Quality Attribute Workshop

NEWS AT SEI

Author

Mario R. Barbacci

This library item is related to the following area(s) of work:

Software Architecture (http://www.sei.cmu.edu/architecture)

This article was originally published in News at SEI on: June 1, 2000

In large software systems, the achievement of qualities such as performance, availability, security, and modifiability is dependent not only on code-level practices (e.g., language choice, detailed design, algorithms, data structures, and testing), but also on the overall software architecture. The quality attributes of large systems can be highly constrained by a system's software architecture. Thus, it is in our best interest to try to determine at the time a system's software architecture is specified whether the system will have the desired qualities.

In previous columns we have written about various components of an emerging "software architecture practice" by drawing analogies to architectural engineering and describing approaches to software architecture representation, quality attributes, and the use of scenarios in architecture evaluations. In this column, I describe one way for combining several of these components into a process that allows early insight into a system's architecture, including its quality-attribute sensitivities, tradeoffs, and risks.

Making Attribute Goals Concrete

Quality attributes are interdependent. For example, performance affects modifiability, availability affects safety, security affects performance, and everything affects cost. Therefore, achieving one quality attribute can affect the other attributes [Boehm 78]. These side effects reflect dependencies among attributes and can be defined by parameters shared among attribute models. If we can identify these parameters, the results from one analysis can feed into the others.

Quality attributes, such as modifiability, performance, and security, are not definitive enough by themselves either for design or for evaluation. They must be made more concrete. Using modifiability as an example, if a system can be easily adapted to have different user interfaces but is dependent on a particular operating system, is it modifiable? The answer is that it depends on what modifications are expected to the system over its lifetime. That is, the abstract quality of modifiability must be made concrete. The same observation is true for other attributes.

For the past two years the SEI has been developing the Architecture Tradeoff Analysis Method (ATAM). ATAM is based on a set of attribute-specific measures of a system-some analytic, based on formal models (e.g., performance and availability), and some qualitative, based on formal inspections (e.g., modifiability, safety, and security). We now have a stable process for carrying out ATAM analyses. The process includes a set of steps and a set of reusable architectural styles and analytic models, called attribute-based architectural styles (ABASs), that we use during an ATAM analysis. The ATAM process will be covered in a future issue of The Architect, so we will not dwell on the method here.

The effectiveness of ATAM depends on having a concrete, well-defined architecture to be analyzed. However, some ATAM benefits can be achieved even if an architecture is not fully defined. Under some circumstances, an organization might wish to identify potential architecture risks while developing a system's architecture. With the sponsorship of the U.S. Coast Guard, we are testing the concept of a "Quality Attribute Workshop" in which architects, developers, users, maintainers, and other system stakeholders, such as people involved in installation, deployment, logistics, planning, and acquisition, carry out several ATAM steps, but focus on system requirements and quality attributes, rather than on the architecture. The objective of the workshop is to identify sensitivities, tradeoffs, and risks and use these as early warnings to the architecture developers.

The workshop is intended as a forum for the discussion of quality attributes and their evaluation. The workshop does not aim at an absolute measure of "architecture quality;" rather the purpose is to identify scenarios from the point of view of a diverse group of stakeholders (e.g., the architect, developers, users, sponsors) and to identify risks (e.g., inadequate performance, successful denial-of-service attacks) and possible mitigation strategies (e.g., replication, prototyping, simulation).

Roadmap Activities

Figure 1 illustrates the Quality Attribute Roadmap, the process we use during the workshops to discover and document quality attribute risks, sensitivity points, and tradeoffs in the architecture, where

- *risks* are architecture decisions that might create future problems for some quality attribute requirement
- *sensitivity points* are architecture parameters for which a slight change makes a significant difference in some quality attribute
- *tradeoffs* are architecture parameters affecting more than one quality attribute

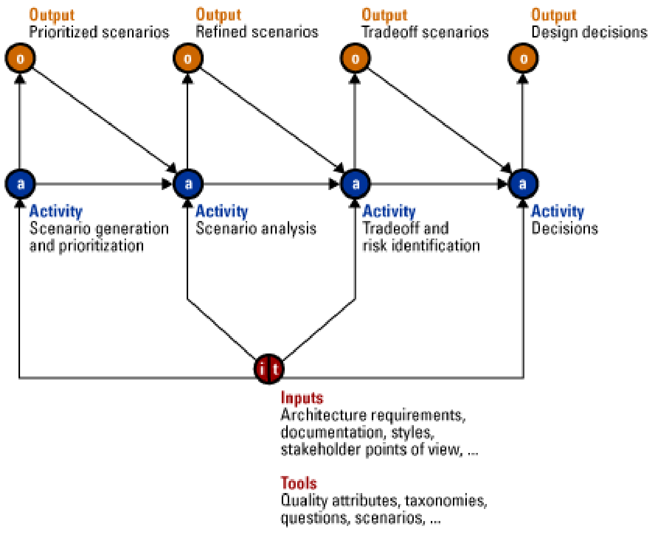
Output
Prioritized scenarios
Output
Refined scenarios
Output
Tradeoff scenarios
Output
Design decisions
o
a
Activity
Scenario generation and prioritization
Activity
Scenario analysis
Activity
Tradeoff and risk identification
Activity
Decisions
i t
Inputs
Architecture requirements, documentation, styles, stakeholder points of view, ...
Tools
Quality attributes, taxonomies, questions, scenarios, ...

Figure 1: Quality Attribute Roadmap

During the workshop we conduct several activities aimed at generating various outputs or products:

- Scenario generation takes place during a facilitated brainstorming process; stakeholders propose scenarios that test the effectiveness of a candidate or conceptual architecture to achieve specific quality attributes within a specific Deepwater mission and geographic context. For prioritization, each stakeholder is assigned a number of votes that she can allocate as desired.
- During scenario analysis, for each of the high-priority scenarios, the stakeholders choose an appropriate architectural style or architectural fragment as an artifact for analysis, and apply the scenario to the artifact. The purpose of the analysis is to

identify important architecture decisions and sensitivity points. As a result of this activity, the stakeholders might decide to conduct additional, more detailed or formal analyses of the scenarios or artifacts, but these activities take place offline, not during the workshop.

- During tradeoff and risk identification, the stakeholders use the results of the analysis activity to identify and document risks-i.e., potential future problems that might impact cost, schedule, or quality attributes of the system. Scenarios to consider include:
  - a single scenario that involves two attributes explicitly or implicitly
  - multiple scenarios about different attributes sharing common factors (e.g., resources, protocols)
  - multiple contradictory scenarios

We use various sources as inputs for the activities, including:

- architecture documentation
- stakeholder points of view
- architecture styles

The stakeholders generate, prioritize, and analyze the scenarios, and identify tradeoffs and risks from their points of view, depending on the role they play in the development of the system, and their expertise on specific quality attributes. As an additional input source, we try to identify known architectural styles because they can expedite the process. Architecture styles are abstractions such as "client/server," "publish/subscribe," "shared memory," "layered," and "pipe and filter," which can be used as drivers for the analysis because they provide "canned" scenarios, known tradeoffs, and likely risks. The results of the analysis would depend on which architecture styles are used.

Finally, there is a collection of tools and techniques that we use to perform a quality attribute analysis:

- scenarios
- quality attribute tables
- questions

These sometimes have different labels, such as "screening questions" or "exploratory scenarios." It is important to be precise in our use of terms to ensure that (a) we share the same understanding, and (b) we can decide what tools to use and when to use them. Thus, prior to the workshops, the participants receive a handbook describing the activities and the tools to be used and as a reminder, they are taken through a short presentation at the beginning of the meeting. The rest of this article details some of the tools and the experiences we have had with the workshops.

Scenarios

Scenarios are used to exercise the architecture against current and future situations:

- Use-case scenarios reflect the normal state or operation of the system. If the system is yet to be built, these would be about the initial release.
- Growth scenarios are anticipated changes to the system. These can be about the execution environment (e.g., double the message traffic) or about the development environment (e.g., change message format shown on operator console).
- Exploratory scenarios are extreme changes to the system. These changes are not necessarily anticipated or even desirable situations. Exploratory scenarios are used to explore the boundaries of the architecture (e.g., message traffic grows 100 times, operating system is replaced).

The distinction between growth and exploratory scenarios is system- or situation-dependent. Anticipated growth in a business application might be a disaster in a deep space probe (e.g., 20% growth in message storage per year).Table 1 shows several representative performance scenarios.

Table 1: Performance Scenarios Scenario Type

Scenario	Type
The communications network is overloaded. Missions are reassigned to reduce traffic.	use case
The LAN is overloaded. Tasks are reassigned to reduce traffic.	use case
The process-to-processor allocation is changed to balance the load.	use case
Data throughput is doubled.	growth
The number of users doubles.	growth
Real-time video data is needed by central office.	exploratory
Important, but not mission-critical, traffic doubles in volume.	growth

There are no clear rules other than stakeholder consensus that some scenarios are likely (desirable or otherwise) and other scenarios are unlikely (but could happen and, if they occurred, it would be useful to understand the consequences).

Quality Attribute Tables

The handbook used in the workshop describes various quality attributes, characterized by stimuli, responses, and architectural decisions that link them. Stimuli and responses are the activities (operational or developmental) that exercise the system and the observable effects, respectively. For example, a stimuli for the "modifiability" attribute could be "change the operating system," and the responses could include "effort to implement" and "number of subsystems affected." The architecture decision in this case might be "use a virtual machine approach." Each attribute is described by stimulus/response/mechanism tables, where the level of detail is appropriate to the state of development and the available documentation. See Figure 2 for an illustration of a table of architecture mechanisms for the modifiability attribute.

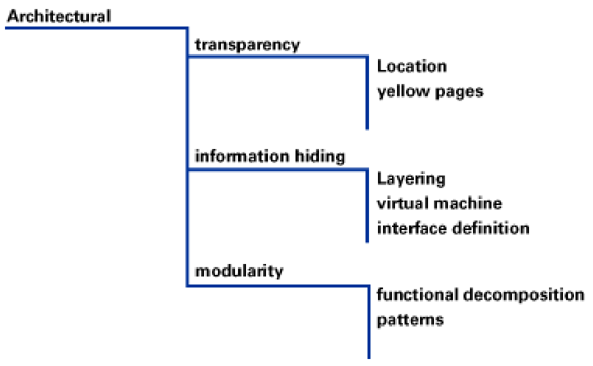

Figure 2: Modifiability Architecture Mechanisms

The attribute tables are used only to suggest stimuli, responses, and mechanisms that might be of interest. They are just a reminder of the kinds of issues we want the participants to take into consideration when generating and analyzing scenarios. We could expect that, depending on the interests of the stakeholders, quality attribute tables might be added, removed, refined, or pruned, as the participants see fit.

Questions

We use various types of questions to collect and analyze information about current and future system drivers and architectural solutions.

- Screening questions are used to quickly narrow or focus the scope of the evaluation. They identify what is important to the stakeholders.

  - Screening questions are qualitative; the answers are not necessarily precise or quantifiable. The emphasis is on expediency.
  - Screening questions can be driven by a quality attribute deemed important to some stakeholders. Sometimes the attribute is clear and explicit (e.g., "the service must be continuous" identifies availability and security as the quality attributes of concern). Sometimes the attribute is implied (e.g., "life-cycle cost must be minimal" suggests modifiability and interoperability as the relevant quality attributes).
  - Screening questions can also be driven by a subsystem or a service deemed important to achieve a quality attribute. For example, once an important attribute is identified by the stakeholders, screening questions can be used to narrow or focus on subsets of the architecture that are relevant to achieving the attribute (e.g., the user authentication subsystem, the message filtering and distribution subsystem).

- Elicitation questions are used to gather information to be analyzed later. They identify how a quality attribute or a service is achieved by the system.
  - Elicitation questions collect information about decisions made; the emphasis is on extracting quantifiable data.
  - Elicitation questions can be driven by an attribute model. We ask for quality attribute-specific information when the answer is a parameter of an attribute model (e.g., message arrival rates are parameters in a model of throughput, repair rates are parameters in a Markov model of availability). These elicitation questions are guided by stimulus/response branches of the quality attribute tables.
  - Elicitation questions can also be driven by architecture styles. We ask for architectural information when the answer is important to determine the "quality" of a particular architecture style choice (e.g., individual latencies are required to compute the performance of a pipe-and-filter architecture). These elicitation questions are guided by the architecture mechanism branch of the quality attribute tables.
- Analysis questions are used to conduct analysis using attribute models and information collected by elicitation questions. Analysis questions refine the information gathered by elicitation.

There is an implied ordering in the questions (i.e., screening > elicitation > analysis) although questioning can be carried out in breadth-first or depth-first order:

- Breadth-first questioning first identifies all important attributes and subsets of the architecture. Then, for each one, questioning elicits all the information that will be used later for analysis.
- Depth-first questioning dives deeply into an important attribute or subset of the architecture before other attributes or subsets of the architecture are considered.

Either order can be used, and the decision might be opportunistic. During a discovery or early analysis exercise, breadth-first might be more appropriate; during an evaluation or detailed analysis exercise, depth-first might be more appropriate.

For each quality attribute of interest, attribute-specific example questions serve as seeds for additional questions about stimuli, response, or architecture mechanisms. Table 2 provides an example of a list of specific questions for security.

Table 2: Security Questions

	Question	**Type**

Requirements	What are the trusted entities in the system and how do they communicate?	Screen
Stimulus/Response	Which essential services could be significantly affected by an attack?	Analysis
	Are there attacks or events that could affect service across the entire integrated system?	Analysis
	Is there a single point from which the entire system is controlled?	Analysis
	For which kind of attacks will recovery be the most difficult?	Analysis
Resistance/Recovery/ Recognition	How is user authentication and authorization information maintained for employees?	Elicitation
	How is access managed for those people who are outside the network?	Elicitation
	What sensitive information must be protected?	Elicitation
	What approach is used to	Elicitation

	protect that data?	
	Which user actions are logged?	Elicitation
	What kind of monitoring and access controls exist at network boundaries?	Elicitation
	What information is permitted through or filtered out?	Analysis

The questions are not meant to be exhaustive; rather they are meant to serve as starting points and as examples for stakeholders to generate additional questions about the quality attribute requirements and the system.

Scenarios and questions contain the explicit or implied attribute stimulus/response /mechanisms that are deemed important. Scenarios and questions might raise doubts or concerns regarding some aspect of the architecture about which we might have to elicit further information to conduct a more detailed analysis. They serve to identify potential risks, such as what risks can arise from decisions made (e.g., choice of middleware), as well as decisions not yet made (e.g., message encoding).

The generation of scenarios can alternate with the generation of questions. For example, screening questions can identify regions of stimuli/responses as sources of use-case scenarios, which in turn might suggest questions about architecture mechanisms involved in the scenario. For example, a screening question might identify message throughput as important; a scenario about message throughput would identify the components involved in the message path. The capacity or speed of some components might be seen as questionable, prompting further elicitation questions (e.g., time required to process a message or choice of queuing policy).

Experience with Quality Attribute Workshops

We have conducted a handful of workshops, and the process is still evolving. As indicated earlier, the intent of the workshops is to encourage an organization to generate and analyze scenarios about a hypothetical system, not necessarily something under development. However, we need something to analyze the scenario against! For example, if a scenario suggests that message throughput is important, we need a sketch of the components and connections that implement the subsystem that processes the messages. Because no such decisions are expected to have been made at the time of the workshop, when we analyze a scenario, the architect can suggest a reasonable or likely candidate architecture for the purposes of the exercise. The stakeholders are not bound to that solution and are not "graded" on the effectiveness of a choice made on the spur of the moment. However, the scenarios, questions, and attribute tables remain with the organization, and they can repeat the exercise using alternative subsystem architectures.

References

[Deepwater] United States Coast Guard Deepwater Project

[Freedberg 00] Freedberg, S. Jr. Coast Guard uses new model to procure new fleet. Daily Briefing Column, GovExec.com (http://www.sei.cmu.eduhttp://www.govexec.com /dailyfed/0400/042400b2.htm) April 24, 2000

[ATAM] The Architecture Tradeoff Analysis (ATA) Method (http://www.sei.cmu.edu /architecture/consulting/)

[Boehm 78] Boehm, B.W., Brown, J.R., Kaspar, H., Lipow, M., MacLeod, G.J. & Merritt, M.J. Characteristics of Software Quality. New York, NY: Elsevier North-Holland Publishing Company, Inc., 1978.

About the Author

Mario Barbacci is a senior member of the technical staff at the SEI. He was one of the founders of the SEI, where he has served in several technical and managerial positions, including project leader (Distributed Systems), program director (Real-Time Distributed Systems, Product Attribute Engineering), and associate director (Technology Exploration Department). Before coming to the SEI, he was a member of the faculty in the School of Computer Science at Carnegie Mellon.

Barbacci is a fellow of the Institute of Electrical and Electronic Engineers (IEEE), a member of the Association for Computing Machinery (ACM), and a member of Sigma Xi. He was the founding chairman of the International Federation for Information Processing (IFIP) Working Group 10.2 (Computer Descriptions and Tools) and has served as vice president for technical activities of the IEEE Computer Society and chair of the Joint IEEE Computer Society/ACM Steering Committee for the Establishment of Software Engineering as a Profession. He was the 1996 president of the IEEE Computer Society. He was the 1998-1999 IEEE Division V Director.

Barbacci is the recipient of several IEEE Computer Society Outstanding Contribution Certificates, the ACM Recognition of Service Award, and the IFIP Silver Core Award. Barbacci received bachelor's and engineer's degrees in electrical engineering from the Universidad Nacional de Ingenieria, Lima, Peru, and a doctorate in computer science from Carnegie Mellon.

The views expressed in this article are the author's only and do not represent directly or imply any official position or view of the Software Engineering Institute or Carnegie Mellon University. This article is intended to stimulate further discussion about this topic.

For more information

Contact Us

info@sei.cmu.edu

412-268-5800