QUality Assessment of System ARchitectures (QUASAR)

Donald Firesmith
Acquisition Support Program (ASP)

Sponsored by the U.S. Department of Defense
© 2006 by Carnegie Mellon University
Topics

What is System Architecture?
Why is System Architecture Critical?
Why Assess the System Architecture?

QUASAR System Architecture Assessment Method:
• Philosophy
• Quality Cases
• QUASAR Process
What is a System Architecture?

Traditional Definition:
the fundamental *structure* of a system in terms of its major components, their relationships to each other and the system’s environment, and the principles governing the creation and evolution of the structure

More General Definition:
the *most important, pervasive, top-level, strategic inventions, decisions, and their associated rationales* about the system including its overall structure (i.e., essential architectural elements, their relationships, and their associated blackbox characteristics and behavior)
Architecture vs. Design

Architecture Decisions:
- **Pervasive** across System Components
- **Strategic** Decisions and Inventions
- **Higher-Levels** of System
- **Huge Impact** on Quality, Cost, and Schedule
- **Drives** Design, Highest-Level Integration, and Integration Testing
- **Driven by Requirements** and Higher-Level Architecture
- **Mirrors** Top-Level Organization of Development Team

Design Decisions:
- **Local** within Individual System Components
- **Tactical** Decisions and Inventions
- **Lower-Levels** of System
- **Smaller Impact** on Quality, Cost, and Schedule
- **Drives** Implementation, Lowest-Level Integration, and Unit Test
- **Driven by Requirements**, Architecture, and Higher-Level Design
Why is Architecture Critical?

Architecture Defines:
- Key System Components
- How Key Components Interact

Architecture Affects:
- Design Decisions
- Implementation Decisions
- Integration Decisions
- Testing Decisions

Architectural Decisions Drive:
- Ultimate System Quality
- Development Costs
- Development Schedule
- Sustainment Costs
- Maintenance and Upgradeability
Why is Architectures Critical?

The quality of the architecture drives the quality of the system:

- Availability
- Interoperability
- Modifiability
- Performance
- Reliability
- Robustness (Error, Failure, and Fault Tolerance)
- Safety
- Security
- Scalability
- Stability
- Testability
- ...

© 2006 by Carnegie Mellon University
Why Assess the Architecture?

Determine System Architecture:
  • Quality
  • Maturity and Completeness
  • Integrity and Consistency
  • Usability

Determine Compliance:
  • Contract Compliance
  • Requirements Compliance

Early Identification of System Architecture Defects:
  • Fix Defects Early
  • Decrease Costs
  • Decrease Schedule
Why Assess the Architecture?

Manage Risks:
- System Architecture Risks
- System Risks

Provide Acquirer Oversight into System Architecture

Develop Consensus:
- Among Developers
- Between Acquirer and Developer Organization

Ensure Specification of Quality Requirements

Help Architects Succeed

Help Program Succeed
How to Assess the Architecture?

Assessment Philosophy
Quality Cases as Foundation
QUASAR Process:
  • Phases
    - System Architecture Assessment Initiation
    - Subsystem Requirements Review
    - Subsystem Architecture Assessment
    - System Architecture Assessment Summary
Assessment Philosophy

Quality Requirements Drive the System Architecture.
Architects should Make Case to Assessors:
• Architects Know Quality Requirement Drivers
• Architects Know What they Did and Why
• Architects Know Where Documented

Safety Cases can Generalize into Quality Cases
(a.k.a., assurance cases) consisting of:
• **Claims:** Architecture Supports Quality Requirements
• **Arguments:** Architects’ Architectural Decisions and Rationales
• **Evidence:** Architects’ Documentation and Witnessed Demonstrations
Assessment Philosophy

Arguments must be Clear and Compelling.
Evidence must be Credible.

