Empirical Software Engineering (2021) 26: 28
https://doi.org/10.1007/s10664-020-09903-x

Strategies to manage quality requirements in agile software development: a multiple case study

Pertti Karhapää, et al. *[full author details at the end of the article]*

Accepted: 11 September 2020 / Published online: 3 March 2021

Abstract
Agile methods can deliver software that fulfills customer needs rapidly and continuously. Quality requirements (QRs) are important in this regard; however, detailed studies on how companies applying agile methods to manage QRs are limited, as are studies on the rationale for choosing specific QR management practices and related challenges. The aim of this study was to address why practitioners manage QRs as they do and what challenges they face. We also analyzed how existing practices mitigate some of the found challenges. Lastly, we connect the contextual elements of the companies with their practices and challenges. We conducted 36 interviews with practitioners from four companies of varying sizes. Since each company operates in different domains, comparing QR management strategies and related challenges in different contexts was possible. We found that the companies apply proactive, reactive, and interactive strategies to manage QRs. Additionally, our study revealed 40 challenges in six categories that companies applying agile methods may face in QR management. We also identified nine contextual elements that affect QR management practice choices and which, importantly, can explain many related challenges. Based on these findings, we constructed a theoretical model about the connection between context, QR management practices, and challenges. Practitioners in similar contexts can learn from the practices identified in this study. Our preliminary theoretical model can help other practitioners identify what challenges they can expect to face in QR management in different developmental contexts as well as which practices to apply to mitigate these challenges.

Keywords Agile software development · Requirements engineering · Quality requirements · Requirements management practices · Non-functional requirements · Agile requirements engineering

1 Introduction

Quality requirements (QRs) are different from functional requirements (FRs) in the sense that they describe non-functional aspects (e.g., maintainability, reliability, performance, security) of a system (Wagner 2013; Wiegers and Beatty 2013). QRs are important because even if the

Communicated by Jon Whittle

promised functionality is delivered, the system will not be regarded as good by users if it does not have, for example, sufficient performance or usability. Agile software development (ASD) is aimed at continuously delivering valuable software to customers. The quality aspects of software are integral to fulfilling customer needs and delivering the promised value. Moreover, considering QRs too late in the development process produces bottlenecks and necessitates further work (Schön et al. 2017). It has been found that adopting agile methods often leads to neglecting QRs when the focus is mostly on delivering working functionality to the customer as quickly as possible (Cao and Ramesh 2008; Ramesh et al. 2010; Curcio et al. 2018). Together with the fact that an increasing number of companies, regardless of size, are adopting agile methods (Rodriguez et al. 2012; VersionOne Inc. 2016), it is important to understand how QRs are or should be managed in ASD. Recent secondary studies on requirements engineering (RE) in ASD have highlighted the importance of QRs and have emphasized that managing QRs in the context of ASD is both challenging and understudied (Schön et al. 2017; Heck and Zaidman 2016; Heikkilä et al. 2015; Inayat et al. 2015; Magües et al. 2016; Alsaqaf et al. 2017; Behutiye et al. 2019). Some of these studies—for example, Inayat et al. (2015) and Alsaqaf et al. (2017)—also concluded that there is a lack of empirical evidence on how QRs are managed in ASD.

Alsaqaf et al. (2019) is the most recent empirical study to have explored in detail the challenges associated with QRs in large-scale ASD. As one of our case companies applies large-scale ASD, we had the opportunity to compare our findings to those of Alsaqaf et al. We were consequently able to determine whether the same challenges exist in any kind of large-scale ASD, or whether other contributing factors are involved. This consideration underscores the importance of accounting for context while studying the management of QRs. However, in our study of the literature, we discovered that many studies of QR management in ASD focus on specific QRs while overlooking a careful consideration of context.

Our research contributes to the body of knowledge of empirical evidence concerning QR management by examining QR management strategies, practices, and associated challenges in four case companies that apply ASD in different contexts. By differences in context, we mean differences in the characteristics of the setting in which software development occurs, such as company size, organizational model, or type of customers. These characteristics are listed as contextual elements in Table 1. By QR management strategy, we mean *a plan containing different practices to handle QRs;* by management practices, we mean the *activities that are carried out to handle QRs in order to realize the plan* (Merriam-Webster 2019). The following three research questions were formulated to determine why practitioners manage QRs as they do, what challenges they face in doing so, and how the contextual elements of the companies are connected with these practices and challenges:

- RQ1: What strategies and practices are used to manage QRs in ASD?
- RQ2: What challenges are associated with the QR management strategies and practices in ASD?
- RQ3: How does context affect QR management strategies, practices, and challenges in ASD?

In this paper, we report on the findings from 36 interviews we conducted with practitioners from four ASD companies. These companies have been using agile methods for several years (from 7 to 13 years). The smallest of the companies employs close to 100 employees; two companies employ close to 1000 employees; and the largest company employs over 100,000

Table 1 Case Companies and Context Adapted from Petersen and Wohlin (2009)

Context facet	Context element	Company A	Company B	Company C	Company D
Product	**Name**	Product A	Product B	Several products	Product D
	Domain	Software development	Telecommunications	Telecommunications	Healthcare
	***Product type**	Commercial software modeling tool	Commercial embedded secure communication device	Common platform software for telecommunications	Contract-based custom software solution
	***Maturity of product**	Long-lived mature product	New product	Long-lived mature product	New product
	***Customization**	General product, same for all customers	Product for specific niche market, same for all customers	General products	Product tailored to different customers
Process	***Characteristics of the software development process (agile, plan-driven, …)**	Agile-based, continuous, and incremental development of features	Scrum-based, continuous, and incremental development of features	Plan-driven and Scrum-based development of features	Mixture of Scrum and Kanban for incremental development of custom software solutions, agile
	Start of agile adoption	2006	2007	2009 to 2011	2012
	***Distributed development**	No	Yes (nationally)	Yes (globally)	No
	***Developers in development of the product or component**	22	50	500	Up to 9
	Team size	Up to 10	6 to 8	9	Up to 9
Organization	**Size (number of employees)**	>1000	~1000	>100,000	~100
	***Hierarchical organizational model**	No	No	Yes	No
	Organizational unit involved in the study	Development team from one location developing one component of the product	Development team in company headquarters developing the product	One local development team from one business line developing a component for that business line	Development team from one location
Market	***Type of customer**	Software developers	National organizations	On component level - internal customers using the same platform for different communication technology services On system level – nations, large organizations	Institutions, business organizations
	Setting	Business-to-user	Niche market	Business-to-business (B2B)	Contractual development, B2B
	Constraints	Competitive market of similar tools	High security and safety, confidentiality, standards and regulations	Servicing multiple networking standards, standards and regulations	Competitive market
	***Access to customers or users**	Direct access to customers and users	Limited access to end users	Access to internal customers, limited access to end users	Direct access to customers and users. Very frequent contacts with customers

employees globally. The focus of the interviews was on the practitioners' ASD processes, QR management strategies and practices, and the challenges of QR management. Since the companies examined in our study are of varying sizes and are situated in different contexts, we also investigated the effect of context on QR management strategies and related challenges. Via this analysis, we were able to create an initial theoretical model depicting the relationship between context and QR management challenges as well as the practices employed by the companies to mitigate these challenges.

The main contributions of the study are as follows: (1) a detailed analysis of how the different case companies manage QRs; (2) an in-depth understanding of the actual challenges faced by practitioners when managing QRs in ASD, which is essential for focusing research efforts on generating solutions to actual problems deemed critical from a practitioner's point of view; (3) a fuller understanding of how contextual elements affect the choice of QR management practices as well as what kinds of challenges the different contextual elements might pose; and (4) a theoretical model of the connections between these contextual elements and QR management challenges and practices. We also discuss our findings in light of those detailed in the existing literature and aggregate the findings in order to facilitate future research on QR management and support practitioners in the management of QRs.

The rest of the paper is organized as follows: Section 2 presents related work. Section 3 describes the research method we followed and introduces the case companies. Section 4 presents our findings regarding the research questions: how the companies applying agile methods manage QRs, what challenges they face, and to what extent contextual elements can explain their choice of practices and the challenges they face. Then, at the end of Section 4, we present our theoretical model. In Section 5, we discuss our main findings and relate them to the scientific literature. We also outline some implications for practitioners and researchers before discussing potential threats to the validity of our findings. Section 6 concludes the paper, providing recommendations for future research.

2 Related work

QRs have been defined as requirements related to quality concerns that are not covered by FRs (Pohl 2016). As such, QRs have often been referred to as non-functional requirements (NFRs). NFRs have a definition very similar to that of QRs; for example, they cover aspects like performance, reliability, and maintainability, which cannot be covered by FRs (Mylopoulos et al. 1992). Other terms have also been used in the scientific literature, like extrafunctional requirements (Shaw 1996) or different -ilities (Glass 1998). Despite such terminological differences, we have used the term QR for all requirements not designated as FRs.

Although several secondary studies (e.g., Schön et al. 2017; Heikkilä et al. 2015; Inayat et al. 2015) aggregated the research on RE in ASD, focus was placed on requirements in general, and as such no studies, except for Alsaqaf et al. (2017), exist that describe QR management. For instance, even though Heikkilä et al. (2015) found that NFRs are often ignored or implemented relying on tacit knowledge, they did not provide a detailed explanation for why this occurs. Likewise, although Inayat et al. (2015) identified some practices—mainly related to testing activities, such as acceptance tests, system quality tests, and "inspective testing" throughout the lifecycle—used to solve challenges posed by NFRs in ASD, they did not provide details of these solutions; moreover, they reported that the usability of these solutions has not yet been explored. Schön et al. (2017) found and briefly described ongoing

work on QR management in ASD in their secondary study, identifying one framework and two methods for managing QRs in ASD. The framework is called the Agile Framework for Integrating Non-Functional Requirements Engineering (AFFINE). This framework introduces a new role into Scrum that is responsible for NFRs (Bourimi et al. 2010). The two methods are (1) Non-functional Requirements Modeling for Agile Process (NORMAP), in which a taxonomy is used to classify requirements as functional or non-functional (Farid 2012); and (2) SENoR for NFR elicitation through workshops utilizing the quality sub characteristics of ISO 25010, and the importance of NFRs for the development project (Nawrocki et al. 2014). These studies described and proposed practices that can help manage NFRs (or QRs); however, they did not focus on a detailed analysis of the QR management strategies or practices that companies are currently using or on the challenges they are presently facing.

The secondary study by Alsaqaf et al. (2017) focus specifically on QRs in ASD. They reported on the challenges covered in the existing literature. They summarized the reported QR management practices in large-scale ASD as well as general challenges to QR management in ASD. The authors outlined 12 elements of agile processes that result in diminished focus on QRs, such as the inability of users' stories to document QRs and the dependency on the product owner (PO) as the single point to collect QRs. The study also inspected different solution proposals for QR management in ASD as well as whether the level of agility of these proposals was considered. They found that the agility factor was considered in only two of the 13 identified proposals. Still, the study did not explain whether the associated challenges were relevant to a specific context. In a more recent empirical study, Alsaqaf et al. (2019) explored the challenges of engineering QRs in ASD. They revealed the challenges practitioners face when engineering QRs, the mechanisms behind these challenges, and the practices applied to mitigate them. Their research questions were very similar to ours; their focus was on large-scale distributed agile projects in the Netherlands. One of our studied companies also applies distributed agile development in large-scale systems. This gave us the opportunity to compare our findings to those of Alsaqaf et al. Additionally, we included small-to-medium size enterprises (SMEs) in our study, which allowed us to compare practices and challenges at different scales of software development. Alsaqaf et al. (2019) also discussed the possibility of traceability between QR management challenges and context. However, in their study, context was not modeled in detail and was related mostly to the domain of the companies.

The adoption of agile methods in large-scale development, which was the setting in Alsaqaf et al. (2019) study, can pose its own set of challenges for the management of requirements, and QRs are no exception. For example, Kalenda et al. (2018) studied this topic both empirically and in the literature. They found, among other challenges, that the distributed nature of scaled ASD places emphasis on constant communication, and that scaling ASD also means scaling RE, which in turn demands some hierarchy in requirements management. Many of the challenges that arise when scaling ASD or when transforming the plan-driven development of large systems into agile development are both interesting and relevant—that said, this topic is mostly outside the scope of our study. What *is* relevant for our study is that Kalenda et al. (2018) determined, from the literature as well as from their empirical results, that when scaling ASD, there is the risk of a loss of quality in code and of an accumulation of technical debt shortly after the transition to scaled agile processes, *if* the processes are not managed properly. Additionally, Kasauli et al. (2017) also studied requirements engineering challenges in large-scale ASD. They also found that communication and knowledge management are important in such

development, and they additionally emphasized the importance of bridging the gap between customers and developers.

In addition to the mentioned studies on QRs in ASD, we found in our own recent systematic mapping study (SMS) on QR management in agile and rapid software development some proposals for QR management (Behutiye et al. 2019). Most of these studies, however, had a narrower scope, focusing only on a particular QR. For example, Azham et al. (2011), Siponen et al. (2005), and Rafi et al. (2015) focused on security in ASD. Azham et al. (2011) stated that Scrum does not address security issues. Meanwhile, Siponen et al. (2005) stated, in their work on integrating security into ASD, that current agile methods lack features to specifically address security risks. Rafi et al. (2015), on the other hand, proposed a modified Scrum method to enhance correctness, usability, and security, and they additionally stated that security and usability are overlooked when focusing on functional features. Usability in ASD is a recurring theme in the literature as well. Butt et al. (2014), Anwar et al. (2014), and Singh (2008) are examples of these studies. Butt et al. (2014) worked toward integrating usability into ASD and found that usability approaches are not followed in agile approaches when focusing on the development of a running system. Anwar et al. (2014) reported that ensuring usability is challenging for distributed teams in SMEs that are applying agile methods. Singh's (2008, p. 556) study on agile methods found that in "traditional Scrum processes […] usability is typically included only as an afterthought, if at all." Usability and security are among the most frequently studied QRs in the scientific literature we reviewed. As the main focus of these publications was on solution proposals, they did not include detailed descriptions of the challenges faced by companies when managing QRs, nor were they all evaluated in an industrial context.

In our SMS, we also identified some solution proposals focusing not on a specific QR but instead on a wider range of QRs. Farid and Mitropoulos (2013) proposed the Non-functional Requirements Planning for Agile Processes (NORPLAN) to improve the planning and prioritization of NFRs. Cannizzo et al. (2008) explained that the implementation of continuous integration (CI) can address QRs, particularly robustness and performance. According to the authors, CI can reveal robustness and performance issues that are not visible in acceptance tests. Similarly, Chen (2015) showed how CI enhances the reliability of releases. Continuous delivery helps to identify errors in deployment earlier through frequent testing. The study by Singh (2008) is one of the few to have deeply analyzed challenges. However, the focus of the study was on usability only. Another study that provided a more detailed analysis of challenges was conducted by Cajander et al. (2013), who also focused on usability as well. The studies that we identified either did not offer a deeper analysis of the challenges faced by companies when managing QRs or had a narrow scope, focusing only on a specific QR.

As a summary, although the literature describes some initial solutions to manage QRs in ASD, the reported proposals often lack empirical validation (Schön et al. 2017; Inayat et al. 2015; Alsaqaf et al. 2017), making it unclear to what degree they are useful in practice or which proposal works in what context. In order to understand and address the challenges that practitioners experience, we must have a better understanding of which challenges are specific to what context and what kinds of practices can mitigate these challenges. This research will shed light on the actual strategies used in practice in different companies, as well as related challenges, and will explore whether—and if so, to what extent—context can explain their choice of strategies or practices and the accompanying challenges.

3 Case study design

We conducted a multiple case study (Runeson et al. 2012) of four separate holistic cases. Runeson and Höst's (2009) guidelines were adopted for designing and conducting the case studies. We focused on specific units of development within each company, which are explained in Section 3.1. In Section 3.2, we describe the data collection procedure and present profiles of all interviewees. Section 3.3 explains the data analysis procedure. In 3.4 we explain the guidelines used for constructing the theoretical model.

3.1 Case and subject selection

The companies in our study are all part of the Q-Rapids[1] project, which is aimed at improving QR management in ASD. This study is also part of the Q-Rapids project. The companies were selected for the project because of their different domains, sizes, and other contextual factors, as the aim of Q-Rapids is to improve the quality of processes and products in any kind of ASD context. The companies have all applied ASD for several years. Based on the checklist and suggestions provided by Petersen and Wohlin (2009) and Dybå et al. (2012), Table 1 presents detailed information about the company contexts in our study so that others can understand and make comparisons to our case companies. We divided context into four different facets, and each facet was further divided into context elements. For example, the Product facet includes the product name (due to confidentiality, the actual names of the products and companies are not given), product type, maturity of the product, and customization, which characterizes the product as a general product, a tailored product, or a product for a specific niche market. The Process facet characterizes the software development process applied in the company, when agile development was adopted, whether or not development is distributed, and the number of developers involved in developing the product. The contextual elements relevant for QR management practices and challenges, as we will show later in the findings, are highlighted in Table 1 with an asterisk (*).

3.1.1 Company A

Company A is an SME developing a long-lived software modeling tool for developers and architects to support software and system engineering. The company is also positioned as a strategic partner of the largest clients of the market in the sectors of Finance, Banking, and Insurance, and it has developed expertise in the field of Digital, Big Data, Analytics, Performance, and Operations. Our case study within Company A was conducted with a software development team in charge of developing one of the modeling language components in that tool. The tool, which has been developed for 25 years, is used by software developers following a model-driven development approach and is also used by the company itself. The development process follows the values of the Agile Manifesto (Fowler and Highsmith 2001), emphasizing working software over documentation and utilizing face-to-face communication and close collaboration with customers. The development process is iterative; however, the company does not follow any formalized method, like Scrum. Even though the content of a release is planned every six months, the company prefers to not follow any pre-defined sprint cycles. In addition, even though development relies heavily on face-to-face communication,

[1] http://q-rapids.eu/

standard daily or weekly meetings are not practiced. Meetings take place when needed. The focus is on being highly responsive to change, staying ahead of the competition, and fulfilling customer needs, which are the drivers when prioritizing requirements. Requirements are managed in terms of features. Features to be developed are elicited by product management following a feature strategy given by the executive managers and are heavily influenced by customer feedback. The features are communicated to and discussed with the development manager, who in turn plans the features for the sprints together with the developers. Unit tests, nightly integration tests, and validation tests are done before release. All faults are reported through automated reports and discussed between developers, the quality assurance team, and the development manager whenever needed.

3.1.2 Company B

Company B is a large enterprise that develops secure communication and connectivity solutions for multiple industry domains. In a bid to achieve efficiency and shorter time-to-market, the company moved to agile and lean software development in 2007. A well-defined Scrum process, "almost directly from the book," as one interviewee stated, is followed in the development of highly secure, reliable, and robust embedded devices. The company aims to identify the needs of the market and to fulfill those needs with special solutions. The company has all their development processes well-documented, which is a requirement in the security and safety-critical domain. The development of one of those devices, which started in 2015, is the unit of focus in our case study. The company uses a standardized process for managing the development process as follows. After the project is initiated, product management approves the product proposal. Elicitation of high-level epics follows this approval. In this case, these high-level epics are refined to features and user stories together with developers. Everything is documented in internal wiki-pages. The user stories are passed on to sprint planning, where they are prioritized in the sprint backlog. A sprint is normally two weeks long in a six-month release cycle. Implementation is supported by automated build and testing, manual code review practices, continuous integration, and acceptance testing. An intelligent dashboard is used to visualize and monitor the status of the development process in real time. The company also applies the standard Scrum practices of daily stand-up meetings, sprint reviews, and retrospectives. There is, however, one difference from the common Scrum practices: Due to the special customer segment (i.e., public safety, law enforcement, military, etc.), the company has limited access to the final users of the embedded products and thus limited communication with them during the actual development of the product. Trial users, who can be company-internal, may act as users to test usability prior to release. Feedback from real users is taken into consideration after release, in the maintenance phase.

3.1.3 Company C

Company C is a global company with several business lines in which development is distributed to many locations across the globe. Several coordinated agile teams develop large systems with complex dependencies. Thus, we can consider development in Company C as very large-scale ASD. Due to the size of the company, there is a need for some hierarchy in the decision making and coordination of development done by several teams. Thus, we classify the company as an organization with a hierarchical organization model in the context table (Table 1). All the different business lines have a global process description to follow; however,

their practices are not identical. Some business lines focus more on software products; while in others, hardware is tightly coupled with the software. The number of employees also varies greatly between business lines. Thus, the process is not exactly the same in all business lines. Even though our main focus of study was on one specific business line, we included interviews from other business lines as well to get a richer picture of QR management in the company. In this paper, we do not go into details of any specific business line but generalize the description. In general, in a business line, development is carried out by teams applying agile methods. Scrum is the most preferred method of the small dynamic teams of nine developers that are formed based on required competences. Features to be built are decided upon and prioritized at the management level, where business opportunities are analyzed. Following that, a separate group of people within that business line is responsible for splitting those features into sub-features and for refining the sub-features into requirements and acceptance criteria. Finally, this same group of people specifies the requirements. Features and sub-features usually require so much effort and different areas of expertise that their development is divided not only into different teams in one location but also into teams at different sites. Thus, one team in one location develops only a small part of a component that is part of a sub-feature, which in turn is part of a feature of the system. Eventually, the POs are responsible for refining the tasks for teams. The PO completes the refinement together with some of the developers, depending on the expertise required for the task. Once this has been done, the PO presents an effort estimation to managers for approval before implementation. Development is accomplished in two-week sprints in either one-month or six-month releases, depending on whether the release is an internal release, a maintenance release, or a customer release. The customer release is typically a six-month release. Testing starts as soon as there is something to test; developers run unit tests before committing code to the CI. A multi-level CI starts with the integration of small parts at the lowest level. Any issues found at this level can be communicated directly back to developers for them to address. The results of several teams are integrated on a higher level in the CI and finally in system testing at the highest level. This practice helps to ensure quality by making it possible to address issues found in lower-level testing as fast as possible.

3.1.4 Company D

Company D is the smallest of the case companies in our study. The company develops customized applications and software solutions for other companies and institutions operating in various areas, such as health care, security, warehouse management, or the space sector. Each project lasts between two and 14 months. The project of focus in our study was in the health care domain. The development employs a mixture of Scrum and Kanban. The company works in close collaboration with customers. Several meetings take place in "scoping sessions" before development, in which the requirements are elicited and an acceptance criterion is formed. The client also participates in intermediate reviews of the product if agreed upon in the contract. The PO sets up the project configuration and development team and defines the tasks for developers to complete in one- to two-week sprints. Sprint length depends on the length of the project but usually lasts one week, from Tuesday to Tuesday. The sprint planning, sprint review, and sprint retrospective meetings are held consecutively within one meeting between sprints. Daily stand-up meetings are also practiced during development. A project Kanban board is utilized to manage the flow of work in the software development process.

3.2 Data collection procedure

Data was collected incrementally through semi-structured interviews by the first two authors. The third author participated in one interview to ensure that the interviews flowed well and for the purpose of researcher triangulation. Interviews were conducted in two rounds. The first round consisted of two sets of interviews. The first set, the initial interviews, were conducted with 13 interviewees in 12 interviews (one interview had two interviewees, see Table 2 in Section 3.2.1): three interviewees from Company A, four from Company B, four from

Table 2 Interviewee Profiles

Comp.	ID	Interviewee role	Experience in years in			Interview length (min)	
			SW dev.	Comp.	ASD	Round 1	Round 2
A	P1**	SW architect, SW developer	11	11	11	48	52
	P2	Project manager	20	6	10	28	66
	P3	Head of research unit	17	11	12	48	
	P22	Executive manager	30	30	13		47
B	P4, P5	P4 – DevOps specialist, Technical leader, Project manager, SW architect	15	2.5	15	54	65
		P5 – Product owner	10	5	7	54	
	P6	Quality manager	21	11	9	63	
	P7	Production tester SW developer	25	6	5	54	68
	P30**	Project manager	25	17	12		51
	P31	Process coach	15	15	6		62
C	P8	Requirements manager	20	20	12	54	
	P9	Product owner	17	4	9	41	
	P10	Agile coach	20	16	11	80	
	P11	Project manager	4	4	4	42	
	P14*	SW developer	16	16	5	43	
	P15*	Test architect	NA	20	7	51	
	P16*	SW developer	22	22	10	58	
	P17*	Product owner	20	10	10	50	
	P18*	Planning manager	20	20	NA	54	
	P19*	Competence developer, Planning manager, Quality specialist	24	24	12	59	61
	P20*	Product owner	20	18	10	45	
	P21	Fault manager	3	33	3		37
	P23**	Transformation expert on processes and tools	NA	1.5	NA		101
	P24	Quality manager, Competence manager	18	25	10		45
	P25	Technical leader, Project manager	12	10	8		48
	P26	Fault manager, Process manager	6	20	6		65
	P27	Quality specialist, Process improvement expert	NA	19	NA		58
	P28	SW engineer, CI developer	6	6	6		54
	P29	SW developer	16	16	6		50
D	P12	Product owner, Scrum master, Chief SW architect	10	10	5.5	68	85
	P13	SW architect, Chief of SW development department	8	8	6	47	

Company C, and two from Company D. Based on the initial analysis of these interviews, we identified the need to have complementary interviews at Company C to get a more detailed view of their QR management practices. Company C is much larger than the other companies, and some of the interviewees came from different business lines with slightly different practices. We conducted seven additional interviews in Company C. These complementary interviews used the same interview script as the initial 12 interviews; thus, they were part of the first round of interviews. Later, a second round of interviews was conducted to confirm, complement, and further deepen our knowledge of QR management in the companies. Some of the interview questions in the second round were specific for each case company to capture all relevant information that was not captured during the first interview round. In the second round, we conducted 17 interviews: three interviews from Company A, four from Company B, nine from Company C, and one from Company D. The distribution of interviewees across companies is also depicted in Fig. 1.

Overall, we conducted 36 interviews with 30 different participants. One interview in the first round included two interviewees, P4 and P5, while other interviewees were interviewed in both the first and second rounds (P1, P2, P4, P7, P12, P19). The interview length for these participants in both round 1 and round 2 are indicated in Table 2. Those participants who participated in the complementary interviews of round 1 are indicated with a star (*).

Each initial interview was divided into three sections. At the start of the interview, each interviewee was given an informed consent form to read and sign before the recording began. The interview then started with warm-up questions, asked regarding the interviewees' expe-

Fig. 1 Interview rounds in the different companies

rience in ASD and the company in question, after which the interviewees were asked about their understanding of QRs. The purpose of this question was to understand how the interviewees perceived QRs and to make sure they understood what we meant by QRs. The rest of the interview questions were defined and structured around different activities of RE according to CMMI for development, version 1.3 (CMMI Product Team 2010). We acknowledge that there are varying views on whether CMMI and ASD work together; however, we used CMMI as a comprehensive checklist, not as a maturity or capability model. With the help of CMMI, we sought to ensure that we covered all the different aspects of RE. We asked questions related to how QRs are elicited, how they are prioritized, how they are documented and validated, and so on. In addition, questions were asked about whether the interviewees saw any connection between QRs and other areas related to software development, such as planning and risk management. Finally, in the last section, we wrapped up the interview by asking about any challenges the interviewees faced in the management of QRs in their current development practices. We did not ask about specific challenges that have been reported in the scientific literature in order not to bias the interviewees into thinking that they might have experienced those challenges. In the second round of interviews, we first confirmed the findings of the first round of interviews, after which we asked about specific details that we failed to capture in the first round. The interview scripts can be found in Appendix 1 and Appendix 2.

The interview scripts were devised by two researchers and reviewed by two additional researchers. All interviews were recorded, stored for internal use by the researchers only, and transcribed for analysis. Transcription was performed by a professional transcription service provider. All data extracted from the interviews were anonymized. We conducted and analyzed all interviews in peers in order to minimize researcher bias.

3.2.1 Subjects

All interviewees were practitioners who were working in software development either directly in the development of products or as a PO or manager with knowledge about QRs or the software development processes. If the company had specific roles related to the quality of the product, these roles was of special interest. Interviewees were selected from varying roles on different levels in the companies to obtain a broad range of perspectives on the management of QRs. From each case company we had one person that was a member of the Q-Rapids project, that we call a champion. The champion helped us in identifying interviewees. In Companies B and D, the champion was a manager; in Company A, the champion was a SW architect; and in Company C, the champion was a transformation expert on processes and tools. A pre-questionnaire was sent to each champion prior to the interviews, in which we asked for domain and context information, as well as for information regarding the methods and tools used in their software life cycle management. The pre-questionnaire was designed to gather initial information for the Q-Rapids project and was also utilized for our case study. Those parts of the pre-questionnaire that were utilized for this case study are presented in Appendix 3. The pre-questionnaires were also employed for triangulation purposes during data analysis.

Table 2 lists interviewee information for each case company. Those champions who also participated as interviewees are indicated with two stars (**). Note that some interviewees have more than one role. Three of the interviewees have N/A (not applicable) in the experience column. P23 is an expert on development processes but has not worked in software

development as a developer. Interviewees P15 and P27 stated that they have worked with software development for longer than they have been employees at the company, but they could not say for how long. Since they both started out in the company as young developers, we can safely assume that their years of experience in software development are equal to their years of experience in the company.

3.3 Analysis procedure

The coding of interview transcripts was completed with NVivo qualitative data analysis software (NVivo 2018). The challenges and practices were coded in a deductive (Miles and Huberman 1994) manner with a start scheme that was devised a priori by two researchers according to the main topics in the interview questions (ways of working, QR understanding, QR elicitation, QR communication, QR documentation, specifying QRs, QR prioritization, QR validation, and related challenges). The concrete practices and challenges emerged during the analysis of the data. Fig. 2 shows a snapshot of coding in NVivo. On the left-hand side, next to the blue bubbles, are the codes of which elicitation is highlighted, and on the right-hand side there are excerpts of interviews coded as relevant for elicitation of QRs. Identifying the

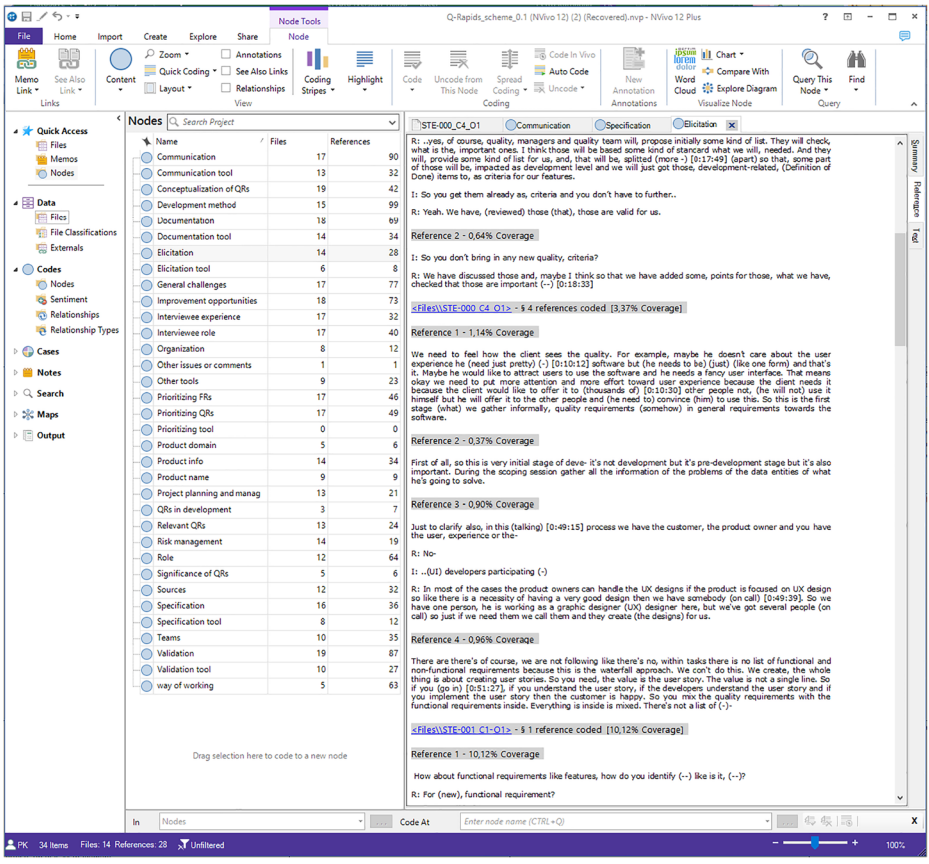

Fig. 2 NVivo used in coding interview transcripts according to themes in interview questions

connection of the practices and challenges to context was achieved in an inductive way through thematic analysis (Cruzes and Dybå 2011). Contextual elements were identified from the interviews and the pre-questionnaire (the contextual elements are presented in Table 1).

Figure 3 shows an example of how we coded parts of interviews related to practices and challenges in the elicitation and analysis of QRs. We coded the practices as shown by text with blue underlining in the yellow boxes and matched them to the activities discussed. Fig. 3 also shows the "together with the client" and "limited access to user scenarios" codes with red underlining. As these were both related to the same concept of access or limited access to users, we classified them under the same theme. Further, we identified contextual elements enabling practices or contributing to inhibiting challenges. In the example in the figure, access to users enables the practice of conducting scoping sessions together with the users and using templates and checklists in the elicitation and analysis of QRs. It also shows how limited access to users poses a challenge for elicitation but can later be mitigated by the practice of value stream mapping sessions (VSM, explained later in Section 4.1.2). Interviewee IDs and parts of the text in the challenge box are hidden to ensure confidentiality.

The information from the pre-questionnaire was also used for triangulation by checking that the information collected through interviews was in agreement. We conducted workshops together with the champions and key roles of the companies to validate the results of our analysis. The number of workshops and participants are presented in Table 3. During the workshops, which lasted approximately 1.5 h, we presented process models of the companies' software development, including QR management, followed by an open discussion on whether the model was accurate and properly fit the company's way of working. In most of the case companies, the input from the workshops required only minor changes in detail regarding, for example, QR management tools and details about QR flows, but these changes did not affect the findings presented in this paper in any other way. In the case of Company C, we needed to update the flow of reporting to correctly order the activities and recurrence timescale. When structuring the findings, we constructed a theoretical model of the relationship between

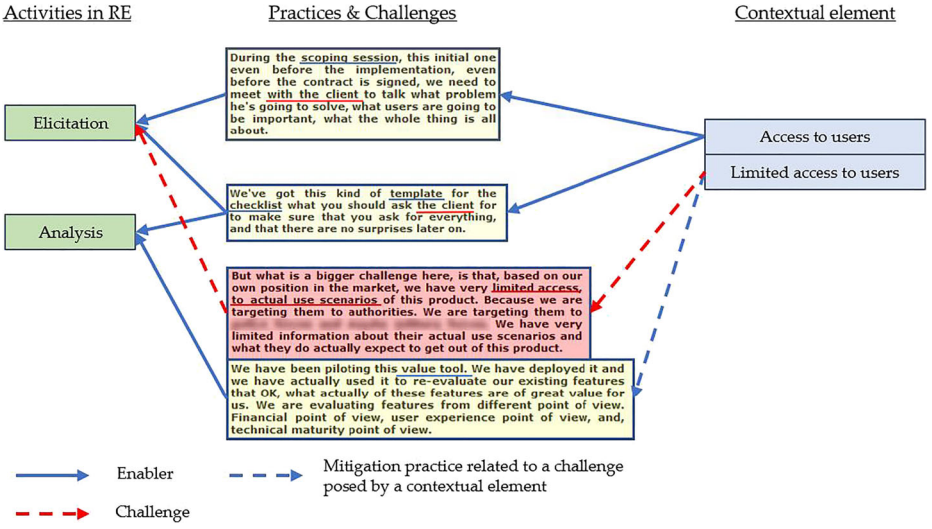

Fig. 3 Example of connecting challenges and practices with contextual elements

Table 3 Workshops for Validating Results of Interviews

	Participants in Workshop 1	Participants in Workshop 2
Company A	4 (P1, P2, P3, P22)	3 (P1, P2, P22)
Company B	3 (P4, P6, P7)	4 (P4, P7, P30, P31)
Company C	3 (P23, P24, P28)	2 (P23, P28)
Company D	2 (P12, champion)	(no workshop)

contextual elements and the reported challenges and included also the practices that the companies apply that have the potential to mitigate the challenges. This was done to better highlight the reasons for the chosen practices and how the practices, challenges, and contextual elements were connected. How we did this is presented in Section 3.4. After we had analyzed the contextual impact, the results were shared with the champions of the companies, who reviewed the results. The results were discussed with the champions when necessary and minor changes to details were done when needed. These results can be used in future research to explore whether the findings are generalizable to other companies in similar contexts.

3.4 Theoretical model formulation

Sjøberg et al. (2008) proposed building software engineering theories in five steps: (1) defining the constructs of the theory, (2) defining the propositions of the theory, (3) providing explanations to justify the theory, (4) determining the scope of the theory, and (5) testing the theory through empirical research. These steps are not usually applied sequentially but are often carried out iteratively and partly in parallel instead.

In our theoretical model, constructs and propositions (steps 1 and 2) were extracted from our results as it is described in Sections 4.1.1 and 4.1.2, respectively. In order to offer explanations for each proposition (step 3), we provide logical justifications based on interpretations of our empirical multiple case study (in Section 4.1.2), which is one of the ways explicitly mentioned in Sjøberg et al. (2008) to perform this task. Regarding the determination of the scope of the theoretical model (step 4), we specify all the contextual factors that define the cases for which the proposed theoretical model applies as attributes of the constructs. Finally, the testing through empirical research of the theoretical model (step 5) has not been done in cases other than the four cases of the present study. Further empirical research should consolidate, extend, or refute the results of the proposed theoretical model.

4 Results

In this section, we present the findings from each case study and the results of the cross-case analysis. We first give an overview of the QR management strategies (*a plan containing different practices to handle QRs*) and practices (*activities that are carried out to handle QRs in order to realize the plan*) in each case company in Section 4.1, with a cross-case comparison in Section 4.1.5 to answer Research Question 1. Then, we present findings regarding challenges in Section 4.2 to answer Research Question 2. We explain the differences in challenges and practices with the help of the different contexts of the companies in Section 4.3 to answer Research Question 3. Finally, in Section 4.4, we present the findings more systematically in the

form of the theoretical model of QR management challenges and practices. In the following text, all quotations from the interviews are in italics.

4.1 Strategies and practices to manage QRs in ASD

Overall, we found that the term QR is a fuzzy concept among practitioners in agile companies. When we asked them about QRs, the discussion tended to focus on quality in general and how it was managed. When seeking to discuss specific RE practices, the discussion tended to focus more on FRs. The interviewees naturally tried to reflect on the practices they were following, which were mostly related to FRs or to the development of features. The same phenomenon was observed regarding the questions about challenges. The interviewees tended to focus on the challenges the interviewees were currently facing in their work, which did not necessarily relate to QRs. The results reported in this section are relevant for QRs.

We characterized the strategies the companies apply as proactive, reactive, and interactive. Table 4 explains these strategies together with examples. The overall strategies applied by each case company mostly feature all of the above characteristics. However, one strategy of QR management can be more dominant than others. We do not intend to classify the strategies as one being better than the other.

4.1.1 QR management in company A

The goal of Company A is to deliver as stable and reliable a product as possible. Thus, the most prominent QRs are stability and reliability. The strategy they apply to reach this goal is mostly reactive and interactive. The only proactive characteristic is that the company roadmap, which contains the company's long-term and short-term goals, is the first source for high-level QRs. The interactive characteristic of their QR management comes from the fact that the developers use the product in the development themselves and can identify concerns regarding the quality of the product. They allow the QRs to evolve during development. "*We are using the tool to develop the tool. That way, the developers can spot any problems.*" The reactive characteristic comes from the practice of not specifying QRs up front and relying heavily on feedback from quality assurance (QA) and users. This is possible because the product is mature and the most important QRs have already been established.

Table 4 Categorization of QR Management Strategies into Proactive, Reactive, and Interactive Strategies

Strategy	Definition (Merriam-Webster 2019)	Meaning and example
Proactive	"creating or controlling a situation rather than just responding to it after it has happened"	QRs are analyzed and specified before development starts (up-front analysis and specification): e.g., Mock-ups and templates are used in meetings together with the client prior to development to elicit and specify important QRs.
Reactive	"acting in response to a situation rather than creating or controlling it"	QRs are allowed to surface later in development: e.g., a minimum viable product can be developed for the users, whose feedback can be used to identify important QRs.
Interactive	"(of two people or things) influencing each other"	General, high-level QRs are identified early, and they are evolved during development: e.g., users validate QRs in review sessions during development, before the product is released.

Instead of explicitly specifying QRs, the company relies on the experience and expertise of the developers to consider these QRs in development. "*It is a trade-off. Do you spend time doing specification, or do you spend time developing?*" Another interviewee said: "*The developers know it. They know the QRs.*" As for FRs or the features in general, specification and documentation of QRs is minimal, as promoted in the Agile Manifesto (Fowler and Highsmith 2001). Company A relies mostly on Word documents, emails, and whiteboards in documentation and communication. "*The QRs are mostly in the heads of the managers, and they are communicated through emails and Word documents,*" one interviewee reported. Another interviewee stated that *"the Word documents can be considered as backlogs."* When the executive board has decided on the priority of features to implement, there is a meeting in which the most suitable architecture is discussed. There is not much more discussion about QRs at this point, unless the feature to be implemented is completely new. In that case, they produce detailed documentation to explore the extent to which the new feature is compatible with existing features or technologies; however, there is still no specification or documentation explicitly for the QRs, even though they are communicated verbally.

In testing, the company utilize both automatic testing in the CI and manual testing to identify quality concerns, and finally, they utilize customer or user feedback extensively. Users can provide feedback through the open source community, and paying customers can utilize a hotline for reporting issues directly to the sales team. These are means to identify "*quality issues,*" so called because they are reported as issues and could be translated into QRs. For example, a report could say that an element of the interface does not scale when changing resolution. This issue could be translated into a QR by stating that all elements in the interface should scale when changing its resolution. The quality-related errors found in testing are discussed between the QA team and the product manager to decide on the priority of the issues. Here, the "*gravity of the issue*" (i.e., the criticality), as some interviewees call it, determines the priority. For example, when a user experiences a crash—i.e., when the software stops working, and the user reports the issue—it receives the highest priority. "*Blocking issue*"—i.e., when the software prevents the user from accomplishing a task—have the second-highest priority. The customer-reported issues are prioritized in the same way as the internally found issues, with the difference being that they always get the highest priority.

Lessons learned

- Company goals and roadmaps can be used for proactive elicitation of high-level QRs
- With a mature product, there is a reduced need to specify all QRs up front
- When developers use the tool in development, they can interactively evolve QRs during development
- Close connection to users enables valuable user feedback for reactive elicitation of QRs
- CI and QA can also be utilized for reactively eliciting QRs

4.1.2 QR management in company B

The most emphasized goal in Company B is to deliver value to the customer: "*We always try to think, what is the value of this feature?*" QRs are important for this goal. The strategy for QR management in Company B is a combination of proactive, interactive, and reactive. It is proactive in the sense that, currently, it includes practices for QR management already in the

initial steps before actual development. Initial roadmaps and high-level epics include QRs that may come from, for example, the regulatory needs of a governmental customer. The interactive part comes from a periodical review/check done by the company to ensure that the development is in line with the goals. The reactive part comes through ongoing test automation in the CI, separate security testing, and trial user tests.

QRs are prioritized already at an early phase when roadmaps and high-level epics are defined. The value to the customer is an important theme in the prioritization. The product manager prioritizes features and requirements together with POs. Features are stored in product backlogs using Jira. At the feature level, QRs are considered in the Definition of Done (DoD) as quality targets, i.e., target values for general quality-related aspects in exit criteria, e.g., testing should reach a certain code coverage. One example of how a QR is considered is that, at the feature level, it is specified that all stability tests need to have been run. These are done through ongoing automated tests. When the features are broken down to user stories and further to lower-level tasks, these user stories and tasks include details that are more technical, and these include QRs related to, for example, performance or power consumption. However, low-level DoDs do not contain quality targets, as in the feature level, but include only items to be done and to ensure that everything has been tested. It is only the higher-level DoD that contains quality targets. The stability and performance QRs are tested in automated ongoing tests in the CI.

As the value to customers is important, it is also considered during development. Company B has so-called value stream mapping (VSM) sessions during development in which they periodically check that the product is in line with the goals, and that they are producing benefits for the end user. "*We start testing the feature. Does this feature work? Is this feature in line with the goal? Does this feature produce any value?*" This also helps Company B to reprioritize QRs. In testing, Company B applies ongoing test automation for performance and stability, and basic acceptance testing is performed after every sprint until release. A separate testing group focuses on security testing only because security is of high importance in Company B. Usability is not tested until a trial phase, which is performed by trial users who are personnel at the same company.

Lessons learned

- When there are standards and regulatory needs, it is necessary to proactively plan QRs up front
- Software internal QRs (stability and performance) can be ensured by extensive ongoing tests
- With limited access to users, an internal VSM session can help to align QR priorities with goals and values
- When security is important, it is advisable to have a team of experts in security for testing
- QRs can be documented as part of DoD and acceptance criteria

4.1.3 QR management in company C

Company C applies a strategy that utilizes mostly proactive and reactive practices for the management of QRs or quality in general. As the system is large and complex, some level of

hierarchy is necessary to coordinate many development teams. Most elicitation, analysis, and specifying of QRs is done up front before implementation together with FRs in the features. Extensive testing in a multi-level CI is practiced to ensure quality as a reactive practice, and review meetings are held periodically to address quality concerns for the future. One interactive practice is also utilized for managing QRs; developers can elicit QRs during development and suggest their inclusion into the backlog.

Development is feature-driven, and all requirements, including QRs, are derived from features. In the initial phase, the elicitation, analysis, and defining of features are done by product managers. Factors such as customers, business opportunities, and the organization's roadmap are taken into consideration in the feature elicitation and analysis from which QRs are derived. In the feature elicitation and analysis, QRs are included as quality criteria for features, just like in Company B. Features are further divided into sub-features before they are split into tasks for development. The sub-features are prioritized by managers for the POs. Thereafter, POs and Scrum masters divide the sub-features into tasks for development teams. When the quality criteria are broken down for the developers, they are written down as part of the DoD as a list of certain practices to follow, certain guidelines to follow, and certain tests to run. Some of the items listed in the DoD relate to QRs—for example, unit testing to ensure robustness.

For the reactive part, extensive testing is practiced to verify that the product meets the desired quality as well as functionalities. Testing is done at several levels, depending on the duration of tests, to speed up the discovery of potential faults. Developers can perform unit tests test when compiling before they push the code to the CI. "*We have three different quality gates in the CI. More parts of the system are integrated in each gate.*" In other words, there are three levels of testing in the CI. In the case of one of the business lines, which is responsible for developing a common platform software, the most prominent QR in integration testing is interoperability with existing systems or legacy systems. Any interoperability issues that arise from testing are communicated back as faults. Lower-level integration-testing faults are communicated directly back to the developers, and the developers are responsible for carrying out root-cause analysis. Higher-level integration faults are communicated back to specific quality roles in a monthly quality meeting, who are responsible for root-cause analysis. These quality roles participate in meetings with managers, in which they present the analysis of the faults, collaboratively decide on actions to take to correct the faults, and agree upon actions to take to prevent these faults in the future. The result may be a process change or an item to put in the product backlog.

In Company C, there is also another practice used to manage quality. There is a small reservation for quality-related initiatives as a percentage of the effort of a sprint. This means that any developer can suggest including quality-related items into the backlog. These quality-related initiatives can regard both process and product improvement. For example, a developer may realize that a certain QR might be easier to ensure with a certain test that is currently not done. He or she can suggest including the implementation of the test as an item in the backlog. As another example, a developer might realize that it would be much easier to pinpoint the root cause of a fault if he or she would have an additional logging of an event in the log file that the system produces. The developer can then suggest including in the backlog, as a quality item, that log file X should incorporate the logs of event Y. The suggestions are inspected, after which management decides whether to include the new suggestion; if so, the suggestion is sent to the developers to implement. In a sense, this is a practice to elicit quality-related items, if not QRs. We call this practice *quality reservation-based elicitation of QRs* when referring to it later in the paper.

Lessons learned

- Distributed development of a large and complex system requires some up-front planning of QRs
- When interoperability with legacy systems might be challenging to elicit up front, a multi-level CI could help
- Knowledge of the experts, i.e., the developers, can be utilized to elicit QRs during development, and the QRs can be proposed for inclusion in the backlog

4.1.4 QR management in company D

The strategy to manage QRs is proactive, interactive, and reactive in Company D; however, in comparison to other cases, Company D utilizes more proactive and interactive practices. QRs in general play an important role in software development in Company D. The visual aspects of the system to be developed are an important topic of the initial scoping sessions together with the client. In these scoping sessions, mock-ups are utilized to present a possible solution for a given problem. The mock-ups are used to elicit requirements, including QRs. "*We present them the mock-up filled with some dummy data. Does it look right? Does it work as it should? Do you find all information you need? Does it respond as you expect?*" If agreement is reached, a contract is made that includes the acceptance criteria, and a second round of scoping sessions is held in which further QRs are elicited with the help of mock-ups. Initial mock-ups, together with notes on client feedback as well as user stories and epics drawn up in the scoping sessions, serve as a baseline for the following implementation process.

The PO is mainly responsible for the prioritization of requirements; however, the prioritization decision is made after consultation with the development team. These requirements are mainly functional, and QRs are included in the FRs. "*For example, the functionality should provide the user some information in tabular form. Then there is a question of how many and which fields do you want to be shown. In what order should they be displayed? How should the rows be organized in the table? Should we include paging, infinite scroll, and so on?*" These details are necessary to be able to provide the requested functionalities. These details are documented in a template that the PO utilizes for specifying some of the QRs. The interviewed PO stated that this template has been designed mainly to help his own work. This way, time is saved, as details are recorded and given directly to developers, avoiding situations in which a developer has to stop work to get additional details from the PO. With the template, the PO can communicate some of the QRs up front and is the main channel of communication of QRs at the beginning of a project. The template is also reusable in other projects.

In addition, QRs are communicated and clarified in the meetings between sprints. Daily stand-up meetings, in which the product owner also participates, are additionally used for discussing QRs or acceptance criteria, if necessary. Additionally, Company D has review meetings with the customers during implementation to check the progress of every sprint or every second sprint, depending on the contract. With the help of this strategy, which includes the client from the beginning to the end, it can be ensured that the customers get what they ask for, and that the result is of sufficient quality. Finally, some QRs are included as acceptance criteria. For example, it might be specified in the acceptance criteria that the user interface

needs to be responsive at a certain resolution, or that the front-end needs to support certain versions of certain browsers, or that support for multiple languages is implemented and working. Some aspects of QRs are included in the DoD; however, the DoD is more generic. The acceptance criteria are more detailed. "*To meet the acceptance criteria is one part of the DoD.*" Meeting the acceptance criteria is mostly achieved through manual testing at the end of development.

Lessons learned

- When accessible, users should be utilized throughout development—in elicitation, analysis, specification, and validation of QRs
- When accessible, users can be consulted during development in intermittent review meetings to verify already elicited QRs and to identify possibly missed QRs
- Mock-ups and quick prototypes are valuable when eliciting QRs from the users, especially if usability is important
- Templates are useful for eliciting important QRs that users might usually omit
- Templates are useful for specifying and document QRs, and for communicating them to developers

4.1.5 Cross-case analysis: Comparison of QR management strategies and practices

This section compares similarities and differences in QR management strategies and practices in the case companies. Many differences are attributable to the companies' contexts, which are further elaborated in Section 4.3. Table 5 summarizes the strategies and practices we found in the case studies. The overall strategies are in one category alone and the practices we categorize according to RE activities in SWEBOK 3.0 (Bourque and Fairley 2014).

What is common to all the studied cases is that they have clear quality goals and manage quality in general. The QRs are tightly coupled with FRs and are included in the acceptance criteria; however, practitioners appear to have a much better understanding of how to manage FRs, since none of the companies have as well-defined a plan for the management of QRs as they do for the FRs. For any specific QRs, the companies' strategy is to complement their development processes with practices specifically oriented toward reaching their quality goals—for instance, having a separate team for testing security or utilizing mock-ups when user experience is important.

All of the case companies utilize proactive practices in their strategies, albeit a bit differently. Companies A to C employ a company roadmap as the first high-level source of QRs. They elicit QRs from the high-level feature or epics. On the other hand, Company D, as their development is contract-based, works closely with the customers, and is able to utilize them directly for QR elicitation with the help of templates. Most of the companies also utilize interactive practices for the management of QRs. Company D works closely with the customers whenever possible, while Company B practices VSM sessions in which they check the results of development with the goals. Company A uses the tool to develop it, and thus developers can notice if any QR-related aspects are not right, while Company C utilizes the expertise of the developers by having them suggest QRs to include in the backlog. For reactive practices, validation was very similar in all companies. All

companies include some QRs in the acceptance criteria, and some are included as items in the DoD. Additionally, all companies utilize CI, to which developers contribute on a daily basis. QRs are distributed in different artifacts rather than being considered as separate items in the product backlog (Table 5).

4.2 QR management challenges

In this section, we present the challenges in managing QRs in ASD as elicited in our interviews. We categorize the individual challenges according to the area of effect. In that way, the categories follow the same structure from SWEBOK 3.0 (Bourque and Fairley 2014) as in the previous section. However, we situated implementation as an additional area of effect, in a category called "Challenges in QR implementation." The category related to specifying QRs was denoted as "Challenges in specifying QRs" instead of "Challenges in QRs specification" in order to avoid confusion with the QR specification artifact. An overview of the challenge categories and the companies can be found in Table 6. After the table, we describe each category and show what kinds of challenges can be expected in each area. We explain the reasons for the challenges and what kinds of consequences these challenges may have. We also reflect on the findings of the QR management practices and elaborate on how these challenges can be mitigated.

None of the challenge categories apply for all companies, as listed in Table 6. Likewise, none of the companies are experiencing challenges in all areas. The effect of the context is explored in Section 4.3 after the challenges have been presented.

4.2.1 Challenges in QR elicitation

Challenges related to the elicitation of QRs were reported in three of our case companies: B, C, and D. One of the companies emphasizes value to customers and users; however, in this company, they have limited access to them. *"We have very limited information about their actual use scenarios and how they use them, what they actually expect to get out of this product."* For this reason, they cannot be sure that they are getting the QRs right. For example, it might be deemed internally in the development organization that processing power is important to make all features work fast, when the context of use would not have required that, but battery power would have been more important. In such a scenario, there is risk of rework. In Company B, the practice of conducting VSM sessions during development, even though not helping elicitation directly, reduces the risk of rework.

In contrast, another company that *does* have access to customers and works closely with them also reported challenges in this area. At this company, part of the challenge came from the fact that customers, or users, rarely pay attention to software-internal QRs, like maintainability or scalability. For example, a customer may order a warehouse management system for one warehouse but fail to mention that, later, the system might need to support decentralized warehouses in different locations. Another example is language support for different languages. If the customer is a national company but fails to mention plans to expand to other countries, and thus, they may not think to ask for language support for different languages. This requirement is not purely internal, since it is also related to the usability of the system. Even though part of the challenge may be the fact that customers or users rarely pay attention to software-internal QRs, the interviewees reported the elicitation itself as a problem: "*How do you make sure that you get all the necessary QRs right from the beginning?*" It is many times the case that the PO does not have enough domain knowledge to be able to know all the important QRs. Even though Company D utilizes a template to collect details about QRs, the PO felt that elicitation techniques could be improved because there is always a risk

Table 5 Practices for QR Management in the Case Companies. RE Activities from SWEBOK 3.0 (Bourque and Fairley 2014)

	Company A	Company B	Company C	Company D
Overall QR management strategy	• Mostly reactive strategy, interactive elicitation • Proactively utilizing high-level features • QRs tightly coupled with FRs • Not explicitly specified • Relying on developers' expertise • Developers elicit QRs during development • Reactively testing in CI and acting on feedback from users	• Mostly proactive and reactive strategy • Proactively utilizing high-level epics and features • Quality targets for high-level features as QRs in acceptance criteria and DoD • QRs tightly coupled with FRs • Reactive testing in CI • Dedicated testing team for security requirements • Interactive VSM sessions	• Mostly proactive and reactive strategy • Proactive elicitation from high-level features • Specifying and analyzing proactively up front • QRs tightly coupled in features • QRs as quality targets for features in acceptance criteria and DoD • Reactive testing in CI • Quality initiative reservation in the backlog	• Proactive and reactive management and interactively evolving QRs • User is central in all phases of development • Proactive elicitation using mock-ups • QRs included in acceptance criteria • Interactively letting users validate QRs during development • Reactive manual testing to validate QRs
RE activity	**Practice in Company A**	**Practice in Company B**	**Practice in Company C**	**Practice in Company D**
Elicitation	• Derived out of high-level features and out of the functionalities • Developers using the tool for developing it • Test automation, manual testing, and customer feedback also utilized	•Derived out of epics and are part of the functionalities •Standards and regulations	• Derived out of features and out of the functionalities • Standards and regulations • Monthly quality review meetings (root causes analysis) • Developers can suggest including quality-related items	• Iterative meetings with clients before development • Utilizes mock-ups to elicit important QRs
Analysis	• No extensive up-front analysis • Discussed in daily and weekly meetings • Prioritized based on "gravity of found issue"	• Initial analysis done when elicited out of the epics • Discussed in daily and weekly meetings • Priorities follow priority of features based on value to customer	• Analyzing and specifying up front by a separate group in each business line • Priorities follow priority of features • Discussed and analyzed in daily and weekly meetings • Analyzed in separate monthly quality review meeting	• Mock-ups and reusable templates together with clients • Discussed in daily and weekly meetings • Prioritized together with functionalities in sprint planning • Discussed with clients in intermediate review meetings
Specifying	• Implicit in the FRs and verbally communicated to developers • Rely on the expertise of the developers	• Implicit in the FRs • Rely on the expertise of developers • Included as tasks to be done for a user story • In acceptance criteria as quality targets for features and as items in the DoD for tasks	• Analyzing and specifying up front by a separate group in each business line • Implicit in features • In acceptance criteria as quality targets and as items in the DoD • Some included as quality items in the backlog	• Implicit in the FRs • Reusable templates • In acceptance criteria as quality targets and as items in the DoD
Validation	• Extensive automatic and manual testing • Developers using the tool themselves while developing it • Users provide feedback through sales team and tool development forum	• Extensive automatic and manual testing • Security tested by a separate team • Usability tested by trial users • Checking the DoD and checking against acceptance criteria	• Extensive testing throughout development • Several quality gates in multi-level CI for interoperability • Checking the DoD and checking against acceptance criteria	• Automatic and manual testing • Reviewing results of development with the customer prior to release • Checking the DoD and checking against acceptance criteria

Table 6 Challenge Categories in the Case Companies

	Company			
Challenge category	A	B	C	D
Challenges in QR elicitation		X	X	X
Challenges in QR analysis	X	X	X	
Challenges in specifying QRs		X		
Challenges in QR implementation	X	X	X	
Challenges in QR validation		X	X	
General QR challenges			X	

that some important QRs might be missed and costly rework would be needed later: "*If you get everything right from the beginning, you just specify and develop. If you miss something, you do rework.*"

Another challenge related to the area of QR elicitation was the dependency between business lines, components, and development teams. One of the business lines in the biggest case company is producing a component for a common platform software. Products of the other business lines utilize this common platform software. Thus, because of the crosscutting nature of QRs, they have dependencies between different business lines, and interoperability between different components and legacy systems becomes a challenge. Interviewees stated that it is difficult to identify these dependencies: "*So, if Business Line X wants a change in our product, how do we make sure that that change does not have adverse effects on Business Line Y?*" Even though the challenge is different, and because of different reasons as in other companies, the consequence is the same—rework: "*Sometimes, when we have successfully satisfied a QR for one business line, it might come as a surprise at some point for another business line when they find out that something is not working any longer.*" The multi-level CI applied in the company helps to identify potential interoperability issues as early as possible.

Summary of challenges, consequences, and mitigation actions in QR elicitation

- Not being able to elicit all important QRs
- Consequence: Development focusing on wrong QRs
- Mitigated in Company B later in development by VSM sessions
- Customers or users might omit important software-internal QRs
- Consequence: Rework
- Mitigated in Company D by utilizing mock-ups and templates in elicitation
- Identifying QRs with dependencies is challenging
- Consequence: Rework
- Mitigated in Company C by multi-level CI to find interoperability issues early on

4.2.2 Challenges in QR analysis

This category includes challenges related to QR analysis. For example, it might be difficult to see the relationship between a specific QR and other requirements, whether a QR is process or product-related, or how to give the QRs proper priority. One of the challenges in this category is—and this is also related to the fact that practitioners find QRs to be a fuzzy concept—that "*it is difficult for us to measure quality.*" Many QRs are difficult to concretize and are more elusive and difficult to measure. When QRs are not easy to analyze and define in detail, they are accordingly not easy to measure; and consequently, they become difficult to specify.

Limited access to customers or users carries over to this area as well. Even if QRs are identified, it is still a challenge to identify what QRs bring the most value to the users; i.e., the challenge is to identify the right QRs to focus on. This makes the prioritization of QRs challenging, and the consequence is that sometimes development might focus on the wrong requirements. For example, if it is not known which communication bands the potential users would need or use, one might implement the use of many different bands, of which some would be unnecessary. One interviewee stated that the goal is to produce the minimum viable product as fast as possible; however, as in the example of communication bands, they would have already gone beyond the minimum viable product. Thus, they can feel that time has been wasted.

Dependencies pose a challenge for analysis too. Even if a dependency is identified, it is difficult to identify the exact effects of addressing that dependency on another component. If one QR is addressed from the point of view of one specific component, it might have adverse consequences on another component.

In our interviews, we found another challenge that also had a big impact on prioritization. This challenge was related to QRs competing for prioritization against FRs. Interviewees in one company said that customer features, and thus the FRs, get more priority than software-internal QRs. Interviewees stated that "*you need to give a good reason why to develop something quality-related,*" "*it is difficult to convince product managers why they should focus on quality,*" and "*it is hard to show the value of QRs in the backlog.*"

Summary of challenges, consequences, and mitigation actions in QR analysis

- Challenging to know what to measure
- Consequence: Unable to identify and know how to monitor and control quality
- No mitigation practice identified from the practices of the companies of this study
- Prioritizing QRs is challenging
- Consequence: Development focusing on wrong QRs
- Mitigated in Company B in VSM sessions by reflecting QRs to goals and targets
- Identifying QR dependencies across components is challenging
- Consequence: Not knowing the effect of a QR on all components
- Mitigated in Company C by multi-level CI to identify dependencies
- Focus on delivering functionality to customers is a challenge for QRs
- Consequence: QRs get less priority compared to FRs
- No mitigation practices identified from the practices of the companies of this study

4.2.3 Challenges in specifying QRs

We found only one challenge that we can categorize as a challenge for the activity of specifying QRs. One company specifies requirements in user stories and tries to also include QRs in those user stories. However, interviewees felt that these user stories are still many times too technical. "*It should not be a technical explanation that says, 'add a button here.' There needs to be the 'why' and the value behind that.*" One consequence that was mentioned, of not having enough background information or detail about a QR in the user story, was that even though functionality—i.e., what needs to happen—is described in detail, it may not say *how*. It might be that there are two ways to implement the "what" part, but only one is preferred. The "why" part could help identify the right "how" part.

Summary of challenges, consequences, and mitigation actions in specifying of QRs

- Use of user stories for specifying requirements is challenging for specifying QRs
- Consequence: QRs are not easy to specify in user stories and unclear QR specification
- No mitigation practice identified from the practices of the companies of this study

4.2.4 Challenges in QR implementation

Most of the individual challenges that were reported by interviewees are related to implementation. Some of these challenges might be seen as belonging to the previous category, Challenges in specifying QRs; however, they were not reported as challenges while specifying QRs. The results of the activity of specifying requirements caused challenges in implementation.

One interviewee in one company reported that "*granularity is a challenge; the specification is not detailed enough.*" That is, specifications are unclear or not always detailed enough for the developers to easily implement. Similarly, in another company, the unclear specification was reported as a challenge several times. However, in this company, the main reason was in some cases that specification was done by a separate group of people without the involvement of developers. Interviewees reported that "*specification is done without collaboration with developers*" and "*specification people and developers are not working together.*"

The developers also expressed that sometimes they get the feeling of "*are we even doing the right thing?*" Contributing to this was that the developers did not have visibility of the original requirements. Interviewees reported that "*the big picture is missing*" and that there was "*no visibility to customers*" in some of the business lines. Because developers have limited visibility to the customers or the original requirements, the developer expects that the specification is precise and unambiguous. However, developers explained that sometimes the specification leaves room for interpretation. The developer might have several options for how to implement a functionality, but how the developer ultimately implements this functionality may have architectural implications or might affect performance. This is why the developers feel it is important that they see the

original requirements or the big picture. As a result, when developers are not involved in carrying out the specification and when they have limited visibility of the original requirements—i.e., they do not see the "why" part—they might not understand the specification as intended. This can lead to an issue whereby the development is focusing on the wrong things or, once again, to more rework later on.

Summary of challenges, consequences, and mitigation actions in QR implementation

- Unclear or insufficiently detailed specification of QRs leaves room for interpretation
- Consequence: Implementation of QRs becomes difficult
- Consequence: Focusing on wrong QRs
- Consequence: Development is slowed down
- Mitigated in Companies B and D by including developers in the activity of specifying QRs
- Mitigated in Company A by increasing developers' visibility to the original requirements

4.2.5 Challenges in QR validation

Interviewees in two of the companies reported challenges that can be considered relevant in the validation of QRs. Again, limited access to users is the cause for one of these challenges. Because of this limitation, one interviewee reported that they receive feedback about QRs too late. Although one company practices VSM sessions and tests the product with internal trial users, the interviewee felt that they cannot be absolutely sure that they have been able to provide all the value a real user would want. The internal trial user can never replace the real end-user because the trial user does not operate the product under the same conditions. Additionally, the internal trial user might be biased.

Other reported challenges were "*limited traceability to the big picture*" and "*some QRs seen first on high-level integration testing.*" The first of these challenges is caused by developers having limited visibility of the original requirements during implementation. In a big organization, when the requirements are split into smaller parts and distributed to many teams, and when the intra-organizational distances start to grow, there is the risk that traceability will be lost. This is related not only to QRs but to any requirements. When a fault is found, the root cause analysis will be slowed down if traceability is lost.

The second challenge is related to the complexity of the dependencies of large complex systems. Even though a multi-level advanced CI system may find faults related to dependencies as early as possible, some dependency issues might still not be discovered until late system testing. When this happens, it produces an additional challenge. Testing a large and complex system takes a long time, and finding the right root cause for a fault might also be time-consuming. During that time, a team of developers might have already moved on to the new tasks in the following sprints; and since teams are dynamic—i.e., they are put together based on needed competences—it might be difficult to find the developer who was developing the part of the component that caused the fault. "*It is difficult to find the responsible guy much later after the sprint has ended,*" one interviewee stated.

Summary of challenges, consequences, and mitigation actions

- Too late feedback regarding QRs
- Consequence: Focus on wrong QRs, or QRs important to the user go unnoticed
- Mitigated in Company B by VSM sessions
- Mitigated in Company B by internal trial users
- Maintaining traceability of QR-related test cases to design artifacts and requirements is challenging
- Consequence: Finding root cause for faults may take a long time
- No mitigation practice identified from the practices of the companies of this study

4.2.6 General QR challenges

The challenges included in this section have a more holistic effect on development, not just on specific parts. They are more general, affect several areas of development, and are context-independent. Two of these reported challenges are related to the tendency of development to focus more on functionality than on the cost of QRs. Interviewees of one company reported that "*because of up-front planning and in case of tight schedules, quality gets less attention,*" and "*the improvement frame has decreased over time.*" Both challenges have effects on several areas of development. The reservation for quality-related improvements in the backlog was in place in one of the companies to enable developers to elicit QRs with regard to, for example, maintainability. However, it was stated by interviewees that these QRs have a limited possibility of getting enough priority and being included in the product backlog. If the QR in question would be related to maintainability, it would have severe effects later. The final challenge included in this category was reported as "*QRs are generally more challenging than FRs.*" This is included here, even though explicitly mentioned in only one interview, because the notion the interviewees had about QRs suggested that QRs are many times difficult to grasp. This confirms that practitioners find the QR to be a fuzzy concept.

Summary of challenges and consequences, and mitigation actions

- More focus on FRs than on QRs
- Consequence: Challenges in all areas of development
- No mitigation practice identified from the practices of the companies of this study
- QRs are a fuzzy concept
- Consequence: Challenges in all areas of software development
- No mitigation practice identified from the practices of the companies of this study

4.3 Context

We will now inspect the differences in QR management and related challenges and explain these differences with the help of the different contexts of the companies. We started by studying which areas in the companies faced challenges, and what kinds of challenges were faced. Next, we studied the differences in the challenges and the contextual elements of the companies in order to determine whether the contextual elements could explain why some companies experience challenges while others do not. When we found a specific contextual element causing a certain challenge, we realized that the same element could also work as a driver for applying a certain practice. That way, we could explain why a specific company was not experiencing a certain challenge while another company was. This led us to map contextual elements to different practices applied in the management of QRs. This way, we identified nine different contextual elements from the four different context facets that may affect the choice of practices or the challenges related to QR management in ASD. From the product facet, the elements were product type, maturity, and customization. From the process facet, the elements were the characteristics of the development process, whether or not development was distributed, and the number of employees involved in the development of the product or component. From the organization facet, the element was whether the organizational model was hierarchical. From the market facet, the element was the type of customers or users and access to customers. Tables 7, 8, 9 and 10 summarizes the contextual elements driving the practices that explain the challenges which are summarized in Tables 11, 12, 13, 14, 15 and 16.

4.3.1 Context and QR management practices

In this section, we start by inspecting the management strategies and practices per RE activity as defined in SWEBOK 3.0 (Bourque and Fairley 2014) and in Table 6. We identified seven different contextual elements affecting the choice of strategy and practices: product type, maturity of the product, customization, characteristics of the development process, distributed development, hierarchical organizational structure, and access to customers or users.

Elicitation of QRs All the companies apply a proactive approach and try to elicit QRs up front before implementation, although differently. Company D utilizes the users of the product for QR elicitation. Ultimately, there will be users using the system whose productivity might depend on the system. For this reason, they need to work closely with the users to develop the right solution. As such, the product and market facets of the context affect the choice of practice in QR elicitation. This would naturally also require that there is access to the users of the product. This is not the case in Company B, which relies on company goals and roadmaps for the product to elicit important QRs. Also, Company A has access to the users; however, the company does not interact with users in the same manner as Company D. The product that Company A develops is a mature tool that already has an established set of requirements. Therefore, the company does not have the same need to elicit specific QRs. The users are more valuable in giving feedback about how the product performs and compares with other, similar tools. The company uses this feedback as a technique to elicit additional QRs. Additionally, the

Table 7 Contextual Elements Driving Practices Used for Eliciting QRs

Contextual elements	Practice applied in case companies
Access to users	Mock-ups and templates for elicitation of QRs
Mature product and access to users	Feedback from product usage for elicitation of new QRs
Limited access to users	Company goals and roadmaps used for elicitation of most important QRs
Large and complex system	Multi-level CI for identification of QRs with dependencies between components

company has adopted a development mode that is highly reactive to this feedback. This can also be the reason why Company A does not experience any challenges in the elicitation of QRs. Company C is producing a large system with complex dependencies. It is challenging for the company to elicit all the QRs that are affected by the dependencies of different components up front, so as an aid it utilizes the multi-level advanced CI to do so.

Analysis of QRs The situation in the analysis of QRs is very similar to elicitation, and for the same reasons. Company D has access to customers or users and utilizes them to analyze the QRs together with the customer using mock-ups and reusable templates. Company B does not do this because of limited access to users, and Company A does not experience the need to do it together with the users due to the maturity of the product. In Company A, there is also another element of the product facet that affects the choice of not analyzing all QRs to the smallest detail: the customization element. The product is the same for all users. This means that the product can be installed in a great variety of environments, and it is virtually impossible to identify all of those environments. Thus, it is easier to address, for example, the main stability issues and address other issues as soon as they emerge. Additionally, the developers use the product to develop it. Company B, which has limited access to users, practices VSM sessions to analyze QRs. All companies evolve the QRs together with FRs during development by analyzing and reviewing the QRs further in daily and weekly meetings, although in Company C this practice is somewhat limited because all requirements are analyzed and specified by a separate internal organization. It could be said that the hierarchical organizational model and distributed development together dictate this choice.

Table 8 Contextual Elements Driving Practices Used for Analyzing QRs

Contextual elements	Practices applied in case companies
Access to users	Mock-ups and templates used further for analysis of QRs
Limited access to users	VSM sessions to analyze QRs
Mature general product	Development process highly responsive to feedback as QRs emerge
Distributed development with hierarchical control mechanism	Most analysis and specification of QRs done up front

Table 9 Contextual Elements Driving Practices for Specifying QRs

Contextual elements	Practices applied in case companies
Distributed development with hierarchical control mechanism	Most analysis and specification of QRs done up front
Characteristics of development process (agile)	Less extensive up-front specification of QRs relying on developer's expertise to implement QRs

Specifying QRs Many of the interviewees stated that they did not specify QRs; however, some of the QRs are still specified in the acceptance criteria, while others are specified as quality-related items and are included in the DoD. Interviewees in Company A reported that they do not exercise extensive specification of QRs, but they instead rely on developers' experience and expertise to consider QRs during implementation. The same applies for Companies B and C. Company B tries to include QRs in user stories but reports doing so as challenging. Company C, on the other hand, aims to conduct the most extensive up-front specifications. The nature of the distributed development affects this choice, since QRs can influence the development of several teams. Company D utilizes a template for specification and documentation of some of the QRs up front and communicates them to developers in this template. However, the company also elaborates QRs further during development. It could be said that most companies avoid specifying QRs too heavily up front, which is in accordance with agile methods. Thus, the characteristic of the development process would influence this strategy.

Validation of QRs CI and acceptance criteria seem to be the chosen practice to ensure the quality of the product in all companies. Multi-level CI appears to be especially useful with large and complex systems. In addition to testing, Companies A and D utilize the users in validation as well. As explained earlier, Company A utilizes user feedback to correct any QR issues. Although this can be seen as a very late reactive practice, since it happens after release, it is less resource-demanding, and the company has adopted a highly reactive development process. In Company D, review sessions are conducted together with the customers or the users, and thus any QR issues can be corrected before release. The contextual elements of access to customers or users and customization explain this difference. Company D is developing a custom solution for a specific problem, while Company A is producing a general product that can be used in many situations and diverse environments. In the case of Company A, it would be an overwhelming task to attempt to address all the contexts of use of the tool to identify all possible quality issues. Thus, it makes more sense for the company to instead react to quality issues found by users as fast as possible after release. If it is not possible to get the final end-users to test the product, then the development organization should have a group of internal people test the product, as is done in Company B.

Table 10 Contextual Elements Driving Practices for Validating QRs

Contextual elements	Practices applied in case companies
Access to users	Review meetings together with users during development
Mature product and access to users	User product usage feedback
Limited access to users	Internal trial users
Large and complex system	Multi-level CI

Table 11 Contextual Elements Contributing to Challenges in Elicitation of QRs

Contextual element	Effect in elicitation of QRs
Limited access to users	Increases difficulty of eliciting right QRs
Large complex system	Increases difficulty of eliciting dependences of QRs

4.3.2 Context and challenges

In this section, we explore the connection between context and the identified challenges in each area of effect and aim to elucidate how the contextual elements may cause challenges. We use the same grouping of challenges as in Table 6. We found five contextual elements that cause challenges in the different areas: access to customers or users, hierarchical organizational model, characteristics of the development process, and type of product.

Contextual elements and challenges in QR elicitation From the interviews, we could identify two contextual elements that cause challenges in QR elicitation. One such challenge comes from the constraint of the market facet, i.e., limited access to customers or users. This has a negative effect on elicitation. If it is difficult to elicit the right QRs together with the customers or users, it is even more so without access to the customers. The other contextual element that causes challenges is the system with complex dependencies of the product-type facet. It is not always easy to see all dependencies up front. Additionally, we found the challenge of how to get all important QRs from the users. However, we do not believe that this challenge is caused by any contextual element but is rather a more general challenge in the elicitation of requirements.

Contextual elements and challenges in QR analysis Two contextual elements cause challenges in QR analysis. The challenge of identifying what is most valuable to customers, which affects the prioritization of QRs, is derived from limited access to customers or users in one case. The other contextual element is the system with complex dependencies. For example, the interoperability QR can be known, but what is not known is exactly how different parts of the product will be affected, and it is thus difficult to analyze dependencies between the tasks of different teams. Additionally, there is one challenge affecting QR analysis that cannot be mapped with a contextual element: QR is a fuzzy concept. Because of this, some QRs can be difficult to concretize and measure. This is, however, a more general challenge.

Contextual elements and challenges in specifying QRs. Regarding the activity of specifying QRs, we found that the only challenge is the difficulty of including QRs in user stories. User stories are widely used in agile methods. Thus, we can say that this challenge originates from the contextual element of the characteristics of the development process. For this challenge, we did not find an effective mitigating practice. The interviewees themselves suggested that user

Table 12 Contextual Elements Contributing to Challenges in Analyzing QRs

Contextual element	Effect in analysis of QRs
Limited access to users	Increases difficulty of identifying QRs of value to users
Large complex system	Increases difficulty of identifying effect of QRs across components of the system

Table 13 Contextual Elements Contributing to Challenges in Specifying QRs

Contextual element	Effect in specification of QRs
Characteristics of development process (agile)	QRs difficult to specify in user stories

stories should include additional information about QRs. How user stories could be extended or enhanced to also include QRs is a topic for further research.

Contextual elements and challenges in QR implementation Most of the challenges were related to implementation. One reported challenge was that specification is not detailed enough. In this particular company, it can be said that the challenges are derived from the characteristics of the development process. The company has chosen not to spend much time on specifying or documenting the QRs. This is in line with the Agile Manifesto (Fowler and Highsmith 2001). In another company, most of the challenges in implementation, like unclear specification, misunderstandings of specification, and the missing "big picture," are also rooted in the activity of specifying the QRs. However, in this company, the requirements were specified without the involvement of developers. Thus, it can be said that this challenge comes from the hierarchical organizational model. The distributed development and plan-driven characteristics of the development process also contribute to these challenges. Requirements are specified up front, and the developers are detached from the original requirements.

Contextual elements and challenges in QR validation The challenges in validation in one of the companies are once again related to limited access to users. Interviewees stated that they received feedback from users quite late, and that they would like earlier user feedback. In addition, the same contextual elements of hierarchical organizational model, distributed development, and characteristics of the development process also contribute to the challenges in validation. Final system-level integration is accomplished late from the perspective of a development team in one company. Because of this, a development team may obtain feedback from integration testing long after the related sprint has ended. At that time, root-cause analysis might take a long time, and it might be difficult to find the developer who was developing a specific part of a component.

Contextual elements and general QR challenges Two of the challenges in this category were related to the focus of development on functional features at the cost of focus on QRs. It could be said that this is inherited from agile development and thus from the characteristics of the development process contextual element. However, in one of our cases, the contextual

Table 14 Contextual Elements Contributing to Challenges in Implementation of QRs

Contextual element	Effect in implementation of QRs
Characteristics of the development process (agile)	Minimum effort for QR specification results in ambiguous specification of QRs
Hierarchy in development organization	Increased difficulty of implementing QRs if developers not involved in specifying QRs
Characteristics of the development process (plan-driven)	Up-front planning decreases response to changes
Distributed development	Increased risk of developers not having visibility to the original requirements

Table 15 Contextual Elements Contributing to Challenges in Validation of QRs

Contextual element	Effect in validation of QRs
Limited access to users	Increases risk of getting valuable user feedback too late
Hierarchy in development organization	Increases difficulty of finding root cause for faults

element of type of customer also contributed to the focus on functionality. If the customers are company-internal—that is, the affected users are developers and engineers—QRs might receive less priority if end-user functionality is not directly affected. The other reported challenge was that QRs are generally more difficult to manage. This challenge has no connection to context. To raise the awareness of quality and QRs among all stakeholders, it could be helpful to shift the focus more toward QRs than functionalities alone.

4.4 Theoretical model

This section presents the results and analysis from our multiple case study in the form of a theoretical model of QR management challenges and practices in ASD. The model was designed to systemize the findings obtained from the present study. For this purpose, we used the methodology proposed by Sjøberg et al. (2008), which was developed to generate theories that can explain or predict phenomena occurring in software engineering. Our model can be used as the basis for explaining the relationship between contextual elements and QR management challenges, and it can thus be used to predict the challenges that companies can face in certain contexts. Further research may consolidate the findings of the present study, yield new findings, or refute some of our propositions.

According to Sjøberg et al. (2008), a theory is formed by introducing its main constructs and then establishing propositions as relationships among them. Constructs can be (1) classes defined as specializations or components of one of the four archetype classes, namely Actor, Technology, Activity, and Software System, or (2) attributes inside these classes. Archetype classes are predefined in Sjøberg et al.'s approach; their purpose is to provide a uniform terminology and structure to formulate theories in software engineering. Propositions relate a source and a target element. The source of a proposition can be a class or an attribute, and the target can be an attribute or another proposition. When the target of a proposition is another proposition, this means that the source element affects the direction and/or strength of the effect of the target proposition.

Before describing the theoretical model, we illustrate its usefulness by means of proposition examples that belong to the resulting theoretical model. On the one hand, the propositions are able to express the relationships between contextual elements and QR management challenges. For instance, the proposition "Limited access to users increases the difficulty of eliciting QRs in ASD" relates a source contextual element, i.e., the limited access to users of the develop-

Table 16 Contextual Elements Contributing to General Challenges Related to QRs

Contextual element	Effect on QRs in development overall
Characteristics of development process (agile)	Increases risk of QRs getting too little priority compared to FRs
Type of customer (internal customers)	Increases risk of QRs getting too little priority compared to FRs

ment team actor, and a target element representing a challenge, i.e., the difficulty of the elicitation activity of the QRs. Furthermore, propositions can also relate practices to the challenges they face. For instance, the previous proposition is itself the target of another proposition, meaning that there exists a source element that affects its strength—that is: "Use of roadmaps and value goals for QR elicitation decreases the difficulty of eliciting QRs when there is limited access to users in ASD."

The rest of the section presents a description of the whole set of constructs and propositions of the theoretical model.

4.4.1 Constructs

In this subsection, we present the constructs that we have defined for each of the predefined archetype classes and illustrate them graphically in Fig. 4.

The diagram shown in Fig. 4 follows a UML-based notation proposed by Sjøberg et al. (2008). Classes are drawn as boxes. Specializations between classes are denoted by the UML generalization arrow, e.g., *Elicitation practice* is a subclass of *QR management practice*. Component classes are depicted as boxes within another box, e.g., *Project* is a component of *Organization*. Attributes are textually represented inside their corresponding classes. As

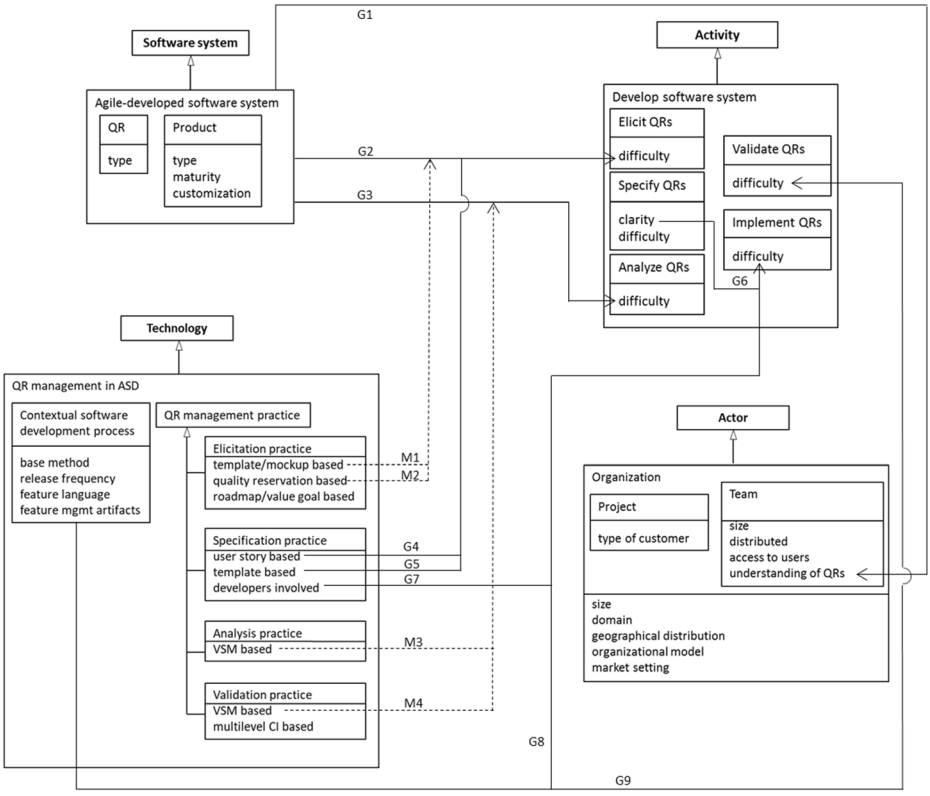

Fig. 4 A theoretical model for QR management in ASD: constructs, general propositions (solid arrows), and their mitigation propositions (dashed arrows)

already done above, terms that appear in the model are written in italics in the text. The arrows in Fig. 4 represent propositions and will be presented in Section 4.4.2.

- *Software system*: This class represents the scope of the study. In our case, we are interested in a specific type of system, i.e., software systems developed using ASD (*agile-developed software system*). A specific agile-developed software system should satisfy certain *QRs* that correspond to a QR *type*. The software system is involved in a software product (e.g., an embedded system, a software tool) that has a *certain* type (e.g., large with complex dependencies), a *maturity* level (e.g., a long-lived mature product, a new product) and a *customization* degree.
- *Technology*: This class represents the focus of the study. Following Sjøberg et al. (2008), this archetype class must be interpreted in a broad sense, and the focus of the study can range from a process model, method, or technique to a tool or language. In our case, the focus is the *QR management process in ASD*. To understand how QRs are managed in a particular context, it is necessary to consider:

    1. The *contextual software development process* that is being followed. From a particular process, it considers the method taken as a basis (*base method*, e.g., Scrum, Kanban), its *release frequency*, the *feature language* used (e.g., natural language), and its *feature management artifacts* (e.g., product backlog).
    2. The *QR management practices* that are being applied. A QR management practice is classified as an *elicitation practice*, *specification practice*, *analysis practice*, or *validation practice*. The aspects to be considered in each practice appear as attributes in the corresponding class.
- *Actor*: The main actor in the theoretical model is the *organization* that manages the development of the software system. According to our study, the context attributes to consider in a particular organization are *size*, *domain*, *geographical distribution*, *organizational model*, and *market setting* (e.g., business-to-user, niche market, business-to-business). In an organization, we identify as crucial actors the *project* and *team*. For a project, we consider the *type* of customer (e.g., national organizations, software developers). For a team, we consider its *size*, whether it is *distributed* or not, whether it has *access to users* or not, and the *understanding of QRs* of its members.
- *Activity*: The main activity studied is the development of a software system (*develop software system*). The important sub-activities of this main activity are the following: *elicit QRs*, *specify QRs*, *analyze QRs*, *implement QRs*, and *validate QRs*. For each of these activities, it is relevant to know about its *difficulty*; and for QR specification, the clarity of its results is also relevant.

4.4.2 Propositions

The propositions obtained from our multiple case study can be classified as general, context, or mitigation propositions. General propositions are those that specify relationships not dependent on contextual aspects and, hence, they are susceptible to appear in any context (e.g., "In ASD, there is no shared understanding of the concept of QR"). Context propositions state relationships in which contextual elements are involved (e.g., "Limited access to users increases the difficulty of eliciting QRs in ASD" is a proposition about contexts in which the team has limited access to users). Finally, mitigation propositions specify cases of elements that can mitigate the negative effects stated by the previous general or context propositions (e.g., "Use of roadmaps and value goals for QR elicitation decreases the difficulty of eliciting

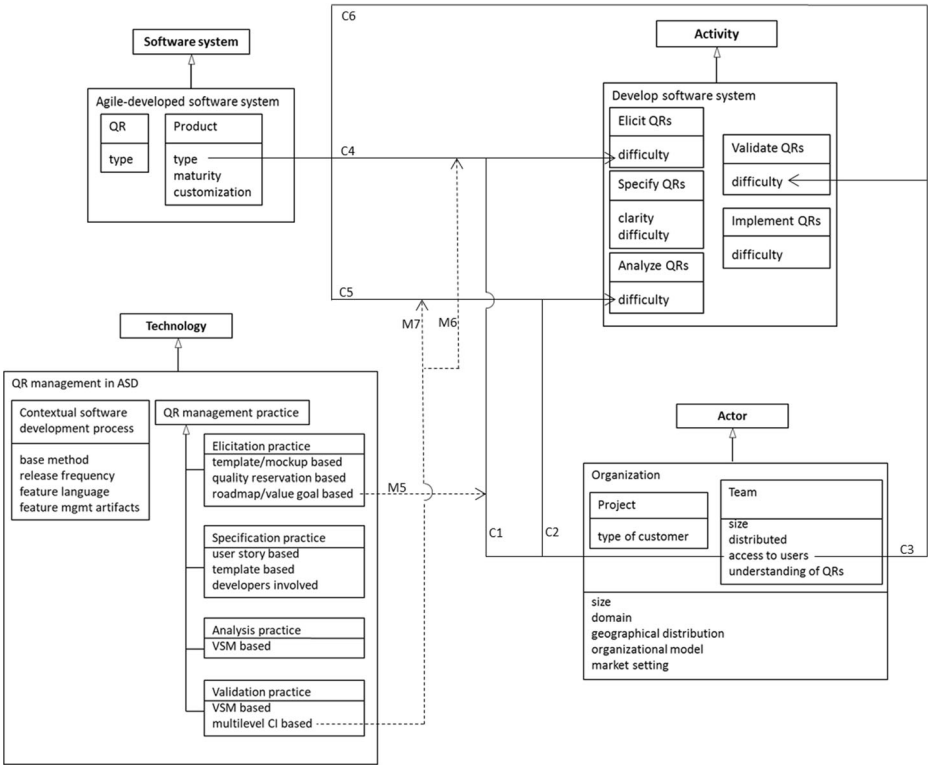

Fig. 5 A theoretical model for QR management in ASD: constructs, context propositions (solid arrows), and their mitigation propositions (dashed arrows)

QRs when there is limited access to users in ASD" is a proposition about mitigating the negative effects on QR elicitation of limited access to users).

The propositions can be graphically represented, following Sjøberg et al.'s (2008) notation, by means of arrows that connect their sources and targets. For readability purposes, we have distributed the representation of our propositions in two figures. Fig. 4 illustrates general propositions (solid arrows) and their corresponding mitigation propositions (dashed arrows), while Fig. 5 illustrates context propositions (solid arrows) and their corresponding mitigation propositions (dashed arrows). The constructs are the same in both figures. The definition of the whole set of propositions is presented in Table 17. Table 17 also provides, for each proposition, an explanation that justifies it in terms of the results of the study. General, context, and mitigation propositions are referred to as G*k*, C*k,* and M*k*, respectively, and their corresponding explanations as EG*k*, EC*k,* and EM*k*, respectively.

5 Discussion

In this section, we discuss the key findings and compare them with the scientific literature. Afterward, based on the findings, we discuss the implications for researchers and practitioners alike. Finally, we discuss the limitations of our research.

Table 17 Theoretical model Propositions and Explanations

General Propositions		Explanations	
G1	In ASD, there is no shared understanding of the concept of QR.	EG1	From the interviews at all companies, it was found that QR is a fuzzy concept among practitioners.
G2	In ASD, it is difficult to elicit all the important QRs.	EG2	From the interviews at one company, it was found that difficulties in eliciting all the important requirements in ASD arise from the observation that customers rarely focus on SW-internal QRs and also from the observation that the PO may not know all the domain QRs due to a lack of domain knowledge.
G3	In ASD, it is difficult to analyze QRs.	EG3	From the interviews at two companies, it was found that QR is a fuzzy concept among practitioners (difficult to concretize), making quality difficult to measure for practitioners.
G4	In ASD, it is difficult to specify QRs using user stories for QR specification.	EG4	The interviewees of the company where user stories are used to specify QRs reported that it is difficult to specify QRs as a challenge.
G5	Use of separate templates for QR specification decreases the difficulty of specifying QRs in ASD.	EG5	The interviewees of the company where a separate template is used to specify QRs did not report as a challenge difficulty in specifying QRs.
G6	In ASD, unclear specification of QRs increases the difficulty of implementing QRs.	EG6	Unclear specification of the QRs was reported as a challenge for the developers to easily implement them by interviewees at two companies.
G7	In ASD, specifying QRs without developers' involvement increases the difficulty of implementing QRs.	EG7	The specification of QRs done by a separate team without developers' involvement was reported by interviewees at one company as a challenge to implementing them.
G8	In ASD, limited visibility of the original requirements of developers increases the difficulty of implementing QRs.	EG8	Limited visibility of the original requirements to developers and the corresponding limited traceability to the big picture was reported by interviewees at one company as a challenge to implementing QRs.
G9	In ASD, limited visibility of the original requirements (or limited traceability to the big picture) of developers increases the difficulty of validating QRs.	EG9	Limited visibility of the original requirements to developers and the corresponding limited traceability to the big picture was reported by interviewees at one company as a challenge to validating QRs.
Context Propositions		Explanations	
C1	Limited access to users increases the difficulty of eliciting QRs in ASD.	EC1	The interviewees of the company with limited access to users reported that this fact increases the difficulty of eliciting QRs in ASD.
C2	Limited access to users increases the difficulty of analyzing QRs in ASD.	EC2	The interviewees of the company with limited access to users reported that this fact increases the difficulty of analyzing QRs in ASD, e.g., it makes it difficult to identify which QRs bring more value to the users.
C3	Limited access to users (too-late feedback from users) increases the difficulty of QR validation.	EC3	Limited access to users was reported by interviewees of one company as a challenge for QR validation because feedback about QRs is received too late.
C4	A large product with complex dependencies increases the difficulty of eliciting QRs in ASD.	EC4	The interviewees of the company with a large product with complex dependencies reported that it is difficult to elicit QRs. Because of the crosscutting nature of the QRs, they have many dependencies between business lines, components, and development teams that are difficult to identify.
C5	A large product with complex dependencies increases the difficulty of analyzing QRs in ASD.	EC5	The interviewees of the company with a large product with complex dependencies reported that it is difficult to analyze the QRs. Because of the crosscutting nature of the QRs, they have many dependencies between business lines, components, and development teams, and it is difficult to analyze the effect of addressing a QR in one component on another component.

Table 17 (continued)

C6	A large product with complex dependencies increases the difficulty of validating QRs in ASD.	EC6	In the company with a large product with complex dependencies, it was reported that it is difficult to validate the QRs since some dependency issues might not be discovered until high-level integration testing, the testing of a large and complex system may take a long time, and finding the cause for a fault may also take a long time.
Mitigation Propositions		**Explanations**	
M1	Use of mock-ups and templates for QR elicitation decreases the difficulty of eliciting all the important QRs in ASD.	EM1	For one of the companies, it was reported that the difficulty to elicit all the important requirements (G2) was mitigated by using mock-ups and a specific template for QR elicitation.
M2	Use of reservation for quality items in backlogs for QR elicitation decreases the difficulty of eliciting all the important QRs in ASD.	EM2	For one of the companies, the reservation for quality items in the backlog was reported to facilitate the elicitation of all the important QRs (G2).
M3	VSM sessions for QR analysis decrease the difficulty of analyzing QRs in ASD.	EM3	For one of the companies, the practice of conducting VSM sessions for QR analysis mitigated the effects of the challenge related to analyzing QRs (G3).
M4	VSM sessions for QR validation decrease the difficulty of analyzing QRs in ASD.	EM4	For one of the companies, the practice of conducting VSM sessions for QR validation mitigated the effects of the challenge related to analyzing QRs (G3).
M5	Use of roadmaps and value goals for QR elicitation decreases the difficulty of eliciting QRs when there is limited access to users in ASD.	EM5	For one of the companies, the practice of using roadmaps and value goals for QR elicitation decreased the challenge posed by the limited access to users to elicit the QRs (C1).
M6	Multi-level CI for QR validation decreases the difficulty of eliciting all the important QRs in cases of a large product with complex dependencies in ASD.	EM6	For one of the companies, the challenge of eliciting QRs in the context of a large product with complex dependencies (C4) was mitigated by testing the system in a multi-level advanced CI instead of spending resources on identifying dependencies in advance.
M7	Multi-level CI for QR validation decreases the difficulty of analyzing QRs in cases of a large product with complex dependencies in ASD.	EM7	For one of the companies, the challenge of analyzing QRs in the context of a large product with complex dependencies (C5) was mitigated by testing the system in a multi-level advanced CI instead of spending resources on analyzing the dependencies in advance.

5.1 Key findings and reflections on the previous literature

We divided this section into three subsections—practices, challenges, and context—for reflecting on our findings in light of the previous literature.

5.1.1 Challenges

In our study, the challenges stated by interviewees were grouped according to area of effect in order to help connect the contextual factors rather than classifying them according to themes of the challenges. Still, we could see that most of the challenges that we found can also be found in the scientific literature. Some of these challenges are: QR is a fuzzy concept among

practitioners (e.g., Alsaqaf et al. 2019), users do not focus on software-internal QRs (e.g., Aljallabi and Mansour 2015), user stories are challenging for QRs (Rodríguez et al. 2009), and feedback regarding QRs is obtained too late (Alsaqaf et al. 2019). A more comprehensive list of all the QR-related challenges reported in the scientific literature can be found in our SMS (Behutiye et al. 2019), and as such there is no need to repeat them here.

Not all of the challenges found in literature were identified in our study, nor did we find any new challenges to add. However, we were able to provide more details for some of the challenges presented in literature to help understand the challenges better. From what can be found in the literature, and from what we did not find, we can point out a few examples. In the literature, it is reported that QR implementation many times relies on tacit knowledge (e.g., Heikkilä et al. 2015). This was the case in some the companies in our study, to some extent; however, it was not explicitly stated as a challenge. Instead, some interviewees said that in the case of unclear specification, they would have liked to have had visibility of the original requirements to help guide them in the implementation of QRs. This can be limited in companies in which developers are not involved in specifying requirements. Additionally, relying on tacit knowledge can be problematic for software development in the domains of safety and security (Knauss et al. 2017). Although one of our companies used Scrum almost by the book, and safety and security were major concerns for this company, it did not adopt any structured framework, such as Safe Scrum (Stålhane et al. 2012) but instead complemented the Scrum process with additional documentation for security and safety and a separate team specialized in testing security. The use of user stories is considered a strength in ASD, but it is difficult to include QRs in user stories (Alsaqaf et al. 2017; Savolainen et al. 2010; Bourimi and Kesdogan 2013). Three of our cases utilized user stories, but the challenge was reported by only one of them.

What is more interesting is that some of the QR management challenges found in our study are very specific to the companies and are often not a challenge in ASD only. For example, one stated challenge was limited access to user scenarios. This was due to the company's attempt to enter a niche market in which it had not yet established good connections with the market's users. The challenge in this regard would have been present irrespective of the mode of development. Complex dependencies across different components of a large system was another company-specific challenge that is directly related to the complexity of the system under development. This complexity would be present in any kind of development. In fact, it might be argued that this particular challenge is less so in ASD that applies CI since less effort is spent up front on something that is very difficult to find or specify up front.

Other interesting points are that the challenges we found were relatively scarce—for example, challenges in the specification of QRs came only from one company; in Company D, interviewees mentioned challenges only in QR elicitation. We did, however, identify the general challenge that QRs are fuzzy, as also found in literature (Alsaqaf et al. 2017), and it is not easy to know what to measure regarding some QRs. Should this be the case, then, consequently, it would be difficult to specify QRs. One reason, at least in one company, could be the fact that it simply chose not to specify QRs in detail. In Company A, the most important QRs of the mature product had already been established, so for this company it was not cost-effective to expend efforts up front on trying to specify something about which it was uncertain. Instead, the company relied on CI and usage feedback. On the other hand, most companies claimed that they specify QRs in DoD and acceptance criteria. Specifying QRs in the acceptance criteria is part of Requirements Specification for Developers (RSD) approach, which has been evaluated in small companies, and which includes an extended acceptance criteria targeted for software-internal requirements (Medeiros et al. 2020). The approach also

includes the use of mock-ups. In Company D, the mock-ups and templates were utilized for elicitation, analysis, and specification of QRs. With the help of the mock-ups and templates, the company actively elicited internal QRs in such detail that they could be specified for developers. In that sense, the practice is similar to RSD. In the largest of our companies, Company C, specification was not always best suited for developers. Perhaps even large-scale development could benefit from the RSD approach.

Not finding as many challenges as would be expected based on the scientific literature and on the number of interviewees may imply that companies applying ASD are becoming increasingly quality aware, and that they are already finding ways to manage QRs. This could also be supported by the finding that many challenges can be seen as non-specific for ASD. Those QR management challenges that were previously ASD-specific are perhaps gradually being resolved, while those that remain are QR management challenges that depend purely on context or that have always been an issue—for example, how to collect all relevant QRs. In cases of large systems with complex dependencies in which some level of hierarchy is need for decision making, there is an increased general need to pay careful attention to aspects such as documentation, communication, and traceability, as well as to generally and carefully align RE with testing through people, processes, and artifacts. This alignment has been the interest of many large software development organizations in the automotive and avionic domains (Karhapää et al. 2017), and QR management could benefit from it.

Since Alsaqaf et al. (2019) is the most recent study, and since one of our companies operates in large-scale ASD, a brief comparison is warranted. We can see similarities in findings regarding QR challenges, however, we have treated these findings somewhat differently in our studies. For example, one of the companies in our study stated that "users do not pay attention to software-internal QRs," highlighting this as a challenge for which the company had developed a mitigation practice. However, in the study by Alsaqaf et al., "customers are not interested in internal QRs" was identified as a mechanism behind "QR elicitation challenges" rather than as a challenge on its own. Still, we can confirm that users' interest in internal QRs is limited. Table 18 lists challenges found by Alsaqaf et al. (2019) and we compare them to our findings and point out similarities.

As can be seen from Table 18, there are similarities but also differences in findings regarding challenges. In addition, it should be noted that not all the challenges that are similar to those of Asaqaf et al. were found in the largest of the case companies of our study. The challenge of too late feedback about QRs was stated in one of the larger SMEs and users limited interest in internal QRs in the smallest SME. One of the most evident dissimilarities concerns the QRs documentation: Alsaqaf et al. found the challenge, "confusion about QR specification approach," and discussed, based on this, the possibility of abandoning the concept of requirement in agile development. According to the authors, for agile practitioners, the concept of requirement included "the user story + the conversation about what is in the user stories + the acceptance criteria" (Alsaqaf et al. 2019, p. 50) and suggested that using this variety of documentation strategy may introduce further inconsistencies into the development process. In our study, the QRs were often included in the acceptance criteria and DoD, and the interviewees did not find this to be problematic. It seemed natural for the practitioners in our study to include certain QRs in the acceptance criteria and others in the DoD.

5.1.2 Practices

In our own SMS about QR management (Behutiye et al. 2019), we identified 143 papers that examined practices, methods, models, frameworks, advice, tools, and guidelines for QR

Table 18 Comparison of Practices Found by Alsaqaf et al. (2019)

Challenges from Alsaqaf et al. (2019)	Similarities to our study
Teams coordination and communication challenges. This category includes the following challenges: - Late detection of QRs infeasibility - Hidden assumptions in inter-team collaboration - Uneven teams maturity - Suboptimal inter-team organization.	Too late feedback regarding QRs was reported in Company B, however, it was not due to communication issues. In the company, they applied VSM sessions to check feasibility of QRs. The three remaining challenge were not found in our study.
Quality assurance challenges. This category includes the following challenges: - Inadequate QRs specification - Lack of cost-effective real integration tests - Lengthy QRs acceptance checklist - Sporadic adherence to quality guidelines	It was confirmed in our interviews that QRs are a fuzzy concept. Practitioners said that it is difficult to know what to measure. This is in line with the first challenge of the category. In Company C, it was stated that some QRs issues might come as a surprise later, however; this was not stated as a testing challenge but rather challenge of eliciting all QR dependencies. The two remaining challenges were not found in our interviews.
QRs elicitation challenges. This category includes the following challenges: - Overlooking sources of QRs - Lack of QRs visibility - Ambiguous QRs communication process	We see the challenge of how to elicit all important QRs from the users as similar to the first challenge. However, this challenge was expressed in the smallest of our case companies, not in the largest. One reason for this could be that the interviewees of the largest company were not in direct contact with the clients. In Alsaqaf et al. (2019) study, Lack of QRs visibility related to the fact that internal QRs are not directly visible to users. We found that according to practitioners, users rarely pay attention on internal QRs. The last challenge of this category was not found in our interviews.
Conceptual challenges of QRs. This category includes the following challenges: - Unclear conceptual definition of QRs - Confusion about QRs specification approaches	The first challenge was confirmed in our study, expressed as QRs are a fuzzy concept. The second challenge we did not find.
Software architecture challenges. This category contains the following challenge: - Unmanaged architecture changes	This challenge did not emerge in our interviews.

management. Out of these studies, 43 focused on practices, with a total of 74 practices of which many have not been validated in industry. In light of this large number of practices, one might expect that all practices that we found in our study have already been included in the literature—indeed, most of them have. For example, the practice of utilizing experts in the field of security can be found in Ayalew et al. (2013) and S. Türpe and Poller (2017); the practice of using mock-ups (or "low-fi prototypes") for QRs is described in Wale-Kolade et al. (2014); the practice of continuous and automated monitoring and testing is discussed in Cruzes et al. (2017), Cannizzo et al. (2008), and Gary et al. (2011); and the practice of using acceptance criteria and DoD, among others, for documenting QRs is detailed in Behutiye et al. (2017). However, not all of these practices were used exactly as they were proposed in the literature, or for the exact same purpose. For example, Wale-Kolade et al. focused on usability, whereas Company A utilized mock-ups and templates to capture any kind of QRs. Cruzes et al. focused only on performance testing, and Gary et al. focused on security testing, yet CI was employed in all our case companies for ongoing testing to fulfill their respective QR needs. Company B utilized VSM sessions periodically, as they explained, to ensure that what they have developed so far was in line with quality goals. Käpyaho and Kauppinen (2015) proposed prototyping with a focus on reviewing the big picture at steady intervals and

reviewing prototypes with a focus on QRs. However, Käpyaho and Kauppinen did not discuss the potential of this practice in the context of limited access to customers. As we have shown, this contextual factor could be the driver for the practice in Company B.

Since similarities do exist between the practices proposed in the scientific literature and the practices found in our case companies, it would seem that these practices have been adapted *from* the literature. However, it would seem that the sources for the practices were rather in their own efforts to solve challenges. For example, in Company D, it was explained that the use of the template for eliciting QRs and details of QRs was developed by themselves in an attempt to solve their needs. The only practice that we did not find in the scientific literature was "developers using the tool to develop it," which was used by Company A. This practice is very specific for a certain type of situation—that is, when a tool is being developed that can be utilized in the development of the next version of that same tool. It is highly unlikely that this practice would be applicable in any other kind of scenario.

Alsaqaf et al. (2019) identified a set of practices in their empirical study. Just like the practices we identified in our study, those discussed by Alsaqaf et al. are very specific to the contexts of the companies they examined and are therefore not directly comparable. However, some similarities exist. In Table 19, we reflect on the practices applied in our case companies with regard to the mitigation practices covered by Alaqaf et al. We compare most of the practices to Company C, since this company operates in large-scale ASD.

Similarities between the practices found by Alsaqaf et al. (2019) and those identified in our study can be evidenced through comparison. However, in our study, these practices did not always surface as practices for mitigating specific challenges. This is probably because the identified challenges are not the same. Likewise, we identified practices in Company C that were not reported by Alsaqaf et al. Company C applied a multi-level CI with different quality gates to catch, for example, internal QR issues as soon as possible. QRs documented in the acceptance criteria and DoD are reported as a practice in our study; however, in Alsaqaf et al., this was regarded as a challenge, one which caused confusion about the specification approach of QRs.

5.1.3 Context

When we grouped the challenges, compared them between different companies, and then compared the contexts of the companies, we could see that some of the contextual elements could explain some of the challenges and also work as drivers behind some of the practices. Likewise, context might explain why none of the identified challenges were universally experienced by all the studied companies. It might be that a specific challenge is not experienced because of the way of working, or because it did not have a major effect in a certain context. A comparison with existing literature, however, is very limited. In our SMS, we found that research into the interplay of context and QR management practices and challenges is almost completely lacking. Alsaqaf et al. (2019) is the only other study that we found that took context into consideration. They explored whether challenges vary in terms of the severity or risks they pose to companies. In their study, they related context mainly to domain. They listed seven business sectors: government, public transport, tax services, commercial/business, transport navigation, banking, and insurance. The type of system was either an information system or an embedded system. Ultimately, they did not find a relationship between context and challenges.

Table 19 Comparison of Practices Found by Alsaqaf et al. (2019)

Practice from Alsaqaf et al. (2019)	Similar practices in our study
Maintaining an assumption wiki-page. A challenge was that developers make assumptions about QRs when the PO has failed to provide sufficient clarity about specifications. These assumptions were collected in wiki-pages. Also, "self-evident" QRs are documented.	Company B did document all requirements and specifications in wiki-pages. Company C (company operating in large-scale ASD) documented all features to the smallest detail in their feature specification. However, interviewees in Company C stated that, sometimes, QRs were not sufficiently specified. In those cases, they relied on additional communication to solve any issues. This occasionally led to delays in development.
Use multiple product backlogs to include requirements of different viewpoints. A challenge was that different stakeholders can have different viewpoints on QRs. With the help of multiple BLs incorporated by developers, the development teams could maintain traceability to the different viewpoints and could escalate possible QR issues to the relevant stakeholders for priority negotiation.	All our companies preferred to work with one BL. Company C previously practiced the use of several BLs. However, having different BLs for different managers and teams resulted in a complex hierarchy of BLs. This hierarchy made traceability more difficult and sometimes introduced delays into development. For this reason, Company C decided to simplify and reduced the number of BLs.
Use automated monitoring tools. To help with conceptual challenges of QRs, QRs that are related to the internal operation of the software were implemented as rules in the monitoring tools.	All our companies utilized automated monitoring tools, especially in the CI, to monitor the quality of the product. Some interviewees stated that QRs can be a fuzzy concept. This usually meant that they did not know what to measure. Thus, the tools did not provide assistance with this specific challenge.
Reserve part of the sprint for important QRs. A challenge was that the PO might neglect requirements that do not have business value. The development teams "found a way to work around this PO behavior" by agreeing with the PO to implement at least one of the important internal QRs per sprint.	This kind of juxtaposition of PO and development team never surfaced from our interviews. However, in Company C, the interviewees stated that it was a challenge to convince managers to focus on something QR-related (not having direct business value). The practice of having a quality reservation in the BL to reserve time for internal QRs is similar to this practice.
Sprint allocation based on multiple product backlogs. This practice was also in place to mitigate conceptual challenges of QRs to ensure that QRs of different viewpoints get priority.	This practice could not be found in any of our case companies. They preferred to use a single product BL.
Establishing preparation team. A practice to mitigate challenges to team coordination and communication, QR elicitation, and conceptual challenges of QRs. Senior analysts and architects and business representatives are together responsible for preparing the product BL and distributing work to different teams in a sensible way based on the nature of the user stories.	This practice is similar to the practice of Company C, where up-front requirements analysis and specification was done by a separate team of experts before distributing work to different teams.
Establishing component teams. Distributed teams were organized around particular components to help resolve QR elicitation challenges.	Company C organized development around particular components and features; however, this was not directly related to QR elicitation but instead to the organization of implementation in a sensible way.
Establishing QR specialists' teams. A practice to mitigate quality assurance and architectural challenges. A team of experts is given the ownership of, e.g., security requirements.	In Company B, where security is of high importance, a team of specialists is used to ensure that security requirements are met. In Company C, teams are dynamic and are put together based on required skills and knowledge.
Innovation and planning iteration. A practice to mitigate quality assurance and architectural challenges. A team can request time to work on activities other than user stories with business value to resolve technical debt from previous sprints.	This practice was not found in our study.

5.2 Implications for research and practice

From all the different challenges experienced by the case companies as well as the various practices employed to mitigate them, we were able to identify sets of practices for each area of software

development that can be applied in ASD. Practitioners can use these results to guide their decision about QR management practices and help to solve challenges that ASD is prone to when managing QRs. In order to keep the text shorter, we included the list of practices in Appendix 4 Beneficial practices for QR management in ASD. As an example, Fig. 6 illustrates the practices that can be applied in the case of access/limited access to users. When there is limited access to users, both QR elicitation and QR validation can benefit from internal trial users while internal VSM session can help in analysis of QRs. When a company does have access to users, the users should be utilized throughout development. Mock-ups and templates can help QR elicitation, analysis, and specifying while reviews meetings together with the customers can help in QR validation.

The relationship between context and QR management, or between context and QR management challenges, is complex and requires further study by researchers. More possibilities exist for drawing connections between contextual elements, chosen practices, and experienced challenges than those we have elucidated in the present study. Moreover, there are likely other contextual elements with similar effects that we did not discover in our study. Additionally, contextual elements could be used to explain why a certain practice is not used, even when it is apparently beneficial, as well as why a certain challenge is not experienced, even though it should be expected. Understanding all these connections would be of significant value in designing both general QR management guidelines and specific QR management processes for practitioners.

5.3 Validity discussion

We classified validity concerns into four different aspects of validity, as proposed by Runeson et al. (2012), in this multiple case study: construct validity, internal validity, external validity, and reliability.

In our study, **construct validity** related mostly to whether interviewers and interviewees possessed the same understanding of the interview questions. The interview questions were created by two researchers and reviewed by two other researchers. All interviews, except the complementary interviews conducted with Company C, were carried out by two researchers. While one researcher asked the interview questions, the other used a checklist to ensure that all necessary information was recorded. Since the researchers and practitioners did not necessarily use the same terminology when talking about a certain concept, preparations were made to explain the terminology whenever needed. For example, in some cases, the practitioners preferred to talk about, and understood better, the concept of NFRs instead of QRs. Still, there was always the possibility that an interviewee would not want to admit that a concept was unfamiliar. For this reason, we encouraged more open discussion instead of short and simple answers. There was also the risk that interviewees would not want to report on the challenges the company was currently facing. This was mitigated by assuring the interviewees that all information would remain anonymous and confidential, and that only a summary of the interviews would be presented to others. In some cases, we observed that some interviewees were very eager to point out challenges—and for this reason, some challenges might have been exaggerated. However, we did not focus on the severity of a challenge reported by one interviewee but instead on the frequency with which the challenge was reported. Additionally, the number of interviewees in the largest company was much larger than in the other companies. This approach was necessary for yielding a fuller understanding of how the different business lines operate. We acknowledge that this might skew the results toward making challenges or practices in the largest company more prominent. However, we did not prioritize challenges based on number of times mentioned, nor did we exclude challenges or

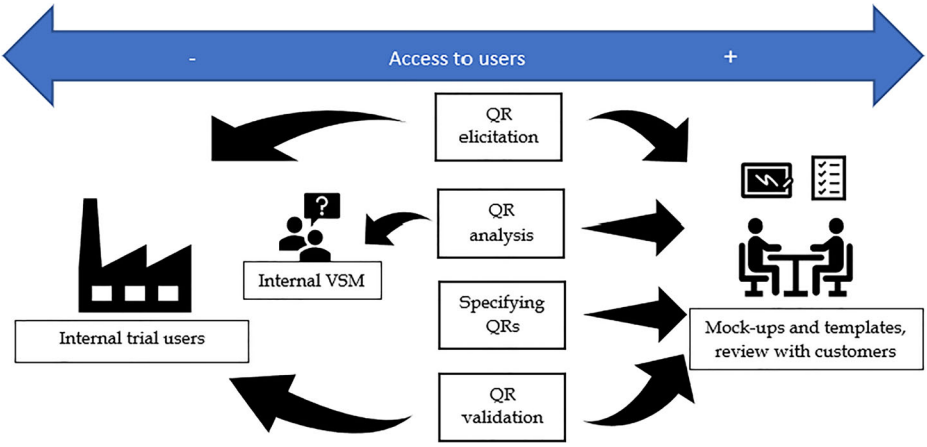

Fig. 6 Practices applied for QR management with limited access to users compared to full access to users

practices mentioned by individual interviewees in smaller companies since we wanted to gather as many QR management related challenges and practices as possible.

Internal validity refers to the causal relationships between factors. Since we were investigating the relationship between context and QR management challenges and practices, internal validity had to be considered. One researcher conceptualized the connections between QR management practices and challenges and the contextual elements of the companies. Two other researchers reviewed the results and used this information to construct a model of the connections as well as to check whether these connections were realistic. As can be seen from our findings, not all challenges were related to the contextual elements. Thus, it can be assumed that factors other than those presented in this study can also affect the challenges. However, the aim was to find those challenges that were most evidently affected by contextual elements. Additionally, it is possible to think of QR type as a contextual factor in itself. Different types of QRs could require different kinds of management practices. For example, practices to manage usability related requirements rarely fit for security requirements, and vice versa. This aspect could be further explored in future work.

External validity must also be considered, since we sought to find connections between context and practices and challenges that others could utilize. All the studied companies operate in different context, thus, there is only one company with a specific context. Yet, some of the practices were applied in all the companies. These practices are typical for or favored in ASD, and thus it can be said that such practices are likely to be found in companies outside our study as well. Some practices are unique in a certain context, and as such it can be argued that they have emerged in response to special needs, or specific challenges, faced by a certain company. However, whether these practices could be effective in mitigating challenges in other companies with similar contexts and similar challenges is an issue in need of further research. None of the challenges were experienced universally by all the case companies. Thus, even though we were able to rationalize the connections between certain elements of the context and the challenges, of which some were obvious, these connections would need to be confirmed with further studies.

For **reliability**, both data collection and analysis were mostly done by peers. Two researchers independently devised coding schemes that were used to code the interview

transcripts. Independent coding was done in a top-down and bottom-up fashion by using the same transcripts. The resulting coding schemes were compared and discussed to form a common understanding of how the transcripts were to be coded. The common coding scheme was then used on another transcript by both researchers, and the results were compared once again. At this point, there were only minor points of discussion, and we were thus able to use the coding scheme on the rest of the interviews. The top-down coding employed the same structure as the interview questions. CMMI for development, version 1.3 (CMMI Product Team 2010), was used in this regard. For this part, other researchers could arrive at the same coding scheme. However, the bottom-up coding is more subjective, and a different set of researchers could generate different results. The results were validated in each company in workshops by the company champions and other key actors who had knowledge about the software development processes. Additionally, the company champions reviewed this manuscript to validate that the included information was accurate.

6 Conclusion and future work

We conducted a multiple case study to investigate strategies and practices for quality requirement (QR) management in agile software development (ASD) and related challenges. Additionally, we identified contextual elements that contribute to the choice of practices and to the emergence of specific challenges. The case study was conducted in four different companies of different sizes and contexts through semi-structured interviews. We found that the practitioners in the studied cases manage quality in general and complement their development processes with any practices necessary for the management of QRs to reach their quality goals. We classified these practices as proactive, reactive, and interactive, and then grouped them as practices in QR elicitation, QR analysis, specifying QRs, and QR validation.

The challenges reported by the interviewees were different depending on the context of the studied companies. We grouped the challenges according to the area of effect as challenges in QR elicitation, QR analysis, specifying QRs, QR implementation, and QR validation. Some challenges were more general and affected several areas of development, i.e., they were context independent. Other practitioners can utilize our findings to identify what kinds of challenges they can expect to face in different contexts.

Many of the challenges reported in the literature were also found in our case companies, albeit with some slight variations. We also determined that not all challenges found in the scientific literature were present in our studied case companies.

We were able to identify a set of practices that have a positive effect on different areas of software development and can as such help mitigate some of the challenges we found. Practitioners could utilize these findings to improve their own QR management practices.

We identified many connections concerning how context drives the choice of QR management practices and how contextual elements can be the source for challenges. However, opportunities remain for the further exploration of these connections. For example, future research could examine why some practices are not used despite being apparently beneficial, or why some challenges are not experienced even though they should be expected to occur. An investigation into these connections would complement the findings of this study and ultimately complete the theoretical model on QR management in ASD. In addition, analyzing whether there are differences in managing QRs depending on the type of QRs and, if so, what those differences are would be interesting.

Acknowledgements We would like to thank all the practitioners who participated in the interviews during this study. This work was supported by the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement 732253.

Funding Open Access funding provided by University of Oulu including Oulu University Hospital.

Appendix 1 Interview script for first round of interviews

1. WARM-UP QUESTIONS:

Less than 10 Minutes Elapsed 0

Q1.1: **Tell us something about your work experience?** [Subject's experience may impact his/her perspective on understanding of quality requirements management in agile and rapid software development].

- ✓ *How long have you been working in the company?*
- ✓ *How long have you been working with the product?*
- ✓ *What is your role in the development/project?*
- ✓ *Do you have any previous experience in software development? If yes, how long and in what roles? [Experience in traditional, agile, and rapid software development before working in this company]*
  - o *How many years of experience in agile and rapid software development?*
  - o *In what roles?*

Q1.2: **Tell us something about the product you are working on?** [The type of product may impact the quality requirements management approach employed in agile and rapid software development]

- ✓ *What is the name of the product?*
- ✓ *What is the domain area of the product? (e.g., financial services, telecom, manufacturing industry, etc.)*
- ✓ *What is the type of product? (e.g., proprietary or open-source)*
- ✓ *What is the size of the product? (e.g., in LOCs)*
- ✓ *What is the programming language used for developing the product?*
- ✓ *Is the product general or customized to different customer segments?*

Around 40 Minutes Elapsed 10

2. OVERALL DEVELOPMENT PROCESS WITH EMPHASIS ON QR MANAGEMENT QUESTIONS:

Q2.1: How would you describe your way of working? [Do you use waterfall, agile, scrum, hybrid, etc.?]

- ✓ *Could you please give us a brief description of the practices and activities that you have or the different processes that you follow from requirements definition to validation?*

Q2.2: How do you understand **quality requirements** (in your company)?

- ✓ *Do you have a shared and established definition/conceptualization of quality requirements?*
- ✓ *What are the constituents/elements of quality requirements typical to your development?*

Q2.3: In your opinion, are quality requirements important for the development/product?

- ✓ *If yes, what are the most important quality requirements in your development?*
- ✓ *If no, why?*

Q2.4: How are **quality requirements taken into consideration** during development?

- ✓ *Do you treat quality requirements and functional requirements at the same time or separately?*
- ✓ *What are the sources of quality requirements in the context of your development?*
  - o *Do you have internal sources (e.g., development teams, customer representatives, maintenance teams), external sources (e.g., customers, end users of the product) for quality requirements?*
  - o *Do you have standards, rules, and laws you need to abide by that bring quality requirements?[e.g., security standards and policies that may affect quality requirements]*
- ✓ *How do you identify and elicit functional requirements?*
  - o *Who is responsible for functional requirements elicitation?*
  - o *What kind of tools, practices, and techniques do you use for eliciting functional requirements?*

- *What are the current challenges in the identification and elicitation process of functional requirements?*

✓ *How do you identify and elicit quality requirements?*
- *Who is responsible for quality requirements elicitation?*
- *What kinds of tools, practices, and techniques do you use for eliciting quality requirements?*
- *What are the current challenges in the identification and elicitation process of quality requirements?*
- *What areas of improvement would you like to see in the requirements elicitation process?*

✓ *How do you communicate quality requirements with customers?*
- *What kinds of tools, practices, and techniques do you use for communicating quality requirements?*
- *What are the challenges of communicating quality requirements with customers?*
- *What areas of improvement would you like to see while communicating quality requirements with customers?*

✓ *How do you communicate quality requirements internally? [Within teams and with other teams?]*
- *What kinds of tools, practices, and techniques do you use for communicating quality requirements internally?*
- *What are the challenges of communicating quality requirements internally?*
- *What areas of improvement would you like to see while communicating quality requirements internally?*

✓ *How do you document quality requirements? [If yes, how?]*
- *What kinds of tools, practices, and techniques do you use for documenting quality requirements?*
- *What are the current challenges regarding documentation of quality requirements?*
- *What areas of improvement would you like to see in the documentation of quality requirements?*

✓ *How do you do requirements analysis?*
- *Do you specify or model requirements?[both functional and quality requirements]*
- *If yes, do you specify or model functional requirements together with quality requirements?*
  - *How do you specify or model them?*
- *If functional requirements and quality requirements are treated separately:*
  - *How do you specify or model functional requirements?*
  - *How do you specify or model quality requirements?*
- *What kinds of tools, practices, and techniques do you use in requirements analysis?*
- *What are the current challenges regarding specification of requirements? [Both functional and quality requirements]*
- *What areas of improvement would you like to see in the requirements analysis process?*

✓ *How do you prioritize quality requirements in your development?*
- *Are there any rules for prioritizing quality requirements?*
- *When and how often are they prioritized?*
- *Who does the prioritization / is responsible for prioritization?*
- *How do you communicate information regarding quality requirements prioritization with relevant stakeholders?*
- *What are the current challenges regarding prioritization of quality requirements?*
- *What areas of improvement would you like to see in quality requirement prioritization?*

✓ *How do you validate the quality requirements?*
- *What are the techniques you use for validating quality requirements?*
- *If review meetings, how often and who are present?*
- *Who determines if quality requirements have been met sufficiently?*
- *How is user feedback regarding QRs taken into consideration during development?*

Q2.5: How does **requirements engineering (requirements development and management process), and in particular quality requirements, relate to other areas?**

- ✓ *Are quality requirements taken into considerations in the risk management process?*
  - o *Does the exclusion of quality requirements increase risk in the project? [If yes, in what ways?]*
  - o *Does risk management introduce new quality requirements*?
- ✓ *How are quality requirements affected in project planning?*
  - o *Do quality requirements affect project planning? [If yes, how?]*
  - o *Are quality requirements taken into account during project planning?*
    - *Are quality requirements budgeted in project planning?*
    - *How does the availability of persons/resources management affect quality requirement?*
- ✓ *How are quality requirements related to project management?*
  - o *Does management see the importance of quality requirements?*
  - o *Are quality requirements taken into consideration during budgeting?*
  - o *How are issues regarding quality requirements communicated between development and project management?*

3. WRAP-UP QUESTIONS:

Around 10 Minutes *Elapsed* 50

Q3.1: Do you see any challenges in the current methods, techniques, tools, or practices regarding quality requirements management?

- ✓ *What areas of improvement would you like to see related to QR management?*
- ✓ *Do you have any suggestions for improvement?*

Q3.2: Are there **any related issues that we missed** that you would like to reflect on?

Appendix 2 Interview script for second round of interviews

Introduction and purpose of interview

Around 5 Minutes Elapsed 5

The goal of our interview today is to first corroborate our current understanding of your software development processes and QR management practices. Then, we want to learn about your experiences using Q-Rapids solutions and related challenges. Lastly, we are interested in how you plan to continue using or want to use Q-Rapids solutions to set the basis for the co-creation of the Q-Rapids software development process in your company.

This interview is a continuation of the interviews in the Q-rapids project, which has the aim of integrating QR management into agile and rapid software development.

The interview will last approximately 1.5 hours. We will begin with a few warm-up questions, after which we will move on to the development process and quality requirements (QRs) management practices to corroborate our findings so far. Following that, we will ask questions about your current use of the Q-Rapids dashboard and related challenges. Questions regarding data gathering will be asked after that. We will end the interview by asking about potential impacts you have observed when using the Q-Rapids dashboard.

We would like to record the interviews to capture all data more accurately. All data will be anonymized by linking data to interviewee IDs and will be kept confidential. You have the right stop the recording at any time. This will not affect your work in the company in any way. An aggregated summary will be presented to you in a workshop after all interviews have been conducted.

1. WARM-UP QUESTIONS, BACKGROUND INFORMATION:

Around 10 Minutes Elapsed 15

Q1.1: **Tell us something about your work experience?** [If the interviewee is one who we have not interviewed previously. If we have interviewed previously, corroborate our findings]

- ✓ *How many years of experience do you have in software development?*
- ✓ *How many years have you been working in the company?*
- ✓ *What project/use case are you working on?*
- ✓ *How many years have you been working with the project/use case?*
- ✓ *What many years of experience do you have in agile development? [Before working in this company and in company]*

Q1.2: **Tell us about your current role?**

- ✓ *What is your role in the software development process?*
- ✓ *What are your tasks in the software development process?*

2. QUALITY REQUIREMENTS MANAGEMENT QUESTIONS:

A NOTE FOR INTERVIEWER
We can use a process picture in this interview question. Print process pictures from D2.1.

Around 30 Minutes Elapsed 45

Q2.1: Case A/**Does this represent your development process and QR management practices (in your company)**? [Need to explain the process model to the interviewee]

- ✓ *We learned that features are drawn of the annual roadmap that have been made by the executive managers, and that these are communicated to product managers mainly in Word documents. How are they communicated? Do you have meetings for this? How frequently?*
- ✓ *When you get issues or requests from the forum or through the sales team, how do you decide about the work to be done on these issues? Do you have meetings for this? How frequently?*
- ✓ *We learned that developers can also be a source for QRs, since they are using the tool. What happens when a developer identifies a quality issue? Does he or she report it somewhere? Do you include these issues as topics in a meeting? What meeting? How frequent?*

- ✓ *We learned that prioritization of QRs is done by product management and also that customer reported issues, like crashes, get the highest priority. How often does the prioritization by product management happen? How and when is this priority communicated to development? How are the customer reported issues prioritized? Who does it? Who decides? How frequently?*
- ✓ *You said that requirements are modeled to the level that code can be auto generated, but still you say that you do not specify requirements: How does this fit together?*
- ✓ *We learned that you use Word documents and a whiteboard to facilitate face-to-face communication. Do you have any meetings for these face-to-face communications?*
- ✓ *Testing of QRs: How is it done specifically?*

Q2.1: **Case B/Does this represent your development process and QR management practices (in your company)**? [Need to explain the process model to the interviewee]

- ✓ *We have learned that you include QRs in the DoD. How exactly are the QRs (which QRs) documented in the DoD? Can you give us examples of these quality-related items in the DoD?*
- ✓ *We have learned that you want to focus on what brings most value to the customer; does this mean that you focus more on quality or functionality?*
- ✓ *Are the QRs visible in the auto generated documents from source code?*
- ✓ *How exactly are the QRs documented in user stories?*
- ✓ *How exactly are the QRs documented in the tools (Jira, SpiraTest, Kibana…)?*
- ✓ *Testing of QRs: How is it done specifically?*

Q2.1: **Case C/Does this represent your development process and QR management practices (in your company)**? [Need to explain the process model to the interviewee]

- ✓ *We have learned that QRs are included in the backlogs; can you give us specific examples of these QRs (performance, maintainability, security, usability, portability…)*
- ✓ *We learned that QR may be included as items in the DoD; can you give us examples?*
- ✓ *We have learned that a quality manager "pushes for quality"; does this mean quality in general or specific QRs?*
- ✓ *We have learned that quality is assured through nightly builds and testing; does this refer to quality in general or are there specific tests for specific QRs?*

Q2.1: **Case D/Does this represent your development process and QR management practices (in your company)**? [Need to explain the process model to the interviewee]

- ✓ *We have learned that before development, you have a scoping session with the customer whereby you can use, for example, a high fidelity mock-up. Do you have any information about what the customer wants before this meeting in order to be able to produce the mock-up? How do you get that information? (Do you create the mock-up before the meeting, or during the meeting?)*
- ✓ *We have learned that QRs are included in the product and sprint backlogs. Are they included there separately, or are they included together with functional requirements? Can you give us specific examples of these QRs (performance, maintainability, security, usability, portability…)*
- ✓ *We learned that QRs are clarified and communicated in each sprint planning, sprint review, and sprint retrospective. How is this clarification done? Is it formal, or informal? How is the communication? Is it just informal communication?*
- ✓ *We have learned that sprint retrospective and sprint review meetings help in checking the completion of user stories. Are QRs also checked in these meetings?*
- ✓ *Can QRs be documented as acceptance criteria? Do you have any examples?*
- ✓ *Do you have any specific tests for quality requirements?*

Appendix 3 Pre-questionnaire

Here we present those parts of the pre-questionnaire in the Q-Rapids project that were sent to the companies and that we utilized for triangulation. Details to be asked from the industrial practitioners were specified by different work packages in the Q-Rapids project. These details were compiled and sent to the practitioners by the Fraunhofer Institute for Experimental Software Engineering (Fraunhofer IESE). They were responsible for the project evaluation. The parts of the pre-questionnaire that are included in this appendix include those details that we, the authors, specified, and that we later used for triangulation in data analysis.

Pre-Questionnaire: "Analysis of Current Situation of the Use Cases"

Executive summary

In the context of the Q-Rapids work package WP5, the Fraunhofer Institute for Experimental Software Engineering (Fraunhofer IESE) kindly invites you to participate in the survey, "Analysis of Current Situation of the Use Cases."

The information that we collect from this survey will be kept confidential and will be published in several project deliverables and made available to all project work packages to support the design and

development of the Q-Rapids framework. Moreover, the information that we collect from this survey will eventually be published as part of scientific publications following the regulations put down in the GA.

The survey includes two parts aimed at eliciting (1) the organization and (2) the processes used to develop each candidate software product that will later be used as basis for a better understanding of the management of technical debt in your organization. Whereas Part 1 is expected to be answered by any person(s) familiar with organizational characteristics and the Q-Rapids project, Part 2 is expected to be answered by the project manager responsible for each selected project and software product to be considered in the Q-Rapids project.

Part 1 – General Organizational Context

The Q-Rapids project aims at developing a framework for managing quality requirements in rapid software development on the basis of example software product(s) data provided by the application partners. In this section, we ask about the general organizational context.

Table 10 Information About Your Organization

Question	Answer
For which application domains does your organization develop software?	
How many employees did your organization have at the end of 2016?	

Part 2 – Product-specific Information

The Q-Rapids project aims at developing a framework for managing quality requirements in rapid software development on the basis of example software product(s) data provided by the application partners. In this section, we ask about the characteristics of the selected product(s).

Table 11 Section About the Selected Project

Question	Answers about the selected project
In which year was rapid software development (or agile, continuous development, etc.) introduced in the project?	

Table 12 Section About the Selected Product

Question	Answers about the selected product
Product Name	
Product Type/Domain	
How many releases have already been completed?	

Table 13 Methods, Standards, Tools, and Responsible People Concerning the Software Product

Please list the methods, standards, and tools used for:	Methods/Standards	Tools	Who is responsible?
Software Life Cycle			

Appendix 4. Beneficial practices for QR management in ASD

Here we list the sets of practices for each area of software development that can be applied in ASD for better management of QRs. The practices are grouped according to the area in which they exert a positive effect.

- R elicitation
  - Access to users/customers
    - Mock-ups and templates for details about QRs
      The mock-ups help identify QRs visible to the user, and the template helps identify software-internal QRs that the users might usually omit.
  - Limited access to users/customers
    - Trial user test
      Despite being a late-phase test, it can be utilized as an additional elicitation technique. Company-internal trial users can test the product to find new QRs. This is especially useful in a situation in which there is limited access to real users.
  - System with complex dependencies
    - Multi-level CI
      This can be utilized as an elicitation technique, especially in fast iterations. CI seems to be the choice of most companies. This practice is particularly beneficial in the development of complex systems in which it is difficult to find all software-internal QRs.
  - The product is a software development tool
    - Developers using the tool they develop
      This might be possible only in rare cases. It can provide the developers with a way to identify any new quality issues in the tool they are developing, which they can accordingly address directly.
  - Agile development utilizing backlogs
    - Quality reservation in the backlog
      In this practice, the developers' expertise is utilized by letting them suggest any QRs they believe should be included. The suggested QRs are analyzed in regular, periodic meetings.
- QR analysis
  - Access to users/customers
    - Template for details of QR
      Useful to use together with the customers to fill in technical details about the QRs.
    - Intermediate/periodic review with customers or users
      The users should be given the opportunity to review the results of development prior to release to spot any QRs that were misunderstood or in the event the customers have changed their minds.
  - Limited access to users/customers
    - Value stream mapping sessions
      The development team stops occasionally, or frequently, to check whether developed features are contributing to the envisioned value that will be delivered to customers or users.

- Specifying QRs
  - Access to users/customers
    - Template for details about QRs
      The same template that was used for the elicitation and analysis of QRs is an effective means for specifying and documenting the details for developers.
  - Agile development with acceptance criteria and DoD
    - QRs in acceptance criteria and DoD
      Most companies have found that some QRs should be included as acceptance criteria to check against before release, while others can be specified as tasks to be done in the DoD to ensure that QRs are met.
- QR validation
  - Access to users/customers
    - Review with customers or users
      In contractual development, it is always a good idea to have a final review session together with the customers or users to ensure that all stakeholders are satisfied.
  - Limited access to users
    - Trial user testing
      Internal trial users can test the product before release to customers. It would be preferable to include customers other than stakeholders who have been involved in the development of the product.
  - Product is a system with complex dependencies
    - Multi-level CI
      CI is also useful in validation to demonstrate that all tests have been passed.
  - Agile development utilizing acceptance criteria and DoD
    QRs in acceptance criteria and DoD
    When some QRs have been specified as tasks to be completed in the DoD, it is easy to check that these QRs have been met. The results of development can be checked against the acceptance criteria before release in acceptance testing.

References

Aljallabi BM, Mansour A (2015) Enhancement approach for non-functional requirements analysis in agile environment. In: Computing, Control, Networking, Electronics and Embedded Systems Engineering (ICCNEEE), 2015 International Conference, pp 428–433

Alsaqaf K, Daneva M, Wieringa R (2017) Quality requirements in large-scale distributed agile projects - a systematic literature review. International Working Conference on Requirements Engineering: Foundation for Software Quality, pp 219–234

Alsaqaf W, Daneva M, Wieringa R (2019) Quality requirements challenges in the context of large-scale distributed agile: an empirical study. Inf Softw Technol 110:39–55

Anwar S et al (2014) User-centered design practices in scrum development process: a distinctive advantage? 2014 IEEE 17th International Multi-Topic Conference (INMIC), pp 161–166

Ayalew T, Kidane T, Carlsson B (2013) Identification and evaluation of security activities in agile projects. In: Proceedings of the 18th Nordic Conference on Secure IT Systems, pp 139–153
Azham Z, Ghani I, Ithnin N (2011) Security backlog in Scrum security practices. Software Engineering (MySec), 2011 5th Malaysian Conference, pp 414–417
Behutiye W, Karhapää P, Costal D, Oivo M, Franch X (2017) Non-functional requirements documentation in agile software development: Challenges and solution proposal. International Conference on Product-Focused Software Process Improvement, pp 515–522
Behutiye W et al (2019) Management of quality requirements in agile and rapid software development: a systematic mapping study. Inf Softw Technol:106225
Bourimi M, Kesdogan D (2013) Experiences by using AFFINE for building collaborative applications for online communities. International Conference on Online Communities and Social Computing, pp 345–354
Bourimi M et al (2010) AFFINE for enforcing earlier consideration of NFRs and human factors when building socio technical systems following agile methodologies. In: International Conference on Human-Centered Software Engineering, pp 182–189
Bourque P, Fairley RE (2014) Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0. IEEE Computer Society Press
Butt S, Ahmad WFW, Rahim L (2014) Handling tradeoffs between agile and usability methods. In International Conference on Computer and Information Science (ICCOINS), pp 1–6
Cajander Å, Larusdottir M, Gulliksen J (2013) Existing but not explicit - the user perspective in scrum projects in practice. In: IFIP Conference on Human-Computer Interaction, pp 762–779
Cannizzo F, Clutton R, Ramesh C (2008) Pushing the boundaries of testing and continuous integration. Agile Conference, AGILE'08, pp 501–505
Cao L, Ramesh B (2008) Agile requirements engineering practices: an empirical study. IEEE Softw 25(1):60–67
Chen L (2015) Continuous delivery: huge benefits, but challenges too. IEEE Softw 32(2):50–54
CMMI Product team (2010) CMMI for Development. Version 1.3. In: Software Eng Inst
Cruzes DS, Dybå T (2011) Recommended steps for thematic synthesis in software engineering. In: 2011 International Symposium on Empirical Software Engineering and Measurement:275–284
Cruzes DS, Felderer M, Oyetoyan TD, Gander M, Pekaric I (2017) How is security testing done in agile teams? A cross-case analysis of four software teams. In: Proceedings of the International Conference on Agile Software Development (XP), pp 201–216
Curcio K, Navarro T, Malucelli A, Reinher S (2018) Requirements engineering: a systematic mapping study in agile software development. J Syst Softw 139:32–50
Dybå T, Sjøberg D, Cruzes DS (2012) What works for whom, where, when, and why? On the role of context in empirical software engineering. In: ACM-IEEE International Symposium on Empirical Software Engineering and Measurement pp 19–28
Farid W (2012) The NORMAP methodology: lightweight engineering of non-functional requirements for agile processes. In: 2012 19th Asia-Pacific Software Engineering Conference, vol 1, pp 322–325
Farid W, Mitropoulos F (2013) NORPLAN: non-functional requirements planning for agile processes. In Proceedings of SouthEastCon, pp 1–8
Fowler M, Highsmith J (2001) The agile manifesto. Softw Dev 9(8):28–35
Gary K, Enquobahrie A, Ibanez L, Cheng P, Yaniv Z, Cleary K et al (2011) Agile methods for open source safety-critical software. Softw: Pract Experience 41(9):945–962
Glass RL (1998) Defining quality intuitively. IEEE Softw 15(3):103–104
Heck P, Zaidman A (2016) A systematic literature review on quality criteria for agile requirements specifications. Softw Qual J:1–34
Heikkilä VT, Damian D, Lassenius C, Paasivaara M (2015) A mapping study on requirements engineering in agile software development. In: 41st Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp 199–207
Inayat I et al (2015) A systematic literature review on agile requirements engineering practices and challenges. Comput Hum Behav 51:915–929
Kalenda M, Hyna P, Rossi B (2018) Scaling agile in large organizations: practices, challenges, and success factors. J Softw-Evol Proc 30(10):e1954
Käpyaho M, Kauppinen M (2015) Agile requirements engineering with prototyping: a case study. In: 2015 IEEE 23rd International Requirements Engineering Conference (RE), pp 334–343
Karhapää P, Haghighatkhah A, Oivo M (2017) What do we know about alignment of requirements engineering and software testing? in: Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering, pp 354–363
Kasauli R, Liebel G, Knauss E, Gopakumar S, Kanagwa B (2017) Requirements engineering challenges in large-scale agile system development. 2017 IEEE 25th International Requirements Engineering Conference (RE), pp 352–361

Knauss E, Liebel G, Schneider K, Horkoff J, Kasauli R (2017) Quality requirements in agile as a knowledge management problem: more than just-in-time. In 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW), pp 427–430

Magües DA, Castor JW, Acuña ST (2016) Requirements engineering related usability techniques adopted in development process. In: SEKE 2016. Proceedings of the Twenty-Eight International Conference on Software Engineering and Knowledge Engineering. Knowledge Systems Institute Graduate School

Medeiros J, Vasconcelos A, Silva C, Goulão M (2020) Requirements specification for developers in agile projects: evaluation by two industrial case studies. Inf Softw Technol 117:106194

Merriam-Webster (2019) [Online]. Available at: https://www.merriam-webster.com/. Accessed 17 April 2019

Miles MB, Huberman AM (1994) Qualitative data analysis: An expanded sourcebook, 2nd edn. Sage, Thousand Oaks, CA, p 1994

Mylopoulos J, Chung L, Nixon B (1992) Representing and using nonfunctional requirements: a process-oriented approach. IEEE Trans Softw Eng 18(6):483–497

Nawrocki M, Ochodek J, Jurkiewicz S, Alchimowicz B (2014) Agile requirements engineering: a research perspective. In: International Conference on Current Trends in Theory and Practice of Informatics, pp 40–51

Nvivo (2018) [Online]. Available at: http://www.qsrinternational.com/nvivo/nvivo-products. Accessed 27 Aug 2018

Petersen K, Wohlin C (2009) Context in industrial software engineering research. In: Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement, pp 401–404

Pohl K (2016) Requirements engineering fundamentals: a study guide for the certified professional for requirements engineering exam-foundation level-IREB compliant. Rocky Nook Inc

Rafi U, Mustafa T, Iqbal N, Zafar WUI (2015) US-Scrum: a methodology for developing software with enhanced correctness, usability and security. Int J Sci Eng Res 6(9):377

Ramesh B, Cao L, Baskerville R (2010) Agile requirements engineering practices and challenges: an empirical study. Inf Syst J 20(5):449–480

Rodríguez P, Yagüe A, Alarcón P, Garbajosa J (2009) Some findings concerning requirements in agile methodologies. In: Product-Focused Software Process Improvement, pp 171–184

Rodriguez P, Markkula J, Oivo M, Turula K (2012) Survey on agile and lean usage in Finnish software industry. In: Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, pp 139–148

Runeson P, Höst M (2009) Guidelines for conducting and reporting case study research in software engineering. Empir Softw Eng 14(2):131

Runeson P, Höst M, Rainer A, Regnell B (2012) Case study research in software engineering: guidelines and examples. Wiley

Savolainen J, Kuusela J, Vilavaara A (2010) Transition to agile development-rediscovery of important requirements engineering practices. 2010 18th IEEE International Requirements Engineering Conference, pp 289–294

Schön E-M, Thomaschewski J, Escalona MJ (2017) Agile requirements engineering: a systematic literature review. Comput Stand Interfaces 49:79–91

Shaw M (1996) Truth vs knowledge: the difference between what a component does and what we know it does. In: 8th International Workshop on Software Specification and Design, p 181

Singh M (2008) U-SCRUM: an agile methodology for promoting usability. In: AGILE'08 Conference, pp 555–560

Siponen M, Baskerville R, Kuivalainen T (2005) Integrating security into agile development methods. In: Proceedings of the 38th Annual Hawaii International Conference on System Sciences, HICSS'05, p 185a

Sjøberg DIK, Dybå T, Anda BCD, Hannay JE (2008) In: Shull F et al (eds) Building theories in software engineering. Guide to Advanced Empirical Software Engineering. Springer

Stålhane T, Myklebust T, Hanssen GK (2012) The application of safe scrum to IEC 61508 certifiable software. In: 11th International Probabilistic Safety Assessment and Management Conference and the Annual European Safety and Reliability Conference, pp 6052–6061

Türpe S, Poller A (2017) Managing security work in Scrum: tensions and challenges. SecSE@ESORICS 2017: 34–49

VersionOne, Inc (2016) 10th annual state of agile development survey. [online]. Available at: https://www.stateofagile.com/#ufh-i-338498988-10th-annual-stateof-agile-report/473508. Accessed April 2019

Wagner S (2013) Software product quality control. Springer, Berlin

Wale-Kolade A, Nielsen P, Päivärinta T (2014) Integrating usability practices into agile development: A case study. Proceedings of 23rd International Conference on Information Systems Development, pp 337–347

Wiegers KE, Beatty J (2013) Software requirements. Pearson Education

Affiliations

Pertti Karhapää[1] · Woubshet Behutiye[1] · Pilar Rodríguez[2] · Markku Oivo[1] · Dolors Costal[3] · Xavier Franch[4] · Sanja Aaramaa[5] · Michał Choraś[6,7] · Jari Partanen[8] · Antonin Abherve[9]

✉ Pertti Karhapää
pertti.karhapaa@oulu.fi

Woubshet Behutiye
woubshet.behutiye@oulu.fi

Pilar Rodríguez
pilar.rodrigues@upm.es

Markku Oivo
markku.oivo@oulu.fi

Dolors Costal
dolors@essi.upc.edu

Xavier Franch
franch@essi.upc.edu

Sanja Aaramaa
sanja.aaramaa@nokia.com

Michał Choraś
mchoras@itti.com.pl

Jari Partanen
jari.partanen@bittium.com

Antonin Abherve
antonin.abherve@softeam.fr

[1] M3S, Faculty of Information Technology and Electrical Engineering (ITEE), University of Oulu, 90014 Oulu, Finland

[2] Department of Languages, Computer Systems and Software Engineering, Faculty of Computer Science, Universidad Politécnica de Madrid, Madrid, Spain

[3] Universitat Politècnica de Catalunya, UPC-BarcelonaTech, 08034 Barcelona, Spain

[4] Department of Service and Information System Engineering, UPC-BarcelonaTech, 08034 Barcelona, Spain

[5] Nokia Oy, 90620 Oulu, Finland

[6] ITTI Sp. z o.o, 61-612 Poznan, Poland

[7] University of Science and Technology, UTP Univerisy of Science and Technology, 5-796 Bydgoszcz, Poland

[8] Bittium Wireless Ltd., 90590 Oulu, Finland

[9] Softeam, 75016 Paris, France