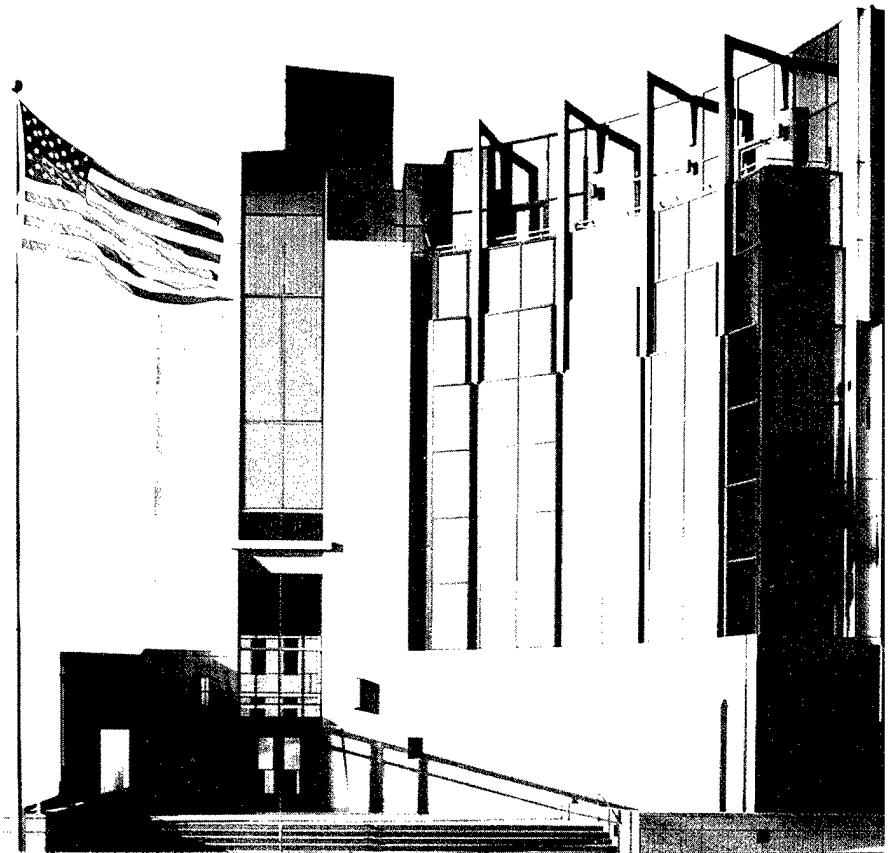
Carnegie Mellon
Software Engineering Institute

Quality Attribute Workshops, 2nd Edition

Mario R. Barbacci
Robert Ellison
Anthony J. Lattanze
Judith A. Stafford
Charles B. Weinstock
William G. Wood

June 2002

TECHNICAL REPORT
CMU/SEI-2002-TR-019
ESC-TR-2002-019



Carnegie Mellon University does not discriminate and Carnegie Mellon University is required not to discriminate in admission, employment, or administration of its programs or activities on the basis of race, color, national origin, sex or handicap in violation of Title VI of the Civil Rights Act of 1964, Title IX of the Educational Amendments of 1972 and Section 504 of the Rehabilitation Act of 1973 or other federal, state, or local laws or executive orders.

In addition, Carnegie Mellon University does not discriminate in admission, employment or administration of its programs on the basis of religion, creed, ancestry, belief, age, veteran status, sexual orientation or in violation of federal, state, or local laws or executive orders. However, in the judgment of the Carnegie Mellon Human Relations Commission, the Department of Defense policy of "Don't ask, don't tell, don't pursue" excludes openly gay, lesbian and bisexual students from receiving ROTC scholarships or serving in the military. Nevertheless, all ROTC classes at Carnegie Mellon University are available to all students.

Inquiries concerning application of these statements should be directed to the Provost, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-6684 or the Vice President for Enrollment, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-2056.

Obtain general information about Carnegie Mellon University by calling (412) 268-2000.

Carnegie Mellon
Software Engineering Institute
Pittsburgh, PA 15213-3890

Quality Attribute Workshops, 2nd Edition

CMU/SEI-2002-TR-019
ESC-TR-2002-019

Mario R. Barbacci
Robert Ellison
Anthony J. Lattanze
Judith A. Stafford
Charles B. Weinstock
William G. Wood

June 2002

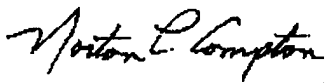
Architecture Tradeoff Analysis Initiative

This report was prepared for the

SEI Joint Program Office
HQ ESC/AXS
5 Eglin Street
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

FOR THE COMMANDER



Norton L. Compton, Lt Col, USAF
SEI Joint Program Office

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number F19628-00-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (http://www.sei.cmu.edu/publications/pubweb.html).

Table of Contents

List of Figures

List of Tables

Abstract

Quality attribute workshops (QAWs) provide a method for analyzing a system's architecture against a number of critical quality attributes, such as availability, performance, security, interoperability, and modifiability, that are derived from mission or business goals. The QAW does not assume the existence of a software architecture. It was developed to complement the Architecture Tradeoff Analysis Method[SM] (ATAM[SM]) in response to customer requests for a method to identify important quality attributes and clarify system requirements *before* there is a software architecture to which the ATAM could be applied. The analysis is based on applying a set of test cases to a system architecture. These test cases include questions and concerns elicited from stakeholders associated with the system. The process of building the test cases allows stakeholders to communicate among themselves, thereby exposing assumptions that may not have surfaced during requirements elicitation. Our experience to date includes multiple QAWs that were held with four different U.S. government acquisition programs.

This is the second edition of a technical report describing QAWs. This report clarifies the context in which a QAW is applicable, provides a rationale for developing the process and describes it in detail, and concludes with a list of lessons learned and a discussion of how these lessons have helped evolve the process to its current state.

1 Introduction

Quality attribute workshops (QAWs) provide a method for analyzing a system's architecture against a number of critical quality attributes, such as availability, performance, security, interoperability, and modifiability, that are derived from mission or business goals. The analysis also provides insights as to how these attributes interact, forming a basis for making tradeoffs between these attributes [Barbacci 95, Barbacci 96, Barbacci 97].

The QAW does not assume the existence of a software architecture. It was developed to complement the Architecture Tradeoff Analysis Method℠ (ATAM℠)[1] [Kazman 00] in response to customer requests for a method to identify important quality attributes and clarify system requirements *before* there is a software architecture to which the ATAM could be applied. In an ATAM evaluation, an external team facilitates short meetings between stakeholders during which scenarios representing the quality attributes of the system are developed, prioritized, and analyzed against the software architectural approaches chosen for the system. The results are expressed as risks, sensitivity points, and tradeoffs. In order to conduct an ATAM evaluation, an articulation of the business drivers and at least an initial draft of the software architecture are required.

The QAW conducts similar activities but earlier in the life cycle of a project. In the QAW, an external team facilitates meetings between stakeholders during which scenarios representing the quality attribute requirements are generated, prioritized, and refined (i.e., adding additional details such as the participants and assets involved, the sequence of activities, and questions about quality attributes requirements). Following the stakeholder meetings, the refined scenarios are converted into test cases which the architecture team analyzes against the system architecture. Then the team documents the results. The test case creation and analysis often takes place over an extended period of time (perhaps months) before the architecture team presents the results of the analysis to the original stakeholders.

The remainder of this report describes the QAW approach. Section 2 describes the sequence of activities in a QAW. These activities include both facilitated meetings and analyses between the meetings. Section 3 describes how QAWs implemented for different organizations vary. Section 4 captures lessons learned during the initial QAWs and summarizes our experience with the QAW approach. The appendix provides an example scenario, its refinement, and a test case derived from the refined scenario.

1 Architecture Tradeoff Analysis Method and ATAM are service marks of Carnegie Mellon University.

2 Process Outline

The QAW process is shown in Figure 1. The process can be organized into four distinct groups of activities: (1) scenario generation, prioritization, and refinement; (2) test case development; (3) analysis of test cases against the architecture; and (4) presentation of the results. These are the four gray ovals in the figure. The first and last segments of the process occur in facilitated one-day meetings. The middle segments take place offline and can continue over an extended period of time. Depending on the application context, the specific roles and responsibilities of the participants in the various activities can be customized, as described in Section 3.

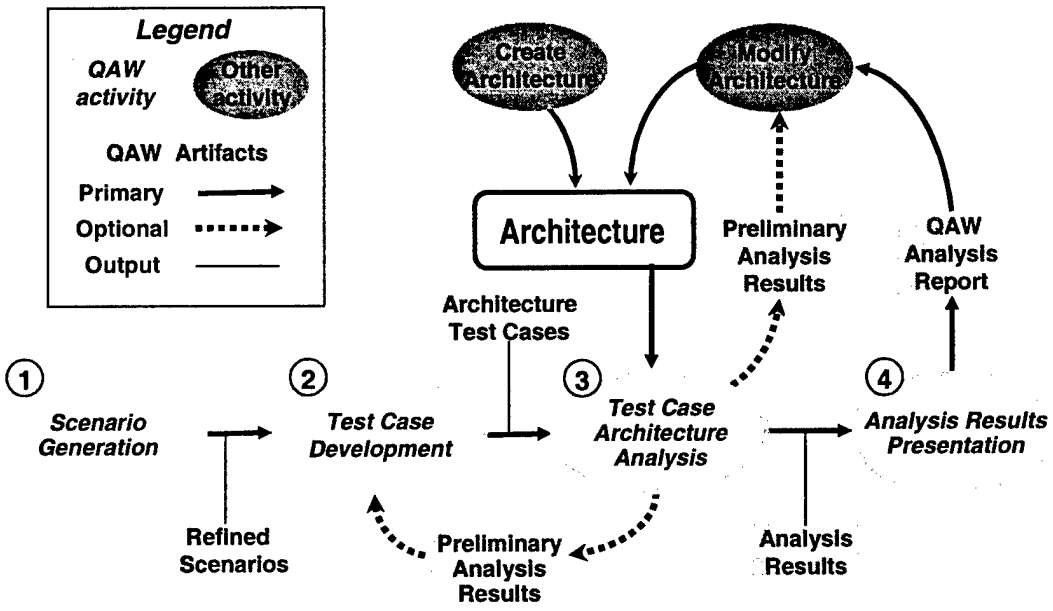


Figure 1: The QAW Process

The process is iterative in that the test case analyses might lead to the development of additional test cases or to architectural modifications. Architectural modifications might prompt additional test case analyses, and so forth. There is a further iteration, not shown in the figure, in which scenarios and test cases are developed in batches, resulting in a continuous cycle of analyses and architectural modifications. Not included in the figure are the planning activities, such as planning the scenario generation meeting. This activity might take a number of interchanges to schedule the meetings, choice of facilities and participants, and so forth.

2.1 Scenario Generation

The first activity in the QAW process is to generate, prioritize, and refine scenarios. In the QAW, a scenario is a statement about some anticipated or potential use or behavior of the system. Scenarios capture stakeholders' concerns about how the system will do its job.

The scenarios are generated during a brainstorming meeting attended by facilitators, stakeholders, and the architecture team. The stakeholders are provided in advance with a workbook containing descriptions of several quality attributes, sample questions to assist in generating scenarios, example scenarios for each quality attribute, and examples of refined scenarios. A typical agenda for this meeting is shown in Table 1.

Time	Activity
8:30 a.m.	Start Welcome and introductions Quality Attribute Workshops overview Business drivers System architecture
10:00 a.m.	Scenario generation and prioritization
noon	Lunch break
1:00 p.m.	Scenario refinement
3:00 p.m.	Wrap up Review scenarios Action items
4:00 p.m.	End

Table 1: Agenda for Scenario Generation

The meeting starts with a facilitation team presenting an overview of the QAW process including QAW activities (the four numbered ovals in Figure 1) and their results. A customer representative then describes the system's mission or business drivers including business context for the system, architectural drivers (quality attributes that "shape" the architecture), critical requirements (quality attributes most central to the system's success), and so forth. The presentation of the business drivers is followed by an overview of the system architecture. The overview addresses technical constraints such as an operating system, hardware, or middleware prescribed for use, other systems with which the system will interact, and planned architectural approaches to address quality attribute requirements. These three presentations set the context for the activities to follow.

During the scenario generation segment of the meeting, individual stakeholders, in a round-robin fashion, propose scenarios or ask questions about the way in which the system will

respond to various situations. Scenarios are used to represent stakeholders' interests and quality attribute requirements to exercise the system against current and future situations.

The quality attributes are characterized by stimuli, responses, and the architectural decisions that link them. Stimuli and responses are the activities (operational or developmental) that exercise the system and the observable effects, respectively. A good scenario clearly states which stimulus causes it and which responses are of interest.

There are several types of scenarios:

1. **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.
2. **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 the operator's console).
3. **Exploratory scenarios** involve extreme changes to the system that may be unanticipated and that may occur in undesirable situations. Exploratory scenarios are used to explore the boundaries of the architecture (e.g., message traffic grows 100 times, requiring the replacement of the operating system).

The distinction between growth and exploratory scenarios is system or situation dependent. What might be anticipated growth in a business application might be a disaster in a deep space probe (e.g., 20% growth in message storage per year).[2] For example, a use-case scenario might be: "Remote user requests a database report via the Web during peak period and receives it within 5 seconds." A growth scenario might be: "Add a new data server to reduce latency in scenario 1 to 2.5 seconds within 1 person-week," and a exploratory scenario might be: "Half of the servers go down during normal operation without affecting overall system availability." Scenarios should be as specific as possible, identifying stimuli, responses, and environment.

Stakeholders are encouraged to generate as many scenarios as possible to represent a wide range of concerns. Questions to clarify scenarios are allowed. However, challenges or arguments about the importance of a scenario are discouraged at this point because the subsequent scenario prioritization takes care of this issue. Scenario generation continues until either the allotted time is exhausted or the stakeholders generate no additional scenarios. This activity takes between one and two hours. Typically, 30 to 40 scenarios are generated.

Only a small number of scenarios can be refined during a one-day meeting. Thus, stakeholders must prioritize the scenarios generated previously by using a voting process. Typically, they are given multiple votes (usually about 30% of the total number of scenarios). They can dis-

2 There are no clear rules other than stakeholder consensus that some scenarios are likely (desirable or otherwise) and others are unlikely. (If the unlikely ones occur, it would be useful to understand their consequences.)

tribute those votes across multiple scenarios or apply all votes to a single scenario. Before the vote, the stakeholders are free to spend some time perusing the scenarios, discussing them in small groups, and combining scenarios that express equivalent concerns. This activity typically takes 30 minutes.

Next, the stakeholders refine the top three or four scenarios to provide a better understanding of their context and detail. To guide the refinement, we use the template shown in Table 2. This template identifies the types of details that usually emerge from the refinement.

Section	Content
Reference Scenario(s)	The scenario that is being refined is listed here. If it is a consolidated scenario, the combined scenarios are listed here in their original form.
Organizations	Organizations involved in or affected by the scenario(s)
Actors/Participants	Individuals involved in or affected by the scenario(s)
Quality Attributes	Quality attributes involved in or affected by the scenario(s)
Context	A description of additional details such as the environment in which the scenario takes place, and the sequence, frequency, and duration of events
Questions	Specific questions that the stakeholders would like to ask the architect and the designers of the system. Typically one or more questions are included for each quality attribute identified above, for example • How will the system prevent unauthorized users from accessing the system? • How will the system store one year of information online? • How will the system respond to user requests within 10 seconds during peak time?

Table 2: Template for Scenario Refinement

For example, a scenario like "a communication relay node failed," does not really capture the consequences or implications of the failure. The refinement activity elicits further details such as the expected operational consequences, the system assets involved, the end users involved, the potential affects of the scenario on system operation, and the exceptional circumstances that may arise. For the above scenario, such details would include which facility or node detects failure, what is the expected automated response to failure (if any), what is the expected manual intervention, which capabilities will be degraded during the outage, and the expected actions taken to return the relay to service. This activity typically takes two hours.

The result of this meeting is a prioritized list of scenarios and the refined description of the top three or four scenarios on that list.

2.2 Test Case Development

The objective of this activity is to transform each refined scenario from a statement and list of organizations, participants, quality attributes and questions into a well-documented test case. The test cases may add assumptions and clarifications to the context, add or rephrase questions, group the questions by topic, and so forth.

Who is responsible for developing test cases depends on the situation. In Section 3, we describe how the method has been applied and who carried out the task (e.g., sponsor/acquirer or development team).

A test case has a context section outlining the important aspects of the case, an issues and questions section stating the stakeholders' concerns, and a utility tree that summarizes the issues and questions.