Architects’ Responsibilities:
• Prepare Quality Cases
• Provide Early Presentation Materials to Assessors
• Present Quality Cases (Make Case to Assessors)
• Answer Assessors’ Questions

Assessor Responsibilities:
• Prepare for Assessments
• Probe Quality Cases
• Determine and Report Assessment Results
Quality Cases – Quality Model

Quality of a system (and system architecture) is defined in terms of a quality model:

Quality Model

- Quality Factor
  - defines a type of the quality of the
  - 1..*
- Quality Subfactor
  - defines a part of a type of the quality of the
  - 0..* to 1..*
- Quality Measure (Measurement Scale)
  - is measured using a
  - 1..*
- System
  - 1..*
Quality Cases – Quality Factors

Quality Model

- Development-Oriented Quality Factor
- Usage-Oriented Quality Factor

Quality Subfactor

- Capacity
- Configurability
- Dependability
- Efficiency
- Interoperability
- Performance
- Utility
- Defensibility
- Soundness
- Security
- Correctness
- Predictability
- Safety
- Survivability
- Operational Availability
- Reliability
- Stability

Quality Measure (Measurement Scale) is measured using a Performance Utility.
Quality Cases – Quality Subfactors

Safety Subfactor

Harm
Accident & Safety Incident
Nonmalicious Agent
Internal Vulnerability
Hazard
Safety Risk

Problem Type

Solution Type

Prevention
Detection
Reaction
Adaptation

Safety Problem Type
Safety Solution Type

Safety Subfactor
Quality Cases - Components

- **Claims**
  Their architecture adequately supports its derived and allocated quality goals and requirements
- **Clear and compelling Arguments**
  - Architecture decisions
  - Associated rationales
- **Supporting Evidence**
  - Official program documentation
  - Witnessed demonstrations

Simplified version of safety case from safety community
Quality Cases - Relationships

- **Quality Subfactor**
- **Quality Factor**
- **Quality Case**
- **System**
- **Subsystem**
- **Claim**
- **Argument**
- **Evidence**

- Quality Subfactor defines a part of a type of quality of a
- Quality Factor defines a type of quality of a
- Quality Case makes the case for the quality of a
- Claim justifies belief in
- Argument supports
- Evidence
Architecture Quality Cases

Architecture Quality Case

Quality Case

System

Architecture

Subsystem

Claim

Argument

Evidence

Architectural Claim

Architectural Argument

Architectural Evidence

makes the case for the quality of an

has an

justifies belief in

supports

justifies belief in

supports
Architecture Quality Case

Architecture Claim justifies belief in

Architecture Argument supports

Architecture Evidence makes architects' case for

Architecture Quality Case makes for the quality of an

Architecture

Architecture Quality Goal

Quality Goal Claim supports

Quality Requirement Claim supports

Architecture Decision justifies

Rationale

Quality Goal

Quality-Related Requirement ensures achieving

Quality Factor Requirement

Quality Subfactor Requirement

Quality Constraint

Subsystem

System

Architecture

Subsystem has an

Concerns

Official Documentation (e.g., Diagrams, Models, and Documents)

Witnessed Demonstrations (e.g., Scenarios, Tests, and Simulations)

System

Subsystem

Architecture

Architecture Quality Case

Architecture Claim justifies belief in

Architecture Argument supports

Architecture Evidence makes architects' case for

Architecture Quality Case makes for the quality of an

Architecture

Architecture Quality Goal

Quality Goal Claim supports

Quality Requirement Claim supports

Architecture Decision justifies

Rationale

Quality Goal

Quality-Related Requirement ensures achieving

Quality Factor Requirement

Quality Subfactor Requirement

Quality Constraint

Subsystem

System

Architecture

Subsystem has an

Concerns

Official Documentation (e.g., Diagrams, Models, and Documents)

Witnessed Demonstrations (e.g., Scenarios, Tests, and Simulations)

System

Subsystem

Architecture

Architecture Quality Case

Architecture Claim justifies belief in

Architecture Argument supports

Architecture Evidence makes architects' case for

Architecture Quality Case makes for the quality of an

Architecture

Architecture Quality Goal

Quality Goal Claim supports

Quality Requirement Claim supports

Architecture Decision justifies

Rationale

Quality Goal

Quality-Related Requirement ensures achieving

Quality Factor Requirement

Quality Subfactor Requirement

Quality Constraint

Subsystem

System

Architecture

Subsystem has an

Concerns

Official Documentation (e.g., Diagrams, Models, and Documents)

Witnessed Demonstrations (e.g., Scenarios, Tests, and Simulations)

System

Subsystem

Architecture

Architecture Quality Case

Architecture Claim justifies belief in

Architecture Argument supports

Architecture Evidence makes architects' case for

Architecture Quality Case makes for the quality of an

Architecture

Architecture Quality Goal

Quality Goal Claim supports

Quality Requirement Claim supports

Architecture Decision justifies

Rationale

Quality Goal

Quality-Related Requirement ensures achieving

Quality Factor Requirement

Quality Subfactor Requirement

Quality Constraint

Subsystem

System

Architecture

Subsystem has an

Concerns

Official Documentation (e.g., Diagrams, Models, and Documents)

