Quality Driven Software Architecture

Peter Hruschka

Atlantic Systems Guild, Langenbruchweg 71
52080 Aachen, Germany
peter@systemsguild.com

Abstract. This short paper introduces "quality driven software architecture" (QDSA) as a method to ensure qualities such as maintainability, modularity, scalability, or extensibility in software architectures and emphasizes the need for a person in charge (i.e. the software architect) to actively manage and control such qualities.

Keywords: Software Architecture, Quality Goals, Architecture Development Process, Evaluating Software Architecture

1. Introduction

Users normally expect a certain qualities from their software systems, even if they do not explicitly ask for it. They often take issues like maintainability, modularity, scalability, extensibility, etc. for granted.

There are various ways to come up with key design decisions for a software architecture. A popular approach is work "domain driven" (as e.g. elaborated in [1] or [2]), to start with fundamental architectural patterns (e.g. [3]) or answer six key questions to create a first sketch of the architecture [4]. While all these approaches have their merits, they often ignore quality requirements too long in the process.

Therefore, we suggest that software architects should start from explicitly negotiated quality goals (or architecture goals). Those are often a subset of non-functional requirements. Despite all the attention that requirements engineering got in industry over the last 15 years the non-functional requirements still tend to be ignored, considered obvious or treated as orphans. And parts of them, the so-called internal qualities (like extensibility, scalability, modifiability, reusability, …) are sometimes in conflict with the project goals and constraints (like time to market, costs, …). Therefore, QDSA makes it the software architect the advocate for such longer term architectural goals. Thus, he or she is an excellent sparring partner for the project manager who is striving to achieve the project goals. Project goals are to determine whether the project was successful, while architecture goals are to determine whether the solutions has been structured in a way to achieve those longer term goals. Successful product development should ensure that both sets of goals are

met and – if there is a conflict between them – that this conflict is openly discussed and resolved early on.

2. Quality Tree as a Starting Point

Clements et. al. [5] describe an excellent process (ATAM – Architecture Tradeoff Analysis Method) to determine how well a software architecture meets given quality goals. The key is to structure the quality goals in form of a quality tree. Concrete scenarios form the leaves of the tree, prioritized from two different points of view, the business value and the architectural challenge (cf. example in Figure 1). Those with high marks in both areas are used by experts to discuss design decisions and suggest measures for the "weak spots".

