CHAPTER

13 Understanding Quality Requirements Engineering in Contract-Based Projects from the Perspective of Software Architects: An Exploratory Study

Maya Daneva[1], **Andrea Herrmann**[2], **and Luigi Buglione**[3]

[1]University of Twente, Enschede, The Netherlands

[2]Herrmann & Ehrlich, Stuttgart, Germany

[3]ETS Montréal/Engineering.IT SpA, Rome, Italy

INTRODUCTION

Quality requirements (QRs) for a software system define quality properties of the system, for example, scalability, security, usability, and performance (Gorton, 2011). Essentially, QRs address issues of concern to applications' users as well as other stakeholders such as the project team, the software maintainers, or the project steering committee. Getting these requirements right in the early stages of a software development project has been known to be instrumental to the design of software architecture and to lead to savings in the later phases of implementation, testing, and maintenance (Gorton, 2011; Pohl, 2011).

In the communities of software architecture and of requirements engineering (RE), the topic of engineering QRs has been investigated for more than a decade; however, primarily in terms of creating methods, representation formalisms, languages, and tools that could support in various ways the process of joint RE and software architecture (Avgeriou et al., 2011; Bachmann and Bass, 2001; Bass et al., 2003; Capilla et al., 2012; Sommerville, 2005). Only recently, this research effort was complemented with studies on the human agents conducting RE and architecture processes (Ferrari and Madhavji, 2008) and on the contexts in which RE and software architecture professionals collaborate. In turn, as noted by Ferrari and Madhavji (2008), few studies exist on the role of agents in project organizations and how these agents cope with QRs in their projects on a day-to-day basis. Specifically, empirical studies on the perspective of software architects (SAs) are relatively recent. Because they are mostly published after 2010, they address only a small number of organizational contexts in which software development might take place; namely, they report on small and mid-sized projects. Moreover, most of them do not address the contextual differences between in-house development projects and contract-based software delivery projects. Yet, in the twenty-first century, most large software system development happens in contractual settings.

In this chapter, we set about closing the gap of knowledge about how SAs cope with QRs in large and contract-based system delivery projects. We achieve this by an exploratory interview-based study with 21 SAs working in large or very large contract-based systems projects in European companies. We identified how SAs cope with the six engineering activities associated with QR, namely QRs elicitation, documentation, prioritization, quantification, validation, and negotiation. Our results revealed the roles that SAs play in these activities in a contract-based project, the interactions between SAs and other project roles, the specific actions SAs take to help discover, document, quantify, validate, and negotiate QRs, and the ways in which the contract orchestrated the QR engineering activities and encouraged the SAs do what they did. We compare our findings with previously published results and draw implications for SAs, for RE specialists, for RE tool vendors, as well as for further research.

This chapter makes three contributions: (1) to software architecture, (2) to RE, and (3) to empirical software engineering. First, whereas the interfaces between software architecture and RE have been the subject of research in the software architecture community for years, *how SAs operate in contract-based context while engineering QRs* has evaded the attention of software architecture researchers. This study narrows the gap of knowledge in the field regarding this question. Second, we inform the RE researchers and practitioners on what contributions SAs bring to the complex social and only partly understood process of engineering QRs in contact-based contexts. While the RE community has acknowledged for a long time that the RE processes are very different for contract-based projects as opposed to in-house projects (Berenbach et al., 2010; Lauesen, 2002), most of the existing empirical RE research was preoccupied with the perspectives of business users, RE specialists, vendors' marketing specialists, and managers, and in fact left the perspective of SAs almost completely unaddressed. If the RE community learns about the SAs' experiences in QR engineering, they might design new methods with and for SAs in order to help both disciplines to collaborate much easier or to collectively come up with criteria for choosing the right method. Third, our chapter responds to two explicit calls: (1) the call by the empirical SE community for more empirical research on which SE process to use in which specific context (Sjøberg et al., 2007) and (2) the particular call by the RE community for more empirical research in the sub-areas of RE (Cheng and Atlee, 2007).

The chapter is structured as follows: We first present the motivation for our study. Next, we describe the context of contract-based software development and summarize related work based on systematic literature reviews published by authors in the fields of RE and of software architecture. Then, we present our research questions, our empirical interview-based research process, its execution, and its results. We end up with our discussion on the results in the light of previously published research, on the implications of our study for practitioners and researchers, and on evaluation of validity threats.

13.1 MOTIVATION

Our motivation for this empirical research is rooted in the trends in the software industry for designing and deploying approaches to joint RE and architecture design (Avgeriou et al., 2011; Bachmann and Bass, 2001; Sommerville, 2005). This trend reflects the understanding that QRs are usually relevant for the architecture and, vice versa, architecture is determining QR satisfaction (Chen et al., 2013). For example, the Twin Peak model (Nuseibeh, 2001) for joint RE and architecture design reinvents the notion of the spiral life cycle for software development by viewing software architecture as a "peak" that along with the "peak" of the requirements is iterated early on. SAs and RE professionals have been experiencing such approaches as key to guiding the architecture design process, capturing design rationale, and supporting traceability between requirements and architecture. This known, both SAs and RE specialists initiated a conversation to understand how each of these professional groups reasons about QRs and contributes to the QRs engineering activities, so that these approaches are leveraged and also improved in a targeted and cost-effective fashion (Avgeriou et al., 2013). However, the concerted empirical research efforts of both research schools and practitioners on understanding the SAs' perspectives on QRs started relatively recently. Therefore, the available empirical evidence and knowledge cover only unevenly the possible range of contextual settings in which SAs and RE professionals work. Specifically, most published empirical evidence until now on how SAs and RE specialists work together has been gained in small and mid-sized projects. The context of large and contract-based software system delivery projects has remained by and large unaddressed. This knowledge gap in our understanding of the SAs' perspective on QRs motivated us to initiate empirical research by collecting and analyzing first-hand information from SAs in the field. In the next two sections, after presenting our context of interest, we report on published empirical studies and indicate that there is an important gap in knowledge.

13.2 BACKGROUND ON THE CONTEXT OF CONTRACT-BASED SYSTEMS DELIVERY

This section summarizes publications on contract-based software development because this is the context of interest in our study. We draw some differences between this context and the context of in-house development that relate to the work of RE staff and SAs.

For the purpose of our study, we consulted literature sources in three streams of research: (i) IT outsourcing (e.g., Furlotti, 2007); (ii) designing inter-firm contracts (Gopal and Koka, 2010; Song and Hwonk, 2012; Wu et al., 2012), and particularly, service-level agreements (SLAs) for IT projects (Goo et al., 2008); and (iii) RE for contract-based systems (Berenbach et al., 2010; Nekvi et al., 2012; Song and Hwonk, 2012). Generally, in the literature on software development contracts, a contract is understood as a legal vehicle that defines the sharing of profit and risk between a software development organization and a business client organization that is committed to delivering a project. A typical contract is composed of two parts (Furlotti, 2007): (i) a *transactional* part, which defines the parties' commitments on tasks, resources, outputs, and remuneration provisions and (ii) a *procedural* part, which defines the rights and processes designed to assist in introducing adjustments, preserving integration, and maintaining shared understanding of the terms and clauses of the contract. Brought together, these parts must address a broad array of issues such as a resulting system's quality, delivery schedule, payment schedule, escalation and termination conditions and procedures, and post-delivery services.

Based on their transactional definitions, software development contracts generally come in three alternative forms: fixed fee, time-and-material, or multistage contacts that combine components of fixed and variable prices (Berenbach et al., 2010; Gopal and Koka, 2010; Kalnins and Mayer, 2004; Lauesen, 2002; Song and Hwonk, 2012; Wu et al., 2012). A fixed-fee contract stipulates a predetermined price that is paid to a software development organization for delivering a software system, while a variable price contract assumes the business client pays the developer the development costs topped up with some profit or payoff. Variable price contracts may differ widely in terms of levels of granularity and sophistication regarding how profit/payoff is defined. Most often, the definition of profit/payoff is tied to project performance or outcomes, and hence the payoff to the development organization depends on the quality of the delivered software system.

Each of the three contract forms is known to work well and be preferred more often in certain contexts than in others. For example, Wu et al. (2012) suggest that fixed-fee contracts fit better in a context of simple software projects that need relatively short development time. In contrast, more complex projects that bear more uncertainty, take longer time, and are tracked based on phases and critical milestones would benefit from variable price contracts, assuming efficient and effective auditing process is set up in place. Also, multi-stage contracts fit long-year large-scale projects and assist their delivery by opening up opportunities for effective use of information gathering and learning during the project, for example, by means of mid-term project evaluations, which would reduce project risk in the periods that follow and would improve team performance because they foster reflections on what works and what does not (Wu et al., 2012). Moreover, critical to the success of every contract-based project are the SLAs that form an addendum to the contract and describe the quality aspects of the products and services to be delivered by the development organization (Goo et al., 2008). They serve the purpose of establishing the parties' expectations, defining contacts in case of user problems, and defining metrics by means of which the effectiveness of the contracted services and processes as well as agreed upon deliverables will be measured and controlled.

From RE and software architecture perspectives, large contract-based systems imply massive documentation processing effort to be invested by RE staff and SAs (Nekvi et al., 2012). As an illustration of a complex project and its contract-related documentation, we refer here to the empirical report of Song and Hwonk (2012), which presents a large-scale infrastructure system that integrates numerous sub-systems, not all of which are software systems but also hardware ones like fire detection and suppression, routers, phone switches, ductwork, and fiber cable. Such a project may entail a contract containing of around 4000 clauses, and systematic processes are needed to be in place for categorizing the contractual requirements from goal-level to design-decision-level. Beyond the contract, the SLA document may well be voluminous too: The Goo et al. (2008) report, a well-designed SLA document, may include up to 11 contractual issues, and each of them might be the subject of a separate document.

Moreover, contract-based projects are usually more complex than in-house development systems because they come with project execution requirements, long-term obligations, and submittals (Berenbach et al., 2010) that are legally binding and impact how the professionals proceed at each step of the project. As these authors recommend, "*clauses dealing with the quality of deliverables must be analysed to create derived requirements that are testable and are acceptable to the client*" (p. 31). Also, these projects' requirements often have to demonstrate compliance with government regulations and standards, which becomes a key RE activity (Nekvi et al., 2012).

Furthermore, Gopal and Koka (2010) indicate that in contrast to in-house development projects, monitoring the software development organization and ensuring compliance to the project execution requirements is an explicit activity that orchestrates the vendor-client relationship at organizational level, for example, the mechanisms of how the software services vendor ensures quality and the returns on the vendor organization's investment in quality.

13.3 EMPIRICAL STUDIES ON THE SOFTWARE ARCHITECTURE PERSPECTIVE ON QRs

This section provides a rundown of how the software architecture perspective on QRs has been studied according to published systematic reviews and empirical studies in the fields of RE and software architecture. Although empirical research on the interplay of software architecture and QRs engineering is gaining momentum, studies in large-scale contract-based project contexts are scarce. To identify related work on SAs' perspective on QRs, we looked at the empirical studies that were included in two systematic literature reviews on QRs (Berntsson-Svensson et al., 2010; Guessi et al., 2012). We also searched the Scopus digital library (www.scopus.com) for publications that came out in the second half of 2012 and in 2013.

The two systematic reviews (Berntsson-Svensson et al., 2010; Guessi et al., 2012) provided examples of projects and organizations described in recent publications by both researchers and practitioners. We found, however, that most of these studies have taken the RE perspective exclusively. Very few studies have included the perspectives of SAs, despite the clear consensus in the RE community that the perspective of SAs is important (Ameller et al., 2012; Avgeriou et al., 2011; Bass et al., 2003; Capilla et al., 2012; Heesch van and Avgeriou, 2011; Poort et al., 2012a; Sommerville, 2005). For example, the review of Berntsson-Svensson et al. (2010) found only 18 studies to be well-documented and stating explicitly the perspectives taken in the empirical research. While preparing for this research, we reviewed these 18 studies and found that none of them considered the SAs' perspective. In the period of 2010-2013, we found six publications (Ameller and Franch, 2010; Ameller et al., 2012, 2013; Heesch van and Avgeriou, 2011; Poort et al., 2012a,b) dedicated specifically to the SAs' perceptions on QRs. These studies agree that the perspectives of SAs and RE specialists on QRs differ. However, the experiences these studies report come mostly from small- and mid-sized projects. In this chapter, we complement these results from the RE perspective and smaller projects with findings from large contract-based projects from SAs' perspective.

13.4 RESEARCH PROCESS

13.4.1 Research objective and research plan

Our study's overall research objective is to understand *how SAs cope with QRs in large and contract-based software system development projects*. We aimed to provide an in-depth understanding of the QRs' engineering activities as experienced by SAs while executing their projects.

In line with the study's objective, we defined the following research questions:

RQ1: How do SAs understand their role with respect to engineering QRs?
RQ2: Do SAs and RE staff use different terminology for QRs?
RQ3: How do QRs get elicited?
RQ4: How do QRs get documented?
RQ5: How do QRs get prioritized?
RQ6: How do QRs get quantified, if at all?
RQ7: How do QRs get validated?
RQ8: How do QRs get negotiated?
RQ9: What role does the contract play in the way SAs cope with QRs?

We note that these questions are open and explorative in nature because (as said earlier) the context of contract-based systems delivery projects is under-researched and no empirical studies on SAs' perspectives on QRs in this context have been published so far. This said, we started without any preconceived ideas about how SAs' behavior should or could be. Instead, we expected to learn such details from the lived and perceived experiences of the participating practitioners in the field. We also note that exploring such questions is a necessary first step to understand what is happening in the field concerning the phenomenon of study (i.e., QRs engineering). Only once this understanding is there, can we motivate and formulate narrower-focused questions for follow-up studies and for quantitative research.

We chose Yin's method of exploratory case study research (Yin, 2008) as the guiding methodological source to plan, execute, and report a multiple case study with 21 software project organizations. Our choice for this research methodology is justified by our research interest and commitment to explore a real-life phenomenon in the context in which it happens. As Bensabat et al. (1987) suggest, a case study is a particularly suitable research method to situations in which (1) researchers study socially constructed processes in systems development projects and seek to analyze the practice and (2) researchers want to answer "how" and "why" questions and comprehend the nature and the complexity of the processes of interest. In our exploratory case study, we expected to earn a rich and contextualized description of the practices, interactions, and roles of SAs. We also expected to understand the influences that the presence of the contract exercises on the professional behavior of the SAs (e.g., the choice of practices, interactions, and roles).

Our case study used structured open-end in-depth interviews with SAs from a wide range of companies and business sectors. The application domains where the practitioners developed software solutions represent a rich mix of fields including telecommunications, real estate management, air transportation, entertainment (online gaming, video streaming), educational services, online travel planning services (e.g., hotel/flight booking), and ERP.

The overall process of performing our study included these steps: (1) composing an interview guide following the guidelines in King and Horrocks (2010), (2) doing a pilot interview to check the applicability of the guide to real-life context, (3) carrying out interviews with practitioners according to the finalized questionnaire, (4) identifying and following up with interviews with those participants who possessed deeper knowledge or a specific perspective, (5) analyzing the data, and (6) preparing the report.

The subsections below present the practitioners involved and the data collection and analysis methods that we used.

13.4.2 The case study participants

The 21 SAs came from 15 companies in the Netherlands, Belgium, Finland, and Germany as presented in Table 13.1. At the time of the interviews, these companies were running large contract-based projects. There were two IT vendors that were large multinational consulting companies responsible for delivering and rolling out multicomponent and interorganizational ERP solutions based on the packaged software of SAP and Oracle. Seven of our interviewees were SAs employed by these two vendors and involved in ERP projects at these vendors' clients' sites. Furthermore, we had two SAs from a large insurance business working with internal IT staff and software development subcontractors on delivering a large custom software solution. Two other SAs were employed in a video-streaming services provider dealing with software component vendors. The rest of the companies were represented by one

Table 13.1 Case Study Participants and Organizations

ID	Business	System Description	Team Size (# of People)	Project Duration (months)
P1.	Large IT Vendor	ERP package implementation (Oracle)	35	18
P2.	Large IT Vendor	ERP package implementation (SAP)	60	15
P3.	Large IT Vendor	ERP package implementation (SAP)	75	18
P4.	Large IT Vendor	ERP package implementation (SAP)	41	12
P5.	Large IT Vendor	ERP package implementation (SAP)	51	12
P6.	Large IT Vendor	ERP package implementation (Oracle)	45	12
P7.	IT Vendor	ERP package implementation (SAP)	40	18
P8.	Software Producer	Online learning environment	22	12
P9.	Software Producer	Sensor system for in-building navigation	35	12
P10.	Software Producer	Online ticket booking application	15	12
P11.	Oil & Gas	Logistics planning application	21	12
P12.	Insurance	Web application for client self-service	61	24
P13.	Insurance	Client claim management and reimbursement app	53	16
P14.	Real Estate	Web application for rental contract handling	42	18
P15.	Air Carrier	Web app for passengers' feedback processing	11	14
P16.	Video Streaming	Viewer recommendation management system	18	18
P17.	Video Streaming	Viewer complaint management system	45	9
P18.	Online bookstore	Order processing system	15	10
P19.	Online game producer	Gaming system	81	21
P20.	Online travel agency	Room deal identification system	45	12
P21	Online game producer	Gaming system	97	19

participant each. We note that while there were SAs who were employed in the same company, no two or more practitioners were engaged in the same project. For example, the two SAs (see participants P12 and P13 in Table 13.1) who were employed in the insurance company were working on two different projects. This allowed us to collect experiences of a total of 21 projects.

The practitioners were selected because they (i) had professional backgrounds pertaining to our research questions and (ii) had the potential to offer information-rich experiences. Also, they demonstrated an interest in exploring similar questions from their companies' perspectives. All 21 SAs had the following backgrounds:

1. They all worked in large projects running in at least three different development locations in one country and had clients in more than two countries.
2. All SAs had at least 10 years of experience in large systems and were familiar with the interactions that happen between SAs and RE staff.

3. All SAs worked in contract-based projects where contracts between parties were established in two steps (Lauesen, 2002): first, a contract was agreed upon for the purpose to get the requirements documented in sufficient detail, so that an SA can use them to work on architecture design. Then, a second contract dealt with the system delivery itself.
4. All participants had "Software Architect" as their job titles (Gorton, 2011) and were employed full-time by their employers. Also, the RE staff and the project managers with whom these SAs worked were employed full-time in their respective jobs. The RE specialists had various job titles, such as business analysts, business system analysts, information analysts; however, they worked in these roles exclusively, in the sense that RE specialists were not senior developers who were taking on RE or project management tasks.

The pricing agreements varied across the participating companies. Some were fixed-price, others variable, and a third group combined fixed-price and variable. Five SAs worked in outsourcing contracts, and 16 were employed on projects where software development subcontractors were participating. All SAs deemed their contracts comprehensive and aligned with the spirit of their projects. (In other words: None suggested that their organization had any issue with the contract.) The projects were considered successful by both parties in the contract. The SAs got to know the first author during various business and research conferences in the period of 2001-2012. Using purposive sampling (King and Horrocks, 2010), she chose the interviewees, based on her knowledge about their typicality. The number of participants was large enough to provide a diversity in viewpoints. We planned the interviews to be "structured" (King and Horrocks, 2010) with regard to the questions being asked during the session. This means the interviewer controlled what topics would be discussed and in which order.

We note that interview-based exploratory case studies usually are intended to promote self-disclosure, and that is what we intended in this work. We collected data via one-on-one interactions of a researcher with each of the interviewees that have exposure to various application domains but also common professional values, backgrounds, and common roles in which they execute their professional duties. As in King and Horrocks (2010), interview studies are not used to provide statistically generalizable results applicable to all people similar to the practitioners in a specific study. The intention of an exploratory case study is not to infer, but to understand, and not to generalize, but to determine a possible range of views. Therefore, in this study we adopt, based on the recommendations in King and Horrocks (2010), the criterion of transferability as a useful measure of validity. Transferability asks whether the results are presented in a way that allows other researchers and practitioners to evaluate if the findings apply to their contexts.

13.4.3 The research instrument for data collection

Because we wanted to understand the tasks of SAs and their interactions with other project stakeholders, we looked at data collection techniques that help extract detailed information about social and cognitive processes through which a professional achieves his or her tasks' goals (Wortham et al., 2011). One prominent technique is the method of semi-structured interviews enabling the interviewer to ask predetermined questions while still remaining flexible, so that the interviewee is not prevented from elaborating on topics that may emerge but were not directly asked about by the interviewer

(King and Horrocks, 2010). Each interviewee was provided beforehand with detailed information on the research purpose and the research process. To help the subjects to verbalize their roles and tasks with respect to QRs, we asked them to think about their current contract-based project. The interviewing researcher then probed each participant for the tacit knowledge used to deal with the QRs engineering in his or her project.

Each interview lasted 35-45 min. Nine interviews took place face-to-face, and 12 on the phone. After each interview, the researcher followed up with each participant to get access to documents related to the study and that the SAs referred to during the interview, for example, job descriptions or standards.

13.4.4 Data analysis strategy

Our data analysis was guided by the Grounded Theory method of Charmaz (2008). It is a qualitative approach applied broadly in social sciences to construct general propositions (called a "theory" in this approach) from verbal data. It is exploratory and well suited for situations where the researcher does not have preconceived ideas, and instead is driven by the desire to capture all facets of the collected data and to allow the theory to emerge from the data. In essence, this was a process of making analytic sense of the interview data by means of coding and constant/iterative comparison of data collected in the case study (Figure 13.1).

Constant comparison means that the data from an interview is constantly compared to the data already collected from previously held interviews. We first read the interview transcripts and attached a coding word to a portion of the text—a phrase or a paragraph. The "codes" were selected to reflect the meaning of the respective portion of the interview text to a specific research question. This could be a concept (e.g., "willingness to pay") or an activity (e.g., "operationalization," "quantification"). We clustered all pieces of text that relate to the same code in order to analyze it in a consistent and systematic way. The results of the data analysis are presented in the next section, after which a discussion is added.

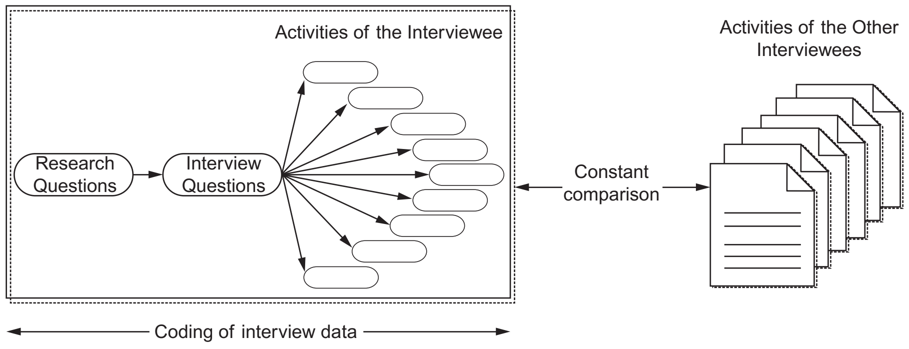

FIGURE 13.1

The GT-based process of data analysis.

13.5 RESULTS

Our findings are presented as related to each research question. As it is common in qualitative studies (Charmaz, 2008), we supplement the observations with interviewees' quotations.

13.5.1 RQ1: How do the software architects understand their role with respect to engineering QRs?

We found that while reflecting on their roles in QRs engineering, SAs grounded their reasoning on their job descriptions in the organizations where they were employed. In the experience of all our participants, their respective companies had job descriptions that listed the most important duties, skills, and competence levels for SAs, including those related to QRs. As said earlier, the SAs' jobs were separate (Gorton, 2011) from those of RE professionals—known as "business analysts," or "systems analysts" —and of software developers—having job titles as "programmer/analyst" or "senior programmer/analyst." This said, no SA from the 21 involved in our study has been tasked with, for example, coding or business requirements specifications.

For the purpose of illustrating the ways in which QRs are mentioned in an SA's job description, we provide below an excerpt from one participant's job description:

> *The software architect is responsible for the technical direction of a project. Makes high level design choices for the software structure, frameworks, protocols, and algorithms. Determines coding practices, development tools, and validation requirements. Performs path-finding and surveys technologies. Interacts with multiple technologists in the company and within the industry as well as between developers and project managers to evaluate feasibility of requirements and determine priorities for development. (Participant P11).*

QRs were included in a variety of responsibilities that go with the SA's job, for example, resolving technical issues related to QRs, evaluating feasibility of QRs (e.g., mapping technology limitations against desired QRs), validating QRs, and communicating technology direction in regard to QRs. These responsibilities were explicitly stated in the SAs' job descriptions, as illustrated below:

> *Understand solution performance and capabilities and drive software requirements, architecture and priorities by evaluating feasibility of requirements. (Participant P15)*
>
> *Ensure Android system needs are factored into future specifications [of functional and QRs] by taking into account Google Android feature roadmap and by path-finding usage models for Android systems. (Participant P21)*
>
> *Lead path-finding work on power, performance, and quality associated with delivering differentiating end user experiences. (Participant P19)*
>
> *Act as a trusted advisor to the business team and play a key role in transforming business plans, and product concepts [functional and QRs], into real world product architectures [software architectures]. As the senior technical leader within the organization the Systems/Software Architect will also engage our major customers in communicating our technology direction and innovative approach. (Participant P18)*
>
> *Serve as the organization's technical authority. Own and champion the product technology roadmap and ensure alignment with business, user and quality requirements. (Participant P10)*

Provide architectural expertise, direction, and assistance to software development teams to conceive and build leading edge products based on a standard architecture. Assess the feasibility of incorporating new technologies in product designs. (Participant P11)

Provide technical leadership in the resolution of complex technical issues related to security, scalability, usability and privacy, across the organization. (Participant P12)

Make high level design choices for the software structure, frameworks, protocols, and algorithms. Determine coding practices, development tools, validation requirements, and benchmarking performance. (Participant P16)

We note that the SAs had a wide range of duties (e.g., creating the architecture design for the system, technology road-mapping, among others) and QRs engineering was only one of the many in the range. Moreover, the SAs' job descriptions did not elaborate in detail of the exact QRs engineering tasks of the SAs. (This in itself confirmed that interviews with the SAs were indeed necessary to understand more in depth the QRs tasks; if the QRs engineering tasks were explicitly detailed in the job descriptions, one might consider carrying out a documentary study; this would be a completely different research design.)

Because all the organizations were mature in terms of project management processes and process-oriented thinking, the roles of the SAs were established and they were clearly visible for their fellow team members. We found four distinctive roles that SAs identified themselves with regarding QRs (see Figure 13.2). These roles are: (1) serving as a "bridge," (2) serving as a "gatekeeper," (3) serving as a "balance-keeper," and (4) providing "QR engineering as a service." We elaborate on these distinctive roles below.

Thirteen of the 21 SAs thought of their role as a "*bridge*" between QRs and the underlying technology that they were dealing with.

You've got to translate what their clients want in terms of working solutions and technology choices. (Participant P1)

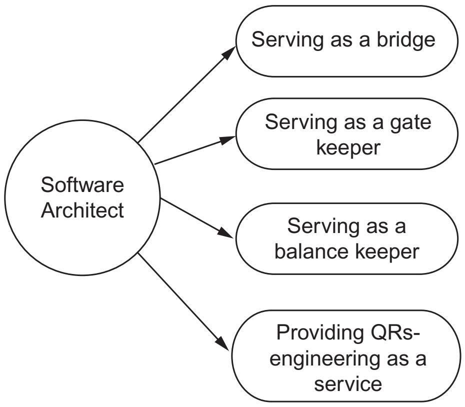
Software Architect
Serving as a bridge
Serving as a gate keeper
Serving as a balance keeper
Providing QRs-engineering as a service

FIGURE 13.2

Roles that SAs play in QRs engineering in contract-based delivery projects, as perceived by the case study participants.

> *You learn pretty quickly to stretch the technology so that it works for those who you call "requirements engineers," I mean here our Business Analysts and their "patrons," the business unit directors. (Participant P2)*
>
> *I have days when I do nothing but "translate" the language of our millions of players into the features that are technologically implementable in the next release. All I do is walk through a list of "user experiences" that our requirements folk want to launch in the market, and I make sure that it all works well at implementation level, that features build upon each other, so that the overall gaming experience and enjoyability are never compromised. That's why I'm coming to work every day. (Participant P19).*

Other seven SAs thought of their roles as "*review gatekeepers*" because of their active involvement in QRs reviews, contract compliance evaluation, and sign off:

> *My review is like a green light for the implementation team's work. (Participant P2)*
>
> *You are the gatekeeper in the game, and you've got to act like this so that you get them focus on the right things. If you do not tell them what the right things are, no one else will. (Participant P11)*
>
> *If you are sick, on vacation, or on a conference, there is no way this can go unnoticed; they will need you, and if there is no one to tell them that what they are doing is right [according to contract and project goals], I mean it pure architecturally; they are going to wait for me so that I give the architecture approval. Otherwise, we run a risk of screwing up things miserably. (participant P15).*

Among those seven who considered themselves as gatekeepers, there were two who explicitly thought of themselves as "*balance keepers*" as well:

> *I watch for the developers to avoid quick-fix solutions and technical debt. In our application area [gaming], this puts the qualities out of balance; you advance, only later to find out user experience is jeopardized… (Participant P19).*

Last, one SA defined his role as a provider of "*QRs engineering as a service*" because he embraced ownership and responsibility for anything related to QRs in his project.

> *I provide QRs engineering as a service to business owners and business analysts and architecture as a service to developers and integration specialists. (Participant P21).*

How did SAs get assigned to projects? Did they choose their projects or did they get assigned? All our participants thought that their interest in working on a particular project in a specific application domain has been considered by their managers. We observed a consensus that it was their domain knowledge that mattered most in the SAs' assignments to projects. Our participants explained this with the important payoffs that domain knowledge brings to a project.

> *There is little point to put you on a SAP Human Resources implementation project, if you earned your credits in Financial Accounting. You leverage your domain knowledge for your client. Unless you want to get exposure to a new domain and start building up expertise in there. (Participant P1).*

Ahead of time, the SAs signaled to the higher management their interest in a specific project. So, the project managers were aware of the willingness of the SAs to join and make a difference.

There is no way you get assigned to a project that includes 80 people and lasts more than a year. You have to want to get there. And if you do not step in, someone else would do, and you would miss on an important opportunity to make a difference with your knowledge and experience. (Participant P7).

13.5.2 RQ2: Do SAs and RE staff use different terminology for QRs?

Because all interviewees had more than 10 years of business experience in their respective sectors, they considered communication clarity with RE staff a non-issue. Even when working with less experienced RE staff, they were of the opinion that the domain knowledge they have is instrumental for spotting missing or incomplete requirements. For example, if one is to develop an online system for processing feedback of clients in the air travel sector, the obvious QR of the highest priority would be scalability.

If your RE person says nothing about it, you are absolutely sure there is something going wrong here. And you better pick up the phone and call them ASAP (as soon as possible) because they may be inexperienced and who knows what else could go wrong beyond this point. Not doing this is just too much risk to bear. (Participant P15).

We do not use the term QRs, not even nonfunctional requirements, we call them ISO aspects, because we are ISO-certified and our system must show compliance to the ISO standards. Also, our requirements specialists have in their template a special section that contains these aspects. We enumerate them one after another. All relevant aspects must be there, security, maintainability, performance, you name it. It's more than 40 aspects. They [the RE staff] know them and we know them, so we have a common ground. (Participant P11).

An interesting observation was shared by those SAs delivering large SAP systems (participants P2, P3, P4, P5, and P7, in Table 13.1). They indicated that the SAP vendor's Product Quality Handbook (SAP, 2000) includes around 400 QRs that are implemented in the SAP's standard software package and that everyone in the project team knows about. If there were QRs that are specific to a particular client in a project and that are unaddressed in the Handbook, then those should be specified and added on top of the 400 that come in the SAP's package.

I open the ISO manual and start from there; and then a kind of expertise we do this for them... (Participant P7)

We make the note that the company SAP is ISO-9001-certified and as such, they have developed and implemented three quality management systems (QMS), all certified according to ISO 9001 (ISO, 2008). These are (1) the QMS for SAP Global Development, which ensures that SAP solutions meet the highest possible standards; (2) the QSM for SAP Active Global Support, which provides tailored services to maintain the quality of installed solutions, and (3) the QMS for SAP IT, which ensures the stability and quality of SAP's internal IT infrastructure; next to this, SAP IT is certified according to the requirements specified in ISO 27001:2005. Because each QMS defines standardized, certified processes for its domain, it enables employees to share and apply best practices within their domain. The three QMSs provide common information (e.g., management review, document control, record control, audits, corrective and preventive actions, and personnel development), thereby ensuring the consistent application of this information across SAP. For more information, we refer interested readers to visit the SAP's corporate Web site (www.sap.com), which provides the current ISO certificates for SAP's QMSs.

How many SAs are enough for engineering QRs? One or two SAs have been involved in their projects. In the case of two SAs being involved, our interviewees consisted of one SA from the vendor and one SA from the client.

You may work with many RE people for the different subject areas, but one architect must be there to aggregate their input and evaluate what it means for the underlying architecture. (Participant P6)

13.5.3 RQ3: How do QRs get elicited?

Our study revealed two types of elicitation approaches being used: (1) checklist-based elicitation and (2) user-experiment-based detection of QRs, which included a variety of techniques centered on active user involvement (see Figure 13.3).

Sixteen of the 21 SAs used checklists in eliciting QRs. These were based on a variety of sources:

i. ISO standards (e.g., 25045-2010, 25010-2011, 25041-2012, 25060-2010).
ii. Architecture frameworks, be they company-specific or sector-specific.
iii. Internal standards (e.g., vendor/client-organization-specific).
iv. Stakeholder engagement standards (e.g., AA1000SES (AccountAbility, 2011).
v. The related new Knowledge Areas from the Project Management Body of Knowledge PMBOK (PMI, 2013).

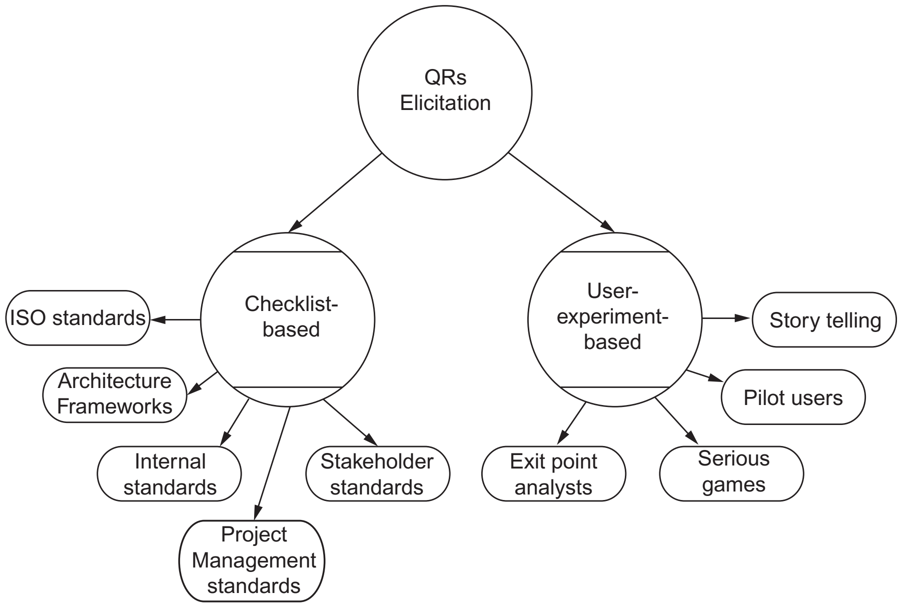

FIGURE 13.3

Elicitation practices as per the experiences of the SAs in the case study.

Regardless of the source, the interviewees agreed that the checklist-supported process is always iterative, because not all QRs could get 100% clear at the same time.

Most quality attributes are known only later, once you nail down the key pieces of functionality. I mean, you first fix the functional requirements and then let the quality stuff grow out of them. (Participant P1).

In the participants' views, the fact that most QRs are not global to the system but act on specific pieces of functionalities imposes an order in the elicitation activities: SAs expect the functional requirements to be elicited first, so that they might use the functional requirements specification as the basis for eliciting QRs.

How otherwise will you know which attribute matters to what piece of functionality? (Participant P9).

We do this top-down, gradually adding up more clarity to our nonfunctional requirements specs. There is no way to know all the details about them in one big batch. (Participant P8).

Five SAs said that QRs are never elicited but are discovered or detected. For example, one SA (who worked in the gaming sector) was involved in an experimental serious-game-based process specifically designed to "detect" QRs:

I have a background in serious gaming for requirements acquisition, and this was instrumental to fixing scalability and repairability, which are two of our most important nonfunctional requirements. (Participant P19).

In a computer-based environment designed for entertaining games but with non-entertainment goals, this SA has set up a QRs elicitation process with fixed starting and finishing points. In this process, users play, and then the SA, the RE staff, and other relevant roles in the project team observe the play and uncover what users want the system to do when it restores itself after a failure.

Two other participants shared that they had "*a bunch of pilot users who volunteer to play the role of guinea pigs; they are passionate about the system and love to tell you where it fails to fulfill their expectations in terms of performance, availability, and user experience*." (Participant P10).

One SA said that he had been using storytelling techniques to uncover QRs, together with his RE counterpart and his clients.

I'm trying to collect stories from the business users that I got to know through our project meetings and this helps me understand how these users make meaning of their role as insurance claim analysts. The business analysts [the RE staff] and I then walk over these stories and analyze what they tell us. Most of the time, I ask questions in the format: "Tell me about a work day when the system worked perfectly for you. Tell me about your day when it was extremely frustrating. When the system was extremely slow, how did it affect you?..." You know, like this I can guess some common "red lines," themes if you want to call them, and I go validate them with other users. (Participant P13).

One other SA (working in the gaming sector) indicated "*exit point analysis*" as the technique that his project team used to discover requirements. Exit points are the points in time when a player decides to stop playing a game by canceling his or her subscription. The game provider on whose projects the SA was employed made its clients fill out an exit survey in order to cancel a subscription. Analyzing the data collected in the survey served the purpose of finding out why a player is quitting a game and what requirements the project team could focus on in order to fix gameplay issues and improve the overall user experience of the game, and thus increase the "player retention rate."

13.5.4 RQ4: How do QRs get documented?

Sixteen of the 21 SAs specified QRs by using predefined templates. Some of them were vendor-specific. For example, in SAP projects, SAs used SAP's standard diagram notation called Technical Architecture Modelling (Groene, 2012), because it has been part of the SAP Architecture Curriculum (Groene et al., 2010). Others were derived based on (i) an ISO standard, (ii) the House of Quality (HoQ) of the Quality Function Deployment (QFD) methodology (Hauser and Clausing, 1988), (iii) the Planguage approach (Gilb, 2005), (iv) the INVEST grid approach (Buglione, 2012), and (v) the contextual design approach (Rockwell, 1999).

The other five SAs were using natural language text that provides at least the definition of each QR, plus information that is needed by the end user when testing its acceptance and the ways to demonstrate that the system meets it. The amount of detail in these specifications varied based on what the SAs deemed important to be provided to them by the RE staff or the users. For example, one SA wanted to hear a story from a user on "*how the user will know the system is slow? A user can tell you: 'If I manage to get one of my phone calls done while waiting, this means to me it's very slow.'*" (Participant P12). Other SAs said they write their QRs definitions next to the functional requirements, if these are specified in a process model or a data model.

> *This way you will know which smallest pieces of functionality and information entities are affected by which QR. (Participant P1).*

13.5.5 RQ5: How do QRs get prioritized?

All SAs agreed that (i) they make QRs trade-offs as part of their daily job on projects and (ii) the project's business case was the driver behind their trade-off decision making. The key prioritization criteria for making the QRs trade-offs were cost and benefits, evaluated mostly in qualitative terms but whenever possible also quantitatively, such as person-months spent to implement a QR. Perceived risk was identified as subsumed in the cost category, because SAs deemed a common practice the tendency to translate any risks into costs to estimate in a contractual agreement.

However, next to perceived cost and benefits, 12 SAs put forward two other QRs prioritization criteria: *client's willingness to pay* and *affordability*. The first is expressed as the level of readiness of client organizations to pay extra charges for some perceived benefit that could be brought by implementing a specific QR. Six SAs elaborated that this criterion alone is instrumental to split up the QRs in three groups. (i) Essential: This group includes those QRs that directly respond to why the client commissioned the system in the first place (e.g., for, a hotel reservation system). It's absolutely essential that a secure payment processing method is provided. (ii) Marginal: This group includes those QRs that are needed, despite clients being unwilling to spend much on them. For example, a user interface feature might be appreciated, but it would be and would be valued at a few hundred euro, rather than thousands of euro. (iii) Optional: This group includes QRs that clients will find enjoyable to have but would not be willing to pay for, such as flashy animation effects in a game system that are and fun, yet not truly a "*money-maker*."

Next, affordability is about whether the cost estimation of a QR is in accordance with the resources specified in the contract and with the long-term contract spending plan of the client organization. This criterion determines whether a QR is aligned with (short-term) project goals and/or (long-term) organizational goals.

Furthermore, we found SAs' experiences differed regarding who ultimately decides on the QRs priorities. Thirteen SAs suggested that in their projects the prioritization is linked to a business driver or a key performance indicator (KPI) in the project and that it is the project's steering committee to decide on the priorities. They deemed the prioritization process "*iterative*" and a "*learning experience*" in which the SAs learn about those QRs that the business drivers "*dictate and how desperately the company needs these QRs*," whereas the RE specialists and the business owners learn about the technical and cost limitations.

> *I educate them [the business representatives] on what risks might look like, why the things that they thought of as being simple turn out to be extremely difficult to implement on our end, and extremely costly, too. (Participant P18).*
>
> *You learn why the system must have certain qualities and how this affects the organizational bottom line... No quality, no company in business; it's that simple... (Participant P16).*

In contrast to these 13 SAs, the other 8 considered themselves as the key decision makers in setting the priorities:

> *You can stretch a technology to a certain point. No matter what your client wants, once in a while you've got to say "no" and push back decisively. The decision making on what's high priority and what's not has been delegated to you, so it's your responsibility to say "no" if you think a [QR] priority is unrealistic. (Participant P11).*

Concerning the requirements prioritization method being used in their projects, 20 SAs suggested no explicit use of any specific method other than splitting up QRs in categories of importance, namely "essential," "marginal," and "optional." SAs named these categories differently, yet they meant the same three. One SA named EasyWinWin (Boehm et al., 2001) as the group support tool that implements collaboration techniques helping the requirements prioritization and negotiation activities in a project. Three out of the 20 shared that nailing down two or three most important QRs is a non-issue, because of the obviousness of these requirements, for example,

> *If you develop a game, it's all about scalability and user experience. You have no way to make your fancy animation work if you cannot get scalable and if you jeopardize user experience. (Participant P19).*

What our participants deemed an issue is the prioritization of those QRs that take a less prominent place from user's perspective, for example, maintainability and evolvability. Such requirements make a lot of business sense from the developers' perspective; however, they are "invisible" for business users. For example:

> *You think making the system evolvable is important to business.... But what if your project's steering committee does not care? They can be very short-term minded. And it's a downhill battle to argue over the priority of this kind of requirements. All they see is the clients' perception on it.... The trouble is that when your manager later on wants to keep you accountable for that... for choices they made some long time ago... years ago... (Participant P13).*

13.5.6 RQ6: How do QRs get quantified, if at all?

All SAs agreed that expressing QRs quantitatively should not happen very early in the project. This was important in order to prevent early and not-well-thought-out commitments. Four out of the 21 said that they personally do not use quantitative definitions of QRs. Instead, they get on board an expert specializing in a specific aspect (e.g., performance, usability, or scalability) and who "*does the quantification job*" for them. For example, in the gaming application domain, one participant revealed that it is common to recruit so-called Enjoyability Evaluation Specialists who generally use quantitative models to measure enjoyability or playability of gaming software, for example, the GameFlow model (Sweetser and Wyeth, 2005) and its variants (Sweetser et al., 2012), which rest on the well-known theory of flow, or some other models that capture the "*entertainment value of a game*," for example, the level of challenge of the task and how it matches the player's abilities (Yannakakis et al., 2006).

Ten other SAs used as a starting point the quantitative definitions that were pre-specified in the contract (e.g., a contract may state explicitly that the system should scale up to serve thousands of subscribers). However, 8 of the 10 warned that more often than not contracts address design-level requirements (Lauesen, 2002), including detailed feature specifications of how QRs are to be achieved, or a particular algorithm rather than required quality attribute value and criteria for verifying compliance. Confusing QRs with design-level requirements was deemed a critical issue in industry; it points to a mismatch in understanding what is really quantified and by using what kind of measures: Design-level requirements are quantified by using product measures and not project measures that are important for contract monitoring purposes. However, often contracts use product and project measures incorrectly, the final effect being that a number of project tasks related to implementing QRs do not get "visible" but "implicit," and therefore no budget is previewed for them and, in turn, the client does not commit to pay.

For example, when a Functional Size Measurement (FSM) method is used in a contract for the purpose of sizing the project at hand, its application would be technically "unsound" if the productivity for a project is calculated by dividing, for example, the number of Function Points (whatever the variant is used, IFPUG, COSMIC, or another ISO 14143-1:2007 compliant technique)—that's a product size measure only for sizing functional user requirements—by the overall project effort, including both functional-requirements-related effort and QRs-related effort. Because "one size doesn't fit all" (i.e., Function Points do not fit for sizing both functional requirements and QRs), what practitioners needed is at least a second unit of measure for sizing QRs (or, as termed in the ISO language, the Non-Functional Requirements). Most recently, the International Function Point User Group (www.ifpug.org) released a new technique for quantifying nonfunctional requirements, called SNAP (Software Nonfunctional Assessment Process) (IFPUG, 2013). The SNAP method adopted the ISO/IEC 25010:2011 product quality model that updated and extended the previous ISO/IEC 9126-1:2001 one. An added value for QRs analysis and sizing that all the existing FSM technique provide is the "elementary process" concept, that is, the "smallest unit of activity" perceivable from a user, self-consistent and independent. The "elementary process" concept is also included in the new SNAP method, which could help a lot when dealing with QRs because it ensures practitioners have a reference point in terms of level of granularity to be considered in QRs sizing.

Furthermore, another part of "missing" effort that becomes only evident when comparing estimates and actual values at the project closure comes from organizational/support processes in the scope of the whole project; such processes are, for example, Project Management, Measurement, Quality Assurance, which refer to another measurable entity that's the project and not its product. In that way (of using product and project measures incorrectly), there is a risk of underestimating the real effort that would be needed for QRs.

Another issue that crystallized in the interviews was the danger of specifying a QR by using a single number. Examples provided by five SAs referred to availability and performance requirements. Too often, these participants observed contracts and project teams having thought of these requirements in terms of "*99% of the time*," "*all the time between 9 am and 5 pm*," or "*in peak hours*." What remains unmentioned, and in turn leads to ambiguity, is the workload and the frequency of interactions of events per period of time. Examples of interactions or events are "*requests by clients*," "*order click by shoppers*," "*hotel bookings per minute*."

> *Without giving a proper context, developers cannot make much sense of availability. I need to go search and double-check what kind of context they [the business managers] are talking about when saying "99% available"; and you cannot be sure you can create a good enough design; what kind of load do they [business managers] mean? 1000 new bookings per minute with the payment processing that may go with it? Or booking cancellations? Or some other tasks the online user may want to do? (Participant P20)*

These SAs thought that suggesting only one number in quantifying a QR severely limits the search space for solution options and trade-offs between those QRs that are important to clients and to developers:

> *Of course, the team can make it work and meet what the contract wants, but that's not the point. You do not want to look myopic, meet your numbers for the sake of them, and find yourself totally out of context. You should not preclude yourself from comparing alternatives and choosing the one that makes sense for the big picture. (Participant P11)*

Next, 7 out of the 21 SAs worked with a team of systems analysts on operationalizing QRs, which in essence meant decomposing them until reaching the level of architecture design choices or of the smallest pieces of functional requirements. Once at this level, the practitioners felt comfortable starting quantifying the QRs, for example, Function Points. However, no common quantification method was observed to be used. Instead, the architect suggested the choice of a method should match the project goal (e.g., toward whatever end quantification was needed in the first place). For example, if quantification of QRs is to serve project management and contract monitoring purposes, then it might be well possible that in the future those organizations experienced in Function-Points-based project estimation would use a way to approximate them in a quantitative way as done, for example, by the newly released SNAP method (IFPUG, 2013). In this way—differently than applying the old VAF (Value Adjustment Factor) to an FP-based evaluation—it's possible to quantify only the nonfunctional-requirements side of a project, which is useful in particular for maintenance projects where the functional side is not present at all (corrective/perfective maintenance) or partly present (adaptive maintenance), as defined in the ISO/IEC 14764:2006 standard (ISO, 2006). A simple example is the adaptation of an existing Web portal to become "accessible" (i.e., the QR in question is "accessibility"), as stated in the Web Content Accessibility Guidelines 2.0 (http://www.w3.org/TR/WCAG20/) or in the Section 508 rules (https://www.section508.gov/).

13.5.7 RQ7: How do QRs get validated?

All SAs were actively involved in QRs validation; 17 considered it part of their job and acted as the contact point for this task, while four SAs said that it's the job of the RE staff to ensure QRs are validated. These four SAs used the RE specialists as contact points on clarifying requirements.

Fourteen SAs participated in requirements walkthroughs with clients led by an RE specialist where clients confirm the functionalities on which the QRs in question were supposed to act. The walkthroughs were deemed part of the client expectation management process that the project manager established. The SAs considered them as the opportunity to inform the clients about those QRs that could not be implemented in the system or could not be implemented in the way the client originally thought:

> *You've got to educate them on what your technology can and cannot do for them and the walkthroughs are how this happens relatively easily. (Participant P1).*

Two of the 14 SAs complemented the walkthroughs with inspection checklists whose purpose was to detect flaws in the QRs. These SAs said they used the checklist at their own discretion, after a walkthrough was over. The SAs did this "*for themselves*," and not because their project managers or RE counterpart asked for it. This was a step—or "*an auxiliary means*" (as one SA put it)—to ensure they did not miss anything important for the architecture design:

> *I do this myself because this way I know I did everything to ensure we do not leave out things unnoticed. This happened too many times in my past projects and I know too well the trouble it brings. (Participant P10).*

The two SAs who were working in the game development business sector presented a perspective of what "the contents of a walkthrough" include in this specific domain. Unlike the other SAs who talked about walkthroughs in the context of functional requirements and quality attributes, these SAs used the term *walkthrough* as applicable at the following levels: (1) the "game play," which is the set of problems and challenges that a player must face to win a game; from RE perspective, the game play is the overarching concept that contextualizes and motivated all functional and QRs; (2) the game story, which is the plot and the character development and which determines how requirements would be grouped together; (3) the functional requirements, specific to each stage of developing of the game story; and (4) the QRs specific to each stage of developing of the game story.

> *Validating your plot is what you should do first. Only after you ensure your game play and story are validated can you go into validating functionals and nonfunctionals; otherwise, it would be a premature action. (Participant P21).*

Three SAs used the HoQ (Hauser and Clausing, 1988) to demonstrate the strength of the relationship between a QR statement and its operationalization in terms of either functional requirements or architecture design choices. We note that the organizations where these SAs worked have been experienced in using the QFD and the HoQ approaches in their large and ultra-large projects, so no additional training of project staff was needed for the purpose of validating the requirements in the SAs' current projects.

> *Once you populate your operational terms in the house of quality, you'll know if things work or don't. If things do not work, then you've got to step back and work on one-on-one basis with those business users or developers who are affected; we do not proceed, unless we first get this done; otherwise it's too risky and you do not want risks to creep into the project. (Participant P12).*

Two SAs validated QRs against internal architecture standards. Should they identify deviations from the standards, they escalate this to both managers and RE staff. In extreme cases, when QRs are grossly misaligned with the architecture standards, this should be brought to the attention of the steering committee, the program director, and the architecture office manager responsible for the project.

> *You have to inform them immediately that things have no way to work as planned, so they get back to negotiation and revise the concept. (Participant P8).*

13.5.8 RQ8: How do QRs get negotiated?

All SAs participated in requirements negotiations, but not all considered themselves as "*true negotiators*." While 16 SAs thought of themselves as the responsible person to run negotiation meetings and to push back if needed, the other five considered themselves to be "*information providers and mentors*" to the team, but not "*truly negotiators on QRs*" and that it's the project manager's responsibility to lead in the negotiation:

> *It's his job to sell it to the other parties. I'm just an internal consultant; what they do with my information is their business. (Participant P10).*

Among those 16 SAs who played the role of "true negotiators." Ten used the business case of the project as the vehicle to negotiate requirements. They considered this a common practice in enterprise systems projects.

> *If you talk money, then you talk business; that's how you get the quality attributes' voices heard. The game is lost, otherwise.... (Participant P20).*

Three SAs who worked on projects where user experience was the most important QR said their goal in QRs negotiation is to prevent the most important QR from becoming suboptimal if other QRs take more resources and attention. These SAs did not use the business case as such, but were considering effort and budget allocation as important inputs to negotiation meetings.

Other three SAs used the HoQ, EasyWinWin (Boehm et al., 2001), or the Six-Thinking-Hats method (de Bono, 1985) to reason about QRs in negotiation meeting. We note that the Six-Thinking-Hats is a general approach to resolving complex issues more efficiently, and companies use it for any negotiation situation, be it QRs related or not. The approach was well received and internalized in the company, and people "*had fun using it as it takes pressure off them in this kind of difficult conversations*" (Participant P16).

Does requirements negotiation happen face-to-face? Eighteen SAs relied on virtual meeting tools (e.g., a conference bridge).

> *The last time I attended a face-to-face meeting was 2 years ago... I can no longer imagine spending budget money on this. (Participant P2).*

Three out of our 21 participants thought of face-to-face meetings as indispensable:

> *I got to face them. Personally. To observe their faces, body language, you name it.... This way I know I can convince them without investing all my energy for the day in this.... (Participant P1)*

What is critical to the requirements negotiation outcomes? The experiences of the 21 SAs varied widely as to what makes the top-three most important skills that matter to negotiation. One common theme across the experiences of the participants was the ability to express themselves in financial terms.

> *Being able to translate architectural choices into budget figures means you are at the top of your game. (Participant P4)*
>
> *Do not get emotional, get financially-savvy and they will listen. (Participant P9)*
>
> *You've got to get good at it [business case terminology]. This is the only way for a large project to operate and negotiate decisions. (Participant P20).*

They suggested that the importance of being able to think in financial terms is consistent with the predominant thinking style that they use in their work, namely, that they think of software architecture as a shared "*mental model*," "*state of mind*," "*project culture*" (in the words of a few participants):

> *Your team delivers on contractual terms, this sets up the stage and the way you look at things; contracts are about money and commitments, about the bottom line, about power receivers and power losers, and you've got to think of architecture in these terms; face the big picture and use it as much as you can when negotiating nonfunctionals. (Participant P20).*

Being gatekeepers and balance keepers, does it happen that architects revise their own "architecture rules" as part of QR negotiation? Twelve of our SAs suggested this happened on regular basis. They considered this part of their job. SAs' knowledge on how to separate those rules that SAs could be flexible with from the rules that call for rigidity was deemed of paramount importance:

> *You have to know where, when, and how to break your own policies. And make sure you write down how you did this, and who you informed on this. You may desperately need this later on, if issues arise in your contractual process. (Participant P3).*
>
> *If you think of yourself as a QR engineering service provider, then you know your clients and you know how to be flexible; and I do not mean it as a negative thing. Think of it as boundary-setting, as a terrain that is constantly subjected to renegotiation.... You have to claim and reclaim your territory and they are free to do the same... so I revise things in the product architecture, and I must say, we have very good reasons to do so.... (Participant P12).*

13.5.9 RQ9: What role does the contract play in the way SAs cope with QRs?

All SAs deemed the contract "*the solid mechanisms*" for regulating the client-developer relationship. Because it determines rights, liabilities, and expectations, they thought of it as "*the specific something*" that sets the backdrop for any QRs engineering activity. Our participants revealed the following elements of the contract that in their experiences directly impacted how QRs engineering took place in their contract-based projects (see Figure 13.4): (1) service levels; (2) pricing, schedule, and payment; (3) warranty and liability; (4) project issue escalation procedures; (5) dispute resolution mechanisms; (6) termination procedures; and (7) information security, including both data security and business recovery planning.

In the SAs' experiences, there were five ways in which the contract influenced how they coped with QRs, as shown in Figure 13.4: (1) The contract enforced the cost-consciousness of the SAs and was used to evaluate the cost associated with achieving the various QRs; (2) the contract stipulated QRs

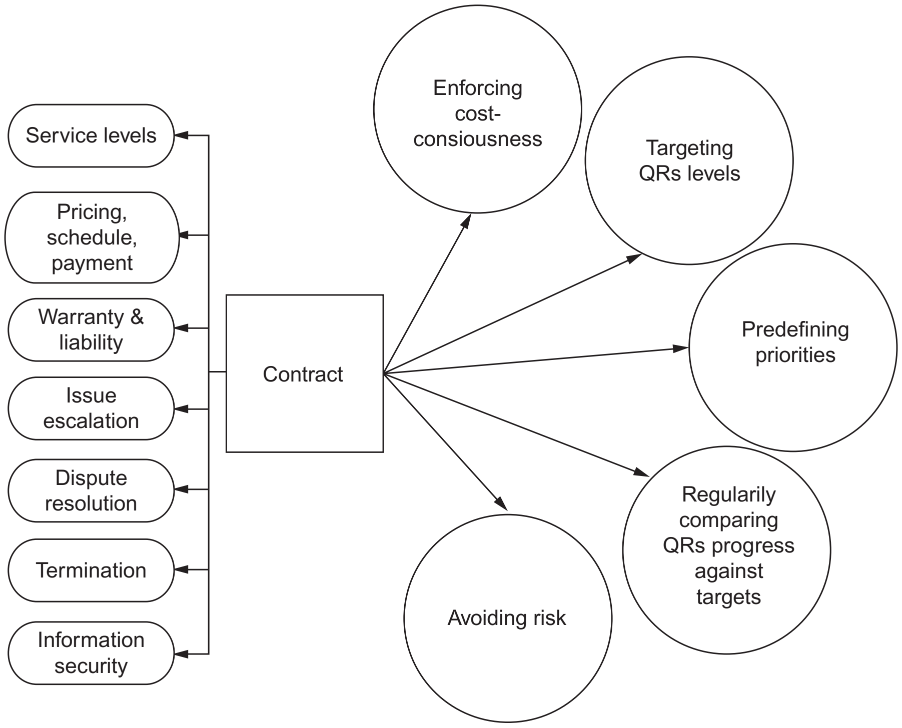

FIGURE 13.4

Contract elements that influenced the behavior of SAs in QRs engineering.

levels, for example, in the SLA part, that were targeted and subjected to discussions with the stakeholders; (3) the contract predefined the priorities for some small but very important set of QRs; (4) the contract instilled the discipline to regularly compare the QR levels in the project against the levels stated in the contract; and (5) the contract encouraged the SAs to practice "risk avoidance" in what they were doing (e.g., by using a checklist on their own discretion when validating QRs).

Eighteen of the 21 SAs agreed that the contract was used on an ongoing basis to stay focused on what counts most in the project. To the SAs, the contact was the vehicle to ensure the system indeed includes "*the right things,*" "*those that they needed in the first place, and that we are billing them for*." Twelve SAs acknowledged that a contract-based context is conductive to understanding QRs as a way to maintain control because every comprehensive contract they had usually comes with SLA specifications, KPIs, and measurement plans that address multiple perspectives (of clients/vendors). For example, the Balancing Multiple Perspectives technique (Buglione and Abran, 2005) is a way to help all involved parties in understanding and validating the right amount of things to do in a contract-based project. Also, in the perceptions of these SAs, the contract helped derive mutual benefits in terms of sharing responsibilities and risks. For example, in the experience of one SAs, the precision in the definitions of those contractual clauses that referred to QRs contributed to clear understanding of risks that, if they remained unnoticed, would later on potentially have brought hidden costs.

Another conception that was shared among the18 SAs was the role of the contract in the escalation of issues related to QRs. Eleven project organizations had in place a Contract Management Committee that was the ultimate resort to resolve disputes on QRs. In the other seven organizations, this role was played by the Vendor Relationship Manager responsible for the contract. The SAs thought of these resources as indispensable in cases of difficult QRs negotiations when the client's business depends on a QR that the current technology may not be able to deliver.

In contrast to these 18 SAs, three participants thought the contract was not that important. They said, it was just "*the beginning of their conversation*" on QRs and not the reference guide to consult on an ongoing basis. They thought it's the people who make the contract work. However, in the experiences of these SAs, it has always been RE staff and project managers who work with the contract, and they usually communicate to everyone else in the event that aspects of the project effort are misaligned with the clauses of the contract.

13.6 DISCUSSION

This section presents a comparison of our findings to those in previously published studies on QRs from SA perspective and a reflection on the implications of our study for practitioners and for researchers.

13.6.1 Comparing and contrasting the results with prior research

Our discussion is organized according to the themes behind our research questions.

13.6.1.1 Role of SAs in engineering of QR

In Ameller et al. (2012), the SAs had indicated a broad diversity of tasks beyond software architecture that they took on (e.g., coding). In contrast to this, our SAs worked full-time as SAs (and not coding or carrying out any downstream development activity). Our case study is in line with Gorton's (2011) statement that "*Most of a software architect's life is spent on designing software systems that meet a set of quality attribute requirements*" (p. 23). Our findings suggest that in contract-based and large/very large projects, the SAs define their role as "a bridge" that connects clients QRs to the architecture design. This difference can be a hint that our SAs came from more regulated environments where terminology, roles, and processes are determined, well communicated, and lived up to. Our findings agree with those of Poort et al. (2012a) on the importance that SAs place on gaining a deep understanding of the QRs and using it to deliver a good quality architecture design.

Regarding how many SAs are enough for engineering QRs, we note that this question has not been yet researched in RE studies. In our interviewees' experiences, it's usually one or two SAs who operate together in a large contract-based project. This is in line with Brooks' most recent reasoning on "the design of design" of large systems (Brooks, 2010) where he explains that a two-person team can be particularly effective where larger teams are inefficient (other than for design reviews where the participation of a large number of reviewers is essential).

13.6.1.2 QRs vocabulary of SAs and RE staff

Our results did not indicate the use of different vocabulary of terms as the issue that preoccupied the SAs, even when communicating with other roles like RE. This contrasts the study of Ameller et al. (2012) where Spanish SAs collectively indicated a broad terminological gap between SAs and RE staff. For instance, a shared glossary of QRs types was missing and was part of the problem. We think the difference between our findings and those in Ameller et al. (2012) may well be due to the fact that our case study projects were happening in regulated organizations where standards defined terminologies that all project team members adopted (including SAs) and assumed ownership over their use.

13.6.1.3 QR elicitation

Our study revealed that all SAs were actively involved in elicitation. This confirms the findings from Poort et al. (2012a) and is in contrast to Ameller et al. (2012). Also, we found checklist-based techniques as the predominant approach in contract-based project context. Additionally, a variety of user-experiment-based techniques are applied for eliciting QRs from end users, like storytelling (Gruen et al., 2002; Snowden, 1999) and pilot users/players that could be considered as ethnography (Bentley et al., 1992). This shows that SAs are well aware of the variety of requirements elicitation techniques that are available and recommended for end-user involvement. This is in contrast with other studies that showed a lack of awareness about RE methods in practice, not only in SMEs (Nikula et al., 2000), but also in larger companies (Klendauer et al., 2012).

13.6.1.4 QRs documentation

We found that standardized forms/templates plus natural language were used most. This is in contrast with Ameller et al. (2012), where the SAs could not agree on one specific systematic way to document QRs, and natural language was the only common practice being used. Why does this difference occur? We assume that it's because of the regulated nature of the contract-based environments and of the use of standards. We think it's realistic to expect that in such contexts, a contract is monitored on an ongoing basis (e.g., all relevant SLAs are well specified and how they would be measured is explicitly defined), and its use forces IT professionals to adopt a sound template-based documentation flow throughout the project (Nicholson and Sahay, 2004). As suggested in Nicholson and Sahay (2004, p. 330), a template is "embedded knowledge" (that "resides in organizing principles, routines and standard operating procedures") and this streamlines the knowledge transfer between clients and vendor's staff (e.g., SAs in our case).

13.6.1.5 QRs prioritization

Two new prioritization criteria crystallized in this study: willingness to pay and affordability. These complement the well-known criteria of cost, benefits, and risk (as stated in Herrmann and Daneva, 2008). We assume that the choice of these criteria could be traceable back to the contract-based project contexts of our study where both vendors and clients have to be clear early on about scope, project duration, and the way they organize their work processes and their impact on each party.

13.6.1.6 QRs quantification

Our results agree with Ameller et al. (2012) on the importance of QRs quantification in practice. However, unlike Capilla et al. (2012), our SAs did not indicate that searching for new or better quantification techniques was their prime concern. Similarly to Poort et al. (2012a), our SAs warned about the pitfalls of premature quantification, meaning that early QRs quantification may be based on too many assumptions about the solution. As in Poort et al. (2012a), our SAs thought that if those assumptions turned out unrealistic, a vendor may find itself in the precarious situation of having committed resources to unachievable QRs. Regarding how quantification happens, the SAs suggest either using a standard (e.g., the ISO usability standard) or engaging an expert in specific type of QRs (e.g., security, scalability). While the first ensures that all tasks of implementing QRs are explicitly accounted for in a project, the second allows for deeper analysis on a single quality attribute and its interplay with others.

An important finding regarding quantification was the SAs' concern about the need to contextualize a QR in the terms of the domain-specific tasks that will be performed by using the system. For example, unambiguously specifying what kind of interactions a QR applies to. This topic so far has been only implicitly touched upon in textbooks in RE (Lauesen, 2002; Pohl, 2011) or in software architecture (Gorton, 2011). For this reason, we think that follow-up studies might be worthwhile to fully understand how to add contextual descriptions to enrich quantified QR definitions.

Another finding relevant to qualification of QRs is the need for a standard definition of the granularity level at which QR statements should be, in order to translate them into a value to be used for size and effort estimation purposes: Because the new SNAP technique (IFPUG, 2013) uses the "elementary process" concept (the smallest unit of activity that is meaningful to the user constitutes a complete transaction, is self-contained, and leaves the business of the application being counted in a consistent state), this concept could possibly be a good candidate to consider as a pointer on which practitioners can ground a sizing unit for properly quantifying QRs.