Witnessed Demonstrations (e.g., Scenarios, Tests, and Simulations)

System

Subsystem

Architecture

Architecture Quality Case

Architecture Claim justifies belief in

Architecture Argument supports

Architecture Evidence makes architects' case for

Architecture Quality Case makes for the quality of an

Architecture

Architecture Quality Goal

Quality Goal Claim supports

Quality Requirement Claim supports

Architecture Decision justifies

Rationale

Quality Goal

Quality-Related Requirement ensures achieving

Quality Factor Requirement

Quality Subfactor Requirement

Quality Constraint

Subsystem

System

Architecture

Subsystem has an

Concerns

Official Documentation (e.g., Diagrams, Models, and Documents)

Witnessed Demonstrations (e.g., Scenarios, Tests, and Simulations)

System

Subsystem

Architecture

Architecture Quality Case

Architecture Claim justifies belief in

Architecture Argument supports

Architecture Evidence makes architects' case for

Architecture Quality Case makes for the quality of an

Architecture

Architecture Quality Goal

Quality Goal Claim supports

Quality Requirement Claim supports

Architecture Decision justifies

Rationale

Quality Goal

Quality-Related Requirement ensures achieving

Quality Factor Requirement

Quality Subfactor Requirement

Quality Constraint

Subsystem

System

Architecture

Subsystem has an

Concerns

Official Documentation (e.g., Diagrams, Models, and Documents)

Witnessed Demonstrations (e.g., Scenarios, Tests, and Simulations)

System

Subsystem

Architecture

Architecture Quality Case

Architecture Claim justifies belief in

Architecture Argument supports

Architecture Evidence makes architects' case for

Architecture Quality Case makes for the quality of an

Architecture

Architecture Quality Goal

Quality Goal Claim supports

Quality Requirement Claim supports

Architecture Decision justifies

Rationale

Quality Goal

Quality-Related Requirement ensures achieving

Quality Factor Requirement

Quality Subfactor Requirement

Quality Constraint

Subsystem

System

Architecture

Subsystem has an

Concerns

Official Documentation (e.g., Diagrams, Models, and Documents)

Witnessed Demonstrations (e.g., Scenarios, Tests, and Simulations)

System

Subsystem

Architecture
Interoperability (Quality) Case

Interoperability Case

Interoperability Argument

Interoperability Claim

Evidence

justifies belief in

supports
Quality Case Diagram

Quality Cases contain a large amount of Information. Claims, Arguments, and a large amount of Evidence are typically text. It is easy to get lost in a large, complex, textual quality case. A quality Case Diagram is a layered UML class diagram that labels and summarizes the parts of a single quality case:

- **Claims:**
  - Quality Goals
  - Quality Requirements
- **Arguments:**
  - Architectural Decisions
  - Rationale
- **Evidence:**
  - Documentation
  - Witnessed Demonstrations
Interoperability Quality Case Diagram

Goal: Architecture Supports Interoperability

- Goal: Architecture Supports Physical Interoperability
  - Requirements: Architecture Supports Physical Interoperability
  - Architecture Decision: One-Way Connections

- Goal: Architecture Supports Energy Interoperability
  - Requirements: Architecture Supports Energy Interoperability
  - Architecture Decision: Layered Architecture

- Goal: Architecture Supports Protocol Interoperability
  - Requirements: Architecture Supports Protocol Interoperability
  - Architecture Decision: Open Interface Standards

- Goal: Architecture Supports Syntax Interoperability
  - Requirements: Architecture Supports Syntax Interoperability
  - Architecture Decision: Modular Architecture

- Goal: Architecture Supports Semantic Interoperability
  - Requirements: Architecture Supports Semantic Interoperability
  - Architecture Decision: Service-Oriented Architecture

- Goal: Architecture Supports Protocol Interoperability
  - Requirements: Architecture Supports Protocol Interoperability
  - Architecture Decision: Proxies and Wrappers

- Goal: Architecture Supports Energy Interoperability
  - Requirements: Architecture Supports Energy Interoperability
  - Architecture Decision: Fly-By-Wire

- Goal: Architecture Supports Semantic Interoperability
  - Requirements: Architecture Supports Semantic Interoperability
  - Architecture Decision: One-Way Connections

