondents foresee specific use cases, while the general way of documenting requirements will continue to rest on natural language. Models can be very useful to increase the feedback speed in some use cases. With more precise notation, there is less room for interpretation, and thus disagreements can surface earlier. Aligning on specific interfaces between components, features, and teams can be very valuable if modeled. Models also enable reasoning on a higher level of abstraction, and by this managing complexity.

Where our respondents disagree, it is for one of the following reasons: models are not useful in all scopes, e.g., when discussing basic technology requirements. Also, models are not the only way of raising the level of abstraction, thus agile transformations can be driven forward without introducing model-driven requirements. In fact, parallel evolution of requirements in different teams will be even harder to merge when relying on models-based representations. Thus, model-driven requirements have only a weak link to the agile transformation in our data.

With respect domain-specific tooling, both companies have ongoing activities to update the tool-landscape for RE and related processes. This has been an enabler for the current agile transformation, thus respondents disagree that it will be solved through the transformation. Also, there is disagreement with respect to whether tooling support should differ between different release trains or even teams.

With respect to the forward-looking requirements engineering aspects found in the first round of interviews; aligning requirements with automated testing, and emergent teams stand out. These two are the ones that respondents mainly link to the agile transformation.

> "Aligning requirements with automated testing, yes I strongly agree that we are trying to do this. /.../ We put a lot of effort on it and it's one of the key drivers in the agile transformation." – R23

Also, without automated tests, continuous integration becomes impossible, and that can be considered a prerequisite of working agile at scale, according to one respondent. The strong disagreement in Fig. 4 for this aspect relates mainly to the fact that this is not positively impacted by the transformation by itself. The strongly disagreeing respondent recognizes a strong focus in the organization on aligning requirements with automated tests, however. In summary, respondents agree that this alignment is important and largely agree that the agile transformation will have a positive impact on this aspect.

Regarding emergent teams, one respondent exemplified how such a way of working is introduced with the transformation.

> "To help with solutions cutting across release trains. /.../ Architecturally, the idea is to form small teams to solve certain issues there and then. When it's solved [the team] dissolves." – R7

In addition, facilitating exploration-driven work, and complementing lightweight pre-development requirements with precise post-development specifications, are perceived as initiatives within the scope of the transformations. We note, however, that agreement is not as strong for these aspects as for the preceding two.

8 Discussion, conclusions, and outlook

In this paper, we investigate the impact of requirements engineering on the goal of automotive companies to increase development speed. We deliberately obtain the perspective of managers and technical leaders to understand their vision about the current and future role of requirements engineering in automotive system engineering. By this, we complement previous works that focus more on operational aspects from a development point of view [4, 25, 47, 52].

Our findings clearly indicate that because of safety and legal concerns, requirements are not optional for automotive systems. However, it is also evident that traditional ways of working are no longer sufficient.

More specifically, for what concerns RQ1 (*Which aspects of the current way of working with requirements impact development speed?*), we discovered that culture and the historical way of working play a crucial role. The requirements engineering style is excessively dominated by safety and legal constraints and development speed is neglected. Rigid engineering processes, decomposition of requirements, and too many levels of abstraction force early design decisions and add unnecessary delays. Also, the current ways of specifying requirements hinder change and the use of requirements as part of legal contracts in the collaboration with suppliers in the value-chain hinders fast collaboration.

For what concerns RQ2 (*Which new aspects should be considered when defining a new way of working with requirements to increase development speed?*),

domain- and context-specific requirements tooling could positively change the way of working in the direction of increasing the development speed. Tools can, however, also become an obstacle if not properly designed and maintained. Moreover, model-based requirements to a larger scale is also one of the top wishes for future ways of working with requirements. Another promising improvement comes from increasing the degree of test automation in the requirements verification. However, relying only on automated tests is insufficient; our respondents emphasize the need for expertly crafted test cases that thoroughly stress the system. One of the larger improvement proposals would be to remove the many organizational silos that exist. These silos are often created for good reasons; to achieve team autonomy and clear divisions of responsibilities, but the separation often leads to a slow workflow, which is excessively based on handovers. If instead a lightweight pre-development requirements engineering approach is combined with precise specifications created post-development, development speed can be increased and collaboration improved throughout the automotive value-chain.

For what concerns RQ3 (*To what extent will either aspects be addressed through the ongoing agile transformation?*), we found different points of view among our respondents. In general, we can say that our respondents are unsure about whether the agile transformation will lead to better balancing of the requirement style. One of the main motivations is that these aspects are already getting attention independently of the agile transformation, which can thus not claim all the credit. The aspects that see the least agreement are domain-specific tooling and model-based requirements engineering. For what concerns domain-specific tooling, we had a wide spectrum of opinions. For what concerns model-based requirements engineering, except in some cases, natural language is the means used for documenting requirements. Models have been identified as mainly useful for providing early feedback (i.e., executable models), and as a way to reduce ambiguity and deal with complexity.

Relevance for practitioners: Our work can be considered a roadmap that can be used by companies in their transformation toward increasing development speed. From our interpretations of the findings, we derive the following advice to practitioners:

- Focus requirement efforts where crucial, e.g., on safety-critical functionality.
- Anticipate and accept that requirements will need to be updated throughout product development. A complete upfront specification is often impossible to have. Better postpone and delegate some decisions to developers.
- Combine lightweight pre-development RE with precise post-development specification.
- Aim for exploration and collaboration based on mutual trust, rather than requirements as contracts to be satisfied, especially in OEM-supplier relationships.
- Consider using model-based RE and especially executable models for having early feedback.

Figure 1 can be considered as a guide in trade-off analysis of where to expend effort when applying improvements.

Relevance for academics: During the study, we found a number of areas that would benefit from further research:

- Developing flexible tools, that are easy to use and maintain, and that can be integrated into the development process, is still a major challenge.
- Traceability is important and current solutions are not satisfactory. There is room for new solutions that are really addressing industry needs.
- Increasing the use of models could be a way forward. However, models cannot completely replace textual descriptions. Balancing models against text as well as how to properly integrate them remain an open challenge.
- Test automation is essential for CI. Our results indicate that in addition to technical aspects, such as test coverage, test execution efficiency, and test case selection, research attention is also needed on how to achieve trust in automated testing.

We particularly encourage multidisciplinary work, since most aspects of the desired future way of working with requirements have been reported to work well in isolation. Yet, their adoption in industry is low, since their interplay is not sufficiently clear (as for example safety concerns and continuous integration or deployment at system level).

Future work: A natural continuation of this work, and of previous research, is to unify the manager and developer perspectives on requirements engineering into a holistic view, thus creating a unified theory of requirements engineering in scaled agile. For this, a replication of this research in other domains within and beyond embedded systems development will be important. Throughout our interviews, it was also clear that when discussing strategic aspects of the current setup of automotive companies, requirements engineering is not the only aspect affecting development speed. We found that especially the relation to architecture [99], the approach to collaboratively constructing and managing system engineering artifacts [97], and the ability to manage safety aspects in continuous software engineering [50] are important topics for further investigation.

Acknowledgements Open access funding provided by University of Gothenburg. We thank all respondents in the study for their valuable input during both rounds of interviews, and for clarifications where needed. We also thank Andreas Karlsson and Caroline Svensson at Knowit AB, for their help with setting up and performing the interviews. This study was performed in collaboration with the Vinnova project Next Generation Electrical Architecture (NGEA), and partially supported by Software Center Proj. 27 *RE for Large-Scale Agile System Development*.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

