# Quality Attribute Workshop Participants Handbook

Mario R. Barbacci
Robert J. Ellison
Charles B. Weinstock
William G. Wood

*January 2000*

SPECIAL REPORT
CMU/SEI-2000-SR-001

Carnegie Mellon
Software Engineering Institute

Pittsburgh, PA 15213-3890

# Quality Attribute Workshop Participants Handbook

CMU/SEI-2000-SR-001
ECS-SR-2000-01

Mario R. Barbacci
Robert J. Ellison
Charles B. Weinstock
William G. Wood

*January 2000*

**Architecture Tradeoff Intitiative**

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

Joanne E. Spriggs
Contracting Office Representative

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 52.227-7013.



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

In large software systems, the achievement of qualities such as performance, security, and modifiability is dependent not only on code-level practices but also on the overall software architecture. Thus, it is in developers' best interests to determine, at the time a system's software architecture is specified, whether the system will have the desired qualities.

With the sponsorship of the U.S. Coast Guard's Deepwater Acquisition Project the SEI has developed the concept of a "Quality Attribute Workshop" in which system stakeholders focus on the analysis and evaluation of system requirements and quality attributes. The purpose of the workshop is to identify scenarios from the point of view of a diverse group of stakeholders and to identify risks (e.g., inadequate performance, successful denial-of-service attacks) and possible mitigation strategies (e.g., replication, prototyping, simulation). Stakeholders include architects, developers, users, maintainers, and people involved in installation, deployment, logistics, planning, and acquisition. This special report describes the process we use to conduct a workshop, information required, suggested tools, and expected outcomes of a workshop.

# 1 Introduction

In large software systems, the achievement of qualities such as performance, availability, security, and modifiability is dependent not only upon code-level practices (e.g., language choice, detailed design, algorithms, data structures, and testing), but also upon the overall software architecture. 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.

A variety of qualitative and quantitative techniques are used for analyzing specific quality attributes [Barbacci 95, Barbacci 96, Rushby 93]. These techniques have evolved in separate communities, each with its own vernacular and point of view, and have typically been performed in isolation.[1] However, the attribute-specific analyses are *interdependent*; for example, performance affects modifiability, availability affects safety, security affects performance, and everything affects cost. In other words, achieving a quality attribute can have side effects on other attributes [Boehm 78]. These side effects represent dependencies between attributes and are defined by parameters that are shared among attribute models. If we can identify these side effects, the results from one analysis can feed into the others.

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 upon 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 goal of modifiability must be made concrete. The same observation is true for other attributes.

This workshop is intended as a forum for the discussion of quality attributes and their evaluation. It is important to point out that we will not aim at an absolute measure of "architecture quality"; rather our 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., prototyping, modeling, simulation).

1. Emerging international standards might eventually bring some clarity to the field [IEEE-610.12, IEEE-1061, ISO-9126]

## 1.1 Roadmap Activities

Figure 1 illustrates the Quality Attribute Roadmap, the process we will use 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.

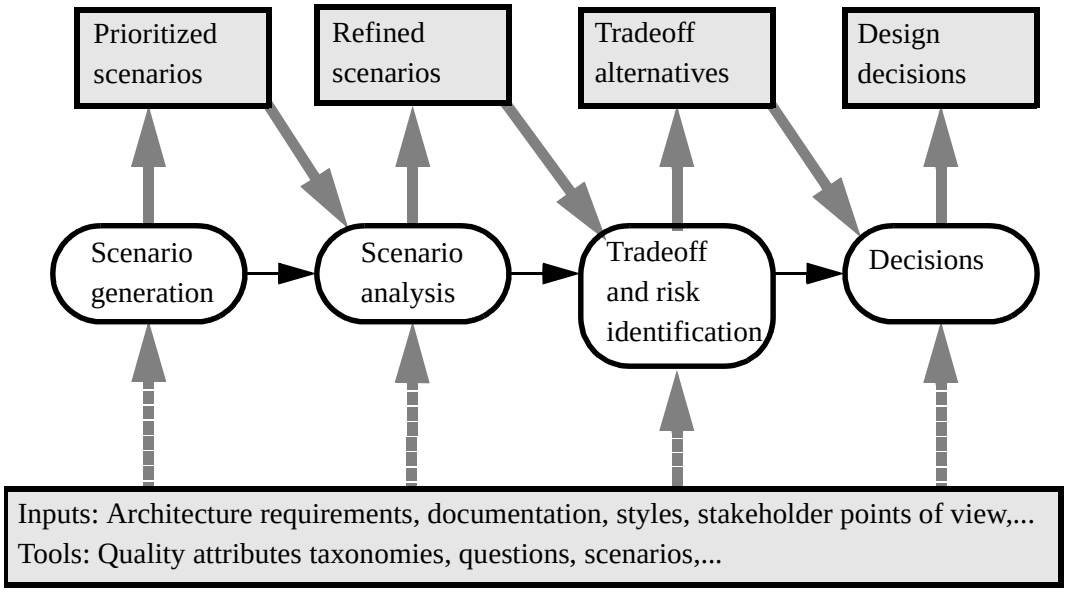


*Figure 1: Roadmap Activities*

Scenario generation takes place during a facilitated brainstorming process; stakeholders propose scenarios that test the effectiveness of a C4ISR 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.[1]

1. 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 off-line, 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

- single scenario that involves two attributes explicitly or implicitly
- multiple scenarios about different attributes sharing common factors (e.g., resources, protocols)
- multiple contradictory scenarios

## 1.2 Roadmap Inputs and Tools

We use various sources as inputs for the activities: architecture documentation, stakeholder points of view, and architecture styles.

Since there are no generally accepted, industry-wide standards for describing a system architecture, the types of questions and scenarios that can be generated are often constrained by the available documentation. In the case of systems documented using the C4ISR Architecture Framework [C4ISR 97], different products or collections of products will differ in their relative value for generating quality attribute-specific scenarios [Barbacci 99]. Depending on the quality attributes of concern, C4ISR products might have to be complemented with additional documents, to address quality attributes concerns that are underrepresented in the framework products.

The stakeholders generate, prioritize, and analyze the scenarios, and identify tradeoffs and risks from their point of view, depending on the role they play in the development of the system, and their expertise on specific quality attributes. In addition to the architect(s), we assume to have participating representative users, developers, and maintainers. Additional participants might include people involved in installation, deployment, logistics, planning, acquisition, etc.

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,” “pipe and filter” that can be used as drivers for the analysis by having “canned” scenarios, known tradeoffs, and likely risks. The results of the analysis would depend on what architecture styles are used.

Finally, we use a collection of tools during the roadmap activities. These tools are described in detail in the remainder of the handbook. However, bear in mind that the sample taxonomies, questions, and scenarios in this handbook show a greater level of detail than expected at this stage. Taxonomies, questions, and scenarios used during the workshop exercises will adapt to the level of detail available.

# 2 Questions, Scenarios, and Quality Attributes

There is a collection of tools and techniques that we use to perform a quality attribute analysis. These come under different labels, e.g., screening questions or exploratory scenarios. While we seem to have an intuitive understanding of what we mean by the terms, it is necessary to be more precise to ensure that a) we share the same understanding and b) we can decide what tools to use and when to use them.

## 2.1 Questions

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

1. **Screening questions** are used to quickly narrow or focus the scope of the evaluation. They are about what is important to the stakeholders.
   - Screening questions are qualitative, and the answers are not necessarily precise or quantifiable. The emphasis is on expediency, on “separating the wheat from the chaff.”
   - 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).
2. **Elicitation questions** are used to gather information to be analyzed later. They are about how a quality attribute or a service that was identified as important is achieved by the system.
   - Elicitation questions collect information about decisions made, and the emphasis is on extracting quantifiable data.

- Elicitation questions can be driven by an attribute model. The reason to ask for some quality attribute-specific information is because 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. The reason to ask for some architectural information is because 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 architecture mechanism branch of the quality attribute tables.

3. **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:

1. **Breadth-first questioning** first identifies all important attributes and subsets of the architecture and then for each one, elicits all the information that will be used later for analysis.

2. **Depth-first questioning** dives deep into any attribute or subset of the architecture identified as important 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.

The results of questioning techniques are captured in a "utility tree" that shows the attributes of concern in sufficient detail to allow generation of appropriate scenarios.

## 2.2 Scenarios

Scenarios are used to exercise the architecture against current and future situations. It is important to distinguish these scenarios from the Modeling and Simulation Master Plan (MSMP) scenarios. The latter are about the operational effectiveness of IDS in performing Coast Guard missions over extended period of time. The quality attribute scenarios are a test of effectiveness of the C4ISR architecture with respect to quality attributes. Thus the scenarios we use are not at the level of IDS missions but at the level of software and hardware systems, subsystems and components.

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 operator console).

3. **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, replace 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).

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 do, it would be useful to understand the consequences).

## 2.3 Quality Attributes

The quality attributes are 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 could be "change the operating system," and the response(s) could include "effort to implement," "number of subsystems affected," etc.

Scenarios and questions contain explicit or implied attribute stimulus/response/mechanisms 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, where risks can arise from decisions made (e.g., choice of middleware) as well as decisions not yet made (e.g., message encoding).

The ordering of the generation of scenarios and questions can alternate. 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, and a scenario about message throughput would identify the components involved in the message path. The capacity or speed of some components might be questionable, prompting further elicitation questions: e.g., time required to process a message or choice of queueing policy.

## 2.4 Generic Questions

These generic questions are suggested as starting points. They must be instantiated for each quality attribute as seeds for additional questions about stimuli, response, or architecture mechanisms.

In addition to these questions, we expect to have between five and seven additional questions for each attribute. The additional questions should be scattered around the specific attribute stimuli, responses, and architecture mechanisms.

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.

*Table 1: Generic Questions - Apply to all Attributes*

| | Question | Type | No, |
|---|---|---|---|
| Requirements | Which are the important services (i.e., services important for some quality attribute requirement)? | Screen | 1. |
| | What services can operate in degraded modes? | Screen | 2. |
| For important services | What are these degraded modes (e.g., X% speed, no training allowed, online upgrade)? | Elicitation | 3. |
| | What are the conditions or events that might lead to a service degradation (e.g., message sent at the wrong time, incorrect operator action, supplier going out of business)? | Elicitation | 4. |
| | What are the consequences of not meeting the quality requirements in various degraded modes (e.g., catastrophe, annoyance, minor inconvenience)? | Analysis | 5. |
| For other services | Are there important services that depend on a non-important service? | Elicitation | 6. |
| | What are the consequences to an important service from the low quality of a non-important service? (e.g., personal email might not be important except that it might be assumed to exist in an important service like training). | Analysis | 7. |

# 3 Dependability

Dependability is defined as “that property of a system such that reliance can justifiably be placed in the service it provides”[Laprie 92]. Depending on the intended application of the system, different emphasis might be placed on different measures of dependability such as availability, the system’s readiness for delivery of service, or reliability, the systems’s continuity of service.

## 3.1 Stimulus

The stimuli are the faults that affect the availability of the system. When a fault propagates across a predetermined boundary, the fault is said to cause a failure.

The definition of fault/failure is recursive -an internal component fault might cause a container component failure and this failure is just a fault inside a bigger container. A fault does not necessarily propagate and cause a failure - it might disappear without trace (e.g., by overwriting an incorrect variable) or it might stay latent and cause a failure at a later time (e.g., by using an uncorrected, wrong value of a variable some time after the incorrect assignment).

Faults can be classified by their **origin** as[1]

- **Internal faults** - these arise inside a component of the system and propagate to other components and possibly to the environment.
- **External faults** - these arise in the environment and propagate into system components.

Fault can be classified by their **types** as

- **Timing faults** - these are expressed as deviations in time (e.g., an event is early or late).
- **Value faults** - these are expressed as deviations in value (e.g., a value is outside bounds).
- **Resource faults** - these are expressed as deviations in resource utilization (e.g., a queue overflows).
- **Halting faults** - these are extreme cases of timing faults (e.g., an event is never seen).

1. The fault origin corresponds to the “source” node in the performance attribute stimuli. Although the questions might be similar, the effects and the analyses thereof are different.

Faults can be further classified by the **amount of time** they are active as **Permanent** or **Temporary** and by their **frequency regularity** as **Periodic** or **Aperiodic.** The latter corresponds to the “frequency regularity” in the performance stimuli. Although the questions might be similar, the effects and the analyses thereof are different.

For purposes of documentation, faults can be further classified by the cause (physical, human generated), nature (accidental, intentional but not malicious, intentional and malicious), and the time of occurrence (during development, during operation).

## 3.2 Response

The response is measured in terms of the availability of the services or modes of operation. A system can operate in various degraded modes, and the response for each mode of operation has to be calculated separately.

The service availability is expressed in terms of

- **Overall availability** - Fraction of time that the system is working.
- **Instant availability** - Probability P(t) that the system is working at time t.
- **Failure rates** - Number of failures per unit time.
- **Repair rates** - Number of repairs per unit time.

The failure domains are similar to fault domains, but they are not correlated; a fault of a given type (e.g., timing) can lead to a failure of a different type (e.g., halting). The perception of the failure by different observers (systems or humans) can be **Consistent** (all observers see the same failure) or **Inconsistent** (observers see different failures, including no failure at all).

For purposes of documentation, failures can be further classified by their consequence (benign, catastrophic).

## 3.3 Architecture

There are two architecture mechanisms to consider depending on whether the detection/containment/correction actions take place during the operation of the system (fault tolerance) [Heimerdinger 92, Jalote 94] or during the development of the system (fault avoidance).

Fault tolerance resources that might be consumed by the detection, containment, and recovery actions for each identified failure can be classified as

- **computation** (e.g., replicated processors, repeated and redundant operations, additional checking steps)

- **storage** (e.g., replicated data and state, saved messages pending acknowledgments)
- **communication** (e.g., replicated paths, repeated transmissions).

Fault tolerance methods that might be implemented include

- **Detection actions** are expressed in terms of the types of faults observable in different components or resources. Faults can change their type as they propagate (e.g., a value fault injected into a component might emerge later on as a timing fault). It helps to know the types the fault can assume and the locations where the fault could be detected because there might be alternative strategies for containment and recovery.
- **Containment and recovery actions** are expressed in terms of locality and degradation of service. It helps to know what containment or repair actions can occur because the system might be degraded and certain obligations might not be met (the system might be degraded in different ways before, during, and after the repair action).

**Fault avoidance** methods include fault prevention, fault removal, and fault forecasting.

- **Fault prevention** simply applies techniques for avoiding the introduction of faults into the system in the first place. An example of a fault prevention technique is the peer design review. Inevitably though, faults will end up in the system, despite prevention attempts.
- **Fault removal** is concerned with eliminating them from the system before they can be triggered during operation. There are three steps to fault removal:
  - **verification** - the process of checking whether the system adheres to its requirements or other verification conditions.
  - **diagnosis** - necessary if the system does not pass its verification conditions. Diagnosis is used to determine why it is not passing those conditions.
  - **correction** - applied to fix the problem once it has been diagnosed.

  Once a fault has been corrected, the entire process should be repeated to ensure that the removal process has not caused further problems. Fault removal techniques span the whole spectrum from unit testing to formal verification of critical portions of the system.
- **Fault forecasting** is concerned with determining how the resulting system is likely to fail. Fault forecasting is either **qualitative** (e.g., a failure modes and effects analysis-FMEA, or a fault tree analysis) or **quantitative** (as measured by predicted reliability and availability).

*Table 2: Dependability Stimulus*

| | |
|---|---|
| Stimulus (Faults) | 1. |
| *Performance: Stimulus/source* | 2. |
| *Performance: Stimulus/frequency regularity* | 3. |
| type | 4. |
| timing | 5. |
| value | 6. |
| resource | 7. |
| halting | 8. |
| persistence | 9. |
| permanent | 10. |
| temporary | 11. |

*Table 3: Dependability Response*

| | |
|---|---|
| Response (Failures) | 1. |
| service availability | 2. |
| overall availability | 3. |
| instant availability | 4. |
| failure rate | 5. |
| repair rate | 6. |
| failure domain | 7. |
| value | 8. |
| timing | 9. |
| resource | 10. |
| halting | 11. |
| failure perception | 12. |
| consistent | 13. |
| inconsistent | 14. |

*Table 4: Dependability Architecture Mechanisms*

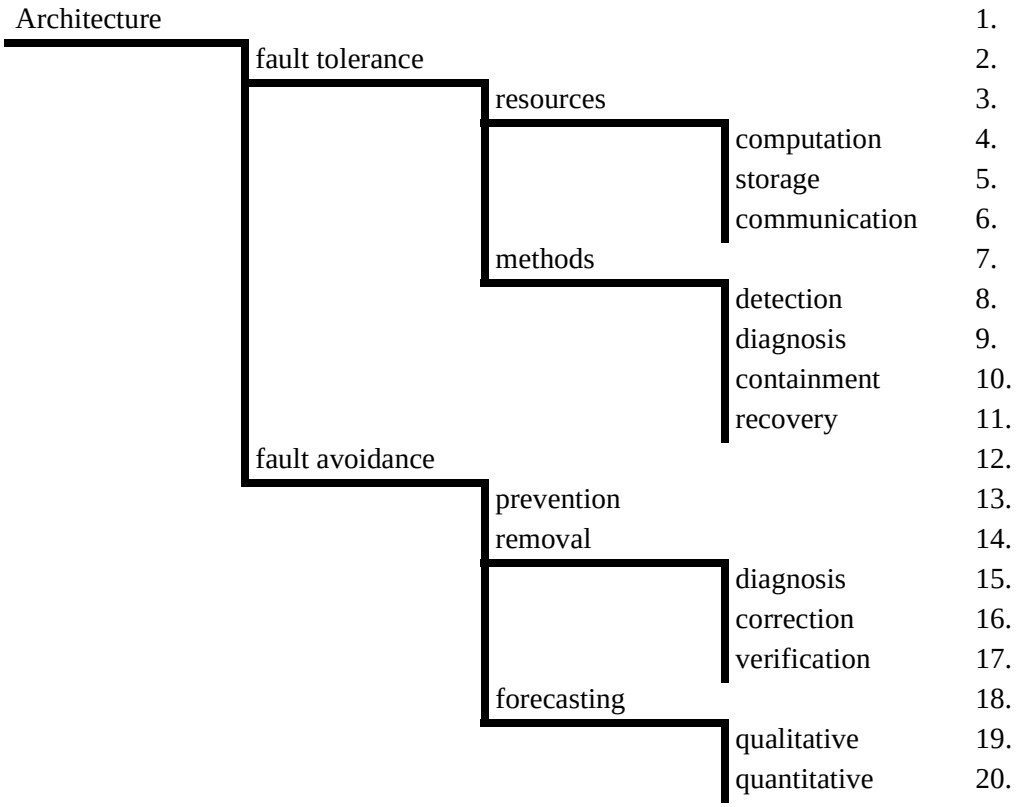

*Table 5: Dependability Questions*

| | Question | Type | No. |
|---|---|---|---|
| Requirements | Which are the important services (i.e., services whose availability is important)? | Screen | 1. |
| | Which of these services can operate in degraded modes (i.e., have varying levels of availability)? | Screen | 2. |
| For Important Services | What are these degraded modes (e.g., up/down, available 50% of the time)? | Elicitation | 3. |
| | What are the conditions or events that might lead to a service degradation (e.g., message sent at the wrong time, incorrect operator action, broken connection)? | Elicitation | 4. |
| | What are the consequences of not meeting the quality requirements in various degraded modes (e.g., catastrophe, annoyance, minor inconvenience)? | Analysis | 5. |
| For Other Services | What are the consequences to an important service of the low quality of a non-important service? For example, an important service like training might assume the existence of a non-important service like personal email for backup course delivery. | Analysis | 6. |
| Stimulus/ Response | What faults can occur in each component or connection involved in a service? | Elicitation | 7. |
| | What services can not be allowed to fail (e.g., authentication services)? | Elicitation | 8. |
| Mechanisms: Fault Tolerance: | What failure detection techniques are used, if any, for each failure mode of a component/connection (e.g., majority voting, leadership)? | Elicitation | 9. |
| | How is a failure or degradation of the service perceived or noted? | Analysis | 10. |
| | What kind of fault recovery techniques are used, if any, for each failure mode of a component/connection (e.g., compensation, reconfiguration)? | Screen | 11. |
| | Could the recovery action introduce new faults (e.g., flushing a queue to recover from running out of memory might lead to timing errors in components waiting for the flushed items)? | Analysis | 12. |
| Mechanisms: Fault Avoidance | For each component or subsystem, what techniques are used to ensure that the design is as fault-free as possible (e.g., modeling, simulation, prototyping, formal verification, peer reviews)? | Elicitation | 13. |
| | What architectural mechanisms are in place to facilitate testing? | Elicitation | 14. |
| | How do you determine when a component or subsystem has been sufficiently tested? | Analysis | 15. |
| | How do you measure the expected reliability/availability of the completed system? | Analysis | 16. |

*Table 6: Dependability Scenarios*

| Scenario | Type | No. |
|---|---|---|
| One processor or server goes down during operation. Half of the processors or servers go down during operation. The network fails during operation. One or more input devices fail during operation. | exploratory | 1. |
| In a distributed system, change the processor on which a specific piece of software runs because of a processor failure. | use case | 2. |
| Degraded operation mode: A user wants to continue to operate the system (perform useful work) in the presence of one of the following kinds of failures: database server, application server, or partial network failure. | use case | 3. |
| The requirements for a subsystem change, and it now has to operate for 24 hours a day, 7 days a week. | exploratory | 4. |
| Error notification: The user wants to define exception conditions corresponding to the data, have the system notify a defined list of recipients by email of the existence of an exception condition, and have the system display the offending conditions in red on data screens. | use case | 5. |

# 4 Security

Security risks can take several forms [Howard 98]. Some risks exist because of the lack of complete knowledge about the environment. Such risks include

- Use of COTS components. Such components may not support security as a critical property. The vulnerabilities of widely used components are also well known to the attacker community.
- Network connections to non-Coast Guard administrated systems. There may be an implied trust of some outside networks so that a successful attack on such a trusted system could make it easier to attack Coast Guard systems.
- General Internet connectivity by employees.
- Multiplicity of users. It can be hard to distinguish “inside” and “outside” users. Account management may be distributed and may not be totally under Coast Guard control.

Other risks may be associated with a particular architecture. Such risks include

- Single points of failure which could be the target of an attack.
- Commonality of components, i.e., a specific vulnerability can be applied against multiple system components or at multiple sites.
- Recovery is too complex, takes too long or requires closely coordinated actions at multiple sites.
- The impact of an attack is not easily contained.

## 4.1 Stimulus

The stimulus for the security attribute are potential attacks. An attack requires internal or external access and has an specific intent or objective. The specific target will usually depends on the objective of the attacker. Given an objective, an attacker will scan a system and design the attack to take advantage of existing vulnerabilities. Specific vulnerabilities in an operating system or in applications are relatively short-lived, but at any given time, there can be system weaknesses which the attacker can exploit.

Some attacks will require **internal access** to assume the identify of a user in order to obtain access to information. Other attacks such as flooding a network or introducing an electronic mail virus can be done with only **external access** to a system.

The attack's **intent** might include **criminal, political, or financial** gain to a person or organization as well as the **increased status** for an individual that is often associated with a computer security breech. The political motivation could be to embarrass the Coast Guard.

A single attack may be part of a larger goal. Many attacker's objective is to get **increased system access** to launch a more damaging attack against the site or be able to attack another site. Thus an attack on a site may begin with attacks on sites which the target site normally trusts. **Theft of resources** in this context is use of computing resources to support non-Coast Guard activities. The remaining impacts are availability, integrity, and confidentiality.

## 4.2 Response

The responses concentrate on the impact of the attack rather than the details of a specific attack.

## 4.3 Architecture

The architectural mechanisms consists of security services to counter the threats. The mechanisms are grouped in three areas: resistance, recovery, and recognition.

**Resistance** includes

- **Access control** - user authorization for access to specific data or functions
- **Strong user** and **system authentication** - use of technology such as public key certificates to provide authentication of both users and systems
- **Administrative procedures** - includes review of modifications, deployment of new equipment, purchasing guidelines
- **Operational security** - system and network administration, responsibilities of operators and system administrators
- **Dispersion** - resources and essential not concentrated physically in one or a limited number of locations (includes redundancy of resources)
- **Diversity** - independent implementation of critical resources to avoid vulnerabilities common to most components
- **Encryption** - use of symmetric key and public key encryption
- **Strong configuration management** and **inventory** of both resources of and data - procedures for restoration of compromised systems
- **Network boundaries** - firewalls, proxy servers, boundary controllers between domains with differing security policies

**Recovery** includes

- **Alternative services** - identification of alternate means to meet the mission
- **Auditing** - recording both user and system actions so as to identify impact
- **Data backup** and **restoration** - identification of data, procedures, and recovery times
- **Reduced services** - continuation of service under attack but with reduced performance or function

**Recognition** includes

- **Auditing** - recognition of attack by impact on data and operations
- **Configuration management** - recognition of attack by changes in configurations
- **Network monitoring** - incident detection

*Table 7: Security Stimulus*

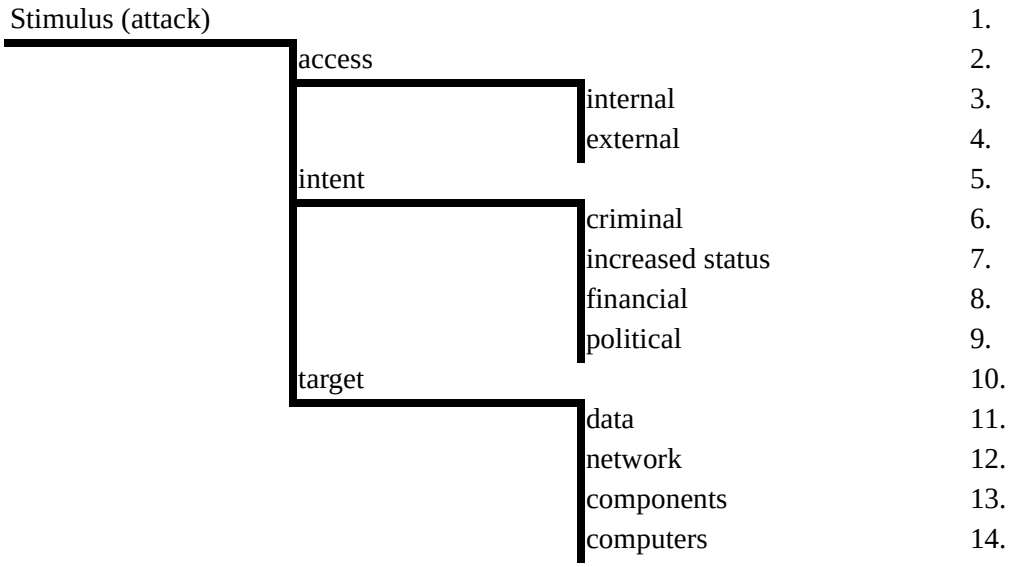


*Table 8: Security Response*

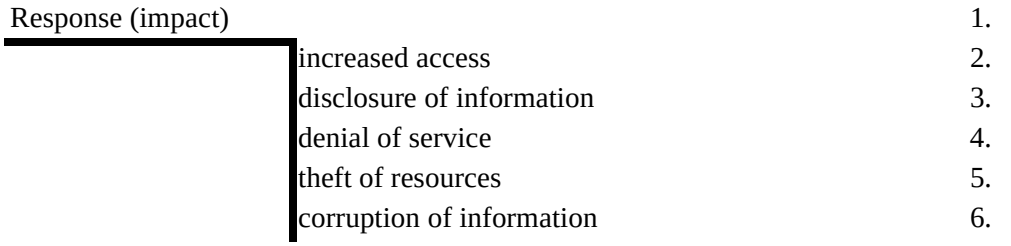

*Table 9: Security Architecture Mechanisms*

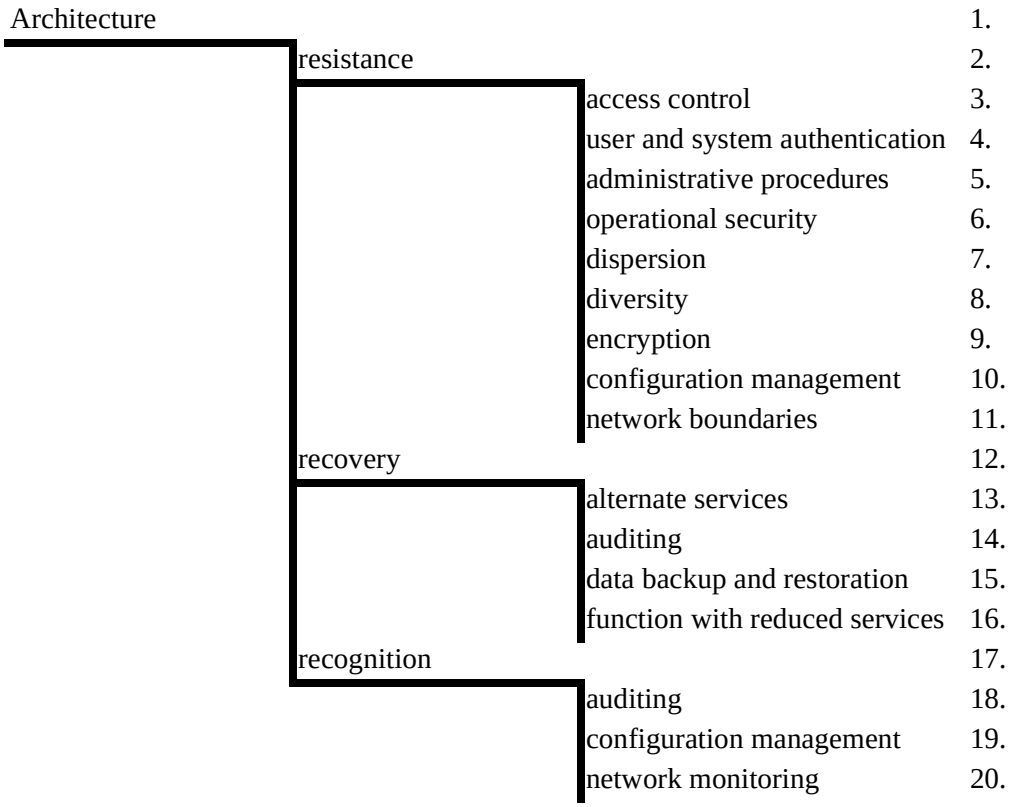

*Table 10: Security Questions*

| | Question | Type | No. |
|---|---|---|---|
| Reqs. | What are the trusted entities in the system and how do they communicate? | Screen | 1. |
| Stimulus/ Response | Which essential services could be significantly impacted by an attack? | Analysis | 2. |
| | Are there attacks or events which could impact a service across the entire Integrated Deepwater System? | Analysis | 3. |
| | Is there a single point from which the entire system is controlled? | Analysis | 4. |
| | For which kind of attacks will recovery be the most difficult? | Analysis | 5. |
| Resistance/Recovery/ Recognition | How is user authentication and authorization information maintained for Coast Guard employees? | Elicitation | 6. |
| | How is access managed for those outside the Coast Guard managed network? | Elicitation | 7. |
| | What sensitive information must be protected? | Elicitation | 8. |
| | What approach is used to protect that data? | Elicitation | 9. |
| | Which user actions are logged? | Elicitation | 10. |
| | What kind of monitoring and access controls exist at network boundaries? | Elicitation | 11. |
| | What information is permitted through or filtered out? | Analysis | 12. |

*Table 11: Security Scenarios*

| Scenario | Type | No. |
|---|---|---|
| A new email virus is publicized but has not yet impacted the Integrated Deepwater System. | use case | 1. |
| A new email virus is publicized and has already impacted the Integrated Deepwater System. | use case | 2. |
| User access control lists need to be changed for managing access for key central information which can be accessed from multiple sites. | growth | 3. |
| Non-Coast Guard personnel require access to sensitive but not classified data which is maintained within a firewall managed system. | use case | 4. |
| An attack is mounted against a key Coast Guard computer. The attacker is able to penetrate the firewall controlling access to internal systems. | use case | 5. |
| An attack is mounted against a key Coast Guard computer and the attacker is a Coast Guard employee and hence there was no penetration of the firewall. | growth | 6. |
| A normally trusted site (non-Coast Guard site) which has regular communications with Coast Guard systems has reported a break-in. | use case | 7. |

# 5 Modifiability

Modifiability considers how the system can accommodate anticipated and unanticipated changes and is largely a measure of how changes can be made locally, with little ripple effect on the system at large.

## 5.1 Stimulus

The stimuli are the future anticipated changes[1] to the system. These changes are classified initially by their origin and can be for the operational, development, and test environments

- **External Changes**. These changes are driven by independent external organizations, such as CANDI vendor upgrades or the availability of new CANDI capabilities.
  - **new COTS version**. Commercial companies release new versions of their products periodically. Some of these are minor releases that fix defects in the previous releases, and others are major releases with significant functionality extensions and interface changes.
  - **new COTS product**. A new COTS product comes on to the marketplace which is directly applicable to the system.
  - **new or improved standard.** A new standard is created or an existing standard is extended.
  - **changed resource**. New resources become available.
  - **new interface.** A new interface is added to the system.

- **Internal Changes**. These are driven from within the organization, such as the addition of new operational functionality, or defect corrections.
  - **defect fix**. A defect has been detected in the system, and this change will fix the defect.
  - **upgraded function**. An existing system function is being upgraded to add new capabilities, or improve quality of service or make the system more easily maintained.
  - **new function.** A new function is added to the system.

1. Since we are discussing this attribute in a large-scale system, the changes may be applicable to the operational environment, the integration and test environment, or the development environment, or any combination of the three.

## 5.2 Response

- **Operational Robustness**. These qualities indicate how well the modifications operate during test and after installation.
  - **number of problem reports** - the problems reported during integration, test, installation, and operation.
  - **mission degradations** - the number of times that a mission was significantly impacted by a change.
- **Effort**. This is the effort required to make the change, in terms of cost, schedule, and people.
- **Complexity**. The complexity of the system increases as the ratio of the number of interfaces to the number of components grows.

## 5.3 Architecture

The architectural parameters measure the flexibility of the architecture to change.

- **Transparency**. This is the ability to move functionality within the system without impacting other functions.
  - **location** - knowing the name of a service allows access without the need to know the specific location.
  - **yellow pages** - knowing the attributes of the service allows access to a service supplier.
- **Information hiding**, **abstraction**. These ensure that the state information is within well-defined containers (objects) and is accessed procedurally, and that the interfaces to obtain this information are well-defined semantically as well as syntactically.
  - **layering** - constructing the system in layers with interfaces between the layers.
  - **virtual machine** - components of the system appear as virtual machines to other components.
  - **interface definition** - concise definition of interfaces between components and layers.
- **Modularity**. The system has been decomposed into modules such that straightforward changes in functionality have straightforward changes to the system to prevent ripple effects.
  - **functional decomposition** - the functions are consistently allocated to components and resources.
  - **patterns** - using architectural styles or defined patterns to capture the system.

*Table 12: Modifiability Stimulus*

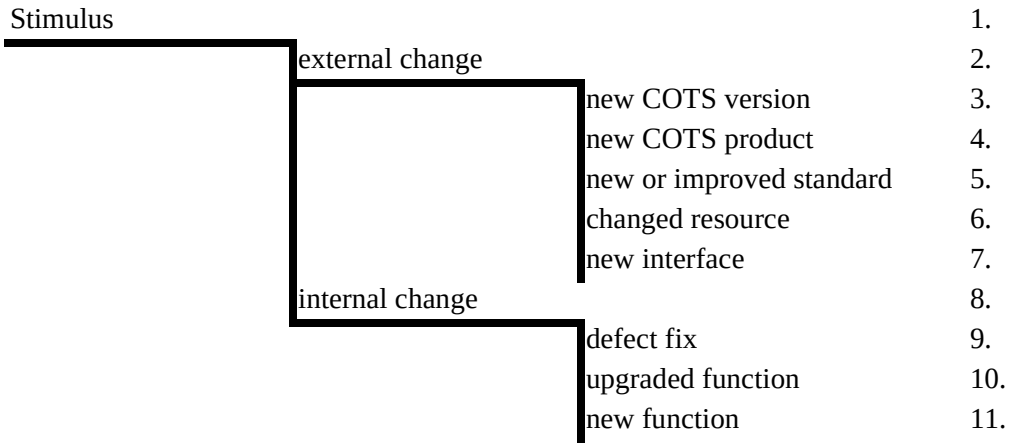


*Table 13: Modifiability Response*

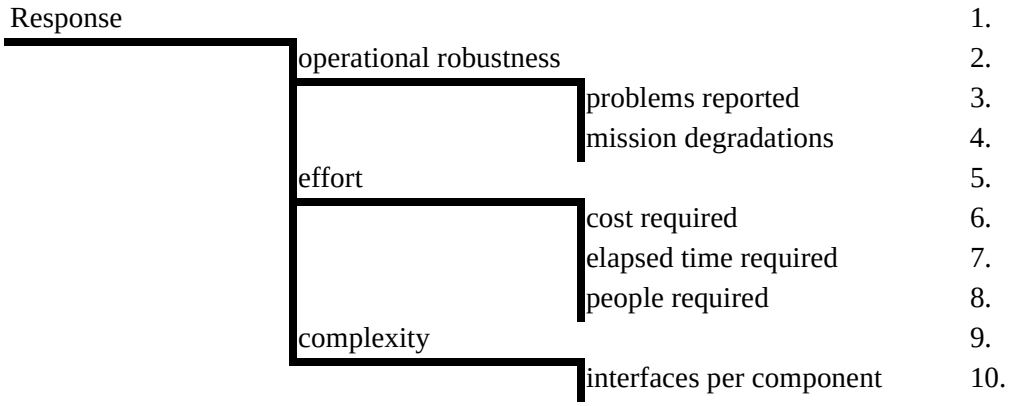

*Table 14: Modifiability Architecture Mechanisms*

| | | | |
|---|---|---|---|
| Architectural | | | 1. |
| | transparency | | 2. |
| | | location | 3. |
| | | yellow pages | 4. |
| | information hiding | | 5. |
| | | layering | 6. |
| | | virtual machine | 7. |
| | | interface definition | 8. |
| | modularity | | 9. |
| | | functional decomposition | 10. |
| | | patterns | 11. |

*Table 15: Modifiability Questions*

| | Question | Type | No. |
|---|---|---|---|
| Requirements | Which are the important anticipated types of upgrades or modifications to the system? | Screen | 1. |
| | Will sequential modifications or upgrades to assets cause the system to operate in a degraded manner for a period of time? | Screen | 2. |
| For important services | What are the limitations on upgrades or modifications (e.g., can only be done offline)? | Elicitation | 3. |
| | What are the conditions or events outside IDS control that might lead to a limitation on upgrades or modifications (e.g., change in standards, suppliers merging, supplier going out of business)? | Elicitation | 4. |
| For other services | How will technology refreshment upgrades (operational, development, test) and new operational capability upgrades be prioritized? | Elicitation | 5. |
| | How will the cost of doing upgrades be predicted? | Analysis | 6. |
| CANDI | What strategy will be used to upgrade versions of systems containing multiple CANDI components released independently? | Screen | 7. |
| | How can the impact of new releases of CANDI components or new CANDI products be effectively evaluated? | Analysis | 8. |
| | How can measures of complexity be used to re-engineer the architecture if changes become very complex? | Analysis | 9. |
| | How will the relationship between defects, fixes, upgrades, and versions be established? | Elicitation | 10. |

*Table 16: Modifiability Scenarios*

| Scenario | Type | No. |
|---|---|---|
| An operational subsystem uses interdependent CANDI components (e.g. OS, ORB, database, window manager, widget manager, tracker), and one or more of these components are upgraded. | use case | 1. |
| A customized component is upgraded regularly and a CANDI replacement becomes available. | growth | 2. |
| An operational defect occurs in a cutter's software while it is at sea and the defect degrades the cutter's mission effectiveness. | use case | 3. |
| An operational defect occurs in an aircraft software while it is on a mission and the defect degrades the aircraft's mission effectiveness. | use case | 4. |
| Upgrade the onboard tracker to use GPS. | growth | 5. |

# 6 Interoperability

The Levels of Information Systems Interoperability (LISI) Capabilities Model provides the basis for the interoperability stimulus/response/architecture [C4ISR 98].

The LISI Capabilities Model and its associated Implementations Options Tables identify the suite of capabilities and available technical implementations for attaining various levels of interoperability, as defined in the LISI Interoperability Maturity Model.

## 6.1 Stimulus

The interoperability stimuli are the systems applications and data exchanges requirements in LISI[1]

> *The applications (A) attribute encompasses the fundamental purpose and function for which any system is built - its mission. For interoperability to occur effectively, similar capabilities or a common understanding of the shared information must exist between systems; otherwise, users have no common frame of reference.*
>
> *The data (D) attribute focuses on the information processed by the system. This attribute deals with both the data format (syntax) and its content or meaning (semantics). The data attribute embodies the entire range of information styles and formats: free text, formatted text, databases (formal and informal), video, sound, imagery, graphical (map) information, etc.*

## 6.2 Response

The response is the LISI Level:

- Enterprise level - systems are capable of operating using a distributed global information space across multiple domains. Multiple users can access and interact with complex data simultaneously.

1. These are two of the four LISI Interoperability Attributes - Procedures, Applications, Infrastructure, and Data (PAID).

- Domain level - systems are capable of being connected via wide area networks (WANs) that allow multiple users to access data. Information is shared between independent applications.
- Functional level - systems reside on local networks that allow data sets to be passed from system to system.
- Connected level - systems are capable of being linked electronically and providing some form of passive electronic exchange.
- Isolated level - systems are isolated or stand-alone.

## 6.3 Architecture

The architecture mechanisms are the procedures and infrastructure support that affect system development, integration, and operation[1]

Items that make up the procedures (P) attribute include

- Standards - includes individual technical standards, architectures, and common operating environments.
- Management - includes aspects of program management, from systems requirements definitions to installation and training.
- Security policy - includes procedures to ensure that proper security precautions are maintained for each implementation.
- Operations

Items that make up the infrastructure (I) attribute include

- Communications and networks
- System services
- Hardware
- Security equipment

1. These are the remaining LISI Interoperability Attributes - Procedures, Applications, Infrastructure, and Data (PAID).

*Table 17: Interoperability Stimulus*

| | | |
|---|---|---|
| Stimulus | | 1. |
| | applications | 2. |
| | | interactive (cross applications) 3. |
| | | full object cut and paste 4. |
| | | shared data (e.g., situation display) 5. |
| | | group collaboration (e.g., white-boards, VTC) 6. |
| | | full text cut and paste 7. |
| | | web browser 8. |
| | | basic operations (e.g., briefings, spreadsheets) 9. |
| | | advisory messaging (e.g., email + attachments) 10. |
| | | basic messaging (e.g., email - attachments) 11. |
| | | data file transfer 12. |
| | | simple interaction (e.g., telemetry, voice, FAX) 13. |
| | data | 14. |
| | | cross enterprise models 15. |
| | | enterprise model 16. |
| | | DBMS 17. |
| | | domain models 18. |
| | | program models and advanced data formats 19. |
| | | basic data formats 20. |
| | | media formats 21. |
| | | private data 22. |

*Table 18: Interoperability Response*

| | |
|---|---|
| Response | 1. |
| enterprise (LISI Level 4) | 2. |
| domain (LISI Level 3) | 3. |
| functional (LISI Level 2) | 4. |
| connected (LISI Level 1) | 5. |
| isolated (LISI Level 0) | 6. |

*Table 19: Interoperability Architecture Mechanisms*

| | | | |
|---|---|---|---|
| Architecture | | | 1. |
| | procedures for data exchanges | | 2. |
| | | multi-national enterprises | 3. |
| | | cross-government enterprises | 4. |
| | | domain (e.g., service, doctrine, training) | 5. |
| | | common operating environment compliance | 6. |
| | | program (e.g., standards procedures, training) | 7. |
| | | standards compliance (e.g., JTA) | 8. |
| | | security profile | 9. |
| | | media exchange procedures | 10. |
| | | manual access controls (e.g. NATO levels) | 11. |
| | infrastructure for data exchanges | | 12. |
| | | multi-dimensional topologies | 13. |
| | | WAN | 14. |
| | | LAN | 15. |
| | | NET | 16. |
| | | two-way | 17. |
| | | one-way | 18. |
| | | removable media | 19. |
| | | manual re-entry | 20. |

*Table 20: Interoperability Questions*

| | Question | Type | No. |
|---|---|---|---|
| Requirements | Which are the important services that must interoperate? | Screen | 1. |
| | What services can operate in degraded levels of interoperability (i.e., at lower LISI level)? | Screen | 2. |
| For important services | What are these degraded levels (e.g., LISI level 2b)? | Elicitation | 3. |
| | What are the conditions or events that might lead to an interoperability degradation (e.g., bandwidth reduction that does not allow sending email attachments would drop LISI from 2a to 1d)? | Elicitation | 4. |
| | What are the consequences of not meeting the interoperability requirements in various degraded modes (e.g., catastrophe, annoyance, minor inconvenience)? | Analysis | 5. |
| LISI | What other systems may need to interoperate with system X? | Elicitation | 6. |
| | What are the specific interoperability characteristics of these systems? | Elicitation | 7. |
| | Projecting a system X profile, what does the Potential Interoperability Matrix reveal (any gaps or shortfalls)? | Analysis | 8. |
| | What strategy will the systems agree to for eliminating interoperability gaps? | Analysis | 9. |
| | What is the assessed LISI level for the new system? | Elicitation | 10. |
| | With which systems (within the existing architecture) is this system potentially capable of interoperating immediately? | Analysis | 11. |
| | With which system(s) does this system need to interoperate (if known)? | Screening | 12. |
| | What interoperability gaps or shortfalls will exists when adding the new system? | Analysis | 13. |
| | What are the implementation options available to eliminate the shortfalls? | Elicitation | 14. |

*Table 21: Interoperability Scenarios*

| Scenario | Type | No. |
|---|---|---|
| Data that was previously transmitted in without encryption now has to be encrypted. | growth | 1. |
| Exchange data with a new external system. | growth | 2. |
| A user attaches screen information to an email message and transmits it to different users using a variety of email systems. | use case | 3. |
| High priority mission data has to be sent to different parties using a variety of messaging systems. | use case | 4. |
| A U.S. Navy committee has released a new version of a communication standard which is used to transmit data critical to IDS operations. | exploratory | 5. |
| Migrate to DII COE Level 5.0 runtime compliance. | growth | 6. |

# 7 Performance

Performance refers to the system responsiveness: either the time required to respond to specific events, or the number of events processed in a given time interval [Klein 93, Lehoczky 94, Smith 90]

## 7.1 Stimulus

The stimuli are factors which lead to some level of performance, as measured by the responses.

- **Mode of operation**. The performance of a system is affected by the **mode** it is in. It can be in **regular** operating mode, or it can be in an **overload** situation.
- **Source of event**. The system is required to respond to events. The **source** of such events can be **internal** (e.g., the system has to respond to an internally generated signal), **external** (e.g., a server has to process a request that arrives from a client process) or via a **clock interrupt**.
- **Frequency of events**. Stimuli can be applied to the system with either **periodic**, or **aperiodic** frequency.

## 7.2 Response

The response of the system is determined by its latency (how long it takes the system to respond to a specific event), its throughput (how many events it can respond to in a given interval of time), and precedence (how the system determines which events to respond to).

- **Latency**. The **response window** is determined by the minimum acceptable latency and the maximum allowable latency (deadline). **Criticality** is a measure of the importance of the function to the system. **Jitter** is a measure of the variation of the time at which a computed result is available from cycle to cycle. Other measurable aspects of latency include the **best, average and worst case behavior** of the system.
- **Throughput**. The **best, average and worst case behavior** of the system is also a key to measuring the system's throughput. **Criticality** is also an important concern and may play a role in determining throughput requirements. The **variability window** measures the interval between the worst case and best case throughput of the system.

- **Precedence**. The system will respond to incoming events according to some precedence. This may be determined by the **criticality** of the event, or some other **ordering** (e.g., order of arrival).

## 7.3 Architecture

Performance architectures are primarily concerned with resources-what they are, how they are consumed, and how contention for them is dealt with.

- **Resource types** can be loosely grouped into four categories: **devices and sensors**, **computing** (CPU) resources, **communication** (network) resources, and **memory** resources.
- **Resource consumption** deals with how these resources are used (for instance, communications bandwidth, memory usage, etc).
- **Resource arbitration** deals with policies and procedures for deciding between competing requests for a particular resource. This arbitration can be determined **offline** (for instance, by pre-computing a **schedule** for a cyclic executive), or **online** (via various priority schemes). There can be several different **queuing policies** employed. A single queue may serve multiple resources or a single resource. Items may be removed from a queue according to their arrival time or any of a number of other policies. Some systems will want to implement a **preemption policy** for scheduled events while others will not. Finally, the architecture will have to implement a **communication policy**. This policy would determine what **network topology** (both physical and logical), **message priorities**, and **delivery guarantees** (if any) are implemented.

*Table 22: Performance Stimulus*

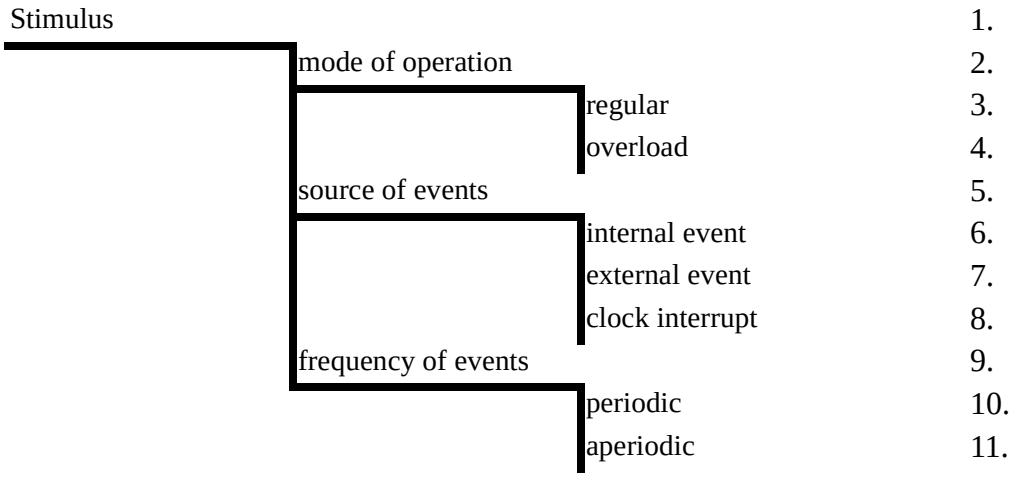


*Table 23: Performance Response*

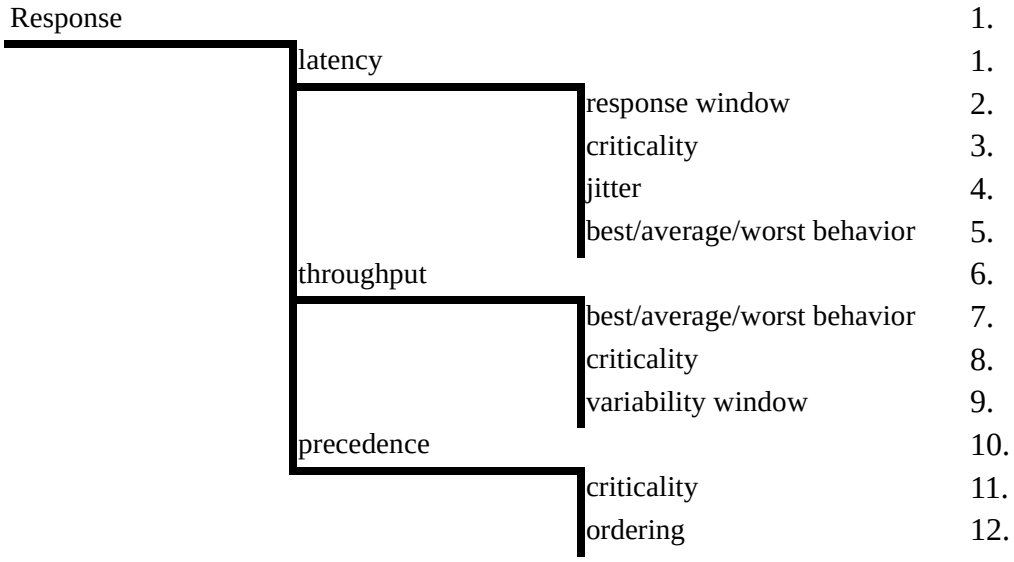

*Table 24: Performance Architecture Mechanisms*

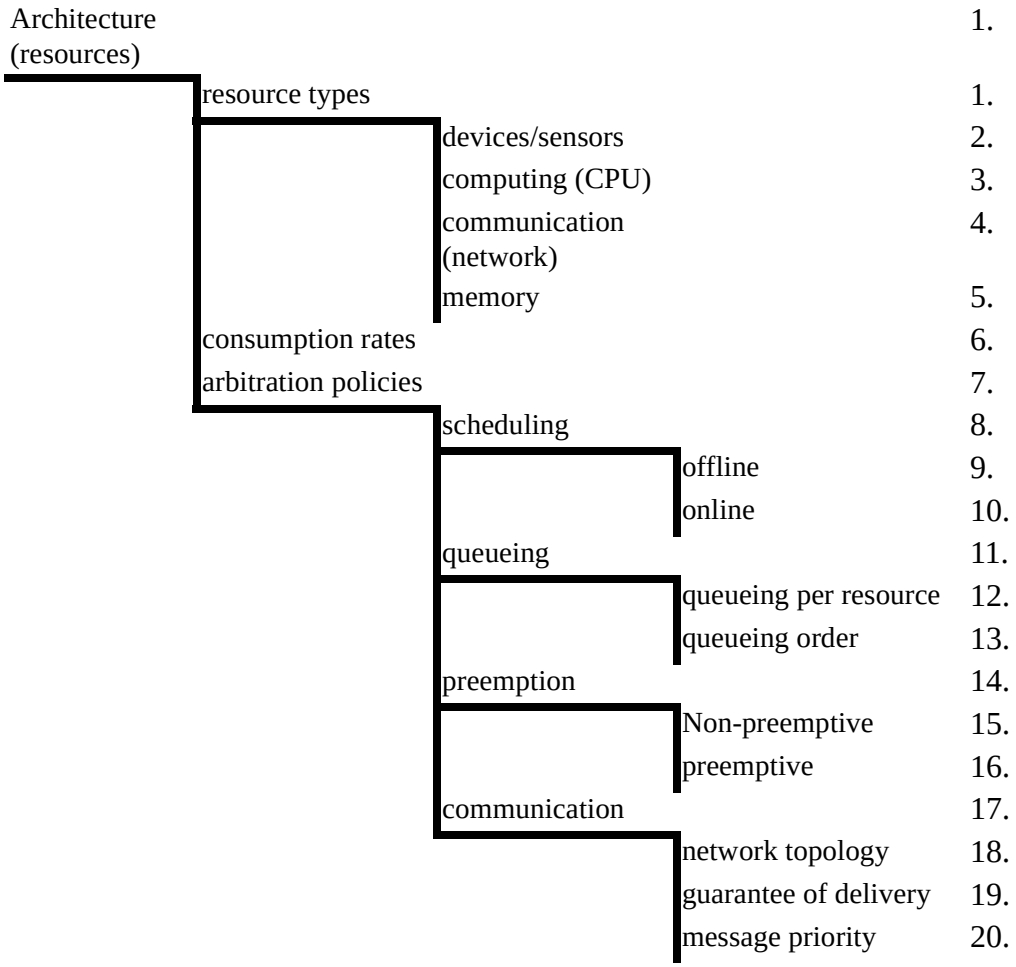

*Table 25: Performance Question*

| | Question | Type | No. |
|---|---|---|---|
| Requirements | Which are the important services from the point of view of performance? | Screen | 1. |
| | What services can operate in degraded performance modes? | Screen | 2. |
| For Important Services | What are these degraded modes (e.g., X% speed)? | Elicitation | 3. |
| | What are the conditions or events that might lead to a performance degradation? | Elicitation | 4. |
| | What are the consequences of not meeting the performance requirements in various degraded modes (e.g., catastrophe, annoyance, minor inconvenience)? | Analysis | 5. |
| | Which are the time-critical components in the system? | Screen | 6. |
| | How do these time-critical components communicate (e.g., protocols, messages received/sent per unit time, message size, processing per message)? | Elicitation | 7. |
| | How do services respond to clients and can they handle multiple clients? | Elicitation | 8. |
| | What are the requirements for which latency is (a) critical and (b) not relevant? | Screen | 9. |
| | How does a particular component or subsystem respond during periods of high stress (e.g., does an overload effect the data flowing through a particular communication link)? | Analysis | 10. |
| | Are there sources of potential resource contention (e.g., specific devices, CPU cycles, network bandwidth, memory usage)? | Analysis | 11. |
| | How does the system arbitrate between components contending for the same resources? Is this accomplished online (i.e., at runtime) or offline (i.e., by precomputing schedules)? | Analysis | 12. |

*Table 26: Performance Scenarios*

| Scenario | Type | No. |
|---|---|---|
| The communications network is overloaded. Reassign missions to reduce traffic. | use case | 1. |
| The LAN on a cutter is overloaded. Reassign tasks to reduce traffic. | use case | 2. |
| Change the process to processor allocation to balance the load. | use case | 3. |
| Data throughput is doubled. | growth | 4. |
| The number of users and/or the number of vessels doubles. | growth | 5. |
| HQ needs real-time video data from a cutter. | explor-atory | 6. |
| Important but not mission-critical traffic doubles in volume. | growth | 7. |

# 8 References


**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-21, ADA307888), Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1995. Available WWW <URL: 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. Available WWW <URL: http://www.sei.cmu.edu/publications/documents/96.reports/96.tr.036.html

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

**C4ISR 97** C4ISR Architectures Working Group, *C4ISR Architecture Framework Version 2.0*. 18 December 1997. Available WWW <URL: http://www.c3i.osd.mil/org/cio/i3/AWG_Digital_Library/pdfdocs/fw.pdf

**C4ISR 98** C4ISR Architectures Working Group, *Levels of Information Systems Interoperability (LISI)*. 30 March 1998. Avalable WWW <URL: http://www.c31.osd.mil/org/cio/i3/AWG_Digital_Library/pdfdocs/lis

**Heimerdinger 92** Heimerdinger, W. L. & Weinstock, C. B. *A Conceptual Framework for System Fault Tolerance*, (CMU/SEI-92-TR-33, ADA258457). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1992. Available <URL: http://www.sei.cmu.edu/publications/documents/92.reports/92.tr.033.html

**Howard 98** Howard, J. & Longstaff, T. *A Common Language for Computer Security Incidents*. Albuquerque, NM: Sandia National Laboratories Report SAND98-8667, Oct. 1998.

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

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

**ISO-9126** ISO Standard 9126, *Information Technology - Software Product Evaluation - Quality Characteristics And Guidelines For Their Use.* Geneva, Switzerland: International Organization For Standardization, 1991.

**Jalote 94** Jalote, P. *Fault Tolerance in Distributed Systems*. Englewood Cliffs, NJ: Prentice Hall, 1994.

**Klein 93** Klein, M., Ralya, T., Pollak, B., Obenza, R., & Gonzales, M. *A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems*. Kluwer Academic, 1993.

**Laprie 92** Laprie, J.C., ed. *Dependable Computing and Fault-Tolerant Systems. Vol. 5, Dependability: Basic Concepts and Terminology in English, French, German, Italian, and Japanese.* New York, NY: Springer-Verlag, 1992.

**Lehoczky 94** Lehoczky, J.P. "Real-Time Resource Management Techniques" *Encyclopedia of Software Engineering***,** Marciniak, J.J (ed.). New York, NY: J. Wiley, 1994. 1011-1020.

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

**Smith 90** Smith, C. U. *Performance Engineering of Software Systems*. The SEI Series in Software Engineering, Reading, MA: Addison-Wesley Publishing Company, 1990.

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*



| | | |
|---|---|---|
| 9. **AGENCY USE ONLY** (leave blank) | 10. **REPORT DATE** July 2000 | 11. **REPORT TYPE AND DATES COVERED** Final |
| 12. **TITLE AND SUBTITLE** **Quality Attribute Workshop Participants Handbook | | 13. **FUNDING NUMBERS** C — F19628-95-C-0003 |
| 14. **AUTHOR(S)** Mario Barbacci, Robert Ellison,Charles Weinstock, William Wood | | |
| 15. **PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)** Software Engineering Institute<br>Carnegie Mellon University<br>Pittsburgh, PA 15213 | | 16. **PERFORMING ORGANIZATION REPORT NUMBER** **CMU/SEI-2000-SR-001 |
| 17. **SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)** HQ ESC/XPK<br>5 Eglin Street<br>Hanscom AFB, MA 01731-2116 | | 18. **SPONSORING/MONITORING AGENCY REPORT NUMBER** **ESC NUMBER |
| 19. **SUPPLEMENTARY NOTES** | | |
| 12.a **DISTRIBUTION/AVAILABILITY STATEMENT** Unclassified/Unlimited, DTIC, NTIS | | 12.b **DISTRIBUTION CODE** |

13. **ABSTRACT** (maximum 200 words)

In large software systems, the achievement of qualities such as performance, security, and modifiability is dependent not only on code-level practices but also on the overall software architecture. Thus, it is in developers' best interests to determine, at the time a system's software architecture is specified, whether the system will have the desired qualities.

With the sponsorship of the U.S. Coast Guard's Deepwater Acquisition Project the SEI has developed the concept of a "Quality Attribute Workshop" in which system stakeholders focus on the analysis and evaluation of system requirements and quality attributes. The purpose of the workshop is to identify scenarios from the point of view of a diverse group of stakeholders and to identify risks (e.g., inadequate performance, successful denial-of-service attacks) and possible mitigation strategies (e.g., replication, prototyping, simulation). Stakeholders include architects, developers, users, maintainers, and people involved in installation, deployment, logistics, planning, and acquisition.This special report describes the process we use to conduct a workshop, information required, suggested tools, and expected outcomes of a workshop.

| | | | |
|---|---|---|---|
| 14. **SUBJECT TERMS** Quality attributes, availability, security, performance, interoperability, attribute architecture mechanisms, attribute stimulus and response. | | | 15. **NUMBER OF PAGES** **PAGE COUNT<br>16. **PRICE CODE** |
| 17. **SECURITY CLASSIFICATION OF REPORT** UNCLASSIFIED | 18. **SECURITY CLASSIFICATION OF THIS PAGE** UNCLASSIFIED | 19. **SECURITY CLASSIFICATION OF ABSTRACT** UNCLASSIFIED | 20. **LIMITATION OF ABSTRACT** UL |