Evidence:
- Wiring Diagram
- Context Diagram
- Allocation Diagram
- Layer Diagram
- Interoperability Whitepaper
- Vendor-Supplied Technical Documentation
- Hardware Schematics
- Configuration Diagrams
- Network Diagrams
- Activity or Collaboration Diagrams

Supports:
- justifies belief in
- justifies belief in

© 2006 by Carnegie Mellon University Version 0.2 QUASAR Method - page 21
Goal:
Architecture Supports Performance
<<claim>>

Goal:
Architecture Limits Jitter
<<claim>>

Goal:
Architecture Limits Latency
<<claim>>

Goal:
Architecture Limits Latency
<<claim>>

Goal:
Architecture Supports Schedulability
<<claim>>

Goal:
Architecture Supports Throughput
<<claim>>

Goal:
Architecture Limits Response Time
<<claim>>

Goal:
Architecture Limits Response Time
<<claim>>

Goal:
Architecture Supports Throughput
<<claim>>

Requirements:
Architecture Limits Jitter
<<claim>>

Requirements:
Architecture Supports Schedulability
<<claim>>

Requirements:
Architecture Limits Latency
<<claim>>

Requirements:
Architecture Supports Throughput
<<claim>>

Requirements:
Architecture Limits Response Time
<<claim>>

Requirements:
Architecture Supports Throughput
<<claim>>

justifies belief in

Architecture Decision:
Real-Time Operating System
<<argument>>

Architecture Decision:
Deterministic Scheduling
<<argument>>

Architecture Decision:
Layered Architecture
<<argument>>

Architecture Decision:
Redundant Hardware
<<argument>>

Architecture Decision:
Load Balancing
<<argument>>
QUASAR Assessment Process

Four Phases:
1. System Architecture Assessment Initiation (SAAI)
   For each Subsystem to be assessed:
2. Subsystem Requirements Review (SRR)
3. Subsystem Architecture Assessment (SAA)
4. System Architecture Assessment Summary (SAAS)

Each Phase consists of 3 Tasks:
- Preparation
- Meeting
- Follow-Through
QUASAR Phases

System Architecture Assessment Initiation

repeat for each subsystem being assessed

Subsystem Requirements Review → Subsystem Architecture Assessment

no

yes

System Architecture Assessment Summary

done
QUASAR Phases and Tasks

System Architecture Assessment Initiation Phase
Prep. Initial Meeting Follow Through

Subsystem 1 Architecture Assessment

Subsystem Requirements Review Phase
Prep. Rqmts Meeting Follow Through

Subsystem Architecture Assessment Phase
Prep. Arch Meeting Follow Through

Subsystem 2 Architecture Assessment

Subsystem Requirements Review Phase
Prep. Rqmts Meeting Follow Through

Subsystem Architecture Assessment Phase
Prep. Arch Meeting Follow Through

Subsystem N Architecture Assessment

Subsystem Requirements Review Phase
Prep. Rqmts Meeting Follow Through

Subsystem Architecture Assessment Phase
Prep. Arch Meeting Follow Through

System Architecture Assessment Summary Phase
Prep. Final Meeting Follow Through

Time (not to scale) →
Quasar Teams

Requirements Team(s) produce the Architecturally Significant (Quality) Requirements produce the Architecture drive the subsystem architectures produce the System Architecture produces the Top-Level Architecture Team make their lead the Subsystem Architecture Teams make their evaluate the architectures' Assessment Team(s) assess the quality of the architecture
System Architecture Assessment Initiation (SAAI) Phase

repeat for each subsystem being assessed

Subsystem Requirements Review → Subsystem Architecture Assessment

System Architecture Assessment Summary

no

done

yes
SAAI Topics

SAAI Phase Objectives
SAAI Phase Principles
SAAI Phase Context
SAAI Phase Overview
  • SAAI Preparation Task
  • SAAI Meeting Task
  • SAAI Follow-Through Task
SAAI Roles and Responsibilities
Discussion
SAAI Phase Objectives

Prepare the teams

Develop Consensus:
  • Scope the Assessments
  • Schedule the Assessments
  • Tailor the Assessment Process and Training Materials
  • Capture Lessons Learned

Produce and Publish Meeting Outbrief and Minutes

Manage Action Items
SAAI Phase Principles

Need to Develop Consensus between Assessors and Assesses

Need to Tailor Process to meet specific Needs of the Overall Assessment

Scope of Assessment should match Project Needs and Resources

Subsystem Assessments must be scheduled to ensure required Resources
SAAl Phase Context