13.6.1.7 QRs validation

Unlike in QRs literature (Berntsson-Svensson et al., 2010; Capilla et al., 2012) where much accent is placed on tools and methods, we found no hint of formal and tool-based validation. No practitioner among our case study participants has even worked in a contractual agreement where a sophisticated automated (model-checking) tool for validating QRs was explicitly stated in the contract or was required by the client organization. Instead, our results suggest that simple and common-sense practices dominate the state-of-the art in engineering QRs: for example, using requirements walkthroughs, documentation reviews, building up communication processes (e.g., escalation if a QR is not timely clarified) around the artifact-development activity are down-to-earth, yet powerful ways to ensure QRs are aligned with clients' expectations and contractual agreements. One could assume the SAs' behavior to remain active and persuasive regarding QRs validation could be due to the explicit contractual agreements (e.g., SLA), controls and project monitoring procedures (e.g., KPI).

13.6.1.8 QRs negotiation

In contrast to RE literature (e.g., Lauesen, 2002) that offers an abundance of negotiation methods, we found that only one: EasyWinWin (Boehm et al., 2001) was mentioned (and by only one SA). SAs hinted to general purpose negotiation techniques, for example. the HoQ and the Six-Thinking-Hats method as being sufficient to help with QRs negotiation. This suggests that it might be worthwhile to explore the kind of support that negotiation methods from other fields (management science, psychology) can offer to QRs negotiation.

13.6.1.9 Contract's role in SAs' coping strategies

Our results suggest that the contract's role in the ways in which SAs executed their QRs engineering activities was fivefold:

1. It made QRs "*first class citizens*" (as one participant put it) in the RE and AD process, as SLA and KPI automatically made them relevant part of the conversations on requirements in the very early project meetings.
2. It instilled an artifact-related discipline in engineering QRs, for example, checklists were used in elicitation, templates were used in documentation, formal document-based walkthroughs took place to validate the QRs.
3. It reinforced the SAs' role in a project organization and redefined how this role is included in all the RE activities concerning QRs.
4. It reinforced SAs' thinking of progress on QRs in terms of levels achieved and also compared against targets.
5. It encouraged SAs' thinking of QRs in terms of business cases and risk avoidance.

These results reflect that SAs are well aware of the possible impacts of a contract on their professional behavior, for example, SAs are thoughtful about what is expected in a contract and how it would be monitored, they assume responsibility to clarify QRs priorities, and they escalate to project managers and/or RE staff when they see project goals threatened. This is in line of other published results (e.g., Gopal and Koka, 2010) suggesting that the better a project organization aligns the aspects of the project effort with the clauses of the contract, the closer the project to deadline and budget.

13.6.2 Implications for practice

Our findings have the following implications. First, to SAs, it suggests that the conversation on QRs starts with the contract, and specifically with the SLA and the business case. Next, it provides a good reason to reflect on what skills in the already existing SAs' skillset need to be added in terms of training, so that more SAs feel at ease with contract-based systems delivery projects—for example, feel at ease with expressing QR trade-offs in financial terms. While it is well-known in the software architecture community that communication skills are the most important ones for practicing architects (Wieringa et al., 2006), these skills alone, however, might not be enough in a large-scale contract-based project. Financially savvy architects with good communication skills might turn out to be a better answer to the large project staffing needs.

Second, our research has some implications for RE practitioners. This study suggests that SAs are in the position to be RE specialists' best helpers when producing the total requirements specification documentation for a project. If organizations let RE specialists work in isolation and let them be unaware of who the SA on their project is, this may mean missing out on an important opportunity to leverage the SA's talents for the purpose of better requirements processes and artifacts. The message of our study is that RE specialists have higher chances to succeed in delivering specifications that meet contractual agreements if they reach out and actively seek the support of their respective SAs. Moreover, our study indicated that SAs have a profound knowledge of contracts and good awareness of how they influence what is happening in the RE processes related to QRs. While to the best of our knowledge there are no studies on the RE specialists' awareness of contracts, it might possibly be the case that SAs are better informed than RE staff on QRs (because SAs perceive themselves as responsible for meeting the SLAs). If this is the case, then RE specialists could tap on SAs' knowledge and consult them on evaluating contractual consequences related to any kind of requirement, both functional and QRs.

Third, to RE tool vendors, our findings suggest vendors are better off thinking about how RE tools should be better embedded into social processes and broader social interaction context. A part of any tool vendor's business is to provide education. This could possibly be positioned in the broader context of contractual agreements and large projects.

13.6.3 Implications for research

To RE researchers, our study suggests that instead of solely focusing on QRs methods, tools, and techniques, it makes good sense to extend existing research by including analysis of QRs processes as socially constructed ones. How a contract shapes the behavior of RE staff and SAs is an interesting question demanding future research. One could think of borrowing theories from other disciplines (e.g., behavior science and management) to explain why RE staff and SAs engineer QRs the way they do.

The line between RE tasks and architectural design is not as clear and unidirectional as is implied by current textbooks. SAs are involved in QR elicitation, prioritization, and validation. But are requirements engineers also participating in architectural design? If yes, how? The closer these two types of activities are found to collaborate, the more pressing the question of whether it makes sense to separate these activities in terms of roles, methods, and tools, or whether they should be merged into one "design" phase.

An interesting finding in our study is the observation that SAs related QRs to business drivers, business cases, KPI, and technical and business constraints, some of which were stated in the SAs' project contracts. This gives us a hint that in the minds of the SAs, interdependencies exist between QRs and the other types of nonfunctional requirements as defined by Gorton (2011) and Kassab et al. (2009), for example, project-related nonfunctional requirements that specify standards and constraints as system access, communication, user/system interface standards, budget, schedule, and scope of the project. What the nature of these interdependencies is and what tacit assumptions SAs make about them while engineering the QRs are research questions that are relevant and worthwhile investigating in the future.

It is also an interesting observation that SAs say that the elicitation of functional requirements must happen first and the QR elicitation second and that all QR must be related to functionalities. It would be interesting to investigate whether this is generally so, or only for specific systems in specific domains. RE methods should take this practical demand into account. Currently, functional and QRs are elicited using separate methods, probably simply for historical reasons. Maybe practitioners need new or integrated methods that closely link functional and QRs to each other. Exploration into this forms a line for future research.

We make the note, however, that all our case study participants were experienced in their respective domains and were well aware of the domain-specific issues and challenges that accompany the development processes in the respective business environments. We, however, did not think this is always the case. There-fore, investigating explicitly what knowledge architects need for handling QRs could be a question for further research.

13.7 LIMITATIONS OF THE STUDY

There are a number of limitations in our exploratory research. In our evaluation of them, we used the checklist of Runeson and Höst (2009) to help analyze the possible threats to validity of our observations and conclusions. Because we completed exploratory qualitative research, the key question to address when evaluating the validity of its results is (Yin, 2008): *To what extent can the practitioners' experiences in coping with QRs could be considered representative for a broader range of projects, and companies?* Our participants had at least 10 years' experience and were active in large contract-based projects. While the projects were suitable for the study, they are not representative for all the possible ways in which engineering of QRs is performed in large contract environments. Following Seddon and Scheepers (2012), we think that it could be possible to observe similar experiences in projects and companies that have contexts similar to those in our study, for example, where large and contract-based projects engage experienced SAs in teams with mature process-oriented thinking. As Seddon and Scheepers (2012) suggest, "if the forces within an organization that drove observed behaviour are likely to exist in other organizations, it is likely that those other organizations, too, will exhibit similar behaviour" (p. 12).

Moreover, we acknowledged that the application domain may have influenced the ways SAs coped with QRs, for example, the game development domain. Therefore, we think that more research is needed to understand the relationship between application domains and the way in which the QRs processes are carried out.

We also accounted for the fact that our case study participants worked on projects delivering systems falling in the class of enterprise information system, for example, only Participant P9 worked on a sensor system. The results therefore reflect the realm of information systems and may not be observable for contract-based projects delivering distributed real-time and embedded systems (e.g., software embedded in hardware components used to assemble a car). Usually, in such projects, more than one discipline deals with the system architecture, which has implications for QRs engineering. Clarifying the differences that may exist between information systems and embedded systems contexts is therefore an interesting line for research. We are willing to carry out follow-up studies and are actively searching for industry collaborators in the area of embedded systems to explore QRs process in it.

Furthermore, we note that the projects of our SAs had no issues with their contracts, nor any issues with their QRs. We assume, however, that the results might have been different if we had included problematic project cases, be it in terms of contracts or in terms of QRs and RE processes. Complementing our results with findings from such cases requires further research.

We also acknowledge the inherent weaknesses of interview techniques. The first threat to validity is whether the interviewees answered our question truthfully. We took a conscious step to minimize this threat by (i) recruiting volunteers, the assumption being that if a practitioner would not be able or willing to be honest, he or she could decline participation at any stage of the research process and (ii) ensuring that no identity-revealing data would be used in the study. Second is the threat that the researcher influences the interviewee. We countered it by paying special attention to conducting the interview in such a way that the interviewee felt comfortable in answering the question. The researcher who interviewed the participants made sure the interviewee did not avoid questions and felt at ease. This created a safer environment and increased the chances of getting a truthful answer rather than one aimed to please the interviewer. Third, it might be possible that an interviewee did not understand a question. However, we think that in our study, this threat was reduced, because the interviewer (Daneva) used follow-up questions and asked about the same topic in a number of different ways. Fourth, we accounted for the possibility that the researcher might instill his or her bias in the data collection process. We followed Yin's recommendations (2008) in this respect by establishing a chain of findings: (i) We included participants with diverse backgrounds (i.e., industry sector, type of system being delivered), and this allowed the same phenomenon to be evaluated from diverse perspectives (data triangulation; Patton, 1999) and (ii) we had the draft case study report reviewed by practitioners from the company that hosted our case study. These SAs read and reread multiple versions of the case study results.

13.8 CONCLUSIONS

This study makes a strong case for exploring the SA's perspective on engineering the QRs in large and contract-based software delivery projects. We reported on the internal working of QRs engineering activities as experienced and perceived by individual SAs. Although there are ample empirical studies on vendor-client relationships in IT contract-based system delivery (e.g., in the field of Information Systems Research), these studies take the level of the vendor or developer organization as a unit of study. Exploratory research at the level of individual practitioner's perspectives is very little. We are not aware of any previously published study that takes the individual SA's perspective to QRs in contract-based context. This exploratory case study therefore contributes unique empirical evidence on the ways in which SAs cope with QRs and the interaction and communication processes they actively get involved in as part of QRs engineering.

While the involvement of SAs and RE staff certainly increases the costs of a project, our study revealed that in the contract-based settings studied the SAs' involvement was a worthwhile investment. Our research found that:

1. SAs had a strong sense of identity in terms of what roles they took on and how those added value to their teams.
2. SAs had clarity on terminology used in engineering QRs and leveraged the presence of standards, business cases, and agreements for the purpose of QRs engineering activities.
3. SAs treated QRs with the same attention and attitude as the architecture design demand.
4. The relationship between RE staff/clients and SAs is actively managed, whereby the contract means embracing responsibilities over QRs (and not abdicating thereof).

Concerning the QRs engineering activities addressed in our research questions, we found that:

1. In the view of SAs, engineering functional requirements precedes and is a prerequisite for QRs engineering activities.
2. QRs elicitation happens mostly by using checklist-based techniques.
3. QRs specification happens mostly by using template-based documentation styles and natural language.
4. Willingness to pay and affordability seems as important prioritization criteria for QRs as cost and benefits are.
5. The contractual agreements and, specifically, SLAs and KPIs, play a role in deciding on QRs priorities and QRs trade-offs.

6. For quantification of QRs to make an impact, it needs to be contextualized; SAs deem knowledge of context critical for coming up with quantification that is meaningful to the project team. Often quantification is done with and by experts specialized in a specific QR (e.g., performance, scalability, enjoyability).
7. QRs validation and negotiation are considered more organizationally and in terms of social interactions with RE staff and clients than in terms of tool-supported processes.

Acknowledgments

The authors thank the anonymous reviewers for their exceptionally helpful comments and suggestions for improving this chapter. We also thank the case study participants. Without their commitment and engaged participation, this exploration would have been impossible.

References

AccountAbility, 2011. Stakeholder Engagement Standard (AA1000SES). http://www.accountability.org/standards/aa1000ses/index.html.

Ameller, D., Franch, D., 2010. How do software architects consider non-functional requirements: a survey. REFSQ 2010, pp. 276-277.

Ameller, D., Ayala, C., Cabot, J., Franch, X., 2012. How do software architects consider non-functional requirements: an exploratory study, RE 2012, pp. 41-50.

Ameller, D., Galster, M., Avgeriou, P., Franch, X., 2013. The role of quality attributes in service-based systems architecting: a survey. ECSA 2013, pp. 200-207.

Avgeriou, P., Grundy, J., Hall, J.G., Lago, P., Mistrík, I. (Eds.), 2011. Relating software requirements and architectures. Springer, Berlin.

Avgeriou, P., Burge, J., Cleland-Huang, J., Franch, X., Galster, M., Mirakhorli, M., Roshandel, R., 2013. 2nd international workshop on the twin peaks of requirements and architecture (TwinPeaks 2013). In: ICSE 2013, pp. 1556-1557.

Bachmann, F., Bass, L., 2001. Introduction to the attribute driven design method. In: ICSE 2001, pp. 745–746.

Bass, L., et al., 2003. Software Architecture in Practice, second ed. Addison-Wesley, NY.

Bensabat, I., Goldstein, D., Mead, M., 1987. The case research strategy in studies of information systems. MIS Quart. 11 (3), 369–386.

Bentley, R., Hughes, J.A., Randall, D., Rodden, T., Sawyer, P., Shapiro, D., Sommerville, I., 1992. Ethnographically-informed systems design for air traffic control. In: Proceedings of the 1992 ACM Conference on Computer-Supported Cooperative Work CSCW'92, pp. 123–129.

Berenbach, B., Lo, R.-Y., Sherman, B., 2010. Contract-based requirements engineering. In: RELAW 2010, pp. 27–33.

Berntsson-Svensson, R., Höst, M., Regnell, B., 2010. Managing quality requirements: a systematic review. In: EUROMICRO-SEAA 2010, pp. 261–268.

Boehm, B., Grunbacher, P., Briggs, R.O., 2001. Easy WinWin: a groupware-supported methodology for requirement negotiation. In: 9th ACM SIGSOFT FSE, pp. 320–321.

Brooks, F.P., 2010. The Design of Design: Essays from a Computer Scientist. Addison-Wesley Professional, NY.

Buglione, L., 2012. Improving estimated by a four pieces puzzle. In: IFPUG Annual Conference. http://www.slideshare.net/lbu_measure/agile4fsm-improving-estimates-by-a-4pieces-puzzle.

Buglione, L., Abran, A., 2005. Improving Measurement Plans from multiple dimensions: Exercising with Balancing Multiple Dimensions - BMP. In: 1st Workshop on "Methods for Learning Metrics", METRICS 2005.

Capilla, R., Babar, M.A., Pastor, O., 2012. Quality requirements engineering for systems and software architecting: methods, approaches, and tools. Requir. Eng. 17 (4), 255–258.

Charmaz, K., 2008. Constructing grounded theory. Sage, Thousands Oaks.

Chen, L., Babar, M.A., Nuseibeh, B., 2013. Characterizing architecturally significant requirements. IEEE Softw. 2013, 38–45.

Cheng, B.H.C., Atlee, J.M., 2007. Research directions in requirements engineering. In: Briand, L.C., Wolf, A.L. (Eds.), International Conference on Software Engineering/Workshop on the Future of Software Engineering. IEEE CS Press, pp. 285–303.

de Bono, E., 1985. Six thinking hats: an essential approach to business management. Little, Brown & Co, Toronto.

Ferrari, R., Madhavji, N.H., 2008. Architecting-problems rooted in requirements. Inf. Softw. Technol. 50 (1-2), 53–66.

Furlotti, M., 2007. There is more to contracts than incompleteness: a review and assessment of empirical research on inter-firm contract design. J. Manag. Gov. 11 (1), 61–99.

Gilb, T., 2005. Competitive Engineering: A handbook for Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage. Wiley, Butterworth.

Goo, J., Huang, C.D., Hart, P., 2008. A path to successful it outsourcing: interaction between service-level agreements and commitment. Decis. Sci. 39 (3), 469–506.

Gopal, A., Koka, B., 2010. The Role of Contracts in Quality and Returns to Quality in Offshore Software Development. Decis. Sci. 41 (3), 491–516.

Gorton, I., 2011. Essential Software Architecture, second ed. Springer, Berlin.

Groene, B., 2012. TAM—The SAP Way of Combining FCM and UML, http://www.fmc-modeling.org/fmc-and-tam (last viewed on Nov 8, 2012).

Groene, B., Kleis, W., Boeder, J., 2010. Educating architects in industry—the sap architecture curriculum. In: 17th IEEE International Conference on Engineering of Computer Based Systems (ECBS), pp. 201–205.

Gruen, D., Rauch, T., Redpath, S., Ruettinger, S., 2002. The use of stories in user experience design. Int. J. Hum.-Comput. Interact. 14 (3-4), 503–534.

Guessi, M., Nakagawa, E.Y., Oquendo, F., Maldonado, J.C., 2012. Architectural description of embedded systems: a systematic review. In: ACM SIGSOFT ISARCS '12, pp. 31–40.

Hauser, J.R. & Clausing, D. (1988), House of Quality. Harvard Business Review Article, 11 pages. May 1, 1988.

Heesch, van, U., Avgeriou, P., 2011. Mature Architecting - A Survey about the Reasoning Process of Professional Architects. In: 9th WICSA, pp. 260–269.

Herrmann, A., Daneva, M., 2008. Requirements Prioritization Based on Benefit and Cost Prediction: An Agenda for Future Research. RE 2008, pp. 125-134.

IFPUG, 2013. Software Non-functional Assessment Process (SNAP)—Assessment Practice Manual (APM) Release 2.1, April.

ISO, 2006. ISO/IEC 14764:2006 Software Engineering—Software Life Cycle Processes—Maintenance. URL: http://www.iso.org/iso/catalogue_detail.htm?csnumber=39064.

ISO, 2008. ISO 9001:2008 Quality management systems—Requirements.

Kalnins, A., Mayer, K.J., 2004. Relationships and hybrid contracts: an analysis of contract choice in information technology. J. Law Econ. Org. 20 (1), 207–229.

Kassab, M., Ormandjieva, O., Daneva, M., 2009. An ontology based approach to non-functional requirements conceptualization. In: ICSEA 2009, pp. 299–308.

King, N., Horrocks, C., 2010. Interviews in qualitative research. Sage, Thousands Oaks.

Klendauer, R., Berkovich, M., Gelvin, R., Leimeister, J.M., Krcmar, H., 2012. Towards a competency model for requirements analysts. Inf. Syst. J. 6 (22), 475–503.

Lauesen, S., 2002. Software requirements: styles and techniques. Addison-Wesley Professional, NY.

Nekvi, M.R., Madhavji, N., Ferrari, R., Berenbach, B., 2012. Impediments to requirements-compliance. In: REFSQ 2012, pp. 30–36.

Nicholson, B., Sahay, S., 2004. Embedded knowledge and offshore software development. Inf. Organ. 14 (4), 329–365.
Nikula, U., Sajaniemi, J., Kalviainen, H., 2000. Management view on current requirements engineering practices in small and medium enterprises (Proceedings of the Australian Workshop on Requirements Engineering).
Nuseibeh, B.A., 2001. Weaving together requirements and architectures. IEEE Comput. 34 (3), 115–117.
Patton, M.Q., 1999. Enhancing the quality and credibility of qualitative analysis. Health Serv. Res. 34 (5 Pt 2), 1189.
PMI, 2013. A guide to the project management body of knowledge (PMBOK), fifth ed. Project Management Institute. http://goo.gl/UNbFam.
Pohl, K., 2011. Software Requirements: Fundamentals, Principles, and Techniques. Springer, Berlin.
Poort, E.R., Martens, N., van de Weerd, I., van Vliet, H., 2012a. How architects see non-functional requirements: beware of modifiability. REFSQ 2012, pp. 37-51.
Poort, E.R., Key, A., de With, P.H.N., van Vliet, H., 2012b. Issues Dealing with Non-Functional Requirements across the Contractual Divide. WICSA/ECSA. pp. 315-319.
Rockwell, C., 1999. Customer connection creates a winning product: building success with contextual techniques. Interactions 6 (1), 50–57.
Runeson, P., Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. Empir. Softw. Eng. 14 (2), 131–164.
SAP AG, 2000. Quality Management Manual for SAP Development, Waldorf, Germany, 2000. SA, http://www.sap.com/solutions/quality/pdf/50010233s.pdf.
Seddon, P., Scheepers, P., 2012. Towards the improved treatment of generalization of knowledge claims in IS research: drawing general conclusions from samples. Eur. J. Inf. Syst. 21 (1), 6–21.
Sjøberg, D.I.K., Dybå, T., Jørgensen, M., 2007. The future of empirical methods in software engineering research. In: Briand, L.C., Wolf, A.L. (Eds.), International Conference on Software Engineering/Workshop on the Future of Software Engineering, pp. 358–378.
Snowden, D., 1999. Story telling: an old skill in a new context. Bus. Inf. Rev. 16 (1), 30–37.
Sommerville, I., 2005. Integrated requirements engineering. IEEE Softw. 22 (1), 16–23.
Song, X., Hwonk, B., 2012. Categorizing requirements for a contract-based system integration project, RE 2012, pp. 279-284.
Sweetser, P., Wyeth, P., 2005. GameFlow: a model for evaluating player enjoyment in games. ACM Comput. Entertain. 3 (3), 1–24.
Sweetser, P., Johnson, D.M., Wyeth, P., 2012. Revisiting the GameFlow model with detailed heuristics. J. Creat. Technol. 3. http://colab.aut.ac.nz/journal/revisiting-the-gameflow-model-with-detailed-heuristics/
Wieringa, R., van Eck, P., Steghuis, C., Proper, E., 2006. Competencies of IT Architects. NAF, Amsterdam, The Netherlands. http://doc.utwente.nl/68444/3/CompetencesOfITArchitects2009.pd.
Wortham, S., Mortimer, K., Lee, K., Allard, E., White, K., 2011. Interviews as interactional data. Lang. Soc. 40, 39–50.
Wu, D.J., Ding, M., Hitt, L., 2012. IT implementation contract design: analytical and experimental investigation of IT payoff, learning and contract structure. Inf. Syst. Res. 24 (3), 787–801.
Yannakakis, G.N., Lun, H.H., Hallam, J., 2006. Modeling children's entertainment in the playwre playground. In: Proceedings of the IEEE Symposium on Computational Intelligence and Games, Reno, USA, May 2006, pp. 134–141.
Yin, R.K., 2008. Case study research: design and methods. Sage, Thousand Oaks.

All in-text references underlined in blue are linked to publications on ResearchGate, letting you access and read them immediately.