Quality Attribute Workshops (QAWs), Third Edition

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

August 2003

TECHNICAL REPORT
CMU/SEI-2003-TR-016
ESC-TR-2003-016

Carnegie Mellon
Software Engineering Institute

Pittsburgh, PA 15213-3890

Quality Attribute Workshops (QAWs), Third Edition

CMU/SEI-2003-TR-016
ESC-TR-2003-016

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

August 2003

Architecture Tradeoff Analysis Initiative

This report was prepared for the

SEI Joint Program Office
HQ ESC/DIB
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

Christos Scondras
Chief of Programs, XPK

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

The Quality Attribute Workshop (QAW) is a facilitated method that engages system stakeholders early in the life cycle to discover the driving quality attributes of a software-intensive system. The QAW was developed to complement the Architecture Tradeoff Analysis Method[SM] (ATAM[SM]) and provides a way to identify important quality attributes and clarify system requirements *before* the software architecture has been created.

This is the third edition of a technical report describing the QAW. We have narrowed the scope of a QAW to the creation of prioritized and refined scenarios. This report describes the newly revised QAW and describes potential uses of the refined scenarios generated during it.

1 Introduction

In software-intensive systems, the achievement of qualities—such as performance, availability, security, and modifiability—is dependent on the software architecture. In addition, quality attributes of large systems can be highly limited by a system's requirements and constraints. Thus, it is in our best interest to try to determine as early as possible whether the system will have the desired qualities. Quality requirements should be described concretely before an architecture is developed.

We distinguish system architecture from software architecture according to the following two definitions:

> *system architecture: the fundamental and unifying system structure defined in terms of system elements, interfaces, processes, constraints, and behaviors [INCOSE 96]*

> *software architecture: the structure or structures of the system, which comprise software elements, the externally visible properties of those elements and the relationships among them [Bass 03]*

Development of software-intensive systems begins with a description of the system's operation and high-level functional requirements, and any constraints on the system, such as legacy or new systems. From these items, the architect derives a system architecture and a software architecture that can then be used to drive detailed design and implementation. (See Figure 1.) The process of creating those architectures is often unstructured.[1] Quality attributes could be missing from the requirements document, and even if addressed adequately, they are often vaguely understood and weakly articulated.

The Quality Attribute Workshop (QAW) is a facilitated method that engages system stakeholders early in the system development life cycle to discover the driving quality attributes of a software-intensive system. The QAW is system-centric and stakeholder focused; it is used before the software architecture has been created. The QAW provides an opportunity to gather stakeholders together to provide input about their needs and expectations with respect to key quality attributes that are of particular concern to them.

1. The problems arise from software, not system engineering practices. In fact, system engineers usually build reliability, availability, and maintainability (RAM) models to evaluate quality attributes, and build simulation models to test communication loads and load sharing.

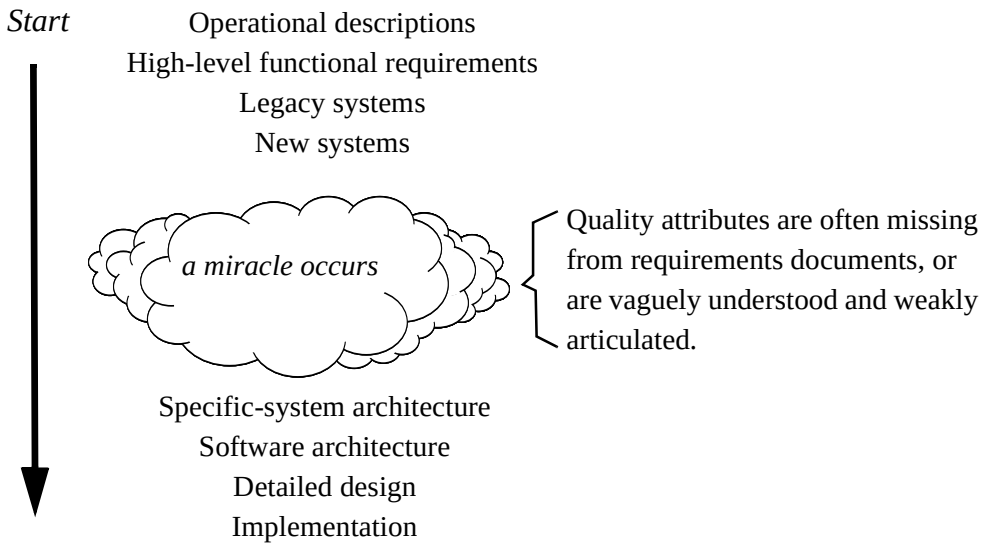

Figure 1: Traditional System Development

Both the system and software architectures are key to realizing quality attribute requirements in the implementation. Although an architecture cannot guarantee that an implementation will meet its quality attribute goals, the wrong architecture will surely spell disaster. As an example, consider security. It is difficult, maybe even impossible, to add effective security to a system as an afterthought. Components as well as communication mechanisms and paths must be designed or selected early in the life cycle to satisfy security requirements. The critical quality attributes must be well understood and articulated early in the development of a system, so the architect can design an architecture that will satisfy them. The QAW is one way to discover, document, and prioritize a system's quality attributes early in its life cycle.

It is important to point out that we do not aim at an absolute measure of quality; rather our purpose is to identify scenarios from the point of view of a diverse group of stakeholders (e.g., architects, developers, users, sponsors). These scenarios can then be used by the system engineers to analyze the system's architecture and identify concerns (e.g., inadequate performance, successful denial-of-service attacks) and possible mitigation strategies (e.g., prototyping, modeling, simulation).

Section 2 describes the motivation for holding QAWs, Section 3 describes the QAW method, and Section 4 describes its benefits. Section 5 concludes this report with some references to experiences using the method. Appendix A provides a template for conducting a QAW, and Appendix B provides an example of scenario refinement.

2 Motivation

Functionality describes the capabilities of the system, and every system has an architecture, regardless of whether it's monolithic. If the only criterion for acceptance were getting the right answer, we would not need to worry about architecture; unstructured, monolithic systems would suffice. However, our concerns often go beyond getting functionally correct answers. A system that is fit for a purpose should meet both functional and quality attribute requirements. If performance goals aren't met, the right answer may not be delivered on time; if the system is too fragile (even though it produces the right answer), it won't be available when needed; if it is easily compromised, it will not be secure enough and its answers won't be protected; if making a modification to one part of the system affects a large percentage of the rest of it, the system won't be easily modifiable. The truth is that qualities like interoperability, modifiability, and portability matter—in some cases as much as functionality does.

The achievement of quality attributes is critical to the success of a system, and a variety of qualitative and quantitative techniques are used for analyzing specific quality attributes [Barbacci 95], [Barbacci 96], [Rushby 93].[2] However, quality attribute goals, by themselves, are not definitive enough 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 which modifications to the system are expected over its lifetime. That is, the abstract quality goal of modifiability must be made concrete.

System-specific scenarios can be used to more clearly describe the quality attributes that are important to the system and what the desired quality attribute responses should be. Scenarios are short stories that describe an interaction with the system that exercises a particular quality attribute. For example, a scenario makes the modifiability requirement above more explicit as follows:

> *"An improved COTS[3] discrete event generator product is available for the system, and the system permits engineers to remove the old discrete event generator and incorporate the new one in less than two person-weeks."*

2. These techniques have evolved in separate communities, each with its own vernacular and point of view, and have typically been performed in isolation. Emerging international standards might eventually bring some clarity to the field [IEEE 90], [IEEE 92], [ISO 01].

3. COTS stands for commercial off-the-shelf.

To be useful, a scenario must, at the very least, have a clear stimulus and response. The stimulus is the part of the scenario that describes an agent or factor that initiates the system to react in some way. In the example above, the stimulus is "*An improved COTS discrete event generator product is available and required for the system.*" The response is the system's reaction to the stimulus. In the example above, the response is "*…the system permits engineers to remove the old discrete event generator and incorporate the new one in less than two person-weeks.*" An important part of the response is a clear response measure. Simply saying "*modify the system to incorporate the new discrete event generator*" would not describe how well the architecture accommodated the modification. Given enough time and money, any modification is possible. However, in this scenario, a response measure of "*two person-weeks*" is imposed, forcing the architect to ensure that the system is modifiable with respect to some very particular and measurable criteria.

In addition to stimulus and response, it is often useful to include in the scenario a third component—the environment in which the scenario takes place. For example, if a system has different modes of operation (i.e., peak vs. low demand, normal operation vs. maintenance mode), the environment helps clarify the scenario. An acceptable or easy-to-achieve response under one environment might be unacceptable or stressful to achieve in a different environment.

System requirements often focus on functionality and provide only vague descriptions of the quality attribute requirements. Certainly, meeting the specified functionality is an important consideration when designing a system. However, a negative side effect of functionality driving architectural design is that the essential quality attributes are often overlooked or omitted from the architectural design. When this happens, the architect often makes assumptions regarding which quality attributes are important to the stakeholders.

System decomposition decisions can also affect the way in which the functionality gets implemented and how well the quality attribute requirements are met. Modifiability, interoperability, availability, and other quality attributes may well be compromised.

In addition, achieving a quality attribute can have side effects on other attributes [Boehm 78]. These side effects can be used to trade off between quality attributes. Promoting one quality attribute requirement usually has an adverse effect on some other quality attribute requirement. Architectural decisions will promote some quality attribute requirements while inhibiting others, thus resulting in quality attribute tradeoff decisions. These tradeoffs are best dealt with in the earliest phases of system development—during the design of the architecture.

The earlier key quality attribute requirements are identified and prioritized, the more likely it is that the essential quality attributes will be built into the system. It is more cost-effective to reason about quality attribute tradeoffs early in the life cycle than later in the life cycle when modifications are often difficult, impractical, or even impossible. There are several challenges when addressing quality attributes in architectural design:

- What is the precise meaning of quality attributes—such as modifiability, security, performance, and reliability—in the context of the system being built?
- How can you discover, characterize, and prioritize the key quality attributes before the system is built?
- How can geographically dispersed communities of system stakeholders be engaged in a disciplined and repeatable way in the discovery and characterization of quality attributes?
- How can all this information be used?

Addressing these issues is the primary purpose of the QAW.

3 QAW Method

The QAW is a facilitated, early intervention method used to generate, prioritize, and refine quality attribute scenarios before the software architecture is completed. The QAW is focused on system-level concerns and specifically the role that software will play in the system. The QAW is dependent on the participation of system stakeholders—individuals on whom the system has significant impact, such as end users, installers, administrators (of database management systems [DBMS], networks, help desks, etc.), trainers, architects, acquirers, system and software engineers, and others. The group of stakeholders present during any one QAW should number at least 5 and no more than 30 for a single workshop. In preparation for the workshop, stakeholders receive a "participants handbook" providing example quality attribute taxonomies, questions, and scenarios. If time allows, the handbook should be customized to the domain of the system and contain the quality attributes, questions, and scenarios that are appropriate to the domain and the level of architectural detail available.

The contribution of each stakeholder is essential during a QAW; all participants are expected to be fully engaged and present throughout the workshop. Participants are encouraged to comment and ask questions at any time during the workshop. However, it is important to recognize that facilitators may occasionally have to cut discussions short in the interest of time or when it is clear that the discussion is not focused on the required QAW outcomes. The QAW is an intense and demanding activity. It is very important that all participants stay focused, are on time, and limit side discussions throughout the day.

The QAW involves the following steps:

1. QAW Presentation and Introductions
2. Business/Mission Presentation
3. Architectural Plan Presentation
4. Identification of Architectural Drivers
5. Scenario Brainstorming
6. Scenario Consolidation
7. Scenario Prioritization
8. Scenario Refinement

The following sections describe each step of the QAW in detail.

3.1 Step 1: QAW Presentation and Introductions

In this step, QAW facilitators describe the motivation for the QAW and explain each step of the method. We recommend using a standard slide presentation that can be customized depending on the needs of the sponsor.

Next, the facilitators introduce themselves and the stakeholders do likewise, briefly stating their background, their role in the organization, and their relationship to the system being built.

3.2 Step 2: Business/Mission Presentation

After Step 1, a representative of the stakeholder community presents the business and/or mission drivers for the system. The term "business and/or mission drivers" is used carefully here. Some organizations are clearly motivated by business concerns such as profitability, while others, such as governmental organizations, are motivated by mission concerns and find profitability meaningless. The stakeholder representing the business and/or mission concerns (typically a manager or management representative) spends about one hour presenting

- the system's business/mission context
- high-level functional requirements, constraints, and quality attribute requirements

During the presentation, the facilitators listen carefully and capture any relevant information that may shed light on the quality attribute drivers. The quality attributes that will be refined in later steps will be derived largely from the business/mission needs presented in this step.

3.3 Step 3: Architectural Plan Presentation

While a detailed system architecture might not exist, it is possible that high-level system descriptions, context drawings, or other artifacts have been created that describe some of the system's technical details. At this point in the workshop, a technical stakeholder will present the system architectural plans as they stand with respect to these early documents. Information in this presentation may include

- plans and strategies for how key business/mission requirements will be satisfied
- key technical requirements and constraints—such as mandated operating systems, hardware, middleware, and standards—that will drive architectural decisions
- presentation of existing context diagrams, high-level system diagrams, and other written descriptions

During this time, facilitators continue to capture key aspects of the presentation for later reference.

3.4 Step 4: Identification of Architectural Drivers

During steps 2 and 3, the facilitators capture information regarding architectural drivers that are key to realizing quality attribute goals in the system. These drivers often include high-level requirements, business/mission concerns, goals and objectives, and various quality attributes.

Before undertaking this step, the facilitators should excuse the group for a 15-minute break, during which they will caucus to compare and consolidate notes taken during steps 2 and 3. When the stakeholders reconvene, the facilitators will share their list of key architectural drivers and ask the stakeholders for clarifications, additions, deletions, and corrections. The idea is to reach a consensus on a distilled list of architectural drivers that include high-level requirements, business drivers, constraints, and quality attributes. The final list of architectural drivers will help focus the stakeholders during scenario brainstorming to ensure that these concerns are represented by the scenarios collected.

3.5 Step 5: Scenario Brainstorming

After the architectural drivers have been identified, the facilitators initiate the brainstorming process in which stakeholders generate scenarios. The facilitators review the parts of a good scenario (stimulus, environment, and response) and ensure that each scenario is well formed during the workshop.

Each stakeholder expresses a scenario representing his or her concerns with respect to the system in round-robin fashion. During a nominal QAW, at least two round-robin passes are made so that each stakeholder can contribute at least two scenarios. The facilitators ensure that at least one representative scenario exists for each architectural driver listed in Step 4.

Scenario generation is a key step in the QAW method and must be carried out with care. We suggest the following guidance to help QAW facilitators during this step:

1. Facilitators should help stakeholders create well-formed scenarios. It is tempting for stakeholders to recite requirements such as “The system shall produce reports for users.” While this is an important requirement, facilitators need to ensure that the quality attribute aspects of this requirement are explored further. For example, the following scenario sheds more light on the performance aspect of this requirement: “*A remote user requests a database report via the Web during peak usage and receives the report within five seconds.*” Note that the initial requirement hasn’t been lost, but the scenario further explores the performance aspect of this requirement. Facilitators should note that quality attribute names by themselves are not enough. Rather than say “*the system shall be modifiable*,” the scenario should describe what it means to be modifiable by providing a specific example of a modification to the system vis-à-vis a scenario.

2. The vocabulary used to describe quality attributes varies widely. Heated debates often revolve around to which quality attribute a particular system property belongs. It doesn't matter what we call a particular quality attribute, as long as there's a scenario that describes what it means.

3. Facilitators need to remember that there are three general types of scenarios and to ensure that each type is covered during the QAW:
   a. use case scenarios - involving anticipated uses of the system
   b. growth scenarios - involving anticipated changes to the system
   c. exploratory scenarios - involving unanticipated stresses to the system that can include uses and/or changes

4. Facilitators should refer to the list of architectural drivers generated in Step 4 from time to time during scenario brainstorming to ensure that representative scenarios exist for each one.

3.6 Step 6: Scenario Consolidation

After the scenario brainstorming, similar scenarios are consolidated when reasonable.

To do that, facilitators ask stakeholders to identify those scenarios that are very similar in content. Scenarios that are similar are merged, as long as the people who proposed them agree and feels that their scenarios will not be diluted in the process. Consolidation is an important step because it helps to prevent a "dilution" of votes during the prioritization of scenarios (Step 7). Such a dilution occurs when stakeholders split their votes between two very similar scenarios. As a result, neither scenario rises to importance and is therefore never refined (Step 8). However, if the two scenarios are similar enough to be merged into one, the votes might be concentrated, and the merged scenario may then rise to the appropriate level of importance and be refined further.

Facilitators should make every attempt to reach a majority consensus with the stakeholders before merging scenarios. Though stakeholders may be tempted to merge scenarios with abandon, they should not do so. In actuality, very few scenarios are merged.

3.7 Step 7: Scenario Prioritization

Prioritization of the scenarios is accomplished by allocating each stakeholder a number of votes equal to 30% of the total number of scenarios generated after consolidation. The actual number of votes allocated to stakeholders is rounded to an even number of votes at the discretion of the facilitators. For example, if 30 scenarios were generated, each stakeholder gets 30 x 0.3, or 9, votes rounded up to 10. Voting is done in round-robin fashion, in two passes. During

each pass, stakeholders allocate half of their votes. Stakeholders can allocate any number of their votes to any scenario or combination of scenarios. The votes are counted, and the scenarios are prioritized accordingly.

3.8 Step 8: Scenario Refinement

After the prioritization, depending on the amount of time remaining, the top four or five scenarios are refined in more detail. Facilitators further elaborate each one, documenting the following:

- Further clarify the scenario by clearly describing the following six things:
  1. stimulus - the condition that affects the system
  2. response - the activity that results from the stimulus
  3. source of stimulus - the entity that generated the stimulus
  4. environment - the condition under which the stimulus occurred
  5. artifact stimulated - the artifact that was stimulated
  6. response measure - the measure by which the system's response will be evaluated
- Describe the business/mission goals that are affected by the scenario.
- Describe the relevant quality attributes associated with the scenario.
- Allow the stakeholders to pose questions and raise any issues regarding the scenario. Such questions should concentrate on the quality attribute aspects of the scenario and any concerns that the stakeholders might have in achieving the response called for in the scenario.

See the example template for scenario refinement in Appendix A. This step continues until time runs out or the highest priority scenarios have been refined. Typically, time runs out first.

4 QAW Benefits

The QAW provides a forum for a wide variety of stakeholders to gather in one room at one time very early in the development process. It is often the first time such a meeting takes place and generally leads to the identification of conflicting assumptions about system requirements. In addition to clarifying quality attribute requirements, the QAW provides increased stakeholder communication, an informed basis for architectural decisions, improved architectural documentation, and support for analysis and testing throughout the life of the system.

The results of a QAW include

- a list of architectural drivers
- the raw scenarios
- the prioritized list of raw scenarios
- the refined scenarios

This information can be used to

- update the organization's architectural vision
- refine system and software requirements
- guide the development of prototypes
- exercise simulations
- understand and clarify the system's architectural drivers
- influence the order in which the architecture is developed
- describe the operation of a system

In short, the architect can use this information to design the architecture. In addition, after the architecture is created, the scenarios can be used as part of a software architecture evaluation. If the Architecture Tradeoff Analysis Method^{SM} (ATAM^{SM})[4] is selected as the software architecture evaluation method, the scenarios generated during the QAW can be incorporated as seed scenarios in that evaluation [Clements 02a].

The QAW lends itself well to the capture of many architecturally relevant materials. Software architectural documentation is a collection of view packets plus any documentation that applies to more than one view [Clements 02b]. Each view packet contains a primary presenta-

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

tion, a catalog of the view's elements (including element behavior), a context diagram, a variability guide, architecture background (rationale, analysis results, and assumptions about the environment), and other information including mapping to requirements.

Several pieces of this information will be gleaned directly from the QAW. For example, scenario generation can lead to the creation of use case diagrams, context diagrams, or their equivalent. Refined scenarios can be documented as sequence diagrams or collaboration diagrams. Stakeholders' concerns and any other rationale information that is captured should be recorded individually in a form that can be included in the appropriate view packet or overview documentation. Details that explain how to transition these artifacts into architectural documentation is the subject of ongoing research.

In addition to the more immediate benefits cited above, the scenarios continue to provide benefits during later phases of development. They provide input for analysis throughout the life of the system and can be used to drive test case development during implementation testing.

5 Conclusions

Quality attribute requirements are the means by which a system is intended to meet its business goals. These requirements must be recognized and considered early in the life cycle, and the system and software architectures must be designed so that their quality attributes are met. To do that, you must understand the quality attribute architectural drivers. The QAW, a method for eliciting and explicitly documenting quality attribute requirements early in the development process, provides a forum for stakeholders to come to consensus about these drivers. If there are a large number of diverse, geographically scattered stakeholder groups, multiple QAWs can be conducted at various stakeholder locations, after which their results can be consolidated.

The QAW also provides mechanisms for capturing other architecturally relevant materials; for example, refined scenarios can be documented as sequence diagrams or collaboration diagrams. Stakeholders' concerns should be incorporated into design rationales. In addition, the QAW might expose assumptions about the environment.

The refined scenarios developed during the QAW have been used to analyze system or software architectures, and the process for conducting the analysis has been adapted to the needs of individual organizations. A number of lessons learned from these experiences are described by Barbacci [Barbacci 02] and Bergey and Wood [Bergey 02].

Appendix A QAW Roles and Template

The QAW is a one-day event facilitated by members of the Software Engineering Institute (SEI[SM])[5] staff. Facilitators capture the raw scenarios during the workshop and guide scenario prioritization and refinement. The method calls for two roles, filled by at least two people:

- QAW lead: The lead ensures that the steps of the method are carried out during the workshop. The lead facilitates discussions, and ensures that the method is carried out in a timely fashion and that the required QAW artifacts are produced.
- QAW scribe: The scribe captures the raw scenarios, their prioritization, the refined scenarios, and any relevant issues that emerge during the workshop. The scribe uses the standard QAW template shown below as a guide during the workshop to ensure that the appropriate information is captured in a consistent manner.

<Organization> Quality Attribute Workshop (QAW) - <date>

1. QAW Overview and Introductions:

<Obtain list of attendees/take notes as appropriate>

2. Business/Mission Presentation:

<Capture driving quality attributes, issues, notes>

3. Architectural Plan Presentation:

<Capture driving quality attributes, issues, notes>

4. Identification of Architectural Drivers:

<Share the information from steps 2 and 3. Then, after a few minutes, ask for clarifications and corrections to the list of architectural drivers. That list will help facilitators ensure coverage during scenario brainstorming.>

5. SEI is a service mark of Carnegie Mellon University.

5. Scenario Brainstorming:

<Elicit raw scenarios from the stakeholder community in round-robin fashion. Use the table below, inserting rows as needed.>

Table 1: Blank Raw Scenario Table

Scenario#	Description	Votes
#1		
#2		
#3		
#4		
#5		
……		

6. Scenario Consolidation:

<Cut and paste the Raw Scenario Table above. Merge similar and duplicate scenarios using stakeholders' input. Cut and paste merged scenarios. Also merge cells in the Scenario# column as necessary.>

7. Scenario Prioritization:

<Prioritize scenarios. Each stakeholder gets votes equal to 30% of the total number of scenarios generated. Add a column titled "Votes.">

8. Scenario Refinement:

<Fully develop the scenario to include details such as how long, how much, how often, when, environment, who, and so forth.>

Table 2: Blank Scenario Refinement Table

Scenario Refinement for Scenario N		
Scenario(s):		
Business Goals:		
Relevant Quality Attributes:		
Scenario Components	**Stimulus:**	
	Stimulus Source:	
	Environment:	
	Artifact (If Known):	
	Response:	
	Response Measure:	
Questions:		
Issues:		

Appendix B Example Scenario Refinement

Table 3: Example Scenario Refinement Table

Scenario Refinement for Scenario N		
Scenario(s):		When a garage door opener senses an object in the door's path, it stops the door in less than one millisecond.
Business Goals:		safest system; feature-rich product
Relevant Quality Attributes:		safety, performance
Scenario Components	**Stimulus:**	An object is in the path of a garage door.
	Stimulus Source:	object external to system, such as a bicycle
	Environment:	The garage door is in the process of closing.
	Artifact (If Known):	system's motion sensor, motion-control software component
	Response:	The garage door stops moving.
	Response Measure:	one millisecond
Questions:		How large must an object be before it is detected by the system's sensor?
Issues:		May need to train installers to prevent malfunctions and avoid potential legal issues.

References

URLs are valid as of the publication date of this document.

[Barbacci 95] Barbacci, M.; Klein, M.; Longstaff, T.; & Weinstock, C. *Quality Attributes* (CMU/SEI-95-TR-21, 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-36, ADA324233). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1996. <http://www.sei.cmu.edu/publications/documents/96.reports/96.tr.036.html>.

[Barbacci 02] Barbacci, M. *SEI Architecture Analysis Techniques and When to Use Them* (CMU/SEI-2002-TN-005, ADA413696). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002. <http://www.sei.cmu.edu/publications/documents/02.reports/02tn005.html>.

[Bass 03] Bass, L.; Clements, P.; & Kazman R. *Software Architecture in Practice, Second Edition.* Boston, MA: Addison-Wesley, 2003.

[Bergey 02] Bergey, J. & Wood, W. *Use of Quality Attribute Workshops (QAWs) in Source Selection for a DoD System Acquisition: A Case Study* (CMU/SEI-2002-TN-013, ADA405848). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002. <http://www.sei.cmu.edu/publications/documents/02.reports/02tn013.html>.

[Boehm 78] Boehm, B. W. *Characteristics of Software Quality.* New York, NY: American Elsevier, 1978.

[Clements 02a] Clements, P.; Kazman, R.; & Klein, M. *Evaluating Software Architectures: Methods and Case Studies.* Boston, MA: Addison-Wesley, 2002.

[Clements 02b] Clements, P.; Bachmann, F.; Bass, L.; Garlan, D.; Ivers, J.; Little, R.; Nord, R.; & Stafford, J. *Documenting Software Architectures: Views and Beyond.* Boston, MA: Addison-Wesley, 2002.

[IEEE 90] Institute of Electrical and Electronics Engineers. *IEEE Standard Glossary of Software Engineering Terminology* (IEEE Standard 610.12-1990). New York, NY: Institute of Electrical and Electronics Engineers, 1990.

[IEEE 92] Institute of Electrical and Electronics Engineers. *Standard for a Software Quality Metrics Methodology* (IEEE Standard 1061-1992). New York, NY: Institute of Electrical and Electronics Engineers, 1992.

[INCOSE 96] International Council on Systems Engineering (INCOSE). *INCOSE Systems Architecture Working Group (Processes & Methods Technical Committee)*. <http://www.incose.org/cmtes/sawg.html> (August 2003).

[ISO 01] International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC). *Software Engineering—Product Quality* (ISO Std 9126). Geneva, Switzerland: ISO/IEC, 2001.

[Rushby 93] Rushby, J. *Critical System Properties: Survey and Taxonomy.* (CSL-93-01). Menlo Park, CA: Computer Science Laboratory, SRI International, 1993.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 2003	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Quality Attribute Workshops (QAWs), Third Edition	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-2003-TR-016

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-2003-016

11. SUPPLEMENTARY NOTES

12.a DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified/Unlimited, DTIC, NTIS

12.b DISTRIBUTION CODE

13. ABSTRACT (maximum 200 words)

The Quality Attribute Workshop (QAW) is a facilitated method that engages system stakeholders early in the life cycle to discover the driving quality attributes of a software-intensive system. The QAW was developed to complement the Architecture Tradeoff Analysis Method℠ (ATAM℠) and provides a way to identify important quality attributes and clarify system requirements *before* the software architecture has been created.

This is the third edition of a technical report describing the QAW. We have narrowed the scope of a QAW to the creation of prioritized and refined scenarios. This report describes the newly revised QAW and describes potential uses of the refined scenarios generated during it.

14. SUBJECT TERMS

quality attributes, scenarios, attribute requirements, attribute tradeoffs, Quality Attribute Workshop, QAW

15. NUMBER OF PAGES

36

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