<table>
<thead>
<tr>
<th>System Architecture Assessment Initiation Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prep.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subsystem 1 Architecture Assessment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subsystem Requirements Review Phase</td>
</tr>
<tr>
<td>Prep.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subsystem 2 Architecture Assessment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subsystem Requirements Review Phase</td>
</tr>
<tr>
<td>Prep.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subsystem N Architecture Assessment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subsystem Requirements Review Phase</td>
</tr>
<tr>
<td>Prep.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subsystem Architecture Assessment Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prep.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>System Architecture Assessment Summary Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prep.</td>
</tr>
</tbody>
</table>

Time (not to scale) →
SAAI Phase Overview

System Architecture Assessment Initiation Phase

- Preparation Task
  - Architecture Assessment Training Materials
  - Architecture Assessment Procedure
- Meeting Task
  - Initial Kick-off Meeting Agenda
  - Initial Kick-off Meeting Notes
  - Architecture Assessment Schedule
  - Architecture Assessment Action Item List
- Follow-Through Task
  - Initial Kick-off Meeting Minutes

Assessment Team

Top-Level Development Team

- Top-Level Development Team discussions
- Top-Level Architecture Team discussions
- Subsystem Architecture Teams discussions
- Top-Level Requirements Team discussions
- Subsystem Requirements Teams discussions
SAAI Preparation Task

Steps:

• Staff Assessment Team
• Train Assessment Team
• Assessment Team Identifies the Top-Level Development Team (Top-Level Requirements Engineers & Architects)
• Assessment Team Trains Top-Level Development Team
• Teams Collaborate to Organize Meeting (attendees, time, location, agenda)
SAAI Meeting Task

Steps:

• Teams Collaborate to Determine Assessment Scope:
  - Architecturally Significant Requirements
  - Subsystems
  - Assessment Resources

• Teams Collaborate to Develop Initial Assessment Schedule

• Teams Collaborate to Tailor Assessment Process

• Assessment Team Manages Action Items
SAAI Follow-Through Task

Steps:

• Assessment Team develops and presents Meeting Outbrief
• Assessment Team develops, reviews, and distributes Meeting Minutes
• Assessment Team tailors and distributes:
  - Assessment Procedure
  - Assessment Training Material
• Assessment Team distributes Assessment Schedule
• Teams obtain Needed Resources
• Assessment Team captures Lessons Learned
• Assessment Team Manages Action Items
SAAI Roles and Responsibilities

The system architecture assessment initiation phase is performed by the following teams:

- Assessment Team
- Top-Level Development Team:
  - Top-Level Requirements Team with input from Subsystem Requirements Teams
  - Top-Level Architecture Team with input from Subsystem Requirements Teams
Assessment Team Membership

Assessment Team Leader
Assessors
Meeting Facilitator
Subsystem Liaison
Subject Matter Experts
Scribe
Assessment Team Responsibilities

Provide Architecture Assessment Training Materials
Provide Architecture Assessment Procedure
Collaborate to Tailor Architecture Assessment Procedure
Collaborate to provide Initial Kick-Off Meeting Agenda
Take Initial Kick-Off Meeting Notes
Collaborate to Develop Assessment Schedule
Produce Architecture Assessment Action Item List
Produce Outbrief and Meeting Minutes
Development Team Membership

Top-Level Requirements Team:
• Lead Requirements Engineer
• System Requirements Engineers
• Subsystem Requirements Engineers

Top-Level Architecture Team:
• Lead System Architect
• System Architects
• Subsystem Architects
Development Team Responsibilities

- Read Architecture Assessment Training Materials
- Read Architecture Assessment Procedure
- Collaborate to Tailor Architecture Assessment Procedure
- Collaborate to provide Initial Kick-Off Meeting Agenda
- Take Initial Kick-Off Meeting Notes
- Collaborate to Develop Assessment Schedule
SAAI Discussion

What is the main objectives of the system architecture assessment initiation phase?
What are the three tasks comprising the SAAI phase?
What teams are involved?
What are the memberships and responsibilities of these teams?
Subsystem Requirements Review (SRR) Phase

System Architecture Assessment Initiation

repeat for each subsystem being assessed

Subsystem Requirements Review

Subsystem Architecture Assessment

done

no

yes

System Architecture Assessment Summary
SRR Topics

SRR Questions for the Attendees
SRR Phase Objectives
SRR Phