The Journal of Systems & Software 199 (2023) 111628

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Aspects of modelling requirements in very-large agile systems engineering[☆]

Grischa Liebel [a,*], Eric Knauss [b]

[a] School of Technology, Reykjavik University, Menntavegur 1, Reykjavik, 102, Capital Region, Iceland
[b] Department of Computer Science and Engineering, Chalmers | University of Gothenburg, Gothenburg, Västra Götaland, Sweden

ARTICLE INFO

Article history:
Received 3 October 2022
Received in revised form 19 January 2023
Accepted 28 January 2023
Available online 2 February 2023

Keywords:
Agile
Modelling
Requirements

ABSTRACT

Using models for requirements engineering (RE) is uncommon in systems engineering, despite the widespread use of model-based engineering in general. One reason for this lack of use is that formal models do not match well the trend to move towards agile developing methods. While there exists work that investigates challenges in the adoption of requirements modelling and agile methods in systems engineering, there is a lack of work studying successful approaches of using requirements modelling in agile systems engineering. To address this gap, we conducted a case study investigating the application of requirements models at Ericsson AB, a Swedish telecommunications company. We studied a department using requirements models to bridge agile development and plan-driven development aspects. We find that models are used to understand how requirements relate to each other, and to keep track with the product's evolution. To cope with the effort to maintain models over time, study participants suggest to rely on text-based notations that bring the models closer to developers and allow integration into existing software development workflows. This results in tool trade-offs, e.g., losing the possibility to control diagram layout.

1. Introduction

Driven by success stories in small-scale software development, agile development is increasingly adopted in large-scale software and systems engineering (Dikert et al., 2016a; Lagerberg et al., 2013; Salo and Abrahamsson, 2008a; Eklund et al., 2014; Berger and Eklund, 2015). However, context factors such as long lead times (Berger and Eklund, 2015), safety criticality (Kasauli et al., 2018), and the scale of development itself make this adoption challenging. In particular, challenges relate to Requirements Engineering (RE), such as building and maintaining a shared understanding of customer value and the system requirements (Kasauli et al., 2017, 2021).

To build and maintain system knowledge over time, models have been used as a suitable means of documentation (Kasauli et al., 2021). Specifically, models are often cited as a way to deal with complexity that arises from the scale of systems (Selic, 1998). However, while the use of models is common in systems engineering (Liebel et al., 2018a), using requirements models is uncommon in practice (Liebel et al., 2018b; Loniewski et al., 2010). In the context of large-scale agile software and systems engineering, we are not aware of any work investigating the use of requirements models in industry.

Therefore, the goal of this paper is to better understand the potential of using requirements models in very large-scale (VLS) agile (Dingsøyr et al., 2014) systems engineering. To do so, we conducted a case study of a single department at Ericsson AB, a large Swedish telecommunications provider, which has long-ranging experience using requirements models in a VLS agile setting. We aim to answer the following research questions (RQs).

RQ1: What sentiments exist for and against the use of requirements models in VLS agile systems engineering?
RQ2: How do different stakeholders use requirements models in VLS agile systems engineering?
RQ3: What are the needs to support the intended use of requirements models in VLS agile systems engineering?

To answer these questions, we collected survey data, followed up with a number of semi-structured interviews to find answers to patterns observed in the survey.

We find that the requirements models at the case department serve as a *boundary object* that relates the agile world in individual teams with the overall waterfall-like process that deals with product requirements and their long-term evolution. While engineers are positive regarding the use of models, many take a practical stance concerning the feasibility of continuously maintaining

☆ Editor: P. Lago.
* Corresponding author.
E-mail address: grischal@ru.is (G. Liebel).

https://doi.org/10.1016/j.jss.2023.111628
0164-1212/© 2023 Elsevier Inc. All rights reserved.

these models over time. To achieve an updated and maintained model, text-based modelling approaches such as PlantUML[1] with certain inherent limitations such as automatic layouting are seen as inevitable. Furthermore, to avoid deterioration of models over time, our study participants suggest generating simple artefacts from the models, e.g., documentation. This would encourage engineers to regularly update the models, as derived artefacts would otherwise become outdated.

2. Related work

There exists a broad body of work on the use of models in industry, and suggestions on how to use models for RE-related activities. In the following, we will discuss this work in detail.

2.1. Use of models in industry

There are numerous case studies reporting how models are used in industry, e.g., Baker et al. (2005), Mohagheghi et al. (2013), Hutchinson et al. (2011a,b), Whittle et al. (2013) and Liebel et al. (2018b).

In a case study at Motorola, Baker et al. (2005) discuss how Model-Based Engineering (MBE) is used at Motorola over a period of 20 years. The authors report several positive effects, such as defect reductions and increases in productivity, but also a lack of tools and tool interoperability, poor performance of generated code, and a lack of scalability of the modelling approach.

Experiences from three European companies with MBE techniques and tools are presented by Mohagheghi et al. (2013) in terms of a qualitative study. The authors find that simulation and testing opportunities are positive aspects of using MBE, while tool problems and the complexity of models are listed as drawbacks.

Hutchinson et al. study the use and adoption of MBE in industry in a series of qualitative and mixed-methods studies (Hutchinson et al., 2011a,b; Whittle et al., 2013; Hutchinson et al., 2014). The overall finding of this study series is that the organisation context and several non-technical topics need to be considered for MBE to succeed. For instance, the authors report that significant additional training is needed for the use of MBE. From their interviews, the authors conclude that especially people's ability to think abstractly seems to have significant impact on their ability to create models. In addition, several technical challenges such as tool shortcomings impede the use and success of MBE.

In a case study at two automotive companies (Liebel et al., 2018b), we find that models are used in automotive RE to improve communication and to handle complexity. However, stakeholders prefer informal models and whiteboard sketches over formal modelling notations.

2.2. Frameworks for using models during RE

Several frameworks and methods have been suggested that include the use of models for or during RE.

Pohl et al. (2012) introduce the SPES 2020 Methodology for the development of embedded systems. During RE in particular, the framework suggests a separation between *solution-independent* and *solution-oriented* diagrams. Practical experiences with SPES are reported in Böhm et al. (2014) and Brings et al. (2017). In Böhm et al. (2014), Böhm et al. present their experiences with SPES in an industrial project at Siemens. The authors apply SPES to a mature, already running train control system, using a specification of "high quality". Findings are that "the high quality of input documents, and cooperation with product experts were considered the most influential success factors". Brings et al. (2017) discuss experiences of using SPES in the area of cyber–physical systems. The authors report that they "identified problems resulting from an increased number of dependencies". and "the need to cope with redundancies caused by properties which are system as well as context properties in a structured manner".

Apart from the SPES framework, there exist several proposed processes and frameworks for requirements modelling, e.g., Vogelsang et al. (2014), Brandstetter et al. (2015), Braun et al. (2014), Fockel and Holtmann (2014) and Berenbach et al. (2012).

Vogelsang et al. (2014) propose to model requirements and architecture in parallel, and evaluate the approach with 15 master students. In particular, the authors propose the use of Message Sequence Charts.

Brandstetter et al. (2015) present a process to perform early validation of requirements by means of simulation, using the control software of a desalination plant as an industrial case. Experiences of using the approach are discussed, but details on the execution of the use case are largely missing.

Resulting from a research project with academic and industrial partners, Braun et al. (2014) propose the use of model-based documentation. For RE, these include goal models, scenario models and function models. To our knowledge, the approach has not been evaluated in terms of an empirical study.

Berenbach et al. (2012) list several requirements they consider essential for a requirements modelling language, such as distinction between process and use case modelling. The authors argue that using UML for requirements modelling has proven to be frustrating. URML is piloted in one commercial project at Siemens, showing that the proposed concepts are useful.

Finally, the Model-Driven Requirements Engineering (MoDRE) workshop series that has taken place since 2011 contains many contributions on how models, in particular in the context of model-driven development, can be used for RE purposes.

2.3. RE in large-scale systems engineering

Initially, agile approaches were focused on small teams developing software (Beck, 1999; Meyer, 2014; Kahkonen, 2004). The success of these approaches have led to their adoption at scale (Dikert et al., 2016b; Lagerberg et al., 2013; Salo and Abrahamsson, 2008b), where non-agile, plan-driven, and stage-gate based processes have been the norm (Pernstål et al., 2012).

Due to their iterative nature, agile approaches are suitable for building systems whose requirements may change; further, experience from early versions of a system can impact later versions (Beck, 1999; Meyer, 2014; Gren and Lenberg, 2020). Gren and Lenberg even argue that the main motivation for choosing agile methods is to be able to respond to changing requirements (Gren and Lenberg, 2020).

However, Heikkilä et al. (2015) find in their mapping study that there is no universal definition of agile RE. Instead, they report that requirements-related agile practices such as the use of customer representatives, prioritisation of requirements, or growing technical debt are particularly hard to manage. The same authors also present a case study at Ericsson, where they investigate the flow of requirements in large-scale agile (Heikkilä et al., 2017). They find that practitioners perceive benefits such as increased flexibility, increased planning efficiency, and improved communication effectiveness. However, the authors also report problems such as overcommitment, organising system-level work, and growing technical debt. In their case study on the use of agile RE at scale, Bjarnason et al. (2011) also report that agility can mitigate communication gaps, but at the same time may cause new challenges, such as ensuring sufficient competence in cross-functional teams. In a case study with 16

[1] https://plantuml.com/.

US-based companies, Ramesh et al. (2010) identify risks with the use of agile RE such as neglecting non-functional requirements or customer inability. A systematic literature review on agile RE practices and challenges reports eight challenges posed by the use of agile RE (Inayat et al., 2015), such as customer availability or minimal documentation. However, the authors also report 17 challenges from traditional RE that are overcome by the use of agile RE. The authors conclude that there is more empirical research needed on the topic of agile RE. Consequently, Kasauli et al. (2021) report on RE challenges in scaled-agile system development that are neither addressed in contemporary RE literature nor by established frameworks for scaled-agile.

Paetsch et al. (2003) suggest that agile methods and RE are pursuing similar goals in key areas like stakeholder involvement and therefore could be integrated in a good way. The major difference is the emphasis on the amount of documentation needed in an effective project. Meyer, in contrast, criticises agile methods for limiting requirements engineering to functional requirements described through (exemplary) scenarios and discouraging upfront planning (Meyer, 2014). In fact, in practice such functional requirements are often described as user stories, e.g. formulated as boilerplate statements: "As a ⟨role⟩ I want ⟨feature⟩ so that ⟨value⟩". Leffingwell (2010). The much more detailed requirements of plan-driven approaches are omitted; instead, agile methods push for a continuous dialogue (with customer representatives or product owners) and comprehensive sets of tests, which are ideally automated (Meyer, 2014).

Given the set of challenges with managing requirements in scaled agile, it is unlikely that user stories and automated tests are enough to enable a shared understanding of requirements in agile at scale. It is therefore that we investigate the use of requirements models in agile.

2.4. Models in agile development

As a final area of related work, several authors have explored how models can be used in agile development, e.g., Rumpe (2002), Ambler (2001), Zhang and Patel (2010), Nakićenović (2012), Hansson et al. (2014) and Alfraihi and Lano (2017).

Ambler (2001) argues that modelling and agile development can go hand in hand. The author describes important aspects to succeed with agile modelling, e.g., using as simple tools as possible, fostering effective communication, and building agile modelling teams. Similarly, Rumpe (2002) argues that modelling can be used as a part of agile methodologies to further increase development efficiency. Concretely, the author suggests to use models for code and test case generation.

Zhang and Patel (2010) report the use of models in combination with agile development at Motorola. The authors report an increase in speed, quality, and a shortening of delivery cycle time. In a similar direction, Nakićenović (2012) reports the combination of modelling and agile development in the banking sector. Doing so, the authors could speed up development and decision making.

A number of further approaches to use models during agile development have been proposed as a part of the Extreme Modelling (XM) workshop series. However, as noted by Hansson et al. (2014), existing work on agile modelling suffers from a lack of empirical evidence on its application in industry. A more recent literature review by Alfraihi and Lano (2017) finds several industrial experience reports of combining agile development and modelling. However, they also note that there is little evidence.

In summary, there is a large body of work on how models are used in industry, including benefits and challenges of using models. Additionally, challenges of agile development and agile RE at scale are studied in considerable depth. Finally, a substantial amount of solution proposals for using models for RE and during agile development exist. However, to our knowledge there are no empirical studies investigating industry cases of successful model use for RE activities.

3. Research method

To address the RQs, we conducted a case study at a department at Ericsson, a large Swedish telecommunications company—in the following referred to as the case department. We embrace a constructivist world view, emphasising that different engineers at the case department have subjective views and opinions on the topic under investigation. The case study is both exploratory and confirmatory in nature. That is, we use a set of propositions we formulated initially and updated throughout the study. At the same time, we included a number of open questions to be investigated as part of the study.

3.1. Case description

We conducted this study in one department at Ericsson AB, a large Swedish telecommunications provider. The department qualifies as VLS agile with respect to the definition provided by Dingsøyr et al. (2014), i.e. that more than 30 Scrum teams develop a single product in parallel based on a scaled agile approach (Heikkilä et al., 2017). A conceptual model of the organisation and the artefacts relevant to this study is depicted in Fig. 1. In the description below, terms appearing as concepts in the diagram are in bold. **Requirement Model** is depicted with dashed lines, as it is a subject of our study and the relations to other concepts are hypothetical.

Cross-functional **teams** are assigned to and independently work on **backlog features** all the way to delivery on the main branch. Specialised coordination roles exist, e.g., for integration or architecture tasks. Scrum sprints are based on a backlog and a hierarchy of **product owners** breaks down **product requirements** and customer visible features to **backlog items**. While these **product owners** represent the customer requirements towards the product development, **system managers (SMs)** provide **system-level requirements**, which are implicitly implemented by the **teams**. These **SMs** also interact with agile **teams** in providing the system-level knowledge. Further products are developed using a similar methodology.

Hardware development at the company is largely decoupled from software development. New hardware becomes available with a regular, but low frequency.

At the time of the study, approximately 200 engineers worked at the case department. Development is closely aligned with existing standards that describe technical solutions in much detail, e.g., 3rd Generation Partnership Project (3GPP) (2021).

System-level requirements are stored in the tool T-Reqs (Knauss et al., 2018), which has been developed in house. T-Reqs allows storing text-based requirements and other artefacts together with code in version control systems such as git, thus bringing these artefacts closer to developers (Knauss et al., 2018). The tool has been used at the case company since 2017.

Requirements models are used to keep track of the system requirements and their relation. This is primarily done in the form of UML activity diagrams, where activities denote requirements and the flow between activities their relation and order. Even though activity diagrams are used, these do not describe detailed behaviour but rather a logical order, i.e., dependencies between requirements. As such, they are different from design models or detailed models of system behaviour. **Requirements models** are created and maintained manually. More details on the used requirements models are presented in Sections 4 and 5.

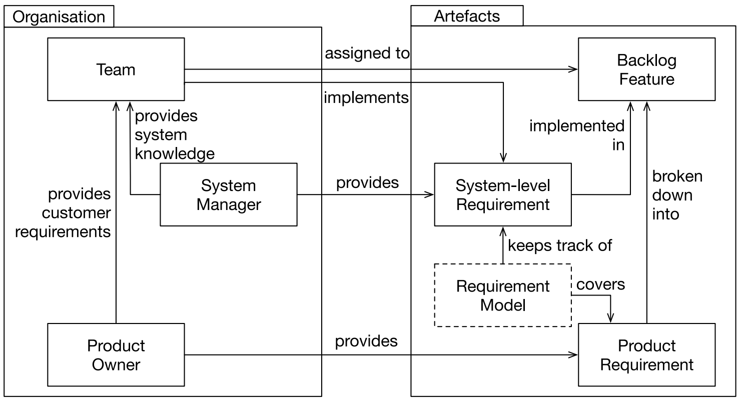

Fig. 1. Conceptual model of the organisation (left package) and artefacts (right package) relevant to this study in UML class diagram notation.

Table 1
Research propositions, with sources and target RQs. D denotes *Discussions*, DK denotes the authors' *Domain Knowledge*.

Nr	Proposition	Source	RQs
P1	Models are created by few experts, and mainly read by them.	Liebel et al. (2018b)	RQ2
P2	Access to models, and especially editing, is rare among testers and developers.	D	RQ2
P3	Model creators are not modelling experts. Therefore, use of modelling languages is ad-hoc and varies across the organisation.	Liebel et al. (2018b)	RQ2
P4	Testers and developers do not see the need/use of modelling requirements.	D	RQ1, RQ2
P5	Only few diagram types (of the UML) are used.	Liebel et al. (2018b)	RQ2, RQ3
P6	Testers and developers do not think that the present models carry important information.	D	RQ1, RQ2
P7	Layouting of diagrams is important to the users.	DK, layout studies	RQ3
P8	Stakeholders believe that modelling should be integrated with existing development tools (e.g., git).	D	RQ1, RQ2
P9	Stakeholders do not believe that the requirements models should be used for automated tasks.	Liebel et al. (2018b)	RQ1, RQ3
P10	The current modelling solution is restricting employees in their work.	Liebel et al. (2018b)	RQ2
P11	Even if a better/good modelling solution would be in place, most stakeholders would not update/maintain the model.	Liebel et al. (2018b)	RQ1, RQ3
P12	Navigating between different diagrams is an important feature.	Liebel et al. (2018b)	RQ3

3.2. Study scope and propositions

From previous research and initial scoping meetings with two contact persons at the case department, we formulated a number of propositions addressing the three research questions. These are depicted in Table 1. For each, we describe the origin of the proposition.

The propositions can be summarised as follows. For RQ2, we envision an organisation in which few experts work with models (P1), while in particular the roles working with lower level of abstractions, testers and developers, do not use the models (P2), do not see the need for them (P4), and do not think that any relevant information is contained in the model (P6). People creating the models are not necessarily experts, resulting in an ad-hoc approach (P3). Few UML diagram types are in use (P5). The existing tool solution for modelling is restricting the employees in their work (P10).

For RQ1, we expect that sentiments towards modelling roughly resemble the current use: a few "power users" of models, but a substantial amount of people not believing in the usefulness of models.

For RQ3, we cover a few important tool decisions, including the need for only few diagram types (P5), the need for layouting capabilities (P7), automation support (P9) and navigation between diagrams (P12). Furthermore, we expected some insights from participants that would not use the models even with better tools (P11), since they might have additional input on what would be the preferred format. For the remaining feature space, we chose an exploratory approach asking several free-text questions to get additional input.

3.3. Survey design, execution and analysis

To evaluate the propositions, we designed an online survey. Our contact persons reviewed the survey design.

After review, our contacts sampled 54 people at the case department, all of which they judged to have sufficient knowledge of the model to answer our questions. We received 33 answers, i.e., a return rate of 61.11%. The participants worked in 16 different areas of the case department, covering various tasks and product aspects, both from functional and non-functional perspective. However, SMs were over-represented among the participants (22 out of 33 participants had an SM role). Finally, the majority of participants had substantial work experience in their role (depicted in blue bars in Fig. 2) and modelling experience (depicted in yellow bars in Fig. 2).

We analysed the survey answers by creating summary statistics and evaluated the propositions in a qualitative manner, i.e., without employing statistical tests or related statistical methods. The first author summarised open-ended questions by assigning topic codes (Saldaña, 2015) to each stanza, i.e., excerpts of text that are assigned a code, then grouping related stanzas together and counting their frequency. The mapping of survey questions to propositions is depicted in Table B.3 in Appendix B. As a form of member checking, we presented the results to our

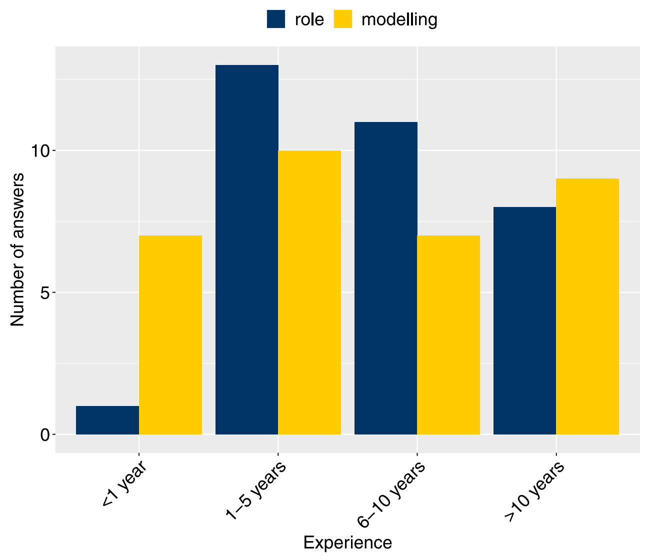

Fig. 2. Model creation and reading frequency. The blue bars refer to work experience in the current role at the company, while yellow bars refer to experience with modelling. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

contact persons, who disseminated the findings in the department. To deal with the over-representation of SMs in our survey, we considered the survey results on a per-role basis, as discussed in Section 4.

3.4. Interview follow-up

Following the questionnaire, we updated and refined our list of propositions and added some open questions (see Appendix C). The open questions relate in particular to contradictions in the survey data. For instance, while the majority endorsed using text-based models, the suggested solution does not support manual layouting, an important feature requested by the majority. We used the proposition and questions as an input for the creation of the interview guide.

Our contact persons recruited five engineers to be interviewed. We requested a varied set of roles and mindsets, to obtain diverse information. In particular, we also asked them to recruit participants who might be skeptics of requirements modelling or modelling in general. While this is a small sample, it nevertheless represents about 10% of the survey sample size, i.e., engineers who are knowledgeable enough in modelling to answer our questions.

We analysed the interview transcripts using the following process. Both authors, Grischa (GL) and Eric (EK), coded all interviews. GL used a list of a-priori codes aligned with the propositions and questions, while EK used open coding. In both cases, the coding followed a content coding approach (Saldaña, 2015). That is, we assigned codes that describe the content of the coded stanza, assigning codes on a per-answer basis. In cases where the interviewees clearly discussed different content, we separated the answer into multiple stanzas which we coded differently. GL piloted the initial a-priori codes on one interview, then modified them according to the pilot. The final a-priori codebook is discussed in Appendix D.

After the first round of coding, we discussed the resulting code distribution and decided to continue a parallel approach. That is, we jointly structured the existing codes in a second-cycle coding approach. We hierarchically grouped the codes obtained from EK's open coding into the different (and much more abstract) a-priori codes, then grouped the resulting clusters according to our three research questions. We then extracted candidate themes, which we validated using all stanzas coded with at least one of the open codes for the theme. Simultaneously, GL analysed the interview data one more time and followed a holistic coding (Saldaña, 2015) approach, writing analytical memos while working through the data. Finally, we integrated the initial themes from the second-cycle coding approach with the themes extracted from the holistic coding and memoing.

3.5. Validity threats

Given the constructivist nature of this case study, we present the threats to validity in terms of transferability, credibility and confirmability (Petersen and Gencel, 2013).

3.5.1. Transferability

Transferability describes to what extent results from the study can be transferred to cases that resemble the case under study (Petersen and Gencel, 2013).

Many of the reported aspects are specific to the case department, e.g., the role of the *SM* that connects agile teams with the system-level view. However, we know from previous work (Kasauli et al., 2021) that similar roles and situations exist in many systems engineering companies. Therefore, we expect that the findings apply in similar cases as well. One exception might be the large emphasis on software development at the case department, which is in contrast to many other systems engineering organisations, where hardware is developed in parallel and thus causes long lead times and longer feedback cycles. Hardware development is not as prominent in our case company, which means the company likely faces challenges that are slightly different to other systems companies. Another characteristic that might limit transferability is the domain of telecommunications systems. In telecommunications, requirements that describe a temporal flow and restrictions on the order of this flow (e.g., the order of messages) are common. This makes activity diagrams a suitable way to express the relations between the requirements, which might not be a suitable choice in other domains.

We used purposeful sampling to select interviewees that had diverse background and at the same time could comment on the use of requirements models. However, we did not reach saturation in all our themes. This means that there might be additional facets or themes, or contrasting ideas that we did not capture. This is a threat to the transferability of our findings.

Our case is a department that is part of a company that uses VLS agile development. As such, it might not fully relect VLS agile as a whole. We believe a department in a VLS company by default faces many situations that are specific to VLS agile development. For instance, the requirements models we studied in this paper have been introduced as a way to address communication and coordination issues arising from VLS agile development. Therefore, we argue that the case department is a valid case for studying VLS agile development. Nevertheless, there is a potential threat to transferability, as we might have missed issues arising from VLS agile development that are not present in the studied case.

3.5.2. Credibility

Credibility describes whether findings are reported truthfully, or have been distorted by the researchers (Petersen and Gencel, 2013).

All interviews were recorded, and data analysis performed on the verbatim transcripts. Additionally, we report quotes for all themes in our qualitative interview analysis. This should ensure credibility of the findings.

Table 2
Evaluation of propositions.

Number	Proposition	Supported by survey
P1	Models are created by few experts, and mainly read by them.	Yes
P2	Access to models, and especially editing, is rare among testers and developers.	Partially
P3	Model creators are not modelling experts. Therefore, use of modelling languages is ad-hoc and varies across the organisation.	No
P4	Testers and developers do not see the need/use of modelling requirements.	Partially
P5	Only few diagram types (of the UML) are used.	Yes
P6	Testers and developers do not think that the present models carry important information.	No
P7	Layouting of diagrams is important to the users.	Yes
P8	Stakeholders believe that modelling should be integrated with existing development tools (e.g., git).	Yes
P9	Stakeholders do not believe that the requirements models should be used for automated tasks. They should instead be used as documentation only.	No
P10	The current modelling solution is restricting employees in their work.	Partially
P11	Even if a better/good modelling solution would be in place, most stakeholders would not update/maintain the model.	No
P12	Navigating between different diagrams is an important feature.	Yes

We performed first-cycle coding and memo writing for both the free-text answers in the survey and the interview transcripts. This should avoid threats to credibility arising from long chains of interpretation in our analysis.

3.5.3. Confirmability

Confirmability describes the extent to which conclusions made by researchers follow from the observed data (Petersen and Gencel, 2013).

To structure our study, we used propositions prior to the survey and in between the survey and interviews. We then evaluated them after each analysis step. Furthermore, survey and interview instruments, as well as the codebooks are available in the appendix to this paper. However, it is a threat to confirmability that the used propositions are not formal hypotheses as used in hypothesis testing, but rather used as guidance only. That is, several of them have been evaluated in a qualitative fashion or based on incomplete answers. For example, in P3 ("Model creators are not modelling experts. Therefore, use of modelling languages is ad-hoc and varies across the organisation".), we did not define what a modelling expert is, and ultimately evaluated this in terms of modelling experience and not in terms of education or another metric. Similarly, in P5 ("Only few diagram types (of the UML) are used".), we did not define what "a few" is prior to analysis. Finally, several items in the questionnaire could potentially be mis-interpreted by participants, or be worded in a suggestive fashion, e.g., "The existing requirements models do not carry important information". To reduce this threat, we reviewed the questionnaire with our contact persons at Ericsson.

4. Exploratory survey

In the following, we present the results of the exploratory survey in terms of descriptive statistics and relations to the propositions. We then discuss the implications of the survey findings.

4.1. Survey findings

The resulting proposition evaluation is summarised in Table 2.

For P1, 5 participants state that they create/modify diagrams at least weekly (see the dark blue bars in Fig. 3). Three of these participants are System Managers (SMs), one is a Developer and one is both a Developer and SM. When consulting the read access/use of diagrams (light yellow bars Fig. 3), these five participants have weekly (4 answers) or daily (1 answer) read access to the diagrams. In the entire sample, only 2 more people stated that they read/use the diagrams on a weekly basis. This seems to confirm

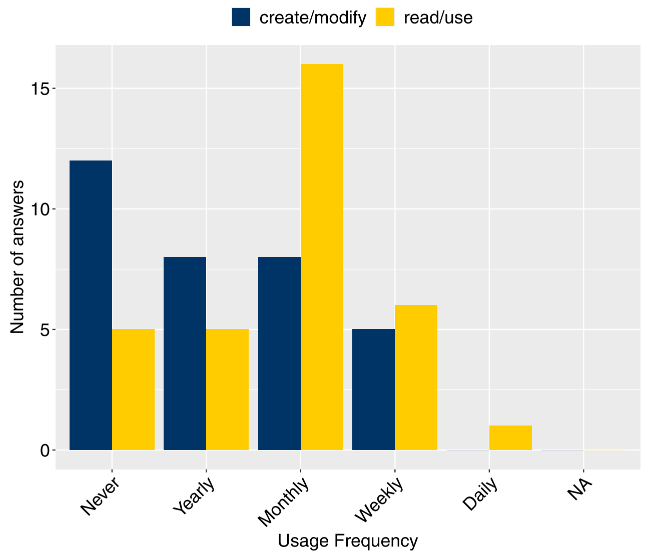

Fig. 3. Model creation and reading frequency. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

our proposition that it is indeed a small group responsible for modification and use of diagrams.

Our data shows a mixed picture for P2 (testers and developers access and modify models rarely). Of the 8 people with development or testing roles (out of 33 participants, see Section 3), 5 state that they read models on a monthly basis, 2 weekly, and 1 yearly. Creation is less common, with 3 stating that they never create or modify models, 2 yearly, 1 monthly, and 2 weekly. This picture does not change significantly if we consider only those who have a pure testing/development role (without addition of Designer/Architect or SM). Overall, these figures do not allow a clear answer as to whether P2 is confirmed or not. The free text answers indicate that people not accessing the models are mainly concerned with the tooling (difficulty of tooling, access to the tool) and the effort it takes to comprehend the models (too much detail, information spread across model layers/navigation).

P3 (model creators are not modelling experts) is not supported by our data and must be rejected. The participants who create/modify models at least weekly all have considerable experience with modelling (at least 5 years). We did however not ask whether they have a formal education, or proceed in an ad-hoc manner.

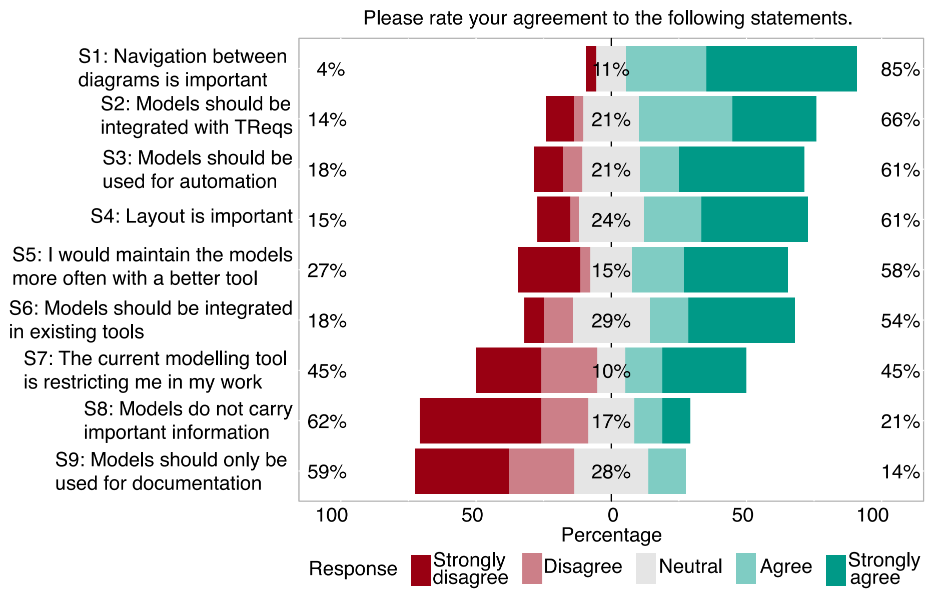

Fig. 4. Agreement to different statements.

For P4 (testers and developers do not see the need of modelling requirements), the survey data shows again a mixed picture. Three survey participants agree or strongly agree that they would update the models regularly if they had a better tool (S5 in Fig. 4). However, two participants strongly disagree and three do not know. There is again no noticeable difference between the pure developer/tester roles and others. Additionally, we do not see a pattern in the answers with respect to how the creation/modification patterns look like at the moment (e.g., "Participants who already modify/create diagrams often would not do it more often"). Interestingly, six people are generally positive towards modelling, and the remaining two neutral. No one opposes modelling per se. The free-text answers for this question do not give a clear justification of the pattern, either. However, one participant stated the concern that the current model is unreliable and therefore not useful, suggesting to assign someone to manage the model.

As expected from the study preparations, due to the domain, behavioural models dominate at the case company (Fig. 5). Activity (18 answers) and sequence diagrams (13 answers) are used by the majority of the participants. However, state machine and use case diagrams follow closely with 9 and 10 participants. Class and component diagrams are used by 4 and 3 participants only.

For P6 (developers and testers do not think that the present models carry important information), it is rather interesting to observe that our proposition must be rejected based on our data. Indeed, 5 testers/developers out of 8 disagree or strongly disagree with the statement that existing models do not carry important information (S8 in Fig. 4). Of the remaining 3 participants, only one agrees, with the other two being neutral or "don't know".

P7, the importance of layout, is clearly confirmed by our participants (S4 in Fig. 4). 20 participants agree or strongly agree, 8 are neutral, and 5 (strongly) disagree. One participant disagreeing noted that requirements should be stated in text, and have as a maximum pictures/models to support its explanation.

Regarding the integration of models into existing tools (P8), the picture is favourable (S2 in Fig. 4). 15 people (strongly) agree that this should happen, 8 are neutral, 5 against, and 5 don't know. Specifically, 19 participants agree that models should be integrated into the existing text-based requirements tool T-Reqs (Knauss et al., 2018), with 4 disagree, 4 don't know, and 6 neutral answers.

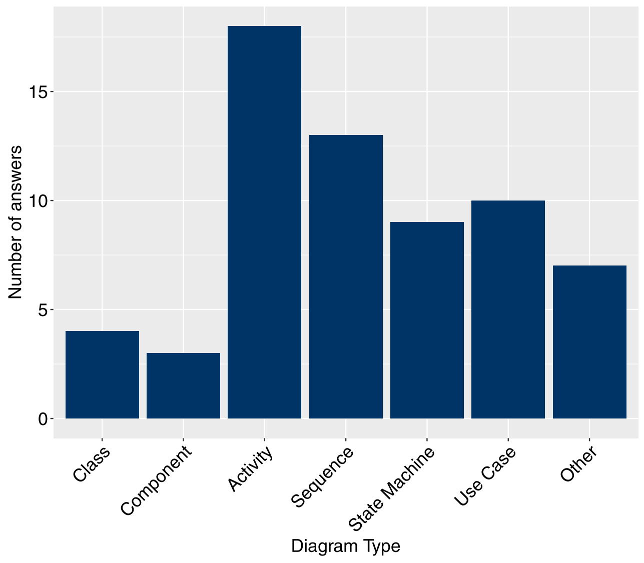

Fig. 5. Diagram usage according to UML diagram types.

From earlier work, we expected that informal modelling without any automation would be favoured by most stakeholders (P9). Interestingly, our results show a different mindset (S3 in Fig. 4): 17 people agree that they should be used for automation, while 6 are neutral, and 5 each disagree or answered do not know.

Regarding P10, there is a disagreement as to whether the current solution is restricting the participants in their work (S7 in Fig. 4): 13 each agree and disagree. 4 do not know and 3 are neutral. While we initially thought that some participants would not feel restricted since they do not use the models, this picture was not confirmed clearly by looking at the disagreeing group: Only 2 of the participants stating that they don't write/modify models are in that latter group.

P11 again contradicted our impression from previous work—we expected that participants would state that they would not update their models even if the tool was better (S5 in Fig. 4). However, 15 participants stated that they indeed would update the model if the tool was better. 6 participants stated that they don't know, 5 were neutral and the remaining 7 disagreed.

P12, that navigation between diagrams is an important feature, got the strongest support in our survey (S1 in Fig. 4). Indeed, 23 participants (strongly) agreed with the statement, 6 participants didn't know, 3 were neutral, and 1 disagreed.

4.2. Survey discussion

Overall, we summarise the survey findings as follows. We find strong support both for working with models (RQ1) and the use of requirements (RQ2) among our participants. With respect to the needs to support the intended use of requirements models (RQ3), layouting and navigation, a focus on activity and sequence diagrams, and close integration into development tools and version control systems surfaced. We discuss these aspects in this section and revisit them in the second, interview-based part of this study with the aim to shed more light on these aspects.

For RQ1 (Sentiments for and against models) we expected a diverse result, based on own experience and literature that propose a divide between model proponents and opponents. Instead, we find that most participants clearly see the value of models. Interestingly, there were a few voices mentioning that text-based requirements would be enough and that models are too complicated to handle. In particular, participants mentioned that the little requirement work per team in the case department could easily be handled in text.

Similarly, the results show that the picture for RQ2 (How do different stakeholders use requirements models?) is not the negative one we expected, especially considering that the requirements models are at a high level of abstraction. While it is true that models are created by few people, and also mainly accessed by them, the majority of our participants see the value of requirements models and also the information contained in existing models. This covers all roles, including testers and developers that do not have an SM role in parallel. Furthermore, the testing and developer roles are also not negative towards modelling. The model creators/maintainers have substantial experience, though we do not know their educational background in modelling. Indeed, the move from the existing modelling tool to an integrated solution is supported, with few exceptions. From free-text comments, we see that there are several factors hindering the use of models at the moment. These include lack of tool access and tool usability, the complicated nature of models, the amount of details and need for constant work related to models, and the outdated information in models. Finally, several statements related to process issues: Participants stated that there was currently no clear direction on whether the models should be kept updated, no process of doing so, and a lack of knowledge how to model and on which abstraction level. This means that the role of reading and understanding the model and then feeding the information into the teams ends up in the hands of a few people (SMs). Participants suggested regular modelling courses for users, clear abstraction levels on what should/should not be in the models, and examples of models that are considered to be of high quality.

Regarding needs to support modelling (RQ3), participants strongly supported the notion that layouting and navigation are key features. Models at the case department often contain multiple requirements in flow charts/activity diagrams, with one requirement per activity node and a text description of each. The entire diagram then gives the context of the requirement, i.e., what happens before and after, and how it relates to other requirements. Often, there are links to other diagrams as well. Therefore, both the layout and the navigation are required to understand how the system behaves as a whole. Our proposition was confirmed that only few diagram types are in active use, mainly activity and sequence diagram. However, there were minor usages of several further types. Free text answers clearly pointed to the fact that any modelling tool needs to be integrated into daily work by using the same tools developers use and by being able to integrate the models with (text-based) version control such as git. While pictures are helpful, the models should theoretically be readable in text, in particular changes to models. Finally, a large share of the participants stated ease of use as one of the main success factors for a modelling tool.

5. Confirmatory interviews

After the survey, several gaps in our understanding remained, in addition to new questions that arose. These gaps directly follow from the survey findings in relation to the propositions in 1 and 2. Since there is a general willingness to work with requirements and models, combined with a sense that current support is lacking, and clear indication of specific needs, clear questions for follow-up in-depth interviews follow (see Appendix C).

In the following, we discuss the findings relating to our three RQs. Given the open nature of interviews, themes in the data can relate to more than one research question. First, we discuss how interviewees see the role of requirements in VLS agile systems engineering in Section 5.1, the role of models in VLS agile systems engineering in Section 5.2, and the use cases arising therefrom in Section 5.3. All these topics relate to RQ1 and RQ2. Finally, we discuss the consequences for tooling (RQ3) in Section 5.3. The summaries at the beginning of each subsection are related to the specific case, and do not necessarily generalise beyond the case department.

5.1. RQ1/RQ2: Requirements in VLS agile systems engineering

Requirements and VLS Agile (RQ1/2)

- System-level requirements are available too late in the process.
- Requirements are an asset when changes are made, but often need to be updated first.
- Importance of requirements: interviewees agree, but have doubts about general sentiment in organisation.

Conclusion: Role of requirements in VLS agile is conceptually unclear.

At the case department, requirements used to be written prior to development. Now, due to the agile transformation at the company, only vague requirements are developed prior to the sprints, which are then shaped and refined in parallel to the development and testing. Some interviewees perceive this as documentation work only, while others see it as a crucial step in invention and in preparing for future maintainability. That is, the role of requirements in large-scale agile development is perceived very differently in the case department. Several interviewees take the standpoint that there are too few requirements, and that those are written too late in the process. They take the traditional development point of view in which upfront requirements analysis guides the development later on, and in which requirements provide the system knowledge. The lack of such requirements is therefore seen as an issue.

> *"[..] someone updates the implementation and suddenly things don't work anymore. And then the problem is you have to determine why. Because a lot of the behaviour of the product is not really based on requirements. We don't have requirements on exactly everything".* – Interviewee 4

None of our interviewees stated that they considered requirements unimportant. However, several of them did express that this was a common belief within the company. That is, that other sources but written requirements are sufficient to obtain system knowledge, e.g., test cases, or annotations to standards (compliance declarations). Since all of the statements were based on beliefs and potential third-hand information, we are however cautious as to the reliability of the information.

Overall, our interviewees agree that requirements on system level are an important asset. However, it remains unclear what their exact role is, i.e., whether they are purely for documentation purposes or should serve as a prescriptive artefact as to what shall be developed. In relation to this role, it is therefore unclear when they should be written and in what quantity. This is in line with our previous work (Kasauli et al., 2017, 2021) that finds that requirements in VLS agile systems engineering are conceptually unclear.

5.2. RQ1/RQ2: Requirements models in VLS agile systems engineering

Requirements Models and VLS Agile (RQ1/2)

- Requirements models are important to understand the big picture.
- Requirements models are hard to keep up to date.
- Some models are increasingly outdated, thus losing value.
- Changes can break a model and require re-design.
- Different modelling styles make shared modelling difficult.

Conclusion: While requirements models provide substantial value, using them successfully in practice is challenging.

Given that the role of requirements is conceptually unclear or at least different from the original, plan-driven process in which requirements were written up front, the role of using models to convey requirements information is also debated at the case company.

Several of our interviewees valued the existing requirements models. They reported that the models serve primarily *as a boundary object* between different *agile islands* and the overall system, providing the long-term knowledge (Kasauli et al., 2020). A common issue in VLS agile systems engineering is that individual methodological islands exist in a company that are disconnected (Kasauli et al., 2017, 2021), e.g., individual Scrum teams and an overall plan-driven process. Having a model that relates system-level requirements to each other can help building bridges between the islands and keep knowledge over a long time. For example, the models can help engineers understand how isolated user stories connect to the overall system behaviour. Furthermore, incoming change requests can be understood better in relation to the current system-level behaviour.

> *"The system model really like defines...[..] we put requirements that tell how something should behave in relation to some other functionality".* – Interviewee 2

However, we also have several interviewees that reject the requirements models, for several reasons. First, while they consider requirements models useful in principle, they differ whether it is worth spending the required effort to create and maintain the models over time. Just as with other forms of documentation, maintenance is essential. If the model becomes outdated, it loses its value to the engineers.

> *"The problem is that the information gets outdated and there are not enough resources to make sure they are correct. And the focus probably lies on other things".* – Interviewee 1

In fact, an interviewee stated that several models at the company are outdated at the moment, and would require a substantial effort to be updated.

> *"The requirement model itself has degraded in many cases to the point where it's useless, totally inaccurate and not up to date".* – Interviewee 4

Also in relation to maintenance effort, one interviewee stated that changes to the requirements can be orthogonal to the way the requirements models are designed, thus leading to substantial maintenance effort up to entire re-designs of a model. For instance, changes that lead to a modification in the system structure could require moving requirements between models or entirely re-designing the information flow in models that depict behaviour or interactions.

> *"Because the model can be modelled in the way that it's hard to fit in my new requirement [..] And then I...okay, should I add to the mess [..] or should I try to remodel this? I think most people just add something with least effort and then the model becomes even harder to add something to".* – Interviewee 2

While only one interviewee mentioned this issue, we considered it critical enough to list it here.

In addition to maintenance and model creation effort, several interviewees highlighted that there is no common way of modelling. Currently, engineers do not get any instructions on how to create a model, how to use it, and how to maintain it. This leads to a multitude of different modelling styles, reluctance to modify a model, and in many cases to teams abandoning the model altogether.

> *"[..] there's so much freedom. And we happen to have different styles depending on which person is doing the work. You try to...you like to model it in a way you like".* – Interviewee 5

We further hypothesise that this also leads to a higher overall effort, since a person used to one model might need additional training to use or modify another model, as it might be modelled following a completely different style.

Finally, several interviewees have reservations towards requirements models due to tooling issues, and integration of the tools into the process. Most of these reservations are similar to tool challenges known from related work, e.g., Hutchinson et al. (2011a,b), Whittle et al. (2013), Liebel et al. (2018b) and Liebel et al. (2018a). For instance, the interviewees mention outdated, heavy-weight tools, and the risk of vendor lock-in.

> *"I have logged into Rhapsody just a few times, but in general that's very slow and so on. So that's not an option to log into it to get information".* – Interviewee 1

Based on these themes, we conclude that requirements models provide substantial value in VLS agile systems engineering. However, practitioners struggle to use them successfully due to challenges in maintaining them and modelling in a consistent style that allows engineers to work on shared models.

Use Cases for Requirements Modelling (RQ1/2)

- Models provide an overview of the requirements and their relationships.
- Models provide valuable information to developers and testers.
  - Many read, few write
  - Potential imbalance (effort/benefit)
  - Potential lack of awareness and appreciation of models

Conclusion: A lightweight approach to requirements models that exposes models to many stakeholders is seen most favourably by the interviewees.

5.3. RQ1/RQ2: Use cases for requirements modelling

As a third theme in our analysis, we discuss the different use cases for requirements models that our interviewees report or discuss, and the roles that relate to these use cases. These are either already in place at the case company today, or the interviewees raised them as desirable or promising. Not all interviewees had a good overview of all stakeholders that actually interact with the models, yet implicit assumptions on which roles should interact with the models existed.

The requirements models at the case department are primarily a collection of flow charts/activity diagrams. Activities are used as containers for textual requirements and their connections depict the connections/traces between requirements. There is typically a main flow and potentially multiple alternative flows, describing error cases. The main value of the model lies in the overview it provides, primarily obtained through the relationships between requirements. Several interviewees state that this overview is something that is hard to achieve with a text-only representation.

"It's impossible to read all of them and understand what the total requirement mass is". – Interviewee 2

The primary use case for the existing requirements models at the studied department is read-only access, to provide valuable information to developers to inform their activities. However, in many cases this information is provided by other roles. There is the widespread idea that mainly the *SM* reads the model in order to then inform other roles and to provide an overview. SMs use the model as a source of information to answer questions regarding the overall system functionality, to investigate how changes affect the system, and to understand if change requests are due to misunderstood requirements, bugs, or actual changed needs. This restricted use of the requirements models has the advantage that other team members do not need to be experts in modelling. However, the disadvantage is that they might not be aware of the models' value and purpose, leading them to believe that updating the model is a waste of time—they do not see that the SM uses the model as a core element in their work.

The degree to which different SMs use the requirements models depends on their personal preferences and the state of the model. As discussed earlier, some models are outdated and therefore no longer used by the SMs.

When discussing future use cases, most interviewees mentioned that all team members should read the model, but not necessarily write.

"Everybody should at least have read access. I cannot see any reason why you should not have read access". – Interviewee 1

Relatively few stakeholders currently modify the model. These are primarily the SMs, who create and update requirements models according to changes in the system, e.g., newly-implemented user stories.

Testers currently benefit from the requirements models to understand which requirements relate to a given object under test. Again, the degree to which they use the models varies. Interviewees also expressed that the model should allow testers better linking of test-related information, such as individual test executions. This is currently not possible, but would enable better integration of work the testers currently have to do in other tools.

5.4. RQ3: Tool features and information content in requirements modelling

Needs for Requirements Modelling in VLS Agile (RQ3)

- Models need to be navigable and searchable.
- Broad and easy access to the models is key to adoption.
- Education and guidance for modelling need to be provided.
- Review mechanisms similar to code reviews can foster adoption.
- Generating artefacts from models can serve as a maintenance incentive.
- Heterogeneity of teams and tools needs to be supported.
- Automation of model layout is a trade-off.

Based on the evidence from our interviews, several hypothetical ways to use requirements models exist. We extrapolate the features necessary in tooling for requirements models, and the information content those models need to have. Note that this section is only partially based on evidence from our interviews, and partially a logical extrapolation based on our expertise in the field. For each theme, we discuss to what extent we do have evidence for the discussion points.

We distinguish four hypothetical scenarios based on our data. These are:

1. Entirely abandoning requirements modelling in favour of using other artefacts as sources of information.
2. Using requirements models as sources of information for the SMs only.
3. Using requirements models as sources of information for developers, with SMs maintaining the model.
4. Use and maintenance of the models by the entire development organisation.

Scenario 1 (no requirements models) makes a discussion about tooling and information content unnecessary.

Documenting knowledge:. Instead, abandoning models raises the question where the information should reside instead at the case department. That is, information on how the overall requirements relate to each other, e.g., in terms of main and alternative flows/scenarios. In our survey and during the interviews, we found several statements that existing documentation such as the user manuals could serve this purpose.

"Yes, you could use [customer documentation] as a requirement if it works, but it has not always worked". – Interviewee 5

Similarly, tests are often raised as a potential source of knowledge that could replace written requirements, both in our data and in related work.

However, our interviewees also raise concerns that tests might not be sufficient. That is, each test expresses exactly one scenario, which means that the overall system behaviour arises from the combination of the entire test suite. Therefore, this overall behaviour is not easily visible.

> *"If you only have the test case, it's not clear really what parts that the test case verifies that our requirements can...and what is just a behavior. That's a risk".* – Interviewee 4

If requirements models are used in some capacity, several important needs arise. Some of these are already present in the current tool solution at the case department, others are lacking according to the interviewees.

Supporting traceability:. In addition to the information being present, requirements need to exist so that testers know what to test, and have a target they can trace to. Currently, the tool T-Reqs (Knauss et al., 2018) fulfils this purpose at the case department, even though one interviewee expressed that the possibilities for tracing are limited. For instance, test executions could not be traced in T-Reqs and could therefore not be addressed in the tracing.

Hence, while traceability capabilities exist in T-Reqs, improvements are necessary.

Traceability capabilities are needed especially if a larger part of the organisation uses the requirements models, i.e., for Scenarios 3 and 4. However, even for Scenario 2 this might be necessary, in case the SMs are creating trace links.

Navigable and searchable information:. In the context of VLS systems engineering, requirements and their relations quickly become complex. Hence, it is important that they can be navigated and searched efficiently.

> *"there is the practical thing of it. And for our requirements to be useful it has to be first of all easy to find and navigate. Because it's a complex model, you can't just...read one model, one short requirement out of context. [..] So you need an easy way to navigate the model".* – Interviewee 4

For instance, links between requirements could be made navigable by using hypermedia with hyperlinks between requirements, as is standard in most RE tools. Currently, this is supported by T-Reqs for textual artefacts. For models, several modelling tools allow for hierarchical models that support hiding information in sub-models, or distributing models and diagrams over several files. However, extracting relevant information from models is difficult (Liebel et al., 2018b), e.g., in the form of a search.

Searching is of particular importance in case the requirements models are used as an important source of knowledge by many stakeholders in the organisation, i.e., Scenarios 3 and 4. If only SMs use the requirements models (Scenario 2), they are often so familiar with the model(s) they work with that search might not be required.

Broad information access:. Several of our interviewees stated that access to requirements information needs to be open to everyone. If only selected roles have access to the information, requirements easily become an abstract concept that many engineers are not aware of or do not consider important. This lowers the overall acceptance of requirements as an important source of knowledge in the organisation.

