

Coping with Quality Requirements in Large, Contract-Based Projects

Maya Daneva, University of Twente

Andrea Herrmann, Herrmann & Ehrlich

Luigi Buglione, Engineering Ingegneria Informatica

// Contracts for delivering large software systems must address issues such as system quality, timelines, delivery cost and effort, and service-level agreements. A study with 20 software architects revealed how they coped with quality requirements in this context. //



IN TODAY'S market, client organizations usually contract IT service vendors to deliver large software systems. In a contract to deliver a large software system, the client and vendor agree to undertake, or refrain

from undertaking, certain actions in the course of delivering the system. The contract regulates the client-vendor relationship by defining each party's rights, liability, and expectations. It addresses a host of related is-

ssues such as system quality, timelines, delivery cost and effort, penalties for mistakes or missed deadlines, and service-level agreements (SLAs).¹⁻⁴

How these issues are settled in a contract often creates various incentives for the participating parties.⁴ Compared to in-house development, in contract-based projects the pressure to align incentives across parties is stronger, and the need to align the parties' understandings of system quality attributes is much greater. So, monitoring the project artifacts' quality is more complex.⁴ Starting a conversation on quality requirements (QRs) as early as possible is therefore crucial. Software architects act as mediators between business analysts and clients on one side and developers on the other, and can constructively influence QRs engineering.

With this in mind, we studied architects' day-to-day coping strategies concerning QRs in large-scale, contract-based systems delivery projects. Here, we present some insights and surprising lessons that challenge the notion of software architects as "technical heroes" and redefine our understanding of QRs engineering from the architects' perspective (for more on this, see the sidebar). Most of our lessons signal that for software architects, moving from in-house to contract-based projects means embracing a different mentality, which might require some preparation to smooth the transition.

The Study

To determine a possible range of views on how architects cope with QRs, we ran a qualitative study in which we interviewed 20 software architects. These participants were from 14 companies in the Netherlands, Belgium, Finland, and Germany.

RELATED WORK IN ARCHITECTS' PERCEPTIONS OF QUALITY REQUIREMENTS

Requirements engineering (RE) considers quality requirements (QRs) and the software architect's perspective important. However, only recently has RE research yielded the first five empirical studies of software architects' perceptions of QRs.¹⁻⁵ The consensus of these studies was that software architects' and RE specialists' perspectives on QRs differ. The software architects' experiences came mostly from small and mid-sized projects. Our research (see the main article) complements these studies with findings from large, contract-based projects.

As in those five studies, we found that software architects feel it's important to gain a deep understanding of the QRs and use this to deliver good architecture design. Our study revealed that all the software architects were actively involved in QRs refinement. This agrees with Eltjo Poort and his colleagues' findings^{4,5} but contradicts those of David Ameller and his colleagues.³

However, regarding most aspects studied, we found significant differences due to

- the different architects' profiles,
- the project organizations' sizes,
- possible incentives in play, and
- the fact that large, contract-based projects are managed differently and take place in more regulated and standardized contexts than smaller projects.

In particular, we found the following differences from previously published results.

In Ameller and his colleagues' study, the software architects took on diverse technical roles and tasks (for example, coding). In contrast, our architects defined their role as a bridge connecting clients' QRs to the architecture design, with social interaction being key. Also, Ameller and his colleagues indicated a broad terminological gap between software architects and RE staff. We found no terminology-related issues because our case study's projects took place in

regulated organizations in which standards defined terminologies explicitly. We also found that software architects mostly used standardized forms or templates plus natural language. This differed from Ameller and his colleagues' study, in which the software architects couldn't agree on a systematic way to document QRs.

Our results match Poort and his colleagues' findings regarding the close attention given to QRs quantification. However, our architects didn't indicate that they were preoccupied with searching for new or better quantification techniques. As in a previous study,⁵ our architects emphasized the pitfalls of premature quantification. Such quantification might be based on too many assumptions about the solution, so a vendor might find itself in a precarious situation if those assumptions are unrealistic. Regarding quantification, our architects suggested either using a standard or engaging an expert in a specific type of QRs (for example, scalability).

We found no study investigating how contracts shape the way in which RE professionals or software architects handle QRs. So, we investigated this to some degree in our study, and we consider it an interesting line of future research.

References

1. R. Capilla et al., "Quality Requirements Engineering for Systems and Software Architecting: Methods, Approaches, and Tools," *Requirements Eng. J.*, vol. 17, no. 4, 2012, pp. 255–258.
2. U. van Heesch and P. Avgeriou, "Mature Architecting—a Survey about the Reasoning Process of Professional Architects," *Proc. 9th Working IEEE/IFIP Conf. Software Architecture*, 2011, pp. 260–269.
3. D. Ameller et al., "How Do Software Architects Consider Non-functional Requirements: An Exploratory Study," *Proc. 20th IEEE Requirements Eng. Conf.*, 2012, pp. 41–50.
4. E.R. Poort et al., "How Architects See Non-functional Requirements: Beware of Modifiability," *Requirements Engineering: Foundation for Software Quality*, LNCS 7195, Springer, 2012, pp. 37–51.
5. E.R. Poort et al., "Issues Dealing with Non-functional Requirements across the Contractual Divide," *Proc. 2012 Joint Working IEEE/IFIP Conf. Software Architecture and European Conf. Software Architecture*, 2012, pp. 315–319.

Each participant served in projects in which parties established contracts in two steps.³ First, the parties agreed on a contract to get the requirements documented in sufficient detail such that an architect could use them to work on the architecture design. Then, a second contract

addressed the system delivery itself. The pricing agreements varied across the companies: fixed-price, variable, or a combination of the two.

Also, each architect worked in large, contract-based projects running in at least three development locations in one country, and had

clients in more than two countries. Finally, each participant had at least 10 years' experience in large systems.

Table 1 lists the projects' details. The contracts' total number of pages (see the rightmost column) included high-level business requirements, a statement of work, key performance

TABLE 1

Study participant details.*

Participant ID	Years of experience	Business	System description	No. of people on team	Project duration (mo.)	Pricing arrangement	Total no. of pages
P1	13	Large IT vendor	ERP package implementation (Oracle)	35	18	FP + V	350
P2	10.5	Large IT vendor	ERP package implementation (SAP)	60	15	FP + V	300
P3	15	Large IT vendor	ERP package implementation (SAP)	75	18	V	250
P4	18	Large IT vendor	ERP package implementation (SAP)	41	12	V	300
P5	10	Large IT vendor	ERP package implementation (SAP)	51	12	V	350
P6	12	Large IT vendor	ERP package implementation (Oracle)	45	12	V	350
P7	13	IT vendor	ERP package implementation (SAP)	40	18	V	330
P8	11	Software producer	Online learning environment	22	12	FP	180
P9	12	Software producer	Sensor system for in-building navigation	35	12	FP	220
P10	14	Software producer	Online ticket-booking application	15	12	FP + V	300
P11	13	Oil and gas	Logistics-planning application	21	12	FP + V	350
P12	10	Insurance	Web application for client self-service	61	24	FP	350
P13	11	Insurance	Client claim management and reimbursement application	53	16	FP	350
P14	12	Real estate	Web application for rental-contract handling	42	18	FP	300
P15	11	Air carrier	Web application for processing passenger feedback	11	14	V	350
P16	13	Video streaming	Viewer recommendation management system	18	18	FP + V	200
P17	10	Video streaming	Viewer complaint management system	45	9	FP + V	250
P18	16	Online bookstore	Order-processing system	15	10	FP + V	300
P19	14	Online-game producer	Gaming system	81	21	FP + V	220
P20	13	Online travel agency	Room deal identification system	45	12	V	300

* ERP = enterprise resource planning; F = fixed-price and V = variable (time and material contract).

indicators (KPIs), SLAs, agreements regarding compliance with applicable laws, and the post-services schedule.

We asked each participant to think of his or her most recent project and provide insights into three areas concerning software architects' coping strategies for QRs:

- interaction with stakeholders (for example, how they negotiated QRs with stakeholders),
- engineering (how they documented QRs), and
- contract compliance (what role the agreements played in how they handled QRs).

The technical aspects of our research process appear elsewhere.⁵

Here, we distill takeaway messages for software professionals transitioning to contract-based system delivery models or searching for practices to adopt or to adapt for their own contract-based QRs engineering. Where applicable, we illustrate the messages with quotes from the participants.

Assume a Mediator's Role

Thirteen participants thought of their role as a bridge between clients' QRs and the underlying technology. As participant P10 put it,

You are there to connect people, to bridge gaps between business "patrons" and developers, to make sure technology catches up with their business demands and that they are aware of the possible choices.

The other seven participants regarded their role as a review gatekeeper because they served most of their time reviewing QRs, giving feedback, and evaluating contract compliance.

Whom did architects speak with most of the time concerning QRs? Seventeen participants went to business analysts (requirements-engineering staff) for clarification on QRs. Two considered the vendor relationship manager responsible for their projects. That was because the vendor relationship manager had the authority to make decisions and negotiate with external parties (vendors) about expected quality levels, and ensured that project execution occurred as per the contract. One participant identified her project manager as the most frequently contacted person.

No participant dealt directly with the user community on an ongoing basis. However, the key business users knew the participants personally because the participants served as mediators in translating technology choices in business terms to non-technical project stakeholders.

Perhaps our most surprising finding is that, although the participants interacted with a diverse audience, they didn't struggle with communication breakdowns due to various QRs interpretations. They attributed this observation to the domain knowledge they had accumulated in their many years of experience in their respective business sectors. For example, when working with less-experienced business analysts, they found their domain knowledge instrumental to spot missing or incomplete requirements. P15, who worked on an online system for processing feedback from an air carrier's clients, said that in this application, scalability was usually regarded as the highest-priority QR. If a specification mentioned nothing about it, he would red-flag that specification and follow up with business analysts and collect details on it.⁵

Use Standards to Ease Communication

The participants who worked in ISO-certified organizations suggested that, next to domain knowledge, knowledge of the adopted ISO standards and mandatory ISO training for everyone in their organization helped them and their business analysts understand each other. As P11 said,

We all are in the habit of looking back to what the ISO-compliant Quality Manual says.

We found that the companies invoked two streams of ISO standards to keep a common interpretation of QRs terminology:

- management systems (for example, ISO standard 9001-27001-20000-1) that are about requirements to be met and
- technical standards (such as 14143-x, 9126-x, and so on) that describe processes and how to do things.

A common concern of the participants, though, was that both streams used QRs terms interchangeably, and it wasn't clear which QR followed the terminology of which stream. So, the participants often coped by making the tacit knowledge explicit by establishing cross-references between the terms traceable to the various standards.

Discover QRs through Refinement

The study participants felt that QRs elicitation included refining any QRs deemed important for the project and mentioned explicitly in the contract. They reported that they took the lead in this process. Fourteen participants working in regulated business domains (such as insurance) championed the

use of checklists to refine QRs. As a basis for the checklists, eight used ISO standards in the SQuaRE (System and Software Quality Requirements and Evaluation) series—for example, 25045:2010, 25010:2011, 25041:2012, and 25060:2010. Four participants used architecture frameworks that were specific to their company, the business sector, or the client organization. Two used stakeholder engagement standards such as AA1000SES. Ten of them created the checklists themselves.

Six participants employed more creative techniques (for example, in video streaming and game design). For example, some used storytelling techniques to gather knowledge in the form of stories. Others employed serious-game-based techniques (used in logistics information systems projects), in which users played a scheduling game by using an early prototype to detect the system's performance and availability requirements.

Use Predefined QRs Templates

Most participants preferred to use predefined templates for QRs documentation. Fifteen participants used templates based on

- the ISO standard,
- vendor-specific standards (for example, in SAP and Oracle projects), or
- Quality Function Deployment (QFD), which translates user-recognizable quality attributes into features for implementation in the system.⁶

The other five participants used plain text to define QRs, plus information on the end user to perform acceptance tests and demonstrate that the system met the QRs.

We assume these results are due to the regulated nature of contract-based environments and the use of standards. So, the continual monitoring of contracts (with well-specified SLAs and explicitly defined measures and metrics) forces IT professionals to adopt a sound template-based documentation flow throughout the project.²

Regardless of the document format, the participants agreed on the importance of ensuring that QRs get reviewed and updated after each milestone in the contract. As P11 stated,

You must update [the QRs] and ensure your commitments to quality levels are still realistic. Otherwise, you open a door for big issues with your clients.

A Business Case Helps to Prioritize QRs

All the participants felt that their project's business case was the key driver in prioritizing QRs. This finding is in line with Frank Buschmann and his colleagues' business value perspective on system quality.⁷ The participants felt responsible for making other stakeholders aware of QRs' tradeoffs. So, they perceived QRs prioritization as an iterative learning experience in which the architects first learn about the QRs needed for the business, along with how urgently the company needs them. Then, the business analysts and stakeholders learn about the technical and cost limitations.

We found four prioritization criteria for making QRs tradeoffs, on the basis of the business case:

- *Cost and benefits* might be estimated quantitatively (for example, the person-months spent to implement specific QRs).

- *Perceived risk* is subsumed in the cost category; that is, risks to QRs are translated into costs.
- *Affordability* determines whether a QR's estimated cost is in accord with the resources specified in the contract and with the client's long-term contract spending.
- *Willingness to pay* is the client's readiness to pay extra for a perceived benefit of implementing or increasing a specific QR.

Although cost, benefits, and risk are well known,³ no studies we know of have covered affordability and willingness to pay. We assume that the choice of these criteria is due to the contract-based nature of the participants' projects. In those projects, the parties had to gain a clear understanding as early as possible of the scope, the project duration, how they organized their work processes, and the penalties for deviation.

Who ultimately decides on QRs priorities? Thirteen participants named the project's steering committee because it linked prioritization to a business driver or a project's KPIs. Seven participants considered themselves the key decision makers on QRs priorities. For example, P13 said,

I must be resolute and very clear about what we can call "must-haves."

P14 said,

I make the decision on the premise that I have a good justification for each choice.

Nineteen participants suggested no explicit use of any method for QRs prioritization other than classifying QRs as essential, marginal, or optional. For three of the 19 (P8,

P19, and P20), naming the top two or three most important QRs wasn't an issue because those QRs were obvious to the client. For example, in the computer game sector, it's a given that the most important QRs are scalability and the user experience.

However, the participants deemed difficult the prioritization of those QRs that were less prominent (from the user's perspective), yet important from the vendor's perspective. For example, regarding maintainability, P13 said,

These requirements matter to you, not to the client. Even [if] you do a brilliant job on maintainability, nobody will pat you on the back and say thank you for this. It's very difficult, I'd say, almost political, to prioritize this kind of QR.

If Quantifying QRs, Start with the Contract

Contract-based projects seem to prompt project organizations to express their QRs quantitatively. All the participants agreed on this, but there was no common approach to quantifying QRs.

Our data suggests that architects either use a size-estimation standard (such as the International Function Point Users Group's nonfunctional-assessment method⁸) or engage an expert in quantifying a specific type of QR (security or scalability, for example). The first approach ensures that a project explicitly accounts for all QRs implementation tasks; the second allows for deeper analysis of a single quality attribute and its interplay with others.⁹ The participants deemed both approaches important to prevent early and poorly conceived commitments. They indicated that if a contract explicitly mentioned a size estimation

standard or an independent expert's evaluation, then it was considered legally binding and they had to apply it.

A common starting point was the contract's prespecified quantitative definitions. For example, a contract might state explicitly that the system should scale up to serve hundreds of thousands of subscribers. However, eight participants experienced that contracts often confused QRs with design-level requirements.³ For instance, instead of QRs, one contract contained detailed feature specifications of how to achieve QRs. This example involved specifying a proprietary influence metric to include in the ranking algorithms of a recommender system that's part of an online video-streaming system. In another case, the contract specified a particular algorithm rather than a required quality attribute value and criteria for verifying compliance. This example involved coding a specific search algorithm for finding hotel deals. The participants considered this issue critical because it signalled a misaligned understanding of what was really quantified and what measures were used.

Use Walk-Throughs to Validate QRs

For our participants, validation ensures that QRs are aligned with the client's expectations and the SLAs. It also confirms that the QRs are technically implementable and that the resulting architecture design satisfies the contractually defined business requirements. Each participant was heavily involved in this process. Sixteen participants considered validation their own responsibility; four deemed it the business analysts' job. We assume that the participants' active and persuasive behavior regarding QRs validation could have been

due to the explicit contractual agreements (for example, SLAs), controls, and project-monitoring procedures (KPIs).

No participant witnessed a contract requiring the use of an automated (model checking) tool to validate QRs. Instead, common-sense, down-to-earth practices prevailed. The participants felt that requirements walkthroughs, documentation reviews, and building up communication processes around artifact development (for example, escalating if a QR isn't clarified in a timely fashion) were simple, yet powerful ways to validate QRs.

Fourteen participants regularly performed requirements walkthroughs with clients and business analysts, in which clients confirmed the functionalities to which the QRs applied. The participants deemed such walkthroughs as part of the client expectation management process that the project manager established.

Other participants used internal architecture standards and QFD to demonstrate the related strength between a QR definition and its operationalization, in terms of features or architecture design choices. If deviations crystalized, the participants took the problem to project managers or a steering committee, depending on how large the misalignments were.

QRs Conflict Resolution Should Be Objective

In the participants' experience, contract-based system delivery included explicit QRs conflict-resolution procedures and using information that was as objective as possible. For most participants, the business case was the most important vehicle for supporting their negotiation positions when resolving conflicting QRs. As P20 noted,

You need to express yourself in money terms that they [the clients] can understand very well.

Ten participants routinely used the business case to justify tradeoffs; three others used effort and budget allocation figures derived from the business case. Other participants used QFD, EasyWinWin (a collaborative process to achieve a win-win situation among parties), or the Six Thinking Hats method¹⁰ to reason about QRs in negotiation meetings. Six Thinking Hats is a general approach to resolving complex issues that companies can use for any negotiation situation.

Your Contract Is a Resource

We were surprised that the participants actively used the contract to achieve different, yet complementary, goals. Every participant considered it an important source of learning about QRs. As P5 explained,

You learn not only about the quality levels in the resulting system but also about the culture of the client. The greater the knowledge of our clients in systems development, the more detailed the terms they use to specify the contract.

Seventeen participants used the contract continually to stay focused on what counted most in the project. They mentioned that the contract helped ensure that the system included “the right things” (P11) and “those that they needed in the first place, and that we are billing them for” (P13).

Twelve participants perceived the contract as a vehicle to maintain control. They believed this was because every comprehensive contract usually came with SLAs, KPIs, and measurement plans that addressed

multiple perspectives (clients or vendors). For example, the Balancing Multiple Perspectives technique⁹ can help the involved parties understand and confirm all the things to do in a contract-based project.

How Contracts Affect Architects

Our data revealed three ways a contract shapes the architects’ coping strategies regarding QRs:

- The contract aroused the participants’ cost consciousness, which led them to habitually estimate QRs’ cost.
- The contract stipulated QRs levels (for example, in the SLA) that the participants discussed with the stakeholders.
- The contract predefined the priorities for a small but important set of QRs.

However, three participants disagreed with that third item in the list. They considered the contract just the beginning of the conversation on QRs, not a reference guide to consult routinely. (They felt it was the people who made the contract work.)

Did any contract-caused incentives motivate the participants to approach QRs the way they did? We found that specific elaborations on price, rewards, and penalties made project organizations default to some choices while setting up their delivery process.

For example, fixed-price contracts created an incentive for vendors to staff projects with the most qualified resources. So, they perceived that engaging experienced architects to engineer QRs end-to-end in a project was conducive to executing the QRs activities in a disciplined and systematic fashion. Vendors’ managers also found it discouraging to

redirect staff to different capacities mid-project.

In addition, fixed-price contracts made the participants conscious about the possibility of facing disputes due to poorly done QRs. This was because the original project scope often didn’t include deliverables that were only implicitly assumed in the project. To counter this, the participants at all times maintained complete awareness of what QRs were and weren’t in scope. As P8 said,


Against this backdrop, I don’t think you can afford ... to ignore QRs.

Applying These Strategies

Because our study was exploratory and aimed to get a snapshot of possible strategies for coping with QRs, we can’t claim that our findings are generalizable. Nevertheless, we think that such strategies might be useful in situations in which managers in large projects hire experienced architects, teams use process-oriented thinking and are aware of what’s in the contract (and how this shapes their process), and standards define the terminology to use for QRs factors.

Large, contract-based system delivery projects favor joint requirements engineering and architecture design. So, they often ensure that architects are integral to QRs processes. Our study participants’ coping strategies reveal four clear payoffs to this strategy.

First, engaging the architects is instrumental in dealing with QRs with the same due diligence that the functional requirements and architecture design demand. Second, mitigating the risks of contract disputes becomes the architect’s responsibility.

ity, who takes leadership in ensuring that QRs tradeoffs align with the contract. Third, leveraging the architect's domain knowledge helps proactively clarify QRs. Finally, socially positioning the architect as a bridge and gatekeeper is conducive to the ongoing conversation on QRs. 

References

1. B. Berenbach et al., "Contract-Based Requirements Engineering," *Proc. 3rd Int'l Workshop Requirements Eng. and Law*, 2010, pp. 27–33.
2. X. Song et al., "Categorizing Requirements for a Contract-Based System Integration Project," *Proc. 2012 Requirements Eng. Conf.*, 2012, pp. 279–284.
3. S. Lauesen, *Software Requirements: Styles and Techniques*, Addison-Wesley, 2002.
4. D.J. Wu et al., "IT Implementation Contract Design: Analytical and Experimental Investigation of IT Value, Learning, and Contract Structure," *Information Systems Research*, vol. 24, no. 3, 2012, pp. 787–801.
5. M. Daneva et al., "Software Architects' Experiences of Quality Requirements: What We Know and What We Do Not Know?" *Requirements Engineering: Foundation for Software Quality*, LNCS 7830, Springer, 2013, pp. 1–17.
6. Y. Akao, *QFD: Quality Function Deployment—Integrating Customer Requirements into Product Design*, Productivity Press, 2004.
7. F. Buschmann et al., "Architecture Quality Revisited," *IEEE Software*, vol. 29, no. 4, 2012, pp. 22–24.
8. *Software Non-functional Assessment Process (SNAP) Assessment Practice Manual*, release 2.2, Int'l Function Point Users Group, June 2014.
9. L. Buglione and A. Abran, "Improving Measurement Plans from Multiple Dimensions: Exercising with Balancing Multiple Dimensions—BMP," *Proc. 1st Workshop Methods for Learning Metrics*, 2005, pp. 205–214.
10. E. de Bono, *Six Thinking Hats*, Little, Brown, & Co., 1985.

ABOUT THE AUTHORS



MAYA DANEVA is a senior scientific staff member of the University of Twente's Services, Cybersecurity and Safety Research Group. She's a specialist in requirements engineering, effort estimation of large systems, and empirical software engineering. Previously, she spent nine years as a business process analyst in the Architecture Group at Telus. Daneva received a PhD in computer science and software engineering from St. Clement Ochrisky University. Contact her at m.daneva@utwente.nl.



ANDREA HERRMANN is a software engineering trainer and researcher at Herrmann & Ehrlich. Her research interests include requirements engineering, project management, and software architecture. Herrmann received a habilitation in software engineering from the University of Heidelberg. Contact her at herrmann@herrmann-ehrllich.de.



LUIGI BUGLIONE is a process improvement and measurement specialist at Engineering Ingegneria Informatica and an associate professor of software measurement at École de Technologie Supérieure. His research interests are software measurement, process improvement, and service management. Buglione received a PhD in management information systems from LUISS Guido Carli University. Contact him at luigi.buglione@eng.it.

IEEE Intelligent Systems

THE #1 ARTIFICIAL INTELLIGENCE MAGAZINE!

IEEE Intelligent Systems delivers the latest peer-reviewed research on all aspects of artificial intelligence, focusing on practical, fielded applications. Contributors include leading experts in

- Intelligent Agents • The Semantic Web
- Natural Language Processing
- Robotics • Machine Learning

Visit us on the Web at
www.computer.org/intelligent



Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.