2.2.1 Context

The context section describes the mission or activity, the assets involved, the geographical region, the operational setting, the players, and so forth. It includes a description over a time interval allowing the operation to play out. In a test case involving a communication relay node failure, for example, the test case may define

- the activities at the time of failure
- what happens immediately after, when the system is reacting to the failure
- degraded operation during the interval when repair is underway
- restoring the system to normal operation

2.2.2 Quality Attribute Questions

This section defines the issues implied by the context and proposes questions that connect these issues to quality attributes. To help focus the analysis, each question is tagged by the quality attributes it addresses, such as performance, security, availability, or modifiability, and with a specific issue of concern within that quality attribute. For example, the issue may be: "how failure is detected." In this case, the questions might be: "What subsystem detects the failure?," "How long does it take to detect the failure?," and "What happens during this interval?" There should be between five and ten issues and questions for each context, covering various quality attributes. In the failure context, it is relatively straightforward to discuss

- performance issues, such as time to detect failure
- availability issues, such as degraded mode of services provided
- interoperability issues, such as how an alternative service might be introduced

- security issues, such as the impact on data integrity

Each test case should cover a few quality attributes, as well as assets, geography, missions, and activities. A checklist could be developed to aid in preparing the test cases to assure coverage across these topics. In addition, with each question, there might be guidance about what is expected in the response to the question. For example, a question concerning the latency of messages between a system's components might also request that the answer be documented by a sequence diagram annotated with time delays along each step in the diagram.

2.2.3 Utility Tree

Each test case has a utility tree that links the questions and issues included in the test case to the important quality attributes. (The template for such a tree is shown in Table 3.) The utility tree provides a visual summary of the quality attributes of importance and the specific issues and questions that pertain to the attributes.

Root	Quality Attribute	Specific Attribute Issue	Question
Utility	performance	... time to send a message...	... frequency of messages?
			... latency of messages?
			... priority of messages?
		... time to perform a critical function...	*... question?*
			... question?
		... time to restart a service...	*... question?*
			... question?
	attribute	*... issue...*	*... question?*
			... question?
	attribute	*... issue...*	*... question?*
			... question?

Table 3: Template for Utility Tree

2.3 Test Case Architecture Analysis

This activity requires the test cases and system architecture documents needed for analyzing the architecture for each test case. Since there are no generally accepted, industry-wide standards for describing a system architecture, the analyses are often constrained by the available documentation. In the case of systems documented using the Command, Control, Communications, Computer, Intelligence, Surveillance and Reconnaissance (C4ISR) Architecture Framework [AWG 97], different products or collections of products will differ in their relative value for analyzing quality-attribute-specific test cases [Barbacci 99].

Depending on the quality attributes of concern, C4ISR products might have to be complemented with additional documents, to address quality attribute concerns that are under-represented in the framework products. In the previous example, when "a communication relay node failed," the analyst might build a sequence diagram showing the behavior of the major system components and the sequences of messages passing between them. For example, the sequence diagram might include an architectural component labeled "traffic manager," which sends messages to other defined components to divert their message traffic, discusses the level of degradation of the network traffic during the failure, and also includes a "load shedding" component.

A typical sequence of steps for conducting this activity would include the following:

1. Review the capabilities of the assets in the test case context and determine how the system will react to the situation.
2. Make and document any assumptions necessary to proceed with the analysis.
3. Determine which architectural views (for example, operational, system, technical, process, behavioral, structural) can best describe how the system will address the issues and their associated questions.
4. Perform the architecture analysis for the selected test cases.
5. If necessary, refine the architecture to help answer the questions.
6. Document the answers as specifically as possible.

The analysis of the architecture for a specific test case would clarify or confirm specific quality attribute requirements and might identify concerns that would drive the development of the software architecture. Some of the test cases could later be used as "seed scenarios" in an ATAM evaluation (e.g., to check if a concern identified during the test case analysis was addressed by the software architecture). The results of analyzing a test case should be documented with specific architectural decisions, quality attribute requirements, and rationale.

2.4 Results Presentation

The results presentation is the final activity in the QAW process. It is a one- or two-day meeting attended by facilitators, stakeholders, and the architecture team. It provides an opportunity for the architecture team to present the results of its analysis and to demonstrate that the proposed architecture is able to handle the cases correctly. In advance, the stakeholders are given a workbook containing a summary of the QAW process, the collection of test cases, and examples of test case analyses. A typical agenda for this meeting is shown in Table 4.

Time	Activity
8:30 a.m.	Start Welcome and introductions QAWs overview Business drivers Architecture plans Purpose of meeting and expected outcomes
9:30 a.m.	Presentation of analysis of test cases #1 and #2
11:45 a.m.	Review of first two test cases and tailoring of afternoon objectives
noon	Lunch
1:00 p.m.	Presentation of analysis of two or more additional test cases
4:00 p.m.	Wrap-up Review of test cases Summary Review future activities
5:00 p.m.	End

Table 4: Agenda for Presentation of Results

The beginning of the meeting recapitulates the QAW process, business drivers, and architectural plans that were presented during the scenario generation meeting. Since the presentation of results might take place a few months after the first meeting, this review serves as a reminder to participants who were involved in the scenario generation and to introduce the concepts to new participants. During the presentation, the facilitators and stakeholders should probe architectural approaches from the point of view of specific quality attribute requirements. In addition to the specific questions included in the test cases, the participants might generate additional quality-attribute-specific questions for high-priority quality attribute requirements. The conclusions, recommendations, and action items resulting from the presentation must be captured in a short report and distributed to the participants.

3 Variants in QAWs

The previous sections described the QAW method in generic terms. The actual application of the method can be tailored to the needs of a specific acquisition strategy and might include incorporating specific documents or sections of documents into the request for proposals (RFP) or contract[3] [Bergey 01]. In this section, we describe how the QAW activities were tailored to the needs of the acquirer.

3.1 Application Before Acquisition

In one application, the QAW method was used in a pre-competitive phase for a large system. Stakeholders involved laboratories and facilities with different missions and requirements. The architecture team (from various facilities) was building the architecture for a shared communications system before awarding a contract to a developer.

- Stakeholders from different facilities held separate meetings to generate, prioritize, and refine scenarios.
- The architecture team turned these refined scenarios into test cases and analyzed the proposed architecture against them.
- The architecture team then presented the results of the analysis, first to a review team, and later, to the original stakeholders.

In yet another pre-competitive case, the customer only wished to generate, prioritize, and refine scenarios as an exercise for the stakeholders. Since the architectural plans were still uncertain, the customer intended to develop and analyze test cases later, after a draft architecture had become available.

3.2 Application During Solicitation and Proposal Evaluation Phases

Figure 2 illustrates a common acquisition strategy. Starting with an initial request for proposals, the acquisition organization evaluates proposals from multiple contractors and chooses one to develop the system.

3 Bergey, J. & Wood, W. *Use of the Quality Attribute Workshop (QAW) in Source Selection for a DoD System Acquisition: A Case Study.* Pittsburgh, PA: Software Engineering Institute, to be published.

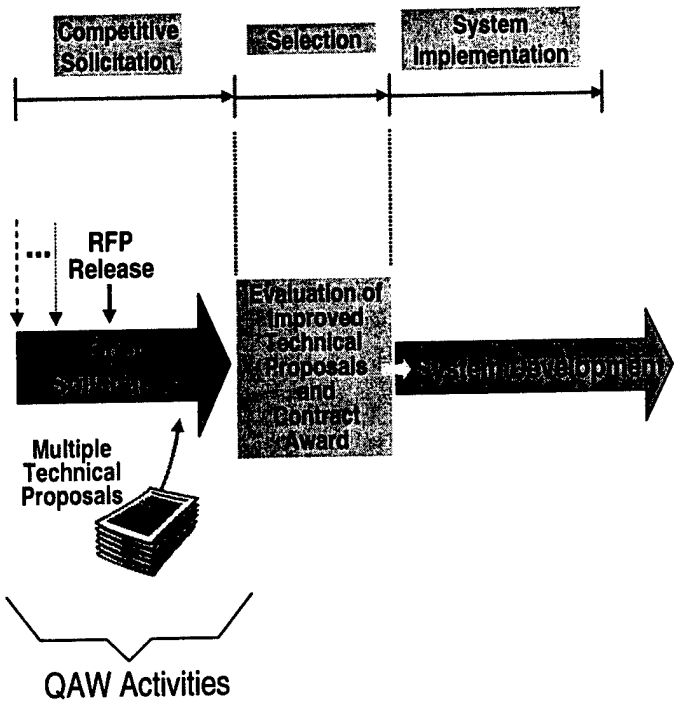


Figure 2: Common Acquisition Strategy

In one application of the QAW method, the system to be developed involved many users in different locations, who access a variety of databases and have different types of responsibilities (e.g., technicians, supervisors, analysts at headquarters). Over time, systems used in different locations had evolved on their own, and the acquirer wanted to integrate all of these legacy systems into one, providing a more consistent but customizable view of the data. To avoid a "big-bang" effect, the acquirer identified three phases in an incremental development plan: initial, intermediate, and final releases.

The acquirer decided to select a contractor based on the contractor's analyses of test cases against its proposed architecture. The QAW activities took place during the competitive selection, and were customized as follows:

- Before the competitive solicitation phase, scenario generation meetings were conducted at three different facilities. These were representative of groups of facilities with similar needs and responsibilities. During the scenario refinement, questions proposed by the stakeholders had to be specific to the system release to which they applied.
- Early in the competitive solicitation phase and prior to the release of the RFP, the acquirer conducted bidders' conferences to inform potential bidders about the need for conducting architecture analysis.

- The acquirer developed several test cases for each type of user, drafted sections of the RFP to incorporate architecture analysis requirements, and included the test cases as government-furnished items (GFIs) in the RFP proper.
- Currently, we are in the middle of the solicitation phase. As part of their proposals, the bidders are expected to conduct an architecture analysis of the RFP test cases, present their results to the acquirer, and write reports consisting of the results of their analysis, their response to requests for clarification, risk-mitigation plans for the risks identified during the presentation, and any new or revised architecture representations.

3.3 Application During a Competitive Fly-off

Figure 3 illustrates a *rolling down select*, a different acquisition strategy. Starting with an initial request for proposals, the acquisition organization awards contracts to a small number of contractors to conduct a competitive fly-off. At the end of the phase, the contractors submit updated technical proposals, including additional details, and the acquirer makes a final down select.

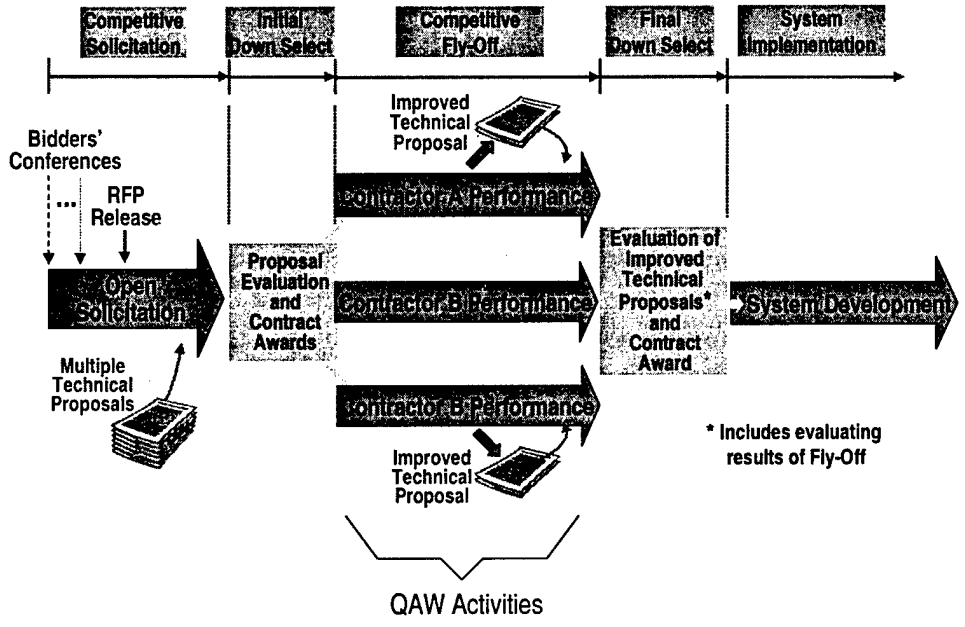


Figure 3: Acquisition Strategy Using Competitive Fly-Off

In this application, the QAW method was used during the Competitive Fly-Off phase (with three industry teams competing) of the acquisition of a large-scale C4ISR system. In this case, the QAW process was customized as follows:

- The scenario generation meetings were conducted with each contractor separately. As a result of these meetings, participants gained an understanding of the process, a list of prioritized scenarios, and a set of refined high-priority scenarios.
- A government technical assessment team (TAT) used these scenarios to develop a number of test cases. Changes were made to hide the identity of the teams and extend the coverage of the scenarios over a set of assets, missions, and geographical regions. An example was developed to make the process more understandable, and copies were distributed to all industry teams.
- The contractors performed the analysis and presented the results in a *dry-run* presentation. There was a large variation in the presentations for these meetings, ranging from performing only one test case in great detail, to performing all test cases in insufficient detail. Each contractor was then informed of how well it did and how it could improve its analysis. The contractors then completed the analysis in a final presentation of the results, allowing them to correct any flaws.

4 Experience and Conclusions

As a result of conducting a number of QAWs, we learned the lessons described below. Most of these lessons were integrated into the method incrementally.

4.1 Lessons Learned from Scenario Generation and Refinement

The scenario generation meeting is a useful communications forum to familiarize stakeholders with the activities and requirements of other stakeholders. In several cases, the developers were unaware of requirements brought up by those with responsibility for maintenance, operations, or acquisition. In one case, potential critics of the project became advocates by virtue of seeing their concerns addressed through the QAW process. We also learned that the facilitation team has to be flexible and adapt to the needs of the customer, as the following observations indicate:

- The process of generating scenarios in a brainstorming session is usually inclusive, but the process for refining the high-priority scenarios might not be. Some stakeholders might feel left out of the refining effort if other, more vocal stakeholders dominate the process. It is the responsibility of the facilitators to make sure that everyone can contribute. The template describing specific details to be identified during the scenario refinement was a great improvement over the initial refinement exercises, because it kept the stakeholders focused on the task at hand and avoided diversions.
- The approach relies on identifying the right stakeholders and asking them to do some preparatory reading and attend the meeting for a day. In one case, the task of inviting these stakeholders fell on the architecture team, which created some awkward situations. The hosts of the meeting need a way of attracting the right people to the meeting. This could include invitations explaining the advantages of participating, and recommendations from upper management to cause interest in attending.
- The scenarios generated in a meeting can be checked against the requirements in two ways. First, unrefined scenarios can be sensitivity-checked against system requirements. Second, refined scenarios can lead to a better understanding of some requirements. Undocumented requirements can be discovered by both means.
- The scenarios generated in a meeting can be checked against the expected evolution of the system over time. In projects planning a sequence of releases, the scenarios should specify

the release to which they apply, ensuring that the projected deployment of assets and capabilities match the scenarios and test cases.

- Some of the scenarios or questions generated during the refinement might not be focused on quality attributes. This is usually an indicator that the issues involved are "hot buttons" for some of the stakeholders. Although we normally try to focus the scenarios on quality attributes, the underlying issues could be important, and on occasion, we have allowed the scenarios and questions to stand.

4.2 Lessons Learned from Developing the Test Cases

Building the test cases from the refined scenarios takes time and effort.

- In one case, the QAW facilitators did not extract sufficient information during the refinement session to build the test cases, and the facilitators had to organize additional meetings with domain experts to better define the context and quality attribute questions. An unintended consequence was that the resulting test case context was far more detailed than if it had been generated during the scenario refinement session. As a result, only portions of the larger test case context were relevant to the test case questions. We learned that having an extremely detailed test case context is not worthwhile. It takes too long to develop, may be hard to understand, and does not lead to focused questions. A test case should not be more than a few sentences.
- Since the software architecture is not yet in place, the questions and expected responses should not force design decisions on the development team. Hence, the questions must be quite general, and the expected responses may suggest architectural representations (for example, "what is the availability of this capability?") but not design solutions (for example, "use triple modular redundancy for high availability").

4.3 Lessons Learned from Analyzing the Architecture Using Test Cases

During the presentation of the test case architecture analyses, questions from the stakeholders often lead to discussions of alternatives and quick lightweight analyses, on the spot. In some cases, the analysis might reveal flaws in the architectures and cause the architecture team to change the design.

Like in the scenario generation meeting, participants are provided with a handbook before the meeting. The handbook includes the test cases and provides a test case analysis example so the participants know what to expect at the meeting. The following observations are derived from conducting a number of QAWs:

- In one case, the initial example given in the participants' handbook was too general. This reduced the level of "buy-in" from participants. We corrected this by developing another example with the right level of detail.
- The test cases generated by the QAW process often extend the existing system requirements. In one case, the new requirements seemed to challenge the requirements elicitation effort and raised concerns of the architecture team. A typical comment was "the system wasn't meant to do that." Some judgment must be made as to which test cases can be handled and at which phase of system deployment. While this can lead to extended arguments within the team, it is a useful exercise, since these concerns must be resolved eventually.
- In another case, the stakeholders were concerned because the process only analyzed a few test cases out of a large collection of scenarios. They wanted to know what was to be done with the remaining scenarios. This issue should be resolved before the scenario generation meeting. One approach is to analyze the architecture incrementally against an ever-expanding set of test cases and, if necessary, adjust the architecture in each increment. However, this approach is constrained by budgets, expert availability, and participants' schedules.

In conclusion, the process for conducting QAWs is solidifying as we continue to hold them with additional customers, in different application domains, and at different levels of detail. The approach looks promising; the concept of checking for flaws in the requirements before committing to development should reduce rework in building the system.

4.4 Lessons Learned from the Results Presentation

In some applications of the QAW, we have conducted the results presentations in two phases: first as a dry-run or rehearsal, and then as a full-scale presentation.

- A dry-run presentation should be conducted when the architecture team making the presentation is unsure about the level of detail required; the precision expected from its answers to the test case questions; how to incorporate other analysis results (for example, reliability, availability and maintainability analysis, or network loading analysis); or what additional architectural documents might be needed.
- The full-scale presentation takes place after "cleaning up" the results of the dry-run presentation. Concerns that arise in the full-scale presentation have to be addressed as potential threats to the architecture.

Appendix A Example Scenario Refinement

A test case usually starts with the details obtained from the scenario refinement, as illustrated in Table 5.

<System/Organization Title> - Scenario Refinement	
Reference Scenario(s)	"Mars orbital communications relay satellite fails."
Organizations	Authorities in multiple organizations and control centers
Actors/Participants	Flight director, mission director
Quality Attributes	Performance (P) and Availability (A)
Context	One of three aero-stationary satellites around Mars fails. Mars surface elements and Mars satellites know that it failed and report the failure to the Earth control element. Traffic rerouting is to be performed and network reconfiguration dictated by flight director, perhaps postponed to limit the possibility of further failure. Service assessments are done at the control center (within two days) using a well-defined decision-making process, leading to the mission director (as the final authority). Multiple missions will be running simultaneously. Currently, we are doing two, but coordination is complex.
Questions	• How long does it take to detect the failure? • Is there a way to send information to Earth for analysis? • Can a crew in the space station help in the transmission? • How long does it take to reconfigure the system? • What redundancy is available? • Can the crew participate in the repair? • Can the customer participate in the notification (e.g., "Please send a message to the other satellite")? • Is there a way for customers to simplify their procedures to handle more missions?

Table 5: Example Scenario Refinement

Appendix B Example Test Case

B.1 Test Case Context and Activities

Humans and robotic missions are present in the Mars surface when one of three aero-stationary satellites has a power amplifier failure. The primary communications payload is disabled for long-haul functions, but the proximity link to other relay satellites and customers on orbit and on the surface still works. Secondary Telemetry and Tele-Command (TTC) for spacecraft health is still working for direct-to-Earth with a low data rate. The remaining two satellites are fully functional. Communications with the crew has been interrupted. The crew is not in an emergency situation at the time of the failure, but reconnection is needed "promptly." The crew on the surface is concentrated in one area, and the other missions in the Mars vicinity are in normal operations or non-emergencies, or are performing mission-critical events. The event occurs late in the development of the communications network, so the system is well developed.

B.2 Quality Attribute Questions

1. **Issue:** Mission safety requires consistent and frequent communications between the crew and Earth. (P, A)
   Questions:
   a. How long does it take to detect the failure?
   b. How long does it take to reconfigure the system to minimize the time the crew is without communication?
2. **Issue:** System operation will be degraded. (P, A)
   Questions:
   a. Is there a way for customers to simplify their procedures so they can handle a larger number of missions with less trouble than coordinating two as they do now?
   b. What redundancy is required?
   c. Is there a way to send information about the degraded satellite back to Earth for analysis?

3. **Issue:** System recovery (P, A)
   Questions:

   a. Can the crew participate in the repair?

   b. Is there any expectation for a human interface between Mars and the Earth (e.g., crew in space station)?

   c. Can the customer participate in the notification (e.g., "Please send a message to the other satellite")?

B.3 Utility Tree

Root	Quality Attribute	Specific Attribute Issue	Question
Utility	Performance	... of communications...	(1b) How long to reconfigure?
		... degraded operation...	(2a) Can decisions be simplified?
			(2c) How is information sent back?
	Availability	... mission safety...	(1a) How long to detect the failure?
		... redundancy...	(2b) What redundancy is required?
		... recovery...	(3a) Can the crew help?
			(3b) Can space station help?
			(3c) Can other assets help?

Table 6: Example Utility Tree

Appendix C Example Results of the Test Case Analysis

The analysis is based on applying a set of test cases to a system architecture. These test cases include questions and concerns elicited from stakeholders associated with the system. In our example, there are two important issues that affect availability and crew safety: satellite locations and monitoring the health of the satellites.

C.1 Satellite Locations

The initial architecture has three operational aero-stationary satellites. Each satellite has visibility over a fixed area of Mars. When the satellite fails, its area of responsibility is no longer in communication with Earth and this area contains the crew.

The risk of a satellite failure is having inadequate communications with the crew on Mars. This is a serious problem but can be alleviated by a number of architectural approaches:

- Move one or two of the other stationary satellites to provide communications with the crew. This will degrade communications between the relocated satellites and the assets in their original area of responsibility. This solution constitutes a tradeoff between the quality of communications in different areas of responsibility.
- Place one or more in-orbit spare satellites. A spare can be moved to the location of the failed satellite and take over its area of responsibility. This solution constitutes a tradeoff between cost (additional satellites) and speed of repair (the more spares, the quicker one could be moved to the desired location).
- Place "feeder antennas" on the surface of Mars to relay communications from the crew to (eventually) another satellite (i.e., rerouting the traffic in case of failure). This solution is a tradeoff between cost (the antennas) and the quality of the communications (the volume and latency of messages).
- Place the satellites in a "slow drift" orbit, such that the loss of a satellite will not cause a complete communications failure. There will be times when one or more of the other satellites will be in sight and times when no satellite is in sight. This is probably the least disruptive solution, provided the periodic (but predictable) loss of communications can be

tolerated. In case of a satellite failure, the crew will have communications when the next satellite drifts over the area.

C.2 Health of the Satellites

Monitoring the health of the satellites will improve the availability of the communications with the crew. A health monitor could help predict imminent failures, detect recent failures, and plan corrective actions.

Without a health-monitoring system, there is a risk that satellite failures could go unnoticed until they are needed to provide communications to the crew on the surface. There is also an additional risk of having extended downtimes due to the long transit time from Earth (e.g., sending a replacement satellite could take months). There are a number of alternatives, which are not mutually exclusive:

- The satellites could exchange periodic health messages among them (e.g., pushing "I am here!" messages and pulling "Are you there?" messages) and inform the ground stations of their health states. This is a tradeoff with performance because of the additional message traffic.
- The satellites could run self-tests and inform the ground stations whenever a problem is detected. This is a tradeoff with performance and probably cost (i.e., extra components to conduct the self-tests).
- The mission customers could notice degradation in communications and alert the communications system operators. This is a tradeoff with usability (e.g., how to avoid false alarms).

References

[AWG 97] Architectures Working Group. *C4ISR Architecture Framework Version 2.0* [online]. <http:// www.c3i.osd.mil/org/cio/i3 /AWG_Digital_Library/pdfdocs/fw.pdf> (1997).

[AWG 98] Architectures Working Group. *Levels of Information Systems Interoperability (LISI)* [online]. <http://www.c3i.osd.mil/org/cio /i3/AWG_Digital_Library/pdfdocs/lisi.pdf> (1998).

[Bass 98] Bass, L.; Clements, P.; & Kazman, R. *Software Architecture in Practice*. Reading, MA: Addison Wesley Publishing Company, 1998.

[Barbacci 95] Barbacci, M.; Klein, M.; Longstaff,T.; & Weinstock, C. *Quality Attributes* (CMU/SEI-95-TR-021, ADA307888). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1995. <http://www.sei.cmu.edu/publications/documents /95.reports/95.tr.021.html>.

[Barbacci 96] Barbacci, M.; Klein, M.; & Weinstock, C. *Principles for Evaluating the Quality Attributes of a Software Architecture* (CMU/SEI-96-TR-036, ADA324233). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1996. <http://www.sei.cmu.edu/publications/documents/96.reports /96.tr.036.html>.

[Barbacci 97] Barbacci, M.; Carriere, S.; Feiler, P.; Kazman, R.; Klein, M.; Lipson, H.; Longstaff, T.; & Weinstock, C. *Steps in an Architecture Tradeoff Analysis Method: Quality Attribute Models and Analysis* (CMU/SEI-97-TR-029, ADA343692). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1998. <http://www.sei.cmu.edu/publications/documents/97.reports /97tr029/97tr029abstract.html>.

[Barbacci 99] Barbacci, M. & Wood, W. *Architecture Tradeoff Analyses of C4ISR Products* (CMU/SEI-99-TR-014, ADA366784). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1999. < http://www.sei.cmu.edu/publications/documents /99.reports/99tr014/99tr014abstract.html>.

[Barbacci 01] Barbacci, M.; Ellison, R.; Stafford, J.; Weinstock, C.; & Wood, W. *Quality Attribute Workshops* (CMU/SEI-2001-TR-010, ADA395197) Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2001. <http://www.sei.cmu.edu /publications/documents/01.reports/01tr010.html>.

[Bergey 01] Bergey, J. & Fisher, M. *Use of the Architecture Tradeoff Analysis Method (ATAM) in the Acquisition of Software-Intensive Systems* (CMU/SEI-2001-TN-009, ADA396096). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2001. <http://www.sei.cmu.edu/publications/documents /01.reports/01tn009.html>.

[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.

[Kazman 00] Kazman, R.; Klein, M.; & Clements, P. *ATAM: Method for Architecture Evaluation* (CMU/SEI-2000-TR-004, ADA382629). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000. <http://www.sei.cmu.edu/publications /documents/00.reports/00tr004.html>.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 2002	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Quality Attribute Workshops, 2nd Edition	C — F19628-00-C-0003

6. AUTHOR(S)

Mario R. Barbacci, Robert Ellison, Anthony J. Lattanze, Judith A. Stafford, Charles B. Weinstock, William G. Wood

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

8. PERFORMING ORGANIZATION REPORT NUMBER

CMU/SEI-2002-TR-019

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

HQ ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2116

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

ESC-TR-2002-019

11. SUPPLEMENTARY NOTES

12.a DISTRIBUTION/AVAILABILITY STATEMENT	12.b DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (maximum 200 words)

Quality attribute workshops (QAWs) provide a method for analyzing a system's architecture against a number of critical quality attributes, such as availability, performance, security, interoperability, and modifiability, that are derived from mission or business goals. The QAW does not assume the existence of a software architecture. It was developed to complement the Architecture Tradeoff Analysis Method℠ (ATAM℠) in response to customer requests for a method to identify important quality attributes and clarify system requirements *before* there is a software architecture to which the ATAM could be applied. The analysis is based on applying a set of test cases to a system architecture. These test cases include questions and concerns elicited from stakeholders associated with the system. The process of building the test cases allows stakeholders to communicate among themselves, thereby exposing assumptions that may not have surfaced during requirements elicitation. Our experience to date includes multiple QAWs that were held with four different U.S. government acquisition programs.

This is the second edition of a technical report describing QAWs. This report clarifies the context in which a QAW is applicable, provides a rationale for developing the process and describes it in detail, and concludes with a list of lessons learned and a discussion of how these lessons have helped evolve the process to its current state.

14. SUBJECT TERMS	15. NUMBER OF PAGES
quality attributes, software system acquisition, software architecture, evaluation, attribute requirements	38
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL