The Need of Complementing Plan-Driven Requirements Engineering with Emerging Communication: Experiences from Volvo Car Group

Ulf Eliasson*, Rogardt Heldal†, Eric Knauss†, Patrizio Pelliccione†
*Volvo Car Group, Sweden
ulf.eliasson@volvocars.com
†Chalmers University of Technology | University of Gothenburg
Department of Computer Science and Engineering, Sweden
heldal@chalmers.se, {eric.knauss, patrizio.pelliccione}@gu.se

***Abstract*—The automotive industry is currently going through an enormous change, transitioning from being pure hardware and mechanical companies to becoming more software focused. Currently, software development is embedded into a V-Model process that defines how software requirements are extracted from system requirements. In recent years, OEMs have come to recognize the importance and opportunities offered by software, which include better management and shorter time-to-market of distinguishing features. Strategies to better utilize software include in-house software development and new ways to collaborate with suppliers. However, in their effort to take advantage of these opportunities, engineers struggle with the formal process imposed on software development. In this paper, we investigate the impact of this struggle on the flow of requirements, including challenges and practices. We found that new ways of working with requirements had emerged that are partly not supported, partly hindered by the old tooling and processes for requirements engineering. Requirements flow both vertical and horizontal in the organization and across the supply-chain. Support for the new way of working should allow us to refine requirements iteratively throughout their life-cycle, handle the discussion of rationales, and to manage assumptions. We found strategies of achieving this to differ not only between OEMs, but also between different divisions inside the OEMs.**

***Index Terms*—beyond plan-driven requirements engineering, emerging communication, automotive requirements engineering**

I. Introduction

The automotive industry is currently making a big change. Traditionally, automotive Original Equipment Manufacturers (OEMs) have been dealing with hardware and mechanical components. Software was first introduced for engine control optimization but has since then spread. Electrical and hybrid vehicles are becoming more important as new, stricter, laws and regulations on emission are introduced, and the public awareness of environmental issues is growing. These new technologies introduce new software intense systems such as batteries, chargers, and electrical engines. Software in active safety systems and, in the near future, autonomous vehicles is combining input from an increasing number of sensors and are making decisions and perform actions that previously were left only for humans. A modern hybrid electrical car can contain up to about 100 Electrical Control Units (ECUs), small to PC-sized computers, connected and communicating over several different types of network technologies while executing gigabytes of software.

Software within a car introduces opportunities. By combining input from existing sensors in new ways and controlling existing actuators, new functions can be introduced significantly faster than if they would need new hardware and mechanical components to be developed as well. This way of developing new functions can cut time to market drastically. To utilize the benefits of software better and keep important domain knowledge within the company, OEMs have started to do in-house software development of key features that should distinguish them from their competitors. At the same time, other functions and the majority of the hardware and mechanical components are still ordered from suppliers. However, the old way of working with the suppliers is nearing its limits as supplier interaction must keep up with the increasing speed within the OEMs.

The automotive OEMs see a need to constantly increase their speed of innovation and to shorten their time-to-market to satisfy new customer expectations. The formal requirement engineering process has not yet been fully adapted to support the faster pace. This has led to engineers embracing new practices, outside the scope of the current requirement engineering process.

In this paper, we investigate challenges emerging at a large automotive company, Volvo Car Group (VCG), within requirements engineering when the need for speed within their development projects increases. We formulated that in our first research question:

> *RQ1: How do requirements flow between stakeholders of automotive (software) engineering?*

The second research question is to understand if there is a need for shorter feedback loops for requirements validation, another important aspect when other parts of the development projects increase its speed.

> *RQ2: Is there a need for shorter feedback loops for requirements validation?*

978-1-4673-6905-3/15/$31.00 © 2015 IEEE

RE 2015, Ottawa, ON, Canada
Industry Paper

The background section, Section II describes works that are related to this paper and provide background information about the automotive domain and specifically VCG. To provide an answer to our research questions, we applied a qualitative research methodology based on semi-structured interviews, as detailed in Section III. Our findings are described in Section IV. The paper concludes with final remarks and future work directions in Section V.

II. Background

A. Communication in Requirements Engineering and Agile Requirements Engineering

There is a growing body of research that studies communication in requirements engineering. While most papers identify problems in RE communication [4] or model collaboration between stakeholders in requirements-centric teams (e.g. [8]), only a few investigate the actual instances and *content* of communication between stakeholders. Studies by Damian and colleagues [8] found that clarification of requirements and communication of changes is the most predominant topics of discussions. Calefato et al. [7] examined the content of requirements communication to compare the level of common ground achieved in computer-supported elicitation vs. negotiation and used certain heuristics to identify clarification communication. Another recent study in agile teams [3] observed verbal communication about requirements in co-located agile teams. The study revealed a number of communication practices related to memory, communication, and learning in a process of shared conceptualization and which supported the team in its RE activities. In this paper, we found that similar practices are used in VCG in order to manage the lack of information or to resolve ambiguities, as well as to have early feedback on defined requirements.

According Inayat et al. [12], agile requirements engineering denotes the "agile way" of planning, executing and reasoning about requirements engineering activities. The authors report a systematic review to identify the adopted requirements engineering practices as well as to understand how agile requirements engineering can solve traditional requirements engineering issues. The findings of this work suggest that agile requirements engineering needs more empirical results to better understand the impact of agile requirements engineering practices. With our work, we want to contribute to that by reporting the experiences and challenges of a large automotive company while transiting from traditional requirements engineering to lean and agile requirements engineering.

B. Automotive Domain

Historically, software was introduced in cars to optimize the control of the engines. Since then the growth of software within the car has been exponential for each year and today not a single function is performed without the involvement of software. According to industry experts, 80% to 90% of the innovation within the automotive industry is based on electronics, as mentioned for instance by Peter van Staa - Vice-President Engineering of Robert Bosch GmbH at the European Technology Congress in June 2014[1]. A big part of electronics is software. Today we can see that the most advanced cars have more/comparable software than fighter airplanes[2]. Cars are also produced in higher volumes than aircraft, which put harder requirements on the technology cost.

Engineers and researchers find new ways of combining the input of existing sensors and new ways of controlling actuators to optimize behavior or introduce completely new functionality. When customers today buy cars, they do not want to pick a mass produced product that is one size fits all that just takes them from A to B. Instead, they want a customized modern transportation system that caters to their wishes and needs specifically. Pushing new features and cool technology is more and more important for the car manufacturers. Moreover, they need to do this before or at least simultaneous as their competitors, otherwise they will fall behind, and customers will move on. This is in line with the general view of manufacturing of the future defined by the Industrie 4.0 Working Group of Germany [13], the Swedish strategic innovation [21], and by the EFFRA European Factories of the Future Research Association [10] that envision a shift towards highly specialized production systems that must cope with a potentially infinite product variations, a high level of customization, and an increasing demand for individualized products.

Another challenge is to meet new laws and regulations when it comes to emission. This has forced manufacturers to seriously consider and release electric and hybrid electric vehicles. This introduced new kinds of software intense systems in the car for controlling electrical engines, batteries, and charging.

Traditionally, a platform for cars has had a long life, 10 to 20 years. While manufacturing cars on the current platform a new was developed and major changes were introduced when launching the new platform. To keep up with the rapid development in other areas, this is not longer feasible. Again this is a trend shared with manufacturing in general [13], [21], [10]. At the same time, many customers expect their car to last and be safe for many years after they acquired it: to buy a new modern car is for many customers a major investment. They expect this without having to care about updates of the car with the latest software, which is common practice in other domains such as with phones.

C. Volvo Car Group

Volvo Car Group is an automotive manufacturer founded in and operating from Gothenburg, Sweden. From the start, their focus has been on safety and quality. Volvo's car development has a varied history with several owners, from the initial Swedish AB Volvo, through being owned by Ford and finally as part of the Chinese Zhejiang Geely Holding Group. The final transition has been roughly aligned with the development of a new platform, thus making the platform independent

[1] http://www.etcwroclaw.eu/files/presentations/peter_van_staa.pdf

[2] As said by Alfred Katzenbach, the director of information technology management at Daimler: http://spectrum.ieee.org/transportation/systems/this-car-runs-on-code

of other OEMs. The new platform has less optimization for usage outside the main brand but is nevertheless more advanced since it is prepared for large-scale hybridization, active safety, and other future functions. Within Volvo Car Group, the experiences reported in this paper are from groups involved in electronics and electric/hybrid propulsion development. However, in such a large organization as the VCG R&D organization experiences and, in certain cases, ways of working and adapted practices varies.

D. Requirement Structure at Volvo Car Group

At VCG, the formal requirement structure is hierarchical. The requirements authors work in requirement management tools. From these tools the formal requirement artifacts on different levels are generated, as shown in Figure 1. The generated documents are used for the formal communication with suppliers, and they are stored for the future to fulfill law, safety and security regulations. In-house engineers have access to the tools and can read the requirements they need to perform their work directly as they are written in these tools. Several roles are involved on different levels, and these are shown in Table I together with roughly their responsibilities in the requirement engineering process. Below we have a high-level description of how the formal requirement structure looks.

For functional requirements, they start as descriptions of what VCG call a car function. These are functions that have customer impact. The function description is then broken down into requirements by a function realizer. They often express their requirements in the form of use cases or scenarios describing the interactions of a customer with the car. A possible scenario might describe the functionality of opening a car with the remote key: *let us assume that a customer is approaching the car, then by pressing the unlock button on his remote key the doors should unlock; finally, he opens a door, sits in the car and close the door*.

These requirements on a function level are then read by a sub-system responsible. In this case, the sub-system that is responsible for the design of the functionality is *locking*. A sub-system might span several ECUs, from different parts of the organization. The sub-system responsible breaks down functional requirements to more detailed functional and non-functional requirements as part of his design of the sub-system. This is often done in cooperation with the software and hardware responsible for the affected nodes. Sub-system requirements are documented in a sub-system wide document, and then further broken down into software and hardware requirements, by the software responsible and hardware responsible for the component. The hardware and software requirements go into separate documents. In the case of supplier implemented nodes, these documents are then sent to the supplier doing the implementation. In the case of in-house development, which is just software, requirements can be managed by the tools used for writing and structuring requirements. However, often parts of the software and the hardware is developed by suppliers even in the cases where the component groups have in-house software development.

TABLE I
SEVERAL DIFFERENT ROLES ARE INVOLVED IN THE REQUIREMENT ENGINEERING PROCESS AT VCG. HERE THE KEY ROLES INVOLVED ARE LISTED WITH A SHORT DESCRIPTION OF THEIR RESPONSIBILITIES.

Role	*Responsibilities*
Function owner	Is the owner of one or several car functions and produces an FRD for the function.
Function realizer	Breaks down the function from the FRD to a slightly more detailed description of how the function should be realized and what sub-systems that should be involved and this is communicated by so-called FRXs.
Sub-system responsible	With the FRXs as input the sub-system responsible should produce an SRD that describes the high-level requirement and design of the sub-system.
Software Responsible	With the SRD as input the software responsible produce a detailed design of the software and also write detailed requirements, produces a document called SWRS.
Hardware Responsible	With the SRD as input the hardware responsible produces a hardware specification with requirements called DPR.
In-house development team	Formally they are seen as the same as a supplier, but sometimes the software and/or hardware responsible is part of the team. They often do not use the SWRS. Instead, they check the requirements from the requirement management systems directly. In-house development is only software development.
Supplier development team	Receives the SWRS and DPR and should deliver an implementation, according to the requirements in these documents.
Tester	Depending on what level the tester is responsible for the tester use the document from their level as input, write a test & verification plan that describe the verification steps to cover the requirement, and then perform verification/validation and report the results.
Safety Engineers	Reviews and audits requirements and design. Can help other requirements authors to formulate safety critical requirements but have no formal role as a requirement author.
Attribute Leader	They handle attributes on car level, e.g. electrical magnetic compatibility (EMC) and write requirements on sub-systems that are affected by the requirements.
System Architect	The system architects are architects responsible for the architecture of the complete electrical system. They write requirements on sub-systems to make them comply with their architectural strategies and rules.

On one side software requirement specifications are needed for formal reasons, i.e. to comply with internal regulations and processes, on the other side they are also used by the group responsible for the final verification and validation.

In parallel with the functional requirements, there are also other kinds of requirements, such as architectural and so-called attribute requirements. Attribute requirements are not visible to the user. They can be things like isolation of the high-voltage system; it is important to make sure that the high-voltage system is safe throughout the lifetime of a hybrid vehicle. This type of requirements usually goes from attribute leaders or system architects to the sub-systems that are affected by them.

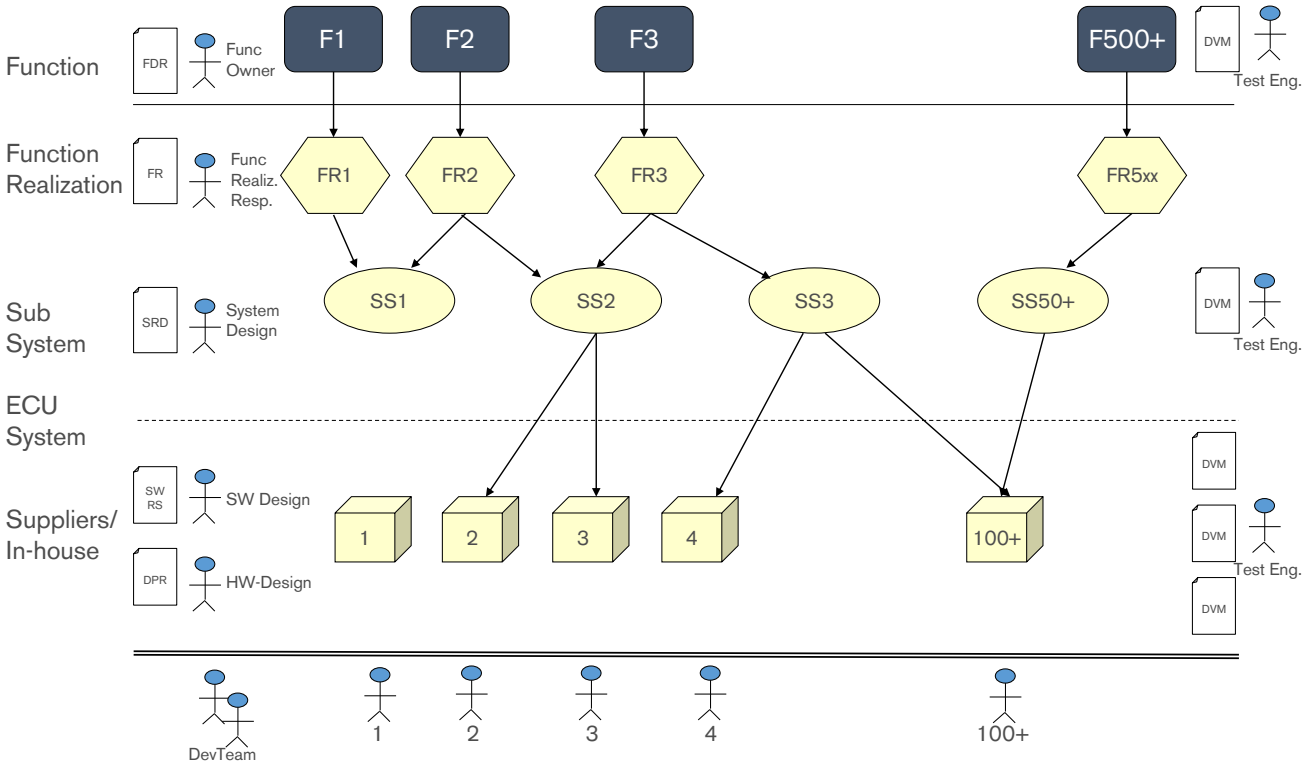

Fig. 1. The (Software) Requirements Hierarchy at VCG, here the version of the Electrical Division. (Adapted from presentation by Kent Niesel, VCG, https://seecgot.files.wordpress.com/2014/06/swc-reqs-test-v001.pptx)

III. Research Method

In our research, we aimed at exploring how information flows within automotive software development. For this goal, we applied a qualitative research methodology, based on semi-structured interviews for data collection and thematic coding for analysis.

To obtain a broad view of the stakeholders in automotive software development and also in-depth knowledge of the development processes and practices within VCG, we relied on discussions within the research team. In particular the first author, who works as part of a team responsible for software process development and the continuous integration tool-chain that is used within the electric and hybrid vehicle group at VCG. He is also currently undertaking his Ph.D. as an industrial Ph.D. candidate in software engineering. This helped us to clarify work practices through his direct involvement in solving technical issues but also through participation in project meetings. This in-depth involvement with the development team within one division within the case company was invaluable in selecting interviewees for the next phase in our data collection.

We conducted a total of 15 semi-structured interviews that fall into four groups. First, we started with 8 more general interviews on information flow in automotive software development. During these interviews, we learned that the management of requirements was one of the main concerns of our interviewees. We then followed up with 3 more interviews that were more focused on the area of requirements engineering. In parallel, we conducted 2 general and 2 requirements-focused interviews with a second automotive OEM, Volvo Group Truck Technology (VGTT). Our cross-functional research team of academic and industrial researchers, as well as practitioners, allowed us to define a suitable interview guide and to get in contact with knowledgeable stakeholders throughout the various abstraction levels of system development. We selected interviewees on different levels, including two software developers, three sub-system responsibles, one function developer, three system architects, one domain architect, one software responsible, one process engineer, one component owner, two technical specialist. The experience of our interviewees in their role ranged from 6 months to 15 years, with an average of 2-3 years. Our interviews were based on a semi-structured interview guide[3] that covered our research questions, but allowed us to pursue related topics our interviewees brought up [18].

The first author conducted the interviews. He benefited from his familiarity with the case company and, with one exception, where we had to rely on meeting minutes, all interviews were recorded. Based on this data, all authors iteratively analyzed the interview data based on the thematic analysis approach [6] as follows: In order to familiarize ourselves with the data, we had at least two researchers listen to each recording. The 8+2

[3]Examples of guiding questions: What is the input for your work? Whom do you give requirements? How do requirements evolve as the project progress?

general interviews at VCG and VGTT were then summarized by at least two researchers in parallel and the summaries were compared and discussed by all authors. The 3+2 requirements engineering specific interviews were completely transcribed (to a total of 46018 words). One of the co-authors then coded the data from the perspective of the research questions and started searching for initial themes by grouping the initial codes. In a series of workshops, we then reviewed the initial themes, regrouped and refined them by cross-checking the interview data with the generated codes and finally established a set of themes, consisting of both challenges and practices. Based on this we report the identified challenges and practices with respect to the flow of requirements in automotive engineering at VCG and in the context of the underlying process.

IV. Findings and Recommendations

The R&D department at VCG aims at drastically reducing the time-to-market. Based on our data collection and analysis, we list nine related challenges for requirements engineers at VCG, together with practices that engineers currently use to tackle these challenges. Based on our knowledge of the field and the case at hand, we then discuss our recommendations for mitigating each challenge. In Table II we summarize our findings that are reported on in detail in this section.

A. How do Requirements Flow Between Stakeholders of Automotive (Software) Engineering? (RQ1)

During its life-cycle, a requirement is influenced by several stakeholders within VCG (see Table I) which we covered to a large degree in our interviews. When asked about the flow of requirements, our interviewees generally started to describe a top-down flow[4]. *Sources of requirements* for this top-down flow include standards, regulations, and laws, in particular with respect to safety.

1) Balancing Under-Specification and Over-Specification of Requirements (Finding 1.1): We observed two and apparently contrasting opinions: some interviewees feel that requirements are under-specified, other that they are over-specified. It emerges then the need and challenge of identifying requirements at the right abstraction level.

Under-specified requirements: Often these requirements are abstract, incomplete and under-specified since they mostly focus on one or a few specific scenarios. A scenario is by definition describing only one or few possible behaviors; thus scenarios need to be generalized to take different facets of reality into account.

> *[On activities that might decrease the number of misunderstandings] But, I think, what would increase the quality and decrease the number of misunderstandings and gaps even more would be, if we as input requirements got functional-requirement instead of use cases. Because a use case is very limited, it describes, a chain of event, what should happen. And then another chain of events. But there exist a lot of other subsets of all these steps in the chain, which have no requirements stated.*
> *— Sub-system responsible*

> *But then we begin to think of variants of that use case: Yes, but what if the user, for example, close the door first? Something that is not described in the use case. Then we go back to the function owner and ask. Then maybe he answers "I haven't thought about that scenario", and maybe he shouldn't have.*
> *— Sub-system responsible*

Over-specified requirements: Often requirement authors feel that they are forced to over-specify and make requirements more detailed than they feel comfortable with. Sometimes they do not care about the exact implementation and sometimes they do not have the knowledge needed to write the requirement on such a detailed level. Especially early in the projects, a lot of assumptions are made.

In some cases, the person writing the high-level functional requirement might not have the detailed knowledge to actually specify the behavior or requirement for all situations. In these situations, the receivers of the requirement have better knowledge and might be the one that is in the best position to decide on what is needed to fill the gaps. This is also a point that has been stressed by our interviewees: in many cases a requirement should give enough freedom for the receiver to construct a good solution, specifically on how it should be detailed and realized.

> *[the component teams] have a lot of opinions about that they would like not to do exactly like [the requirement says], they want to do it a little bit different. And [the system group] do not have any opinions on that. If a signal should be called this or that, for us it is the same. Rather, we would have liked to just say: When this error occurs the battery should act like this e.g. open the contractors [...]. Then they can decide, what does it mean to open the contractors, how fast do they want to do it [...], maybe they already have a general way of handling it. [...] The problem is that our requirement goes directly into the software requirement specification [...] that they send to their supplier. So then it needs to be at a certain level of detail, considering that there will be no further refinement of the requirement*
> *— Sub-system responsible*

Our interviewees would prefer if the receiver could refine the requirements further if needed, since the receiver has the knowledge necessary, such as details about the component structure, the exact behavior of the hardware, and the surrounding components. However, it appears that then a dialog with the sender of the requirement would be appreciated.

a) Practice(s): In order to balance the trade-off between ***under-specifying and over-specifying requirements***, the common practice today is to exploit the ***personal network*** to get input from known and knowledgeable people, such as authors

[4]We describe the direction of requirement flows with respect to the organizational hierarchy.

TABLE II
IDENTIFIED CHALLENGES, CURRENT PRACTICES TO HANDLE THEM AND OUR RECOMMENDATIONS.

Challenge	*Practices*	*Recommendations*
F1.1: Balancing under-specification and over-specification of requirements	• Personal network • Oral communication • Assumptions	• Networking • Infrastructure for communication and feedback
F1.2: Synchronizing cross-function and cross-system requirements	• Personal network	• Continuous integration and deployment
F1.3: The cost for changing requirements increases over time	• Workarounds	• Defer commitment
F1.4: Avoid premature commitment	• Workarounds • Oral communication	• On demand / Just-in-time RE • Rationale as by-product
F1.5: Requirements need context	• Personal network • Oral communication	• Rationale and context as by-product
F2.1: Slow feedback cycle on requirements	• Personal network • Oral communication	• Short iterations/agile • Virtual verification early
F2.2: Balancing the need for oral communication and thorough documentation of requirements	• Personal network • Oral communication	• On demand / Just-in-time RE • Rationale as by-product
F2.3: Find the right person for getting or giving feedback or information	• Personal network • Expert seeking	• Facilitate networking within company & supply-chain
F2.4: Sufficiently fast supplier interaction	• Personal network	• New business cases / opportunities for suppliers

of high level requirements as well as with peers. Information is basically collected through ***oral communication***. If this is not possible, e.g. due to deadlines, engineers are aware that they need to make ***assumptions***, which they mark for future clarification and for which they try to collect all available information. However, sometimes even the assumptions made are not completely clear, and there is the risk to perceive those assumptions as "complete" knowledge.

b) Recommendation(s): The need to rely on personal networks for clarifying requirements in large-scale software development has been reported before, e.g. in the scope of software ecosystems [15]. Based on the findings above and these previous works, we believe that there are two main directions of improvements: (i) bringing groups together to identify the right abstraction level on demand, and (ii) creating strong communication and feedback infrastructure to facilitate the (oral) information flow. Also, a framework for highlighting open issues and assumptions would be very useful; this would make a reader aware that the author is not certain about the specific part of the requirement.

2) Synchronizing Cross-Function and Cross-System Requirements (Finding 1.2): It is worth pointing out that functions owned by some parts of the organization stimulate the identification and definition of additional requirements to different parts of the organization. Such ***cross-functional requirements*** can be only partly anticipated since they emerge once function owners start to work in parallel. Function owners will then contact and coordinate with their peers about such emerging requirements. These requirements establish a ***horizontal flow***, in addition to the previously mentioned ***top-down flow***.

To better explain this concept we report a citation from a system constructor at VCG:

> *Then requirement can come from other directions as well, e.g., base technology, attribute requirements can directly impact us, even architectural requirement and cross-requirements from other sub-systems, or rather from other functions.*
> *— System responsible*

It is particularly difficult to manage these requirements since they can originate from all over the organization, and they can appear late in the projects as the functions are developed in parallel at different parts of the organization. Let us consider, for instance, the case of a requirement involving a signal that is already provided by a different function. These cross-requirements also create dependencies on deliveries, such that a function dependent on the behavior of another function or sub-system needs the other to be delivered before or at the same time otherwise the function will not work when integrating in the car. With cross-requirement, possible going back and forth between sub-systems or even components within this can create a mesh of dependencies that need to be integrated at the same time, or the system will fail.

a) Practice(s): To deal with the ***horizontal requirements flow*** of ***cross-function*** and ***cross-system requirements***, our interviewees referred to using their ***personal network*** to learn about potential synergies and conflicts from peers.

b) Recommendation(s): Technical dependencies are a well-known problem in large-scale software development [9], [16]. Especially in agile development, such dependencies can only partly be anticipated, and a strong culture of knowledge sharing is needed to deal with them efficiently [20]. The existence of cross-function requirements is one reason for the need for continuous requirements clarification [14]. Based on our findings, we suggest investigating whether continuous integration and deployment together with simulation can mitigate the challenge of ***horizontal requirements flow***.

3) The Cost for Changing Requirements Increases Over Time (Finding 1.3): Interviewees expressed that the cost for changing a requirement changes over time as the project progress. Early in a project, the cost for changing is low, as things have not yet been set in stone. As the project progress, the time and cost for changing design or requirement grows. Now, changes need to be reviewed and approved by more stakeholders and on a higher level in the organization. As

the project progress, the suppliers wish to be compensated for changes increases as well. This creates frustration as it is in this part of the project where most of the work is done, and issues are discovered when the implementation work has started.

> *[...] as the project progress, new gatekeepers and decisions fora where a change must be approved are introduced. [...] And the further the project progress the suppliers start to demand payment for changes, and that also introduces additional friction.*
> *— System responsible*

a) Practice(s): Our interviewees highlight that in the very end of the project, the high pressure to resolve serious issues and show-stoppers helps to speed up the process for changing a requirement significantly.

> *But then it is also so, if you are very far into the project or maybe even after start of production and you find some error, which is considered to be critical, then suddenly the decision paths are very short, if it is something that really have to be done.*
> *— Sub-system responsible*

Thus, it appears that ***workarounds*** are possible and that perhaps the way of working could be changed so that changes become easier during the time when they are most likely to occur.

b) Recommendation(s): Requirements will need to be updated, refined and changed as the project progress and new knowledge is generated. This is related to the cone-of-uncertainty concept [5, pg. 311], where it is argued that uncertainty is reduced as the project progresses. Lean software development, therefore, advocates to make decisions as late as possible [17], which becomes even more important when time-to-market of research and development is to be further shortened. Processes and ways of working need to take this into account and future research should investigate whether new ways of working across organizations and new business agreements between suppliers and OEMs can be designed to support this.

4) Avoid Premature Commitment (Finding 1.4): The way requirements are documented at VCG (as well as at many other OEMs) has been defined as a plan-driven development process in mind (i.e. the V-Model). The documents are rigidly structured with respect to the system architecture, which helps to address complexity. As a consequence, depending on where a requirement is written it will be interpreted as being a software or hardware requirement. Thus, when introducing a new requirement, a requirement author often needs to make an implicit decision about whether a requirement refers to hardware or software. In the communication with suppliers, there is currently no artifact that describe requirements on a component level.

[17].

> *At Volvo we have several requirement documents towards our suppliers, but in essence we have the [hardware requirement specification] and [software requirement specification]. [...] And sometimes we put software requirements in the hardware requirement specification [...] but in nine cases out of ten we want to have the requirement on the ECU in itself, then it is the responsibility of the supplier to divide the requirements between software or hardware.*
> *— Sub-system responsible*

The same challenge also applies to the decision, whether a software requirement is implemented by in-house teams or by suppliers. Our interviewees feel that they need to do a ***premature commitment*** when they document requirements and by this forestall design decisions that are (or should be) the responsibility of the receiver of the requirement.

a) Practice(s): Our interviewees mention ***workarounds*** such as specifying requirements as software requirements and then relying on ***oral communication***, mail, or in some cases comments in the requirement to notify the receiver that this implied decision can be overrun.

b) Recommendation(s): It appears that the current practices are somewhat fragile which can lead to additional efforts during the development, e.g. when receivers are not notified about their larger design space and need to default to the implied decision. We would, therefore, recommend that requirements should be made independent from such design decisions. By default leaving them open for the receiver of the requirements to decide allows for more efficient requirements communication and reduces waste.

5) Requirements Need Context (Finding 1.5): The interviewees agree on the fact that without knowing the proper context, it is hard to know exactly how the requirement should be interpreted, or what the rationale behind a requirement is.

> *[The requirements] often lack context, purpose, which might be one of the biggest problems, that we work without context. We do not know why something should be a certain way, and then you can't maintain them.*
> *— ECU responsible*

a) Practice(s): Our interviewees mention that to counter this, they search for requirement authors by leveraging their ***personal network*** and engage in direct ***oral communication***. They, however, admit that this is often hard in the boundaries between organizations, e.g. between supplier and OEM. It can also happen that the author is no longer working at the organization or has forgotten the reason for specifying a requirement in a specific way.

b) Recommendation(s): We recommend creating lightweight facilities to store context for a requirement beside the requirements. This could include, but is not limited to, comments from the author, rationale behind decisions, e-mails and other documents showing the discussion around a requirement or standards and specifications that is a source. In line with suggestions by Schneider [19], documentation of context could happen as a by-product and on demand, and by this avoid the ***over-specification*** of requirements and support continuous clarification.

B. Is There the Need for Shorter Feedback Loops for Requirements Validation? (RQ2)

1) Slow Feedback Cycle on Requirements (Finding 2.1): Engineers writing requirements for others to implement feel that the time between writing requirements and getting feedback or validation on whether their requirements/design is correct is too long, especially on requirements towards suppliers. In certain cases, the time between writing or updating a requirement and it can be validated can be close to one year.

Engineers writing requirements for others often feel that they need face to face or at least direct communication with the receiver to make sure the requirements are understood correctly

> *[On time for feedback on requirement] It takes a long time [...] and it usually goes full circle, that they have the software ready, it is put in the car, it is tested together in the environment it should be in, and then you discover that this and this is probably wrong. [...]*
> *— Sub-system responsible*

a) Practice(s): In certain cases, requirement authors try to mitigate these problems by personally asking for feedback ***personal network, oral communication***. Some engineers, for example, feel that they need to go to the testers to make sure that requirements are verified correctly and to get the feedback on if it works or not. However, it is not always the case that these questions can be answered with certainty until the requirement has been implemented, delivered, and tested.

Our interviewees also expressed the desire of having a framework supporting early validation of requirements and design to drastically reduce the time to receive feedback.

b) Recommendation(s): In order to provide faster feedback on requirements, we suggest offering requirement authors a virtual environment that allows exploration and simulation of requirements, to increase their confidence in a requirement. Also, a more agile process with shorter iterations and less early commitments to requirements and design can help.

2) Balancing the Need for Oral Communication and Thorough Documentation of Requirements (Finding 2.2): Engineers and requirement authors feel that the only way to be certain that a requirement is interpreted, tested, or otherwise used in the correct way is to talk directly and make sure. There are however doubts on the extent to which oral communication can cover all information needs in large-scale development that involves multiple suppliers.

Oral communication is preferred and in some cases, our interviewees could not imagine that it could work otherwise.

As a consequence of this ***communicative culture***, our interviewees expressed that they will ask for clarification in case of uncertainties or ambiguities. However, in certain cases, they rely on ***assumptions*** to fill the blanks. This happens in a number of situations, a deadline approaching can make it impossible to have the time to clarify everything. In certain cases, even if you ask for clarification you do not get a better answer, or just the point that you point out is fixed but the rest still does not make sense for you. After a while, you either have to give up and go ahead with your own assumptions and guesses or you have to take it to a manager at a level above. This is often something you want to avoid because many people does not feel comfortable with this, and it cost time and energy, maybe even more than just try to fix it yourself.

a) Practice(s): Our interviewees rely on their ***personal network*** and direct ***oral communication*** with other stakeholders. However, as discussed in the challenges above, this direct communication is not sufficiently integrated with processes and tools for requirements documentation.

b) Recommendation(s): We recommend to perceive requirements as a volatile and living entity that can and will evolve over the development life-cycle. Research should investigate suitable ways of capturing the essence of requirements related discussions in process and tool to provide good management of requirements knowledge.

3) Find the Right Person for Getting or Giving Feedback or Information (Finding 2.3): Our interviewees expressed that it was important to know who the right person to ask is, because in many cases things are not documented. Especially motivation and rationale for a requirement is in many cases missing, which makes it impossible to know why a requirement is specified in a specific way, and without context it is hard to interpret the requirement. The context, however, is hard to clarify since this might involve different persons working in different parts of the organization.

> *[On what to do if you think there are gaps in the information provided] you have to look in people's memories. So I start with someone, and they might answer that they do not know, so I ask who do you think know? Then I try with the next one and keep on doing that until I get what I need. [...] That is why I often talk a lot with the testers, often they are the ones who knows how it turned out to be.— Sub-system responsible*

a) Practice(s): Our interviewees stated that they rely on their ***personal network*** but sometimes struggle with ***expert seeking***, especially when they have no direct contact with the required expertise.

b) Recommendation(s): Based on these findings and related work [15], we recommend to facilitate networking within the organization and supply-chain.

4) Sufficiently Fast Supplier Interaction (Finding 2.4): Our interviewees agree that achieving sufficiently fast supplier interaction is challenging and that this interaction is significantly slower than interaction inside the organization. This is due to two interacting findings:

1) Documentation of requirements for subcontractors needs to be more detailed.
2) Communication and information sharing is harder or lacking because external subcontractors cannot walk around at Volvo to gain knowledge.

These two findings interact: If communication and information sharing were easier, requirements would not need to be

documented in such detail. If requirements were less detailed, interaction across organizational boundaries might increase.

Our interviewees express that the communication culture with suppliers differs, some partners in the supply chain are more open to changes to requirements than others, especially in the early phases of the project, where mutual understanding that the requirements need to be corrected or clarified exists. However, late changes are expensive for the supplier, which is problematic because often the need for changes to the requirements will only be identified late (see Finding 1.4). In particular, it leads to the following impediments for more efficient and faster requirements engineering:

- Suppliers lack knowledge about the environment of their component within the car;
- Suppliers sometimes lack domain knowledge;
- Suppliers offer only late feedback on requirements validation, mainly due to late integration of their contributions.

> *I think that sometimes the supplier needs to understand the overall behavior to be able to implement a good function. — Sub-system responsible*

a) Practice(s): Partly, our interviewees have a ***personal network*** that stretches to the suppliers.

b) Recommendation(s): Based on these findings we suggest that research should focus on how to base collaboration across different organizations within automotive software development more on trust and less on formal requirements (which, as discussed before will become more and more volatile) and rigid business agreements. Such research should include the definition of strong business cases for suppliers to allow for more accepting more changes, giving more feedback, and exchanging more intermediate artifacts.

C. Triangulation of Findings

The research we report on in this paper has focused on investigating challenges and practices for speeding up requirements engineering within a single automotive company, Volvo Car Group (VCG). To better understand the extent to which our findings are specific to this company, we conducted additional semi-structured interviews with a second company, Volvo Group Trucks Technology (VGTT). This comparison was enabled by existing research collaboration between both companies through the Software Center, which is a Scandinavian research center where companies and universities work together to accelerate adoption of novel approaches to software engineering. In this section, we will highlight similarities and differences based on these additional interviews, supported by quotes from our interviews at VGTT.

Both VCG and VGTT are in the process of changing their way of working and how requirements flow between the various stakeholders of automotive software development. As with VCG, we also see engineers at VGTT emphasizing the need for faster feedback cycles on requirements, especially when it comes to validation of requirements. Our interviewees at VCG referred to their ***personal network*** and ***oral communication*** as enablers for faster feedback cycles and for making the requirements engineering process keep up with the desired pace. We see similar practices at VGTT. However, VGTT is currently introducing additional practices to better support this, including ***cross-functional teams*** to support direct communication and to reduce handovers as a process developer at VGTT explains in the following quote.

> *That's the thought, to avoid these handovers [of documents], at least try to not having to do them for every sprint. [...] The idea is that specification and implementation people work together in parallel to create both the specification and the software [...] — Process developer*

When it comes to subcontractors, engineers in both companies think that the feedback cycle for requirements they send is far too long. To mitigate this challenge and to allow for more agility in the process, VGTT is changing its way of working with suppliers. Instead of only relying on contracts with subcontractors for delivering certain functionality according to specifications at a certain point in time, VGTT also starts to ***bring suppliers in-house*** and to pay for their development time as an alternative:

> *This in practices means that we pay our subcontractors for development time. — Process developer*

This has the consequence that a change of requirement or specification does not need new negotiations. Instead, VGTT can decide to pay for extra development time and take responsibility for any delays that implementing a desired change would mean.

The challenges of ***over specification*** and ***premature commitment*** that we found at VCG resonate strongly with VGTT: both companies have processes that rely very much on having comprehensive specifications upfront. In some cases, this is driven by safety standards, e.g. ISO26262, and therefore necessary, but our interviewees at VGTT see potential for improvement:

> *We need some documentation for after a project is completed. However, instead of writing specifications before starting development, it is better to implement and get feedback than waste time and effort on just believing and check that one are in agreement with early requirements. — Process developer*

We conclude that the challenges faced by both companies are comparable, while approaches to mitigate them and to improve the flow of requirements differ. We hope that this paper helps to facilitate a discussion on how to exchange knowledge within automotive software development, which becomes even more challenging when considering variability.

V. Conclusion and Future Work

Software has become a key driver for innovations, not only in automotive engineering. At the same time, traditional software companies like Google [11] and Apple [1], [2] are pushing into the automotive domain. To enable traditional automotive OEMs to compete with these new players, the

lead time for new functions needs to be drastically reduced, which can only be achieved by finding new ways of developing automotive software.

In this paper, we investigate how increased development speed and complexity affect requirements engineering in automotive engineering. We found that the current processes and tools do help with managing the complexity and engineering automotive products of the highest quality, but fail to sufficiently support achieving shorter lead times and faster time-to-market. In our interviews, we found engineers to work around rigorous processes by relying on informal communication. This is crucial to achieve a shared understanding of the context each stakeholder should operate in and direct communication is needed to understand and interpret requirements and the underlying rationale.

A reoccurring challenge we found included that requirements are hard to specify on a suitable level of detail. There is a trade-off between over-specifying and under-specifying requirements. On the one hand, high-level requirements are easily under-specified and do not cover all important variants. On the other hand, the current processes and tools seem to force over-specification of lower level requirements.

Another finding was that as different functions are divided between different groups in the R&D organization, one do not know all dependencies between them in the outset. The dependencies only become clear after the design of the function realization or even after the implementation has started.

One of the key issue we found is that the feedback cycle on requirements is too long. Our interviewees indicate that more in-house development and better simulation environments are required to address this challenge.

Finally, requirement authors felt frustrated that the friction for changing requirements is highest during the time of the project where most of the development happens.

Based on our findings and related work, we believe that future research needs to investigate the following questions:

- How to bring cross-functional groups together to identify the right abstraction level for requirements?
- How to create a communication and feedback infrastructure for supporting quick and smooth flow of information?
- How to implement continuous integration and deployment together with simulation to mitigate the challenges of horizontal requirement flows?

Such research will help to change the way of working with suppliers to a more open and flexible ecosystem. For this, it will be crucial to not only specify the formal requirements but also the rationale behind them. It is also important to facilitate networking within the organization and across the supply-chain.

In future, we plan to investigate how the automotive engineering ecosystem can be improved and to what extent our findings are relevant for other domains of embedded software engineering and software ecosystems.

ACKNOWLEDGEMENTS

The work acknowledges support by the ASSUME-project Vinnova, the Vinnova-project NGEA, the Software Center initiative (software-center.se), Volvo Car Group and Volvo Group Truck Technology.

REFERENCES

[1] Apple gears up to challenge tesla in electric cars. *Wall Street Journal*, Feb 13, 2015. Last visit: 2015-March-10.

[2] Apple hiring automotive experts to work in secret research lab. *Financial Times*, Feb 14, 2015. Last visit: 2015-March-10.

[3] N. N. B. Abdullah, S. Honiden, H. Sharp, B. Nuseibeh, and D. Notkin. Communication patterns of agile requirements engineering. In *Proc. of the 1st Workshop on Agile Requirements Engineering*, pages 1–4, New York, USA, 2011. ACM.

[4] A. Al-Rawas and S. M. Easterbrook. A field study into the communications problems in requirements engineering. In *Proc. of Conf. on Professional Awareness in Software Engineering*, London, 1996.

[5] B. Boehm. *Software Engineering Economics*. Prentice-Hall, 1981.

[6] V. Braun and V. Clarke. Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3:86–94, 2006.

[7] F. Calefato, D. Damian, and F. Lanubile. Computer-mediated communication to support distributed requirements elicitations and negotiations tasks. *Empirical Software Engineering*, Oct. 2011.

[8] D. Damian, I. Kwan, and S. Marczak. Requirements-driven collaboration: Leveraging the invisible relationships between requirements and people. In I. Mistrk, A. van der Hoek, J. Grundy, and J. Whitehead, editors, *Collaborative Software Engineering*, pages 57–76. Springer Berlin Heidelberg, 2010.

[9] C. R. de Souza and D. F. Redmiles. An empirical study of software developers' management of dependencies and changes. In *Proc. Int'l Conf. Software Engineering (ICSE)*, pages 241–250. ACM, 2008.

[10] EFFRA European Factories of the Future Research Association - a MANUFUTURE initiative. Factories of the future multi annual roadmap for the contractual ppp under horizon 2020, 2013.

[11] E. Guizzo. How the google self-driving car works. *IEEE Spectrum*, Oct 18, 2015. Last visit: 2015-March-10.

[12] I. Inayat, S. S. Salim, S. Marczak, M. Daneva, and S. Shamshirband. A systematic literature review on agile requirements engineering practices and challenges. *Computers in Human Behavior*, 2014.

[13] Industrie 4.0 Working Group - Final report of the Industrie 4.0. Securing the future of german manufacturing industry - recommendations for implementing the strategic initiative industrie 4.0, april 2013, online at: http://www.forschungsunion.de/pdf/industrie_4_0_final_report.pdf.

[14] E. Knauss, D. Damian, J. Cleland-Huang, and R. Helms. Patterns of continuous requirements clarification. *Requirements Engineering Journal (REEN)*, pages 1–21, 2014.

[15] E. Knauss, D. Damian, A. Knauss, and A. Borici. Openness and Requirements: Opportunities and Tradeoffs in Software Ecosystems. In *Proceedings of 22nd International Requirements Engineering Conference*, pages 213–222, Karlskrona, Sweden, 2014.

[16] I. Kwan and D. Damian. The hidden experts in software-engineering communication (nier track). In *Proc. Intl Conf. Software Engineering (ICSE)*, pages 800–803. ACM, 2011.

[17] M. Poppendieck and T. Poppendieck. *Lean Software Development: An Agile Toolkit*. Addison-Wesley Professional, 2003.

[18] P. Runeson and M. Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Softw. Eng.*, 14:131–154, 2009.

[19] K. Schneider. *Rationale Management in Software Engineering*, chapter Rationale as a By-Product, pages 91–109. Springer, Berlin, Heidelberg, 2006.

[20] N. Sekitoleko, F. Evbota, E. Knauss, A. Sandberg, M. Chaudron, and H. H. Olsson. Technical Dependency Challenges in Large-Scale Agile Software Development. In G. Cantone and M. Marchesi, editors, *Proc. of Int'l Conf. on Agile Softw. Dev.*, volume 179 of *LNBIP*, pages 46–61, Rome, Italy, 2014. Springer.

[21] Teknikfretagen produktionsforskning. Made in sweden 2030 - strategisk innovationsagenda fr svensk produktion, online at: http://www.vinnova.se/PageFiles/0/Made%20in%20Sweden%202030_produktion.pdf.