1. Agner LTW, Soares IW, Stadzisz PC, Simão JM (2013) A brazilian survey on UML and model-driven practices for embedded software development. J Syst Softw 86(4):997–1005
2. Ågren SM, Knauss E, Heldal R, Pelliccione P, Malmqvist G, Bodén J (2018) The manager perspective on requirements impact on automotive systems development speed. In: 2018 IEEE 26th international requirements engineering conference (RE), pp 17–28. https://doi.org/10.1109/RE.2018.00-55
3. Ali N, Sharafl Z, Gueheneuc Y, Antoniol G (2012) An empirical study on requirements traceability using eye-tracking. In: Proceedings of the 28th IEEE international conference on software maintenance (ICSM'12). IEEE, pp 191–200. https://doi.org/10.1109/ICSM.2012.6405271
4. Alsaqaf W, Daneva M, Wieringa R (2017) Quality requirements in large scale distributed agile projects—a systematic literature review. In: Proceedings of 23rd international working conference on requirements engineering. Foundation for Software Quality (REFSQ), Essen, pp 219–234
5. Baker P, Loh S, Weil F (2005) Model-driven engineering in a large industrial context—motorola case study. In: Briand LC, Williams C (eds) Model driven engineering languages and systems. Lecture notes in computer science, vol 3713, pp 476–491
6. Beck K (2000) Extreme programming explained: embrace change. Addison-Wesley Professional, Boston
7. Berger C, Eklund U (2015) Expectations and challenges from scaling agile in mechatronics-driven companies—a comparative case study. In: Proceedings of 16th internaional conference on agile processes in software engineering and extreme programming (XP '15), pp 15–26
8. Bjarnason E, Runeson P, Borg M, Unterkalmsteiner M, Engström E, Regnell B, Sabaliauskaite G, Loconsole A, Gorschek T, Feldt R (2014) Challenges and practices in aligning requirements with verification and validation: a case study of six companies. Empir Softw Eng 19(6):1809–1855. https://doi.org/10.1007/s10664-013-9263-y
9. Bjarnason E, Smolander K, Engström E, Runeson P (2016) A theory of distances in software engineering. Inf Softw Technol 70:204–219
10. Bjarnason E, Wnuk K, Regnell B (2011) A case study on benefits and side-effects of agile practices in large-scale requirements engineering. In: Proceedings of 1st WS on agile requirements engineering
11. Boehm B (1981) Software engineering economics. Prentice-Hall, Upper Saddle River
12. Böhm W, Junker M, Vogelsang A, Teufl S, Pinger R, Rahn K (2014) A formal systems engineering approach in practice: An experience report. In: Proceedings of the 1st international workshop on software engineering research and industrial practices. ACM, pp 34–41
13. Braun P, Broy M, Houdek F, Kirchmayr M, Müller M, Penzenstadler B, Pohl K, Weyer T (2014) Guiding requirements engineering for software-intensive embedded systems in the automotive industry. Comput Sci Res Dev 29(1):21–43. https://doi.org/10.1007/s00450-010-0136-y
14. Broy M, Damm W, Henkler S, Pohl K, Vogelsang A, Weyer T (2012) Introduction to the SPES modeling framework. In: Pohl K et al (eds) Model-based engineering of embedded systems. Springer Berlin, pp 31–49
15. Carrillo de Gea JM, Nicolás J, Fernández Alemán JL, Toval A, Ebert C, Vizcaíno A (2012) Requirements engineering tools: capabilities, survey and assessment. Inf Softw Technol 54(10):1142–1157. https://doi.org/10.1016/j.infsof.2012.04.005
16. Chow T, Cao DB (2008) A survey study of critical success factors in agile software projects. J Syst Softw 81(6):961–971
17. Cleland-Huang J (2012) Traceability in agile projects. In: Software and systems traceability. Springer London, pp 265–275. https://doi.org/10.1007/978-1-4471-2239-5
18. Crispin L, Gregory J (2009) Agile testing: a practical guide for testers and agile teams, 1st edn. Addison-Wesley Professional, Boston
19. de Gea JMC, Nicolas J, Aleman JLF, Toval A, Ebert C, Vizcaino A (2011) Requirements engineering tools. IEEE Softw 28:86–91
20. de Lucia A, Oliveto R, Tortora G (2008) IR-based traceability recovery processes: an empirical comparison of one-shot and incremental processes. In: Proceedings of the 23rd IEEE/ACM international conference on automated software engineering. IEEE Computer Society, pp 39–48. https://doi.org/10.1109/ICPC.2011.34
21. de Oliveira Neto FG, Horkoff J, Knauss E, Kasauli R, Liebel G (2017) Challenges of aligning requirements engineering and system testing in large-scale agile: a multiple case study. In: Proceedings of 4th international workshop on requirements engineering and testing (RET@RE), Lisbon, Portugal
22. Demuth A, Kretschmer R, Egyed A, Maes D (2016) Introducing traceability and consistency checking for change impact analysis across engineering tools in an automation solution company: an experience report. In: IEEE international conference on software maintenance and evolution (ICSME'16), pp 529–538. https://doi.org/10.1109/ICSME.2016.50
23. Dikert K, Paasivaara M, Lassenius C (2016) Challenges and success factors for large-scale agile transformations: a systematic literature review. J Syst Softw 119:87–108
24. Eklund U, Holmström Olsson H, Strøm NJ (2014) Industrial challenges of scaling agile in mass-produced embedded systems. In: Proceedings of international workshop on agile methods. Large-scale development, refactoring, testing, and estimation, pp 30–42
25. Eliasson U, Heldal R, Knauss E, Pelliccione P (2015) The need of complementing plan-driven requirements engineering with emerging communication: experiences from volvo car group. In: IEEE 23rd international conference requirements engineering. IEEE, pp 372–381
26. Espinoza A, Garbajosa J (2011) A study to support agile methods more effectively through traceability. Innov Syst Softw Eng 7(1):53–69. https://doi.org/10.1007/s11334-011-0144-5
27. Fagerholm F, Guinea AS, Mäenpää H, Münch J (2017) The right model for continuous experimentation. J Syst Softw 123:292–305

28. Feiler P, Gabriel RP, Goodenough J, Linger R, Longstaff T, Kazman R, Klein M, Northrop L, Schmidt D, Sullivan K, Wallnau K (2006) Ultra-large-scale systems: the software challenge of the future. Software Engineering Institute, Pittsburgh
29. Figueiredo MC, De Souza CR (2012) Wolf: Supporting impact analysis activities in distributed software development. In: Proceedings of the 5th international workshop on cooperative and human aspects of software engineering (CHASE), pp 40–46. https://doi.org/10.1109/CHASE.2012.6223019
30. Fitzgerald B, Stol KJ (2017) Continuous software engineering: a roadmap and agenda. J Syst Softw 123:176–189. https://doi.org/10.1016/j.jss.2015.06.063
31. Fitzgerald B, Stol KJ, O'Sullivan R, O'Brien D (2013) Scaling agile methods to regulated environments: an industry case study. In: Proceedings of 35th international conference on software engineering, pp 863–872
32. Fowler M (2006) Continuous integration. http://martinfowler.com/articles/continuousIntegration.html, http://martinfowler.com/articles/continuousIntegration.html last visit: 2016-01-12
33. Gayer S, Herrmann A, Keuler T, Riebisch M, Antonino PO (2016) Lightweight traceability for the agile architect. Computer 49(5):64–71. https://doi.org/10.1109/MC.2016.150
34. Gibbs GR (2008) Analysing qualitative data. Sage, Thousand Oaks
35. Graaf B, Lormans M, Toetenel H (2003) Embedded software engineering: the state of the practice. IEEE Softw 20(6):61–69. https://doi.org/10.1109/MS.2003.1241368
36. Haasis S (2016) Systems engineering for future mobility. In: RE conference. https://www.hood-group.com/fileadmin/projects/hood-group/upload/Images/REConf/2016/vortraege/mittwoch/auditorium/Keynote-Systems_Engineering_for_future_mobility.pdf
37. Hanssen GK, Haugset B, Stålhane T, Myklebust T, Kulbrandstad I (2016) Quality assurance in scrum applied to safety critical software. In: International conference on agile software development. Springer, pp 92–103
38. Heikkila VT, Damian D, Lassenius C, Paasivaara M (2015) A mapping study on requirements engineering in agile software development. In: 41st Euromicro conference on software engineering and advanced applications (SEAA '15), pp 199–207
39. Heikkilä VT, Paasivaara M, Lasssenius C, Damian D, Engblom C (2017) Managing the requirements flow from strategy to release in large-scale agile development: a case study at Ericsson. Empir Softw Eng 22(6):2892–2936
40. Helming J, Koegel M, et al (2009) Traceability-based change awareness. In: Proceedings of the 12th international conference on model driven engineering languages and systems (MODELS'09), pp 372–376. https://doi.org/10.1007/978-3-642-04425-0_28
41. Heumesser N, Houdek F (2003) Towards systematic recycling of systems requirements. In: Proceedings of 25th international conference on software engineering (ICSE), Portland, pp 512–519
42. Hoda R, Noble J, Marshall S (2009) Negotiating contracts for agile projects: a practical perspective. In: Proceedings of international conference on agile processes and extreme programming in software engineering (XP), pp 186–191
43. Hohl P, Münch J, Schneider K, Stupperich M (2017) Real-life challenges on agile software product lines in automotive. In: Proceedings of international conference on product-focused software process improvement (PROFES), pp 28–36
44. Houdek F (2017) Automotive future and its impact on empirical requirements engineering. In: Keynote at 6th international workshop on empirical requirements engineering at RE 2017
45. Humble J, Farley D (2010) Continuous delivery: reliable software releases through build, test, and deployment automation. Pearson Education, London
46. Hutchinson J, Whittle J, Rouncefield M, Kristoffersen S (2011) Empirical assessment of MDE in industry. In: 33rd International conference on software engineering (ICSE '11), pp 471–480
47. Inayat I, Salim SS, Marczak S, Daneva M, Shamshirband S (2015) A systematic literature review on agile requirements engineering practices and challenges. Comput Hum Behav 51:915–929
48. Jastram M (2014) How the REQIF standard for requirements exchange disrupts the tool market. Requir Eng Mag. https://re-magazine.ireb.org/articles/open-up. Accessed 13 July 2019
49. Kahkonen T (2004) Agile methods for large organizations-building communities of practice. Agile Dev Conf 2004:2–10
50. Kasauli R, Knauss E, Kanagwa B, Nilsson A, Calikli G (2018) Safety-critical systems and agile development: a mapping study. In: Proceedings of Euromicro SEAA
51. Kasauli R, Knauss E, Nilsson A, Klug S (2017) Adding value every sprint: a case study on large-scale continuous requirements engineering. In: Proceedings of 3rd workshop on control requirements engineering, Essen, Germany
52. Kasauli R, Liebel G, Knauss E, Gopakumar S, Kanagwa B (2017) Requirements engineering challenges in large-scale agile system development. In: IEEE 25th international on requirements engineering conference (RE). IEEE, pp 352–361
53. Katumba B, Knauss E (2014) Agile development in automotive software development: challenges and opportunities. In: Jedlitschka A, Kuvaja P, Kuhrmann M, Männistö T, Münch J, Raatikainen M (eds) Proceedings of 15th international conference on product-focused software process improvement (Profes '14), Springer, Helsinki, LNCS, vol 8892, pp 33–47. https://doi.org/10.1007/978-3-319-13835-0_3, http://link.springer.com/chapter/10.1007/978-3-319-13835-0_3
54. Kirstan S, Zimmermann J (2010) Evaluating costs and benefits of model-based development of embedded software systems in the car industry–results of a qualitative case study. In: Workshop C2M: EEMDD "From code centric to model centric: evaluating the effectiveness of MDD"
55. Knauss E, Pelliccione P, Heldal R, Ågren M, Hellman S, Maniette D (2016) Continuous integration beyond the team: a tooling perspective on challenges in the automotive industry. In: Proceedings of ESEM '16. ACM, pp 43:1–43:6
56. Knauss E, Yussuf A, Blincoe K, Damian D, Knauss A (2016) Continuous clarification and emergent requirements flows in open-commercial software ecosystems. Requir Eng J 23(1):97–117
57. Laanti M, Salo O, Abrahamsson P (2011) Agile methods rapidly replacing traditional methods at nokia: a survey of opinions on agile transformation. Inf Softw Technol 53(3):276–290
58. Lagerberg L, Skude T, Emanuelsson P, Sandahl K, Ståhl D (2013) 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 international symposium on empirical software engineering and measurement, pp 348–356
59. Larman C, Vodde B (2017) Large-scale scrum: more with less. Addison-Wesley, Boston
60. Leffingwell D (2016) SAFe® 4.0 reference guide: scaled agile framework® for lean software and systems engineering. Addison-Wesley Professional, Boston
61. Liebel G, Marko N, Tichy M, Leitner A, Hansson J (2016) Model-based engineering in the embedded systems domain: an industrial survey on the state-of-practice. Softw Syst Model. https://doi.org/10.1007/s10270-016-0523-3
62. Liebel G, Tichy M, Knauss E, Ljungkrantz O, Stieglbauer G (2018) Organisation and communication problems in automotive requirements engineering. Requir Eng J 23(1):145–167. https://doi.org/10.1007/s00766-016-0261-7 online first: 2016

63. Liebel G, Tichy M, Anjorin A, Lorber F, Knauss E (2017) Modelling behavioural requirements and alignment with verification in the embedded industry. In: Proceedings of 5th international conference on model-driven engineering and software development (MODELSWARD '17), Porto, Portugal, pp 427–434. http://www.scitepress.org/DigitalLibrary/PublicationsDetail.aspx?ID=0rSONMuAFkA=&t=1
64. Lindvall M, Muthig D, Dagnino A, Wallin C, Stupperich M, Kiefer D, May J, Kahkonen T (2004) Agile software development in large organizations. Computer 37(12):26–34
65. Loniewski G, Insfran E, Abrahão S (2010) A systematic review of the use of requirements engineering techniques in model-driven development. In: Petriu D, Rouquette N, Haugen O (eds) Model driven engineering languages and systems. Lecture notes in computer science, vol 6395, pp 213–227. https://doi.org/10.1007/978-3-642-16129-2_16
66. Mäder P, Egyed A (2015) Do developers benefit from requirements traceability when evolving and maintaining a software system? Empir Softw Eng 20(2):413–441. https://doi.org/10.1007/s10664-014-9314-z
67. Marczak S, Damian D, Stege U, Schröter A (2008) Information brokers in requirement-dependency social networks. In: Proceedings of 16th international requirements engineering conference (RE), pp 53–62
68. Meyer B (2014) Agile! The good, the hype and the ugly. Springer, New York
69. Mohagheghi P, Gilani W, Stefanescu A, Fernandez MA, Nordmoen B, Fritzsche M (2013) Where does model-driven engineering help? Experiences from three industrial cases. Softw Syst Model 12(3):619–639
70. Mohagheghi P, Dehlen V (2008) Where is the proof? - a review of experiences from applying mde in industry. In: Schieferdecker I, Hartman A (eds) Model driven architecture—foundations and applications. Lecture notes in computer science, vol 5095, pp 432–443
71. Mohamad M, Liebel G, Knauss E (2017) Loco coco: automatically constructing coordination and communication networks from model-based systems engineering data. Inf Softw Technol 000:1–15. https://doi.org/10.1016/j.infsof.2017.08.002
72. Neely S, Stolt S (2013) Continuous delivery? Easy! Just change everything (well, maybe it is not that easy). In: Proceedings of agile conference (AGILE), pp 121–128
73. Neill CJ, Laplante PA (2003) Requirements engineering: the state of the practice. IEEE Softw 20(6):40–45. https://doi.org/10.1109/MS.2003.1241365
74. Olsson HH, Alahyari H, Bosch J (2012) Climbing the" stairway to heave"—a mulitiple-case study exploring barriers in the transition from agile development towards continuous deployment of software. In: 2012 38th Euromicro conference on software engineering and advanced applications, IEEE, pp 392–399
75. Paasivaara M, Lassenius C (2016) Challenges and success factors for large-scale agile transformations: a research proposal and a pilot study. In: Proceedings of the scientific workshop proceedings of XP2016. ACM, p 9
76. Pelliccione P, Knauss E, Heldal R, Ågren SM, Mallozzi P, Alminger A, Borgentun D (2017) Automotive architecture framework: the experience of volvo cars. J Syst Archit 77:83–100. https://doi.org/10.1016/j.sysarc.2017.02.005
77. Pernstål J, Magazinius A, Gorschek T (2012) A study investigating challenges in the interface between product development and manufacturing in the development of software-intensive automotive systems. Int J Softw Eng Knowl Eng 22(07):965–1004
78. Pohl K, Hönninger H, Achatz R, Broy M (2012) Model-based engineering of embedded systems: the SPES 2020 methodology. Springer, New York
79. Ramesh B, Cao L, Baskerville R (2010) Agile requirements engineering practices and challenges: an empirical study. Inf Syst J 20(5):449–480
80. Rempel P, Mäder P (2015) A quality model for the systematic assessment of requirements traceability. In: Proceedings of the 23rd ieee international requirements engineering conference (RE'15), pp 176–185. https://doi.org/10.1109/RE.2015.7320420
81. Runeson P, Höst M (2009) Guidelines for conducting and reporting case study research in software engineering. Empir Softw Eng 14(2):131
82. Salo O, Abrahamsson P (2008) Agile methods in european embedded software development organisations: a survey on the actual use and usefulness of extreme programming and scrum. IET Softw 2(1):58–64
83. Saunders B, Sim J, Kingstone T, Baker S, Waterfield J, Bartlam B, Burroughs H, Jinks C (2018) Saturation in qualitative research: exploring its conceptualization and operationalization. Qual Quant 52(4):1893–1907. https://doi.org/10.1007/s11135-017-0574-8
84. Savolainen J, Kuusela J, Vilavaara A (2010) Transition to agile development-rediscovery of important requirements engineering practices. In: 18th International requirements on engineering conference. IEEE, pp 289–294
85. Sengupta S, Kanjilal A, Bhattacharya S (2008) Requirement traceability in software development process: an empirical approach. In: Proceedings of the 19th IEEE/IFIP international symposium on rapid system prototyping (RSP'08). IEEE, pp 105–111. https://doi.org/10.1109/RSP.2008.14
86. Sikora E, Tenbergen B, Pohl K (2012) Industry needs and research directions in requirements engineering for embedded systems. Requir Eng 17(1):57–78. https://doi.org/10.1007/s00766-011-0144-x
87. Sikora E, Tenbergen B, Pohl K (2011) Requirements engineering for embedded systems: an investigation of industry needs. In: Berry D, Franch X (eds) Requirements engineering: foundation for software quality. Lecture notes in computer science, vol 6606, pp 151–165
88. Sikora E, Tenbergen B, Pohl K (2011) Requirements engineering for embedded systems: an investigation of industry needs. In: Proceedings of the 17th international working conference on requirements engineering: foundation for software quality, REFSQ'11. Springer, Berlin, pp 151–165. http://dl.acm.org/citation.cfm?id=1987360.1987383
89. Stupperich M, Schneider S (2011) Process-focused lessons learned from a multi-site development project at daimler trucks. In: Proceedings of 6th international conference on global software engineering (ICGSE), Helsinki, Finland, pp 141–145
90. Unterkalmsteiner M, Feldt R, Gorschek T (2014) A taxonomy for requirements engineering and software test alignment. ACM Trans Softw Eng Methodol 23(2):16:1–16:38. https://doi.org/10.1145/2523088
91. Unterkalmsteiner M, Gorschek T, Feldt R, Klotins E (2015) Assessing requirements engineering and software test alignment, Äîfive case studies. J Syst Softw 109:62–77. https://doi.org/10.1016/j.jss.2015.07.018
92. van der Valk R, Pelliccione P, Lago P, Heldal R, Knauss E, Juul J (2018) Transparency and contracts: continuous integration and delivery in the automotive ecosystem. In: 40th International conference on software engineering: software engineering in practice track (ICSE-SEIP 2018). IEEE/ACM, Gothenburg

93. Weber M, Weisbrod J (2003) Requirements engineering in automotive development: experiences and challenges. IEEE Softw 20(1):16–24. https://doi.org/10.1109/MS.2003.1159025
94. Weber M, Weisbrod J (2002) Requirements engineering in automotive development-experiences and challenges. In: Proceedings of IEEE joint international conference on requirements engineering (RE '02), pp 331–340
95. Wiklund K, Sundmark D, Eldh S, Lundqvist K (2013) Impediments in agile software development: an empirical investigation. In: Proceedings of product-focused software process improvement, pp 35–49
96. Wohlrab R, Knauss E, Steghöfer JP, Maro S, Anjorin A, Pelliccione P (2018) Collaborative traceability management: a multiple case study from the perspectives of organization, process, and culture. Requir Eng. https://doi.org/10.1007/s00766-018-0306-1
97. Wohlrab R, Pelliccione P, Knauss E, Larsson M (2019) Boundary objects and their use in agile systems engineering organizations. J Softw Evol Process. https://doi.org/10.1002/smr.2166
98. Wohlrab R, Pellicione P, Knauss E, Gregory S (2018) The problem of consolidating re practices at scale: an ethnographic study. In: Proceedings of 24th international working conference on requirements engineering: foundation for requirements engineering (REFSQ), Utrecht, The Netherlands
99. Wohlrab R, Pellicione P, Knauss E, Heldal R (2019) On interfaces to support agile architecting in automotive: an exploratory case study. In: Proceedings of IEEE international conference on software architecture (ICSA), Hamburg, Germany
100. Zijdemans SH, Stettina CJ (2014) Contracting in agile software projects: state of art and how to understand it. In: Proceedings of international conference on agile processes and extreme programming in software engineering (XP), pp 78–93

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.