Requirements Eng (2012) 17:255–258
DOI 10.1007/s00766-011-0137-9

Quality requirements engineering for systems and software architecting: methods, approaches, and tools

Rafael Capilla · Muhammad Ali Babar ·
Oscar Pastor

Received: 25 July 2011 / Accepted: 12 October 2011 / Published online: 29 October 2011

Abstract Requirements engineering and software architecture are quite mature software engineering sub-disciplines, which often seem to be disconnected for many reasons and it is difficult to perceive the impact of functional and non-functional requirements on architecture and to establish appropriate trace links for traceability purposes. In other cases, the estimation of how non-functional requirements, as the quality properties a system should pose, is not perceived useful enough to produce high-quality software. Therefore, in this special issue, we want to highlight the importance and the role of quality requirements for architecting and building complex software systems that in many cases require multidisciplinary engineering techniques, which increases the complexity of the software development process.

Keywords Quality attributes · Software architecture · Systems engineering

R. Capilla (✉)
Universidad Rey Juan Carlos de Madrid, Madrid, Spain
e-mail: rafael.capilla@urjc.es
URL: http://www.sait.escet.urjc.es/rafael

M. Ali Babar (✉)
IT University of Copenhagen, Copenhagen, Denmark
e-mail: malibaba@itu.dk
URL: http://malibabar.wordpress.com

O. Pastor (✉)
Universidad Politécnica de Valencia, Valencia, Spain
e-mail: opastor@dsic.upv.es
URL: http://www.pros.upv.es

1 Introduction

A quality attribute is a non-functional requirement (NFR) of a system such as reliability, modifiability, performance, and usability [7, 10]. Recently, it has been widely acknowledged that quality attributes of large-scale software intensive systems depend more on the overall architecture of such systems than on other factors such as the platform, programming models and languages, and data structures. That is why it is considered important to pay significant attention to the engineering tasks related to quality attributes such as eliciting, specifying, and modeling quality requirements for systematically designing and rigorously analyzing systems and software architectures.

A large number of software intensive systems are developed without rigorously and systematically addressing quality attributes at the architecture design stage. Quality attributes are many times used informally because of a general lack of supportive methods, and tools for systematically eliciting, specifying, and representing the desired set of quality properties of systems. In addition, engineering complex and multidisciplinary software systems requires a balance of the qualities of its major functional and critical parts, which complicates the evaluation and the selection of the right quality attributes, and its implementation in the final system.

The main objective of this special issue is to highlight the role of engineering quality requirements for architecting systems and provide a venue to report the state-of-the-art research on developing, evaluating, and deploying methods, approaches, and tools for eliciting, specifying, and modeling quality requirements [8, 9, 11, 13]. For this special issue, we sought for high-quality research reporting other types of requirements (e.g., application, hardware, platform, business) that are also used to drive the key decisions for designing and evaluating [3] a software or system architecture.

2 Quality requirements in architecture

Software architectures represent the cornerstone of any software design process. It is a commonly observed problem that many software architects tend to implement the functional requirements in the architecture, but they tend to compromise the quality requirements that are required by the key stakeholders. In addition, goal-oriented approaches play an important role in requirements engineering as they are used to transform goal-models into architecture models [1] with model-driven development architecture approaches [12].

Also, poor-quality designs or low-quality systems are many times delivered because the significant architectural design decisions may not have been taken into account during early evaluation of the architecture. Moreover, many organizations do not formally review architectures with respect to the desired quality attributes [4], as this is an activity considered important for engineering complex software systems.

During the last decade, there have been several significant efforts [9] to provide architectural support for quality-related issues including elicitation, specification, and representation of quality attributes in a precise manner. However, architecture challenges arise because the required quality attributes are informally stated during requirements elicitation and analysis, and hard to validate when the software system is ready for delivery. Hence, there is no synthesis of the research and practice reporting methods, approaches, and tools to support quality attributes elicitation, specifications, and modeling for designing and evaluating software and system architectures.

The evaluation of the quality attributes in the software architecture is also an important activity before the system is delivered, as architecture evaluation methods attempt to select the best design choices to achieve a balance of the desired qualities. As modern systems are more and more complex (e.g., pervasive, self-adaptive, service-oriented systems, or autonomic computing), they require a continuous modification of the quality contracts and Service Level Agreements (SLAs). Therefore, the current architecting and systems engineering practice must reflect how these quality properties are addressed in the development process. To this point, some of the challenges we have identified relating requirements and architecture are the following:

- How to elicit, specify, and represent quality requirements to drive the selection and the shape of a system's architecture?
- How should quality attributes be specified and represented in the system architecture of using a particular architecture viewpoint?
- How do the modeling approaches affect the design and analysis of system and software architecture?
- How to establish and evolve connections between quality attributes and design decisions and capture rationale for those connections?
- How to transform goal and soft-goal models to architecture models?

3 The role of quality requirements in systems engineering

While it is recognized that quality requirements play an important role for engineering modern applications, there is relatively less realization that there is acute shortage of appropriate and practical methods, approaches, and tools for engineering complex systems that demand robust and high-quality software architectures. Mobile applications, complex Web-based systems, self-adaptive software, and real-time systems demand high-quality software to avoid failures. Also, the increasing popularity of cloud-computing software poses a new set of challenges when it comes to dealing with quality requirements in engineering systems for cloud-computing environments [6]. As a result, the combination of a diversity of technologies makes more complex to evaluate and balance the relevant quality attributes and support different levels of quality as well.

In addition, quality attribute scenarios are becoming popular in terms of eliciting and specifying the quality requirements [5], but there is a general paucity of research on guiding users to select the most appropriate technique for quality attribute scenarios [2]. It is also a common observation that prioritizing, managing, and benchmarking quality requirements are challenging areas, which need specific methods and tools. Some other research questions that should draw immediate attention on the software and system engineering communities are the following:

- Which elicitation techniques are better to discover the most significant quality attributes for a particular types of systems (e.g., reliability for safety critical systems or availability for Web-based systems).
- How requirements related to a specific platform or technology (e.g., J2EE, SOA) are architecturally significant for system development and how their potential impact on architecting a system or software can be aligned with business goals?

These and many other research questions related to the role of NFR in the systems engineering practice constitute the motivation of this special issue for which we have selected the following three papers.

4 The articles in this issue

The three papers that appear in this special issue attempt to answer some of the research challenges stated before, such as we describe below.

The paper "*Deriving Software Architectural Models from Requirements Models for Adaptive Systems: The STREAM-A approach*" by J. Pimentel, M. Lucena, J. Castro, C. Silva, E. Santos and F. Alencar focuses on a strategy (called STREAM-A) to transition from requirements to architecture models and applied to adaptive systems. This approach uses goal models based on the so-called *i-star* framework to design and evolve systems that require some kind of adaptation and transforms *i-star* models to architecture models. As adaptive systems require of certain capabilities to accommodate and optimize the changes in the environment, many of them motivated by quality concerns, the evolution and alignment of new requirements to architectural and system changes are challenging, in particular when runtime capabilities are needed. The transformations proposed in this paper helps software engineers to perform a smooth transition from requirements to architecture models. The so-called STREAM-A process applies horizontal transformations (i.e., the source and target models are in the same level of abstraction, such as requirements to requirements), to requirement models represented as *i-star* goal models. In addition, vertical transformations (i.e., models that are on different abstraction levels, such as requirements to architectures) map the result of horizontal transformation onto architectural models like Acme. In this work, architectures using the Acme Architecture Description Language (ADL) can be associated with the rationale of the architecture and their design decisions and explains strategic rationale diagrams of systems. Refinements from soft goals to architecture models focused on *adaptability* are also used to connect better requirements and architecture models.

The paper "Setting Quality Targets for Coming Releases with QUPER—an Industrial Case Study" by R. Berntsson Sevenson, Y. Sprockel, B. Regnell, and S. Brinkkenper, discusses a quality performance model used to estimate quality targets for architecting with quality requirements. The proposed model uses qualitative reasoning to assist the decision-making activity and set targets of quality requirements for new product releases. The authors use a case study in the e-payment domain to validate their approach and they investigate how the model performs in terms of *usability* and *usefulness*. Because QUPER has been developed in close collaboration with industry, the relationship between quality and cost is highlighted in the paper as a relevant factor in competitive environments. Upgradability of the e-payment system architecture as e-payment solution (EPS) is studied using QUPER to prioritize the quality requirements and estimate better the cost barrier. The authors identify breakpoints for quality indicators of the EPS and which would be the overall cost barrier of the total software optimization budget. Interesting lessons learnt refer to the applicability of QUPER to *upgradability* and *performance* requirements and to improve the decision-making process (e.g., for release planning).

The third paper "*An Integrated Strategy to Systematically Understand and Manage Quality in Use for Web Applications*" by P. Lew, L. Olsina, P. Becker and L. Zhang deals with an evaluation-driven strategy for understanding and improving software quality requirements in a systematic way (SIQinU). The paper discusses the case of Web applications (WebApp) from an internal/external quality perspective to evaluate and find potential *usability* problems. The authors evaluate external quality properties (i.e., as inherent properties of Web applications) to derive a benchmark to improve subsequent evaluations in current usability. Potential relationships between external qualities (EQ) and actual usability (QinU) are defined to enhance Web applications. The SIQinU strategy consists of 6 integrated phases starting and ending with QinU (actual usability) evaluation. Finally, a case study describes a defect reporting of a WebApp and focused on assessing the actual usability to provide recommendations as improvement actions for the external quality.

5 Summary

The increasing demand of high-quality software and the interplay of different software and hardware technologies for engineering modern and complex software systems is one of main drivers to cater the quality needs of systems. Such NFRs must be systematically elicited, specified, and evaluated at the architecture level, as software architects use them to make the best design decisions when several alternatives are available. Therefore, the elicitation, specification, and evaluation of quality requirements in the early stages of the architecture development process are considered important for the success of software development. Moreover, certain software systems that demand runtime capabilities need to check continuously the quality level of the system based on current contextual conditions new requirements. Hence, quality properties are becoming more and more relevant for many application domains as these are continuously revisited when the environment changes. The articles described in this special issue address many of the challenges discussed before and state the importance of quality properties in different application domains for software and systems engineering development.

Acknowledgments We would like to thank all the people that contributed to the preparation of this special issue and Professor P. Loucopoulos, as Requirements Engineering Journal co-Editor-in-Chief, for giving us this opportunity. We also give thanks to Professor W. Robinson, co-Editor-in-Chief for his accurate comments and review of this Editorial. Finally, we want to thank the reviewers who contribute with their effort in reviewing the articles of this special issue.

References

1. Alencar FMR, Marín B, Giachetti G, Pastor O, Castro J, Pimentel JH (2009) From i* requirements models to conceptual models of a model driven development process. The practice of enterprise modeling, second IFIP WG 8.1 working conference (PoEM 2009), pp 99–114
2. Ali Babar M, Biffl S (2006) Eliciting better quality architecture evaluation scenarios: a controlled experiment on top-down vs. bottom-up. ISESE 2006:307–331
3. Ali Babar M, Capilla R (2008) Capturing and using quality attributes knowledge in software architecture evaluation process. In: Proceedings of 1st international workshop on managing requirements knowledge (MARK'08). IEEE Computer Society, ACM DL, pp 53–62
4. Ali Babar M, Gorton I (2009) Software architecture review: the state of practice. IEEE Comput (COMPUTER) 42(7):26–32
5. Bass L, Clements P, Kazman R (2003) Software architecture in practice, 2nd edn. Addison Wesley, Reading
6. Chauhan MA, Ali Babar M (2011) Migrating service oriented systems to cloud computing: an experience report, the 4th international conference on cloud computing
7. Chung L, Sampaio do Prado Leite JC (2009) On non-functional requirements in software engineering. Conceptual modelling: foundations and applications 2009:363–379
8. España S, Condory-Fernández N, González A, Pastor O (2009) Evaluating the completeness and granularity of functional requirements specifications: a controlled experiment. In: 17th IEEE international requirements engineering conference (RE 2009), IEEE Computer Society, pp 161–170
9. Kazman R, Bass L (1994) Toward deriving software architectures from quality attributes. Software Engineering Institute technical report CMU/SEI-94-TR-10
10. Laplante P (2009) Requirements engineering for software and systems, 1st edn. CRC Press, Redmond
11. Paech B, Dutoit A, Kerkow D, von Knethen A (2003) Functional requirements, non-functional requirements, and architecture should not be separated. In: Proceedings of the international workshop on requirements engineering: foundations for software quality. Essen, Germany, September 9–10, 2002. Elsevier BV, Amsterdam, pp 102–107
12. Pastor O, Molina JC (2007), Model-driven architecture in practice: a software production environment based on conceptual modeling. Springer, New York
13. Stephenson M, McDermid J (2005) Deriving architectural flexibility requirements in safety-critical systems. IEEE Proc Softw 152(4):143–152