> *"I think it's important to be used and to be...maybe have good qualities, I think it's good if it can be [..] easily accessed. That seems like a crucial point I think".* – Interviewee 2

While easy tool access can help acceptance of the models in any case, especially for Scenario 3 and 4 (developers use the models at least as a source of information) this feature is crucial. Previously, the case department used IBM Rhapsody, which required engineers to set up a remote environment to open the tool. This turned out to be a large obstacle and only few engineers accessed the model, effectively limiting the access to the SMs.

> *"Well, first of all it has to be accessible, both for the people who need to do updates of the model, and also for the people then who are supposed to read the model, read the requirements, the designers and testers. I mean if it's very easy to access, people will do it. If it's hard to access, people will not".* – Interviewee 4

Need for education and guidance:. Similar to coding guidelines that exist in most organisations, *modelling* and *requirements* guidelines need to be in place that ensure a common approach to modelling and requirements. While important for any kind of scenario in which requirements and models play a role, this guidance becomes more important when many people are supposed to edit models, i.e., for Scenario 4 in particular. Several of our interviewees raise this point.

> *"[..] everybody has access. But that of course means that there should be guidance. So when should you use it? And why should you use it? And who should use it for what reason?"* – Interviewee 1

As a variant of guidelines, several interviewees also suggest mentors at the company that can support others in modelling-related questions.

> *"So need to be one or a couple of people that really know how to model that you can ask for 'Okay, can we have an hour and come to a conclusion how we should model this, my problem?'".* – Interviewee 2

Need for review mechanisms:. While guidance and education can improve the quality of models, enforcing standards could become difficult. Hence, mechanisms are required to do so. Drawing from experience both in RE and in programming, we believe that both reviews (similar to code reviews) and automated analyses (similar to requirements heuristics or static code analyses) have the potential to enforce model quality. Reviews are also brought up several times by interviewees as a reason to rely on textual modelling tools like PlantUML.

> *"But if you instead did it in a pure text, then you can use your standard merge tools. You can do standard diff to see what has been updated. If someone wants to do a review, I mean we have today code review tools we use for everything else".* – Interviewee 4

While, on first sight, reviews might only appear necessary once a large number of people modifies and creates requirements models, i.e., in Scenario 4, even in Scenario 2 (SMs work with models) reviews might be helpful to establish a more aligned way of modelling in the case department.

Code generation: Carrot and stick:. Existing models at the case company are in many cases outdated or of low quality. While guidelines, mentors, and model reviews could help addressing this, several interviewees suggested that artefacts could be generated from the models in a form of lightweight model-driven engineering process. This would encourage people to read and update the models, as they serve as a ground truth. In turn, generation could help enforcing guidelines. For instance, using elements that are semantically wrong would lead to incorrect artefact generation or errors during the generation process.

"[..] maybe it can just generate a text. [..] I know that in Rational Rose you could generate a Word document." – Interviewee 3

Due to the use of the requirements model as a read-only source of information, the model is disconnected from the final product. Nothing is generated from the model that is used further in development. This means that the end product could potentially be completely contradicting the model. This bears the risk that the model deteriorates. If code or other important artefacts would directly be generated from it, this would not happen.

However, it is important to note that generation of artefacts requires a well-balanced approach: None of the interviewees expressed the desire to follow a strict generation approach, where, e.g., the entire code is generated based on models. Hence, generation should remain a tool that allows engineers to see some direct benefit of the models, and to obtain quick feedback of sorts, without dictating their entire workflow.

Generation of artefacts could be used from Scenario 2 onward. Especially in Scenario 3 and 4, we believe some kind of automation might be necessary to reach broad acceptance of the requirements models in the case department.

Supporting heterogeneity:. In VLS systems engineering, development work is organised in a number of different ways. For instance, component teams are common, where different teams focus on their individual components. The case department instead structures work by expertise, e.g., having teams that focus on quality attributes such as availability. This heterogeneity leads to different approaches, and to different needs. Furthermore, heterogeneity and independence of teams is encouraged by the use of agile practices. In turn, this heterogeneity requires highly flexible tools and notations, or independence within the teams to choose their own tools and notations for requirements and requirements modelling. Forcing a single tool/style/approach on all teams will likely lead to resistance. Nevertheless, advertising success stories from individual teams might pave the way for others to adopt similar approaches.

"We have different areas with different needs, but my area is very functional". – Interviewee 4

With respect to requirements modelling, supporting heterogeneity might also mean accepting that some teams choose to abandon modelling entirely, either due to preference, or due to a mismatch with their way of working. The heterogeneity is independent from the different scenarios: for all scenarios that use requirements models, different needs exist across the department and therefore need to be supported.

Abstraction level:. The case company needs to be compliant with specifications from 3GPP (3rd Generation Partnership Project (3GPP), 2021). This standard contains many technical details, that often need to be discussed or referenced in the requirements, e.g., to discuss required additions.

"[..] somewhere I would say between 75% and maybe up to 90% depending on a bit what area we are in, the requirements are specified by 3GPP, we implement the standards. And then we don't need to re-specify this. But sometimes we need to clarify this because the standard might not be very clear on certain details. So we might need to annotate it and say 'okay, in this case the value should be this or we do like that', when the standard is not clear enough". – Interviewee 4

This leads to requirements or requirements models on a low level of abstraction. Any notation or specification format used at the case company needs to support this level of abstraction. For example, to make a requirements model understandable, it might be necessary to use hierarchical models to hide details. A modelling tool would thus need to offer sophisticated hierarchy and decomposition features.

However, given that the requirements at the case department are closely-aligned with the standard, this also means that engineers have an easier time choosing the right level of abstraction, something that is otherwise challenging (Liebel et al., 2018b).

As with the heterogeneity across the case department, the level of abstraction relates to the product and is therefore independent of a specific scenario for the use of requirements models.

Importance of layout:. Finally, several of the interviewees raise the advantage of textual modelling languages like PlantUML, as they can be integrated into traditional text-based environments such as git or diff. However, they come at the cost of relying on automated layouting, e.g., through GraphViz[2] in the case of PlantUML. The layout of a graphical model can contain important information, and reflect the intent of the modeller (Störrle, 2011). Expressing this information is therefore no longer possible with automated layouting. Our interviewees have differing views as to whether this is a problem or not.

"I don't think that will be any problem. [..] It could be better like that. I see it, as long as I can show the flow, you have space to write, you could show the relation of the different parts, I think it should be okay". – Interviewee 5

"I think that's crap [automatic layout]. You must be able to...an automatic thing is good, but then you should click on a button 'I want an automatic suggestion'. And then you should be able to fine-tune it and it should stay that way. [..] Because the tool will never know what my intention was". – Interviewee 1

One interviewee suggested that, while automatic layout might not work for complex models, conventions or adjustments to automatic layout could be possible.

This last point clearly shows the complex trade-off between the simplicity of tools, and features that might be considered essential by some. Once again, layouting is largely independent from the chosen requirements modelling scenario. However, flexibility in choosing a layout might encourage more engineers to use the models.

6. Discussion and conclusion

We conducted a case study in one department at Ericsson AB, a large Swedish telecommunications provider, investigating the use of models for RE purposes. We conducted a survey with 33 participants, followed by 5 semi-structured interviews.

With respect to RQ1 (What sentiments exist for and against the use of requirements models in VLS agile systems engineering?), we find that our study participants consider requirements models useful and valuable. While several interviewees mentioned that sentiments against these models exist in the case department, we did not directly interact with anyone supporting this view. However, we also find that creating and maintaining requirements models at a sufficiently high level of quality is challenging. Several participants maintain that many existing models have deteriorated over time and are no longer useful. An additional point worth highlighting is that different modelling styles make it difficult to jointly work on models, something that might be harder to unify compared to, e.g., writing style

2 https://graphviz.org/.

in textual requirements. Finally, one interviewee mentioned that the nature of some changes to requirements models can require entire models to be re-drawn. This either leads to substantial overhead, or it causes resistance to make changes in the models, especially if changes are made by other engineers than the model creator.

The use of existing requirements models at the case department (RQ2, How do different stakeholders use requirements models in VLS agile systems engineering?) is primarily by SMs, in their role as providers of system-level knowledge and as a boundary between incoming change requests, system requirements, and work in individual agile teams. While several engineers in other roles use the models as well, mainly in a read-only fashion to answer their questions on intended system requirements, complex tooling used in the case department in the past has prevented a broader consumption of the models. Interviewees expressed the desire that all engineers should at least read the models. They further express confidence that their tool T-Reqs supports this, in which models are stored in textual format alongside code and textual requirements in git repositories. This allows easy access through tools engineers use on a daily basis, as well as easier review in terms of textual diff. This overall use of models is similar to what Zhang and Patel (2010) refer to as "live design documentation" and "continuous modelling".

Our findings with respect to RQ1 and RQ2 allow for reasoning about the information content and tooling needs (RQ3, What are the needs to support the intended use of requirements models in VLS agile systems engineering?) regarding requirements modelling. The value of the models at the case department is primarily in providing an overview of the system requirements and especially their connections, something that is difficult to express in a suitable manner in text. However, in order for the models to reach their full potential, the contained information needs to be up to date and of high quality. This, in turn, requires a broad access to the models by all stakeholders, at least in read-only fashion. Furthermore, education and guidance in how to create and maintain the models is essential, potentially also in the form of mentors at the case department. In terms of tool features, navigation and search are essential. Furthermore, interviewees expressed the desire to incorporate the requirements models in their regular code review workflow, e.g., by adding them to the git repository in textual form. Using generation of artefacts from models could be a way to further incentivise their use. However, no interviewee expressed the desire to generate substantial code from the models. We believe there are several reasons for these sentiments against code generation. First, the models currently express requirements. While these requirements are at a considerable level of technical detail, generating code would likely require an even lower level of abstraction and more design details. Given that models are already expensive to create and maintain, engineers could be cautious of adding more detail to the models. This level of detail is, we believe, also the key difference to other accounts of modelling in large-scale agile systems engineering that highlights automation as a key benefit of models, e.g., in Zhang and Patel (2010) and Lantz (2014). For example, Zhang and Patel (2010) refer to simulation capabilities. Similarly, Lantz (2014) refers to Simulink models used for code generation, and simulation, both of which require substantial technical details and, in many cases, custom code added to the models. Second, several of our interviewees mentioned bad experiences with modelling tools in the past. This could have shaped their perceptions regarding code generation and models at a detailed level of abstraction.

Using textual models is considered an advantage by most study participants, due to easier tooling and the possibility to integrate the models into existing tools. However, a number of issues arise due to the reliance on textual modelling, most notably the loss of manual layout capabilities at the case department. Interviewees suggested workarounds such as a standard layout, where the main flow is always displayed to the left, while alternative flows are drawn to the left of the main flow. Finally, we note the importance of supporting heterogeneity at the case department, with different needs and preferences in the agile teams.

Our study clearly shows the usefulness of models during RE, if used for well-motivated use cases. We outline four potential scenarios, i.e., (1) not using requirements models at all, (2) using requirements models as a tool for selected roles to answer questions and inform development teams, (3) using requirements models that are maintained by selected roles, but widely used in a read-only fashion, and (4) broad use, both in reading and writing, across the case department. Notably, Scenarios 3 and 4 require a substantially higher effort, e.g., in terms of training and reviews. However, if successful, requirements models could become a powerful information source and tool to handle change.

Furthermore, the study shows that simple modelling tools that are close to the engineers in terms of workflow and tooling have the potential to be successful, while heavy-weight modelling tools do not reach their full potential due to difficulties in accessing and using the tools, and resistance to do so regularly. Finally, we find several trade-offs that exist when tailoring models and modelling tools to an organisation, e.g., sophisticated modelling tool features such as hierarchy and manual layout vs. simple, text-based modelling tools.

CRediT authorship contribution statement

Grischa Liebel: Conceptualization, Methodology, Investigation, Formal analysis, Writing – original draft, Writing – review & editing, Visualization, Project administration. **Eric Knauss:** Conceptualization, Methodology, Validation, Investigation, Formal analysis, Writing – original draft, Writing – review & editing, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that has been used is confidential.

Acknowledgements

We would like to thank our contacts at Ericsson AB for the fruitful collaboration and constructive input at all stages of this work, and the study participants for their valuable contributions. Furthermore, we express our gratitude to João Araujo for feedback on the manuscript draft. Parts of this work were supported by Software Center Project 27 on RE for Large-Scale Agile System Development.

Appendix A. Questionnaire

This section shows the questionnaire we ran with the case department.

A.1. Page 1

Rationale:

In this questionnaire, we aim to investigate the current use of requirements models in [Case Department], as well as the needs for future use. The reason to do so is the planned replacement of IBM Rational Rhapsody. At the same time, this questionnaire is part of an academic research project at Chalmers & Gothenburg University in collaboration with [Case Company], investigating the use of models for requirements engineering in industry.

Target group:

We target all individuals at [Case Department] that come into contact with requirements models. This means both people who create models and those who access them.

Practicalities:

The survey consists of 20 questions and will take approximately 15 min to answer. Your answers are treated completely anonymous. The survey ends on [end date].

Thank you very much for your time!

If you have any questions, please contact the creators of this survey: Grischa Liebel (grischa@chalmers.se) or Eric Knauss (eric.knauss@cse.gu.se) for any research-related questions, and [Contact Person 1] or [Contact Person 2] for any [Case Company]-related questions.

A.2. Page 2

1. What is your main role?
   (Free text)
2. In which system area do you work?
   (Free text)
3. How long have you been working with [Product]?
   (Free text)
4. What are your main work tasks?
   (Options, multi-selection)
   - Line management
   - Requirements specification
   - Architecture definition
   - Design definition
   - Software implementation
   - Testing
   - Customer support
   - Security
   - Quality management
   - Process improvement
   - Organisation improvement
   - Other:

Models

In the following, we refer to "models" in the sense of UML or similar notations.

5. How much experience do you have in creating models (in years)?
   (Mandatory, free text)
6. How often do you on average create/modify models of requirements?
   (Mandatory, single-selection: Never, Yearly, Monthly, Weekly, Daily)
7. How often do you on average read/use models of requirements without modifying them?
   (Mandatory, single-selection: Never, Yearly, Monthly, Weekly, Daily)
8. How often do you on average access (modify/read) models of requirements created outside your team?
   (Mandatory, single-selection: Never, Yearly, Monthly, Weekly, Daily)
9. Do you personally use IBM Rational Rhapsody for creating/reading models of requirements?
   (Mandatory, Yes/No)
10. Do you have any additional comments on this page?
    (Optional, free text)

A.3. Page 3

11. Which of the following statements correspond to your understanding of the terms diagram/model?
    (Mandatory, Multi-selection)
    - There is no difference between a diagram and a model.
    - A model contains all entities and relations, while a diagram is a (partial)
    - visualisation of this model.
    - Multiple diagrams can exist for the same model.
    - A diagram always has a graphical representation, while a model may not.
    - Other: **(Free text)**
12. Are you in general positive or negative towards the use of models to express requirements? (Note that this question is not used to rate your answer. Instead we aim to find out what the general view of models is in the organisation.)
    (Mandatory, single-selection: Positive, Negative, Neutral/I don't know)
13. Which diagram types do you use (creation or usage) for expressing requirements? **(Optional, Multi-selection, with an example diagram for each diagram type)**
    - Class diagrams
    - Component diagrams
    - Activity diagrams
    - Sequence diagrams
    - State Machine diagrams
    - Use Case diagrams
    - Others (Free text)
14. Please rate your agreement to the following statements.
    (Mandatory, 5-point Likert scale from Strongly Disagree to Strongly Agree, plus an "I don't know" option)
    - The layout of a diagram is important to me.
    - The existing requirements models do not carry important information.
    - The access to requirements models should be integrated with existing development tools.
    - The access to requirements models should be integrated with T-Reqs. Requirements models should be used for automated tasks (e.g., verification, artefact generation).
    - Requirements models should only be used for documentation.
    - The current modelling tool is restricting me in my work.
    - I would update/maintain the requirements models more frequently if I had a better tool.
    - Navigation between diagrams is an important feature.
15. Do you have any additional comments on this page?
    (Optional, free text)

Additionally, we displayed the following two questions iff a participant stated in question 6 (model creation) that he/she created/used models at least monthly:

Table B.3
Evaluation of propositions.

Prop Nr	Used questions
P1	Q1, Q4, Q6, Q7, Q8, Q9, Q17
P2	Q1, Q4, Q6, Q7, Q8, Q9, Q17
P3	Q5, Q6 + free text
P4	Q1, Q4, Q6, Q7, Q8, Q12, Q14.2.7
P5	Q13 + free text
P6	Q1, Q4, Q14.2
P7	Q14.1
P8	Q14.3.4
P9	Q14.5
P10	Q14.6.7, Q12, Q9
P11	Q14.7
P12	Q14.8

16. For what purposes do you create models of requirements? **(Optional, free text)**
17. Who is looking at the models of requirements you create? **(Optional, Multi-selection)**
    - Myself
    - Other developers in the same team
    - Other teams
    - Product owner for my team
    - Product owners of other teams
    - CPI writers
    - System managers
    - Customer support
    - Others:

A.4. Page 4

18. What features/properties does a modelling tool need to offer?
(Optional, free text)
19. Currently, keeping the models/diagrams of requirements up to date is challenging. Do you have any suggestions how engineers could be motivated to maintain these models/diagrams more frequently?
(Optional, free text)
20. How would the ideal situation regarding requirements modelling look like in the future?
(Optional, free text)
21. Do you have any other comments (e.g., alternative ideas to modelling, tool suggestion)?
(Optional, free text)

Additionally, we displayed the following question iff a participant stated "yes" in question 9 (use of IBM Rhapsody):

22. Which features in IBM Rational Rhapsody are important to you?
(Optional, free text)

A.5. Page 5

Thank you for completing this questionnaire!
We would like to thank you very much for helping us.
Your answers were transmitted, you may close the browser window or tab now.

Appendix B. Proposition evaluation

See Table B.3.

Appendix C. Interview propositions and questions

In the following, we discuss the updated propositions and open questions guiding the interview design. The open questions relate in particular to contradictions in the survey data. Propositions are accompanied with *How*-questions, as we aim to understand their outcome in detail, not simply corroborate them or not.

C.1. Propositions

- The current ad-hoc use of models is insufficient. Either modelling should be abandoned, or a clear process (with clear stakeholders, tasks and abstraction levels) and guidelines (including courses on modelling) are needed. (*How could such a process look like?*)
- Information in models is outdated in many areas. This needs a centralised effort to be fixed, replacing the tool does only treat the symptoms. (*How could the case company proceed? How do first steps look like?*)
- A number of stakeholders/tasks have been forgotten when considering the use of models and the tool integration. (*Who are these stakeholders? What are their needs?*)
- Potential users need a clearer motivation for using (and in particular updating) models. (*How could we motivate them?*)
- A lightweight modelling approach is sufficient for the case company. They require only very few model elements of activity diagrams (activity nodes with text, relations between them) and few model capabilities. (*Which of the features of modern modelling tools are still required?*)

C.2. Questions

- Using PlantUML allows only automated layouting. How do stakeholders view this trade-off between text-based integration into T-Reqs, and losing capability to modify the layout? How does the simple approach relate to other modelling capabilities?
- Automation using existing models was supported in the survey. How should this automation look like? What aspects could/should be automated?
- What kind of information is needed in the models? What is a suitable level of abstraction?
- Working with models is currently cumbersome. How can the experience be improved? What does it mean to be easy to use for a modelling tool?
- Information needs clearly differ between stakeholders. What information needs exist for specific stakeholders?

Appendix D. A-priori codebook

- role, experience (demographic codes)
- modellingProcess, stakeholdersOfModels, informationContent, motivationForModels, trade-off, abstraction, modelAutomation, toolFeatures

References

3rd Generation Partnership Project (3GPP), 2021. Technical Specification Group Services and System Aspects; Release 16 Description; Summary of Rel-16 Work Items (Release 16). Tech. Rep. 21.916.
Alfraihi, H.A.A., Lano, K.C., 2017. The integration of agile development and model driven development: A systematic literature review. In: The 5th International Confrence on Model-Driven Engineeing and Software Development.
Ambler, S.W., 2001. Agile modeling: A brief overview. In: Practical UML-Based Rigorous Development Methods-Countering Or Integrating the Extremists, Workshop of the PUML-Group Held Together with the UML 2001. Gesellschaft für Informatik e. V..

Baker, P., Loh, S., Weil, F., 2005. Model-driven engineering in a large industrial context - motorola case study. In: Briand, L.C., Williams, C. (Eds.), Model Driven Engineering Languages and Systems. In: Lecture Notes in Computer Science, vol. 3713, pp. 476–491.

Beck, K., 1999. Extreme Programming Explained: Embrace Change. Addison-Wesley.

Berenbach, B., Schneider, F., Naughton, H., 2012. The use of a requirements modeling language for industrial applications. In: 20th IEEE International Requirements Engineering Conference. RE, pp. 285–290.

Berger, C., Eklund, U., 2015. Expectations and challenges from scaling agile in mechatronics-driven companies – a comparative case study. In: Lassenius, C., Dingsøyr, T., Paasivaara, M. (Eds.), Agile Processes in Software Engineering and Extreme Programming. Springer International Publishing, Cham, pp. 15–26.

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 the 1st Workshop on Agile Requirements Engineering, AREW '11. Association for Computing Machinery, New York, NY, USA, pp. 1–5. http://dx.doi.org/10.1145/2068783.2068786.

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.

Brandstetter, V., Froese, A., Tenbergen, B., Vogelsang, A., Wehrstedt, J.C., Weyer, T., 2015. Early validation of automation plant control software using simulation based on assumption modeling and validation use cases. Comp. Syst. Inform. Model. Q. (4), 50–65.

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.

Brings, J., Bellendorf, J., Keller, K., Kempe, M., Kurt, N., Palm, A., Daun, M., 2017. Applying the spes modeling framework. In: Joint Proceedings of REFSQ-2017 Workshops, Doctoral Symposium, Research Method Track, and Poster Track Co-Located with the 22nd International Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2017).

Dikert, K., Paasivaara, M., Lassenius, C., 2016a. Challenges and success factors for large-scale agile transformations: A systematic literature review. J. Syst. Softw. 119, 87–108. http://dx.doi.org/10.1016/j.jss.2016.06.013.

Dikert, K., Paasivaara, M., Lassenius, C., 2016b. Challenges and success factors for large-scale agile transformations: A systematic literature review. J. Syst. Softw. 119, 87–108.

Dingsøyr, T., Fægri, T.E., Itkonen, J., 2014. What is large in large-scale? A taxonomy of scale for agile software development. In: International Conference on Product-Focused Software Process Improvement. Springer, pp. 273–276.

Eklund, U., Holmström Olsson, H., Strøm, N.J., 2014. Industrial challenges of scaling agile in mass-produced embedded systems. In: Dingsøyr, T., Moe, N.B., Tonelli, R., Counsell, S., Gencel, C., Petersen, K. (Eds.), Agile Methods. Large-Scale Development, Refactoring, Testing, and Estimation. Springer International Publishing, Cham, pp. 30–42.

Fockel, M., Holtmann, J., 2014. A requirements engineering methodology combining models and controlled natural language. In: IEEE 4th International Model-Driven Requirements Engineering Workshop (MoDRE '14). pp. 67–76.

Gren, L., Lenberg, P., 2020. Agility is responsiveness to change: An essential definition. In: Proceedings of the Evaluation and Assessment in Software Engineering. pp. 348–353.

Hansson, S., Zhao, Y., Burden, H., How mad are we? Empirical evidence for model-driven agile development. In XM 2014–Extreme Modeling Workshop.

Heikkilä, V.T., 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. pp. 199–207. http://dx.doi.org/10.1109/SEAA.2015.70.

Heikkilä, V.T., 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, 2892–2936. http://dx.doi.org/10.1007/s10664-016-9491-z.

Hutchinson, J., Rouncefield, M., Whittle, J., 2011a. Model-driven engineering practices in industry. In: 33rd International Conference on Software Engineering (ICSE '11). pp. 633–642.

Hutchinson, J., Whittle, J., Rouncefield, M., 2014. Model-driven engineering practices in industry: Social, organizational and managerial factors that lead to success or failure. Sci. Comput. Program. 89 (Part B), 144–161, SI: Success Stories in Model Driven Engineering.

Hutchinson, J., Whittle, J., Rouncefield, M., Kristoffersen, S., 2011b. Empirical assessment of MDE in industry. In: 33rd International Conference on Software Engineering (ICSE '11). pp. 471–480.

Inayat, I., Salim, S.S., Marczak, S., Daneva, M., Shamshirband, S., 2015. A systematic literature review on agile requirements engineering practices and challenges. Comput. Hum. Behav. 51, 915–929, Computing for Human Learning, Behaviour and Collaboration in the Social and Mobile Networks Era. http://dx.doi.org/10.1016/j.chb.2014.10.046.

Kahkonen, T., 2004. Agile methods for large organizations-building communities of practice. In: Agile Development Conference. IEEE, pp. 2–10.

Kasauli, R., Knauss, E., Horkoff, J., Liebel, G., de Oliveira Neto, F.G., 2021. Requirements engineering challenges and practices in large-scale agile system development. J. Syst. Softw. 172, 110851.

Kasauli, R., Knauss, E., Kanagwa, B., Nilsson, A., Calikli, G., 2018. Safety-critical systems and agile development: a mapping study. In: 2018 44th Euromicro Conference on Software Engineering and Advanced Applications. SEAA, IEEE, pp. 470–477.

Kasauli, R., Knauss, E., Nakatumba-Nabende, J., Kanagwa, B., 2020. Agile islands in a waterfall environment: Requirements engineering challenges and strategies in automotive. In: Proceedings of International Conference on Evaluation and Assessment in Software Engineering (EASE), Trondheim, Norway. pp. 31–40. http://dx.doi.org/10.1145/3383219.3383223, https://arxiv.org/abs/2106.11187.

Kasauli, R., Liebel, G., Knauss, E., Gopakumar, S., Kanagwa, B., 2017. Requirements engineering challenges in large-scale agile system development. In: 2017 IEEE 25th International Requirements Engineering Conference. RE, IEEE, pp. 352–361.

Knauss, E., Liebel, G., Horkoff, J., Wohlrab, R., Kasauli, R., Lange, F., Gildert, P., 2018. T-reqs: Tool support for managing requirements in large-scale agile system development. In: 2018 IEEE 26th International Requirements Engineering Conference. RE, IEEE, pp. 502–503.

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: 2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement. pp. 348–356. http://dx.doi.org/10.1109/ESEM.2013.53.

Lantz, J., 2014. Using models to scale agile mechatronics development in cars: case studies at volvo car group. In: Proceedings of the 18th International Software Product Line Conference-Volume 1. p. 20.

Leffingwell, D., 2010. Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise. Addison-Wesley Professional.

Liebel, G., Marko, N., Tichy, M., Leitner, A., Hansson, J., 2018a. Model-based engineering in the embedded systems domain: an industrial survey on the state-of-practice. Softw. Syst. Model. 17 (1), 91–113. http://dx.doi.org/10.1007/s10270-016-0523-3.

Liebel, G., Tichy, M., Knauss, E., 2018b. Use, potential, and showstoppers of models in automotive requirements engineering. Softw. Syst. Model. http://dx.doi.org/10.1007/s10270-018-0683-4.

Loniewski, G., Insfran, E., Abrahão, S., 2010. A systematic review of the use of requirements engineering techniques in model-driven development. In: International Conference on Model Driven Engineering Languages and Systems. Springer, pp. 213–227.

Meyer, B., 2014. Agile! Bertrand MeyerThe Good, the Hype and the Ugl. Springer.

Mohagheghi, P., Gilani, W., Stefanescu, A., Fernandez, M.A., Nordmoen, B., Fritzsche, M., 2013. Where does model-driven engineering help? experiences from three industrial cases. Softw. Syst. Model. 12 (3), 619–639.

Nakićenović, M.B., 2012. An agile driven architecture modernization to a model-driven development solution. Int. J. Adv. Softw. 5 (3, 4).

Paetsch, F., Eberlein, A., Maurer, F., 2003. Requirements engineering and agile software development. In: WET ICE 2003. Proceedings. 12th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Vol. 2003. pp. 308–313. http://dx.doi.org/10.1109/ENABL.2003.1231428.

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.

Petersen, K., Gencel, C., 2013. Worldviews, research methods, and their relationship to validity in empirical software engineering research. In: 2013 Joint Conference of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement. IEEE, pp. 81–89.

Pohl, K., Hönninger, H., Achatz, R., Broy, M., 2012. Model-Based Engineering of Embedded Systems: The SPES 2020 Methodology. Springer Science & Business Media.

Ramesh, B., Cao, L., Baskerville, R., 2010. Agile requirements engineering practices and challenges: an empirical study. Inf. Syst. J. 20 (5), 449–480. http://dx.doi.org/10.1111/j.1365-2575.2007.00259.x.

Rumpe, B., 2002. Agile modeling with the uml. In: International Workshop on Radical Innovations of Software and Systems Engineering in the Future. Springer, pp. 297–309.

Saldaña, J., 2015. The Coding Manual for Qualitative Researchers. Sage.

Salo, O., Abrahamsson, P., 2008a. Agile methods in European embedded software development organisations: a survey on the actual use and usefulness of extreme programming and scrum. IET Softw. 2 (6), 58–64.

Salo, O., Abrahamsson, P., 2008b. 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.

Selic, B., 1998. Using uml for modeling complex real-time systems. In: Languages, Compilers, and Tools for Embedded Systems. Springer, pp. 250–260.

Störrle, H., 2011. On the impact of layout quality to understanding uml diagrams. In: 2011 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC). IEEE, pp. 135–142.

Vogelsang, A., Eder, S., Hackenberg, G., Junker, M., Teufl, S., 2014. Supporting concurrent development of requirements and architecture: A model-based approach. In: Model-Driven Engineering and Software Development (MODELSWARD), 2014 2nd International Conference on. IEEE, pp. 587–595.

Whittle, J., Hutchinson, J., Rouncefield, M., Burden, H., Heldal, R., 2013. Industrial adoption of model-driven engineering: Are the tools really the problem?. In: Moreira, A., Schätz, B., Gray, J., Vallecillo, A., Clarke, P. (Eds.), Model-Driven Engineering Languages and Systems. In: Lecture Notes in Computer Science, vol. 8107, pp. 1–17.

Zhang, Y., Patel, S., 2010. Agile model-driven development in practice. IEEE Softw. 28 (2), 84–91.

Grischa Liebel is an Assistant Professor at Reykjavik University, Iceland. He is the director of the Center for Research on Engineering Software Systems (CRESS) at Reykjavik University. Grischa's research focuses on human values in software engineering, typically with emphasis on requirements, models, and development processes. He holds a Ph.D. from Chalmers University of Technology, Sweden. Contact him at grischal@ru.is.

Eric Knauss is an Associate Professor at Chalmers | University of Gothenburg, Sweden. His research focuses on managing requirements and related knowledge in large-scale and distributed software projects. He holds a Ph.D. from Leibniz Universität Hannover, Germany. He is member of program and organisation committees of top conferences and reviewer for top journals. Contact him at eric.knauss@cse.gu.se.