The Role of Quality Attributes in Service-based Systems Architecting: A Survey

David Ameller[1], Matthias Galster[2], Paris Avgeriou[3], Xavier Franch[1]

[1]Universitat Politècnica de Catalunya (UPC), Barcelona, Spain
{dameller, franch}@essi.upc.edu
[2]University of Canterbury, Christchurch, New Zealand
mgalster@ieee.org
[3]University of Groningen, The Netherlands
paris@cs.rug.nl

Abstract. Quality attributes (QA) play a fundamental role when architecting software systems. However, in contrast to QA in traditional software systems, the role of QA when architecting service-based systems (SBS) has not yet been studied in depth. Thus, we conducted a descriptive survey to explore how QA are treated during the architecting of SBS. Data were collected using an online questionnaire targeted at participants with architecting experience. Our survey shows that QA and functional requirements of SBS are mostly considered equally important. Also, QA are usually treated explicitly rather than implicitly. Furthermore, dependability and performance appear to be the most important QA in the context of SBS. Our results partially show that general findings on QA also apply to the domain of SBS. On the other hand, we did not find a confirmation that QA are primary drivers for the architecting of SBS, or that certain application domains would focus on particular QA.

Keywords: quality attributes; service-based systems; survey; architecting

1 Introduction

Quality attributes (QA) are characteristics that affect the quality of software systems [1]. Quality attribute requirements are requirements that refer to these quality attributes. For example, demanding a response time of 1 millisecond for a particular function of a system is a quality attribute requirement referring to performance (the QA). QA tend to be more difficult to achieve because they are often not explicitly described by stakeholders, exhibit trade-offs, or are subjective. It has been acknowledged that QA affect the architecting of software systems and thus should be considered from the very start of a software project [2, 3]. One type of software system that has become popular in industry is that of service-based systems (SBS) [4]. In contrast to conventional software systems, the role of QA in the context of SBS has not yet been studied extensively. However, quality is a top challenge in SBS engineering [2, 5]. Even though proposals for quality attributes in SBS exist, there is a lack of empirical studies that investigate QA in practice [6]. To contribute to the understanding of the role of QA in SBS by providing insights into how QA are treated in practice, we present a descriptive

survey. Using the GQM approach [7], the goal of our survey is defined as to *analyse and characterise* (purpose) the *role of QA* (issue) in *SBS architecting* (object) from the perspective of *practitioners and researchers with practical architecting experience* (viewpoint). Section 2 summarizes our research method. Section 3 presents the results of our study. In Section 4 discuss our results. We conclude the paper in Section 5.

2 Research Method

Surveys collect qualitative and quantitative information to provide a "snapshot" of the current status related to a phenomenon [8]. To ensure rigor, repeatability and to reduce researcher bias, we designed a survey protocol following the template proposed for evidence-based software engineering[1]. We defined three research questions:

- RQ1: *How important are QA compared to functionality when architecting SBS*?
- RQ2: *To what extent are QA specific to application domains of SBS*?
- RQ3: *What kind of architectural decisions are used to address QA in SBS*?

Current literature, such as [9, 10, 11], suggests that QA drive the design of software architectures. RQ1 investigates if this is also the case for SBS, or if QA are treated as factors that suggest the use of a service-based solution in the first place but are not considered architecture drivers (service-orientation claims to achieve "qualities", such as interoperability, flexibility, reusability [12]). For RQ2, we aim at providing guidance for software architects by focusing on the QA that are most important for a certain application domain (e.g., healthcare, telecommunication). Finally, RQ3 investigates the transition from QA to architectural decisions by relating QA to the architecture decision types and categories proposed by Kruchten [13].

Survey design. We conducted a descriptive survey. We required participants to have practical experience in architecting SBS. This included practitioners from industry, researchers, and participants with mixed background (e.g., participants that moved to academy after working in industry, practitioners with part-time academic positions). Participants were recruited through several cycles of advertising (e.g., LinkedIn groups, conferences, and online communities), between May and September 2011.

Data preparation and collection. All survey questions[2] on the online questionnaire referred to one particular project that participants had worked on in the past. For most questions, participants could provide comments to complement their answer.

Data analysis. To ensure the quality of the data obtained from the questionnaire, we applied sanity checks to find obvious errors in data. We used descriptive statistics to analyze the data [14]. Free text answers were coded [15] and underwent content analysis [16] involving all four authors.

Internal validity. Confounding variables could bias our results [17]. Thus, we applied exclusion and randomization [9]. Exclusion means that participants who are not

[1] http://www.dur.ac.uk/ebse/resources/templates/SurveyTemplate.pdf.

[2] All questions can be found at www.essi.upc.edu/~dameller/publications/ecsa13-ap.pdf.

sufficiently experienced were excluded from the study. Randomization means that we used a sampling technique which led to random participants. Furthermore, to mitigate the risk of ambiguous and poorly phrased questions, we piloted the questionnaire in multiple iterations. Another limitation is that participants might not have answered truthfully to the questions [9]. Thus, we made participation voluntary and anonymous. Finally, the protocol was reviewed by external reviewers.

External validity. External validity is concerned with generalizing the results to the target population. We assume that our results are applicable to a population that meets the sampling criteria of our survey (i.e., architects with experience in SBS).

3 Results

We obtained 31 valid responses. From these, 18 participants (58%) had experience in both academia and industry. 10 participants (32%) had only experience in industry, whilst 3 (10%) were participants from academia.

RQ1: How important are QA compared to functionality when architecting SBS?

Importance and explicitness of QA. Functionality and QA were considered equally important by most respondents (Figure 1). When asked whether QA were considered implicitly or explicitly, most respondents (71%) answered "explicitly" (see Figure 2). To identify dependencies between the importance of QA and their implicit or explicit nature, we created a cross-tabulation (Table 1). Eighteen respondents (58%) considered QA and functionality equally important and at the same time made QA explicit. In 6 cases (20%), QA were not made explicit and QA were considered less important than functionality. In all 4 answers (12%) where QA were more important than functionality, QA were made explicit. Fisher's exact test led to $p < 0.001$ which means that there is a statistically significant relationship between the importance of QA and their implicit or explicit nature. Thus, there is a high probability that projects which treat functionality and QA equally important also make QA explicit.

Table 1. Cross-tabulation of the importance of QA and their implicit or explicit nature

	QA explicit	**QA implicit**	**Total**
QA were AS important AS functionality	18 (58%)	3 (10%)	21 (68%)
QA were LESS important THAN functionality	0 (0%)	6 (20%)	6 (19%)
QA were MORE important THAN functionality	4 (12%)	0 (0%)	4 (13%)
Total	22 (71%)	9 (29%)	31 (100%)

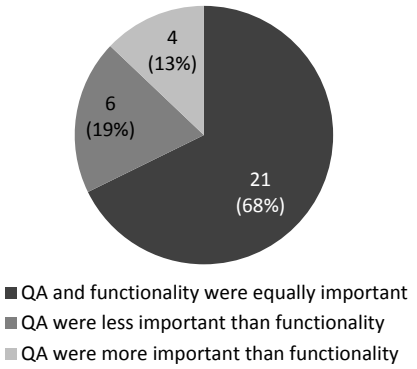

Fig. 1. QA compared to functionality

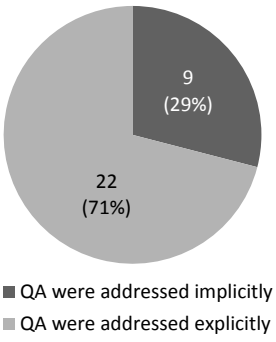

Fig. 2. Implicit / explicit nature of QA

Impact of role on how QA are perceived. Even though all participants had architecting responsibilities in the project for which they answered the questions, they had different roles. Architects and designers were the majority (17 participants or 55% of all participants). Additional roles included 3 project managers (10%), 2 developers (7%), and 1 participant of each of the following roles: consultant, quality engineer, analyst, industrial researcher, unit manager and standards developer. Cross-tabulations are shown in Table 2 and Table 3. Three participants did not provide any role. Thus, the total number in Table 2 and Table 3 is 28. Fisher's exact test indicated a dependency between the role of participants and the importance of QA (p = 0.078). Given the number of architects that considered QA as equally important compared to functionality, this dependency means that architects and designers tend to treat QA and functionality equally important. Furthermore, 71% of architects and designers treated QA explicitly (not statistically significant; Fisher's exact test led to p = 0.151).

Table 2. Cross-tabulation of the importance of QA and the role of participants

	Architect	**Other**	**Total**
QA were AS important AS functionality	14 (82%)	6 (55%)	20 (71%)
QA were LESS important THAN functionality	2 (12%)	3 (27%)	5 (18%)
QA were MORE important THAN functionality	1 (6%)	2 (18%)	3 (11%)
Total	17 (100%)	11 (100%)	28 (100%)

Table 3. Cross-tabulation of the nature of QA and the role of participants

	Architect	**Other**	**Total**
QA explicit	12 (71%)	8 (73%)	20 (71%)
QA implicit	5 (29%)	3 (27%)	8 (29%)
Total	17 (100%)	11 (100%)	28 (100%)

RQ2: To what extent are QA specific to application domains of SBS?

To answer RQ2, we used responses to the question about the most important QA that participants had experienced. During analysis we mapped all QA stated by participants in terms of scenarios to QA for SBS as defined by the S-Cube quality model [18]. This

was done through content analysis where combinations of three researchers categorized each QA. Figure 3 shows the frequency distribution of QA. We grouped data-related quality attributes from the S-Cube quality model (data reliability, completeness, accuracy, integrity, validity). Dependability and performance are the most frequently addressed QA. As not all participants provided a complete scenario, the total number in Figure 3 is 28.

Fisher's exact test did not reveal any correlation between QA and domains (p = 0.456). We analyzed the correlation between the QA and their importance, and we found that except for dependability and performance which tend to be considered more important than functionality, there is no correlation between other QA and their importance (p = 0.983).

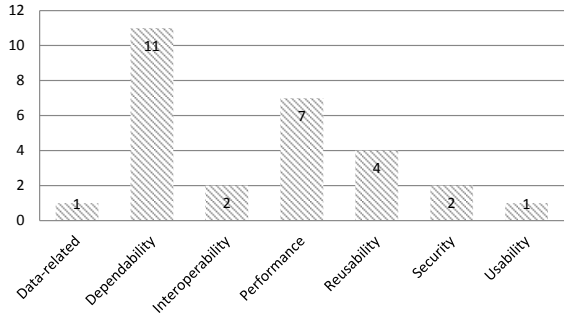

Fig. 3. Frequency distribution of QA

RQ3: What kind of architectural decisions are used to address QA in SBS?

We used two classifications to differentiate the kinds of decisions. First, we used Kruchten's taxonomy of decision types [13]. Second, we classified decisions based on decision categories: *Ad-hoc*: Solution that is specific to a concrete problem of the project (e.g., the architect decides to create a separate service to store sensitive information about the users to improve the security of the system). *Pattern*: Reusable and widely-known architectural solution (e.g., the decision to use of the Model-View-Controller pattern). *Technology*: A piece of implemented software that fulfills some required functionality (e.g., the use PostgreSQL instead of other DBMS).

Assigning decisions made to accommodate QA to types and categories of decisions was made based on a content analysis involving all authors. Figures 4 and 5 show the results. One decision was not classified because the participant did not provide a description for it. We found a correlation between decision types and decision categories (Fisher's exact test: p = 0.018): 83.3% of technology decisions are existence decisions, 69.2% of the ad-hoc decisions are property decisions, and 54.5% of pattern decisions are also property decisions.

QA and decision classification. We tried to find correlations between the decision classifications and the QA mentioned by the participants. The results were not significant. We obtained p = 0.835 (for types) and p = 0.741 (for categories).

QA treatment and decision documentation. As part of analysing the types of decisions made to accommodate QA, we studied if these decisions were actually

documented or treated implicitly. There is a correlation between treating QA explicitly and documenting decisions (Fisher's exact test: $p = 0.022$, Table 4). All participants that treated QA explicitly also documented the decisions to accommodate this QA. Also, all participants that did not document decisions treated QA implicitly. We also found a relationship between the importance of a QA and if decisions have been documented ($p = 0.112$, Table 5). Note that only 26 participants provided information about the degree of documentation of their architecture decisions.

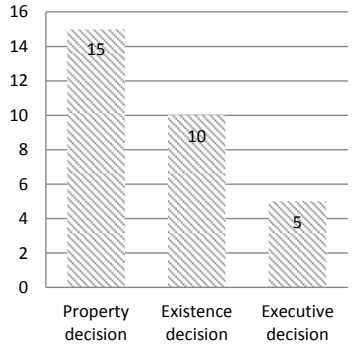

Fig. 4. Decision types

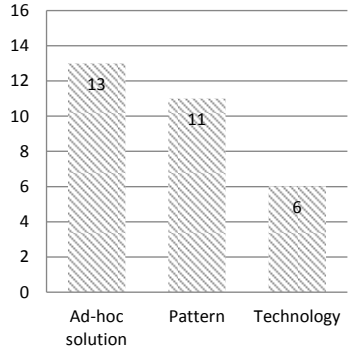

Fig. 5. Decisions categories

Table 4. Cross-tabulation of the nature of QA and documentation

	Not documented	Documented	Total
QA explicit	0 (0%)	18 (78.3%)	18 (69.2%)
QA implicit	3 (100%)	5 (21.7%)	8 (30.8%)
Total	3 (100%)	23 (100%)	26 (100%)

Table 5. Cross-tabulation of the importance QA and documentation

	Not doc.	Documented	Total
QA were AS important AS functionality	1 (33.3%)	17 (73.9%)	18 (69.2%)
QA were LESS important THAN functionality	2 (66.7%)	3 (13.0%)	5 (19.2%)
QA were MORE important THAN functionality	0 (0%)	3 (13.0%)	3 (11.5%)
Total	3 (100%)	23 (100%)	26 (100%)

4 Discussions of Results

Literature argues that QA are important and a major challenge when architecting SBS [2]. Our study showed that 71% of the participants indicated that QA were treated explicitly. This could be an indicator that special attention is paid to QA because they pose a major challenge. On the other hand, general literature about software architecting and design claims that QA drive the architecture [3]. We found that QA were rarely more important than functionality. However, stating that QA drive the architecture is different from stating that QA are more important than functionality. Also, we had indicators that QA were treated as global influence factors or architectural drivers for

high-level architectural decisions. This also indicates that using a service-based solution is not only a technology-driven decision but has sound rationale based on QA.

We found that the majority of participants treated QA and functionality as equally important in the context of SBS, in contrast to [9] who argued that QA are more important than functional requirements. In another study, van Heesch and Avgeriou [10] said that more than 80% of participants indicated that quality requirements play a prominent role during architecting. A similar result can be found in our study with practitioners in the context of SBS as only 19% of our participants indicated that QA were less important than functionality.

In [19] the authors conducted a survey to evaluate a catalogue of non-functional properties for SOA. The study found that security was prioritized as being absolutely essential in a quality model for SOA. However, our study showed that security only occurred in two projects. Reusability or dependability, two main features of SBS were not found to be relevant non-functional characteristic in SOA in [19].

A study in the embedded systems industry [20] studied how quality requirements are handled in practice. The study found that usability and performance are the most important quality aspects. While in our study dependability and performance are the most important QA, with usability being the least important QA. The difference in the importance of usability could be due to the nature of embedded systems versus SBS.

Non-functional requirements as seen by architects were studied by Poort et al. [21]. The study found that if architects are aware of non-functional requirements, they do not adversely affect project success. This is in line with our results that most participants consider quality attributes explicitly and at least equally important as functionality.

5 Conclusions and Future Work

We presented the results of a survey to study the role of QA in the context of SBS architecting. Our study provides empirical evidence of the current state of practice. Future work includes the extension of this study by gathering more responses, and a more detailed analysis of QA in industry, for example, by using case studies rather than broad surveys can be used to confirm or refute the findings of our study.

Acknowledgments

This work has been supported by the Spanish project TIN2010-19130-c02-01 and NWO SaS-LeG, contract no. 638.000.000.07N07.

References

1. IEEE Computer Society Software Engineering Standards Committee, IEEE Standard Glossary of Software Engineering Terminology, 1990.
2. Q. Gu, P. Lago, Exploring Service-oriented System Engineering Challenges: A Systematic Literature Review, Service Oriented Computing and Applications, 3 (2009) 171-188.
3. I. Ozkaya, L. Bass, R. Sangwan, R. Nord, "Making Practical Use of Quality Attribute Information," IEEE Software, 25(2), 2008.
4. L. O'Brien, L. Bass, P. Merson, Quality Attributes and Service-Oriented Architectures, Technical Report, SEI CMU, Pittsburgh, PA, 2005.
5. L. O'Brien, P. Merson, L. Bass, Quality Attributes for Service-oriented Architectures, in: International Workshop on Systems Development in SOA Environments, IEEE Computer Society, Minneapolis, MN, 2007, pp. 1-7.
6. S. Mahdavi-Hezavehi, M. Galster, P. Avgeriou, "Variability in Quality Attributes of Service-based Software Systems: A Systematic Literature Review," Information and Software Technology, 55 (2), 2013, pp. 320-343.
7. V. Basili, G. Caldiera, D. Rombach, The Goal Question Metric Approach, in: J.J. Marciniak (Ed.) Encyclopedia of Software Engineering, John Wiley & Sons, New York, NY, 1994, pp. 528-532.
8. C. Wohlin, M. Hoest, K. Henningsson, Empirical Research Methods in Software Engineering, in: R. Conradi, A.I. Wang (Eds.) Empirical Methods and Studies in Software Engineering, Springer Verlag, Berlin / Heidelberg, 2003, pp. 7-23.
9. U. van Heesch, P. Avgeriou, Mature Architecting - A Survey about the Reasoning Process of Professional Architects, WICSA, 2011, pp. 260-269.
10. U. van Heesch, P. Avgeriou, Naive Architecting - Understanding the Reasoning Process of Students - A Descriptive Survey, ECSA, 2010, pp. 24-37.
11. F. Bachmann, L. Bass, Introduction to the Attribute Driven Design Method, ICSE, 2001, pp. 745-746.
12. T. Erl, Service-Oriented Architecture (SOA): Concepts, Technology, and Design, Prentice Hall, Upper Saddle River, NJ, 2005.
13. P. Kruchten, An Ontology of Architectural Design Decisions in Software-intensive Systems, in: 2nd Groningen Workshop on Software Variability, Groningen, The Netherlands, 2004, pp. 54-61.
14. B. Kitchenham, S.L. Pfleeger, Principles of Survey Research - Part 6: Data Analysis, ACM SIGSOFT Software Engineering Notes, 28 (2003) 24-27.
15. M.B. Miles, A.M. Huberman, Qualitative Data Analysis, 2nd ed., Sage Publications, Thousand Oaks, CA, 1994.
16. K. Krippendorff, Content Analysis: An Introduction to its Methodology, 2nd ed., Sage Publications, Thousand Oaks, CA, 2003.
17. M. Ciolkowski, O. Laitenberger, S. Vegas, S. Biffl, Practical Experiences in the Design and Conduct of Surveys in Empirical Software Engineering, in: R. Conradi, A.I. Wang (Eds.) Empirical Methods and Studies in Software Engineering, Springer Verlag, Berlin / Heidelberg, 2003, pp. 104-128.
18. A. Gehlert, A. Metzger, Quality Reference Model for SBA, Deliverable #CD-JRA-1.3.2, S-Cube, 2009, pp. 64.
19. H. Becha, D. Amyot, "Non-functional Properties in Service Oriented Architecture – A Consumer's Perspective," Journal of Software, 7(3), 2012, pp. 575-587.
20. R.B. Svensson, T. Gorschek, B. Regnell, Quality Requirements in Practice: An Interview Study in Requirements Engineering for Embedded Systems, REFSQ, 2009, pp. 218-232.
21. E. Poort, N. Martens, I. van de Weerd, H. van Vliet, How Architects see Non-Functional Requirements: Beware of Modifiability, REFSQ, 2012, pp. 37-51.