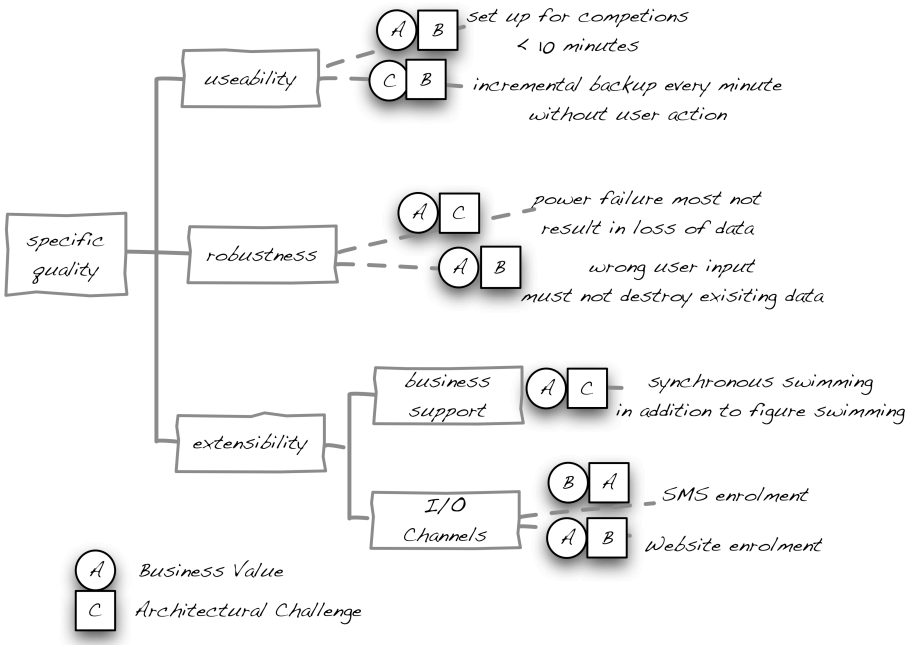

Fig. 1. Example for a Quality Tree with Scenarios

3. Changing ATAM to a Constructive Process

In the QDSA-approach of ARC42 [6], we suggest to use the ATAM techniques as part of the architect's normal, iterative architecture tasks (cf. figure 2).

The task "clarify requirements & constraints" – among other things – is to create the quality tree. This encourages the software architect to gain an excellent understanding of the requirements, especially the non-functional ones, which are often neglected in requirements documents. Since these quality requirements are the

essential design drivers a solid understanding prior to making key architecture decisions is very helpful. In the two key tasks in the middle of figure 2 the software architect uses this detailed knowledge about the design drivers to come up with an adequately balanced design, always aware of the potential tradeoffs he or she has to make.

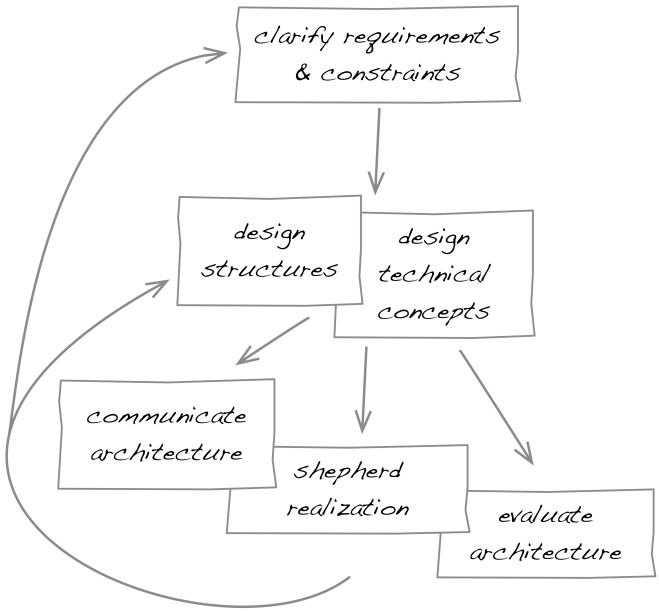

Fig. 2. The Iterative Architecture Development Cycle of ARC42

The three feedback tasks at the bottom of figure 2, especially the task "evaluate architecture" constantly check alignment with the defined goals. While ATAM is more of a one-shot task, these task becomes part of everyday work. Normally an hour every two weeks for evaluating is enough to stay on track.

4. Summary and the Way Ahead

The next steps in the development of QDSA are to support software architects further by suggesting best practices, i.e. constructive strategies and policies for selected quality goals to be applied under given constraints in the tasks "design structures" an "design technical concepts". Then the task "evaluate architecture" should become a formal routine quality check, since it should not come up with any surprises.

Acknowledgments. Thanks to Gernot Starke for jointly developing the QDSA approach and the ARC42 portal for architects.

References

1. Evans, E.: Domain Driven Design, Addison Wesley (2003)
2. Nilsson, J.: Applying Domain Driven Design & Patterns, Addison Wesley (2006)
3. Buschmann, F., Henney, K., Schmidt, D..: Pattern Oriented Software Architecture Volume 4 – A Pattern Language for Distributed Computing (2007)
4. Starke, G., Hruschka, P .: Software-Architektur kompakt, 2nd edition, Springer (2011) (in German)
5. Clements, P. et al. .: Evaluating Software Architectures, Addison Wesley (2001)
6. The ARC42 Portal for Software Architects, http://www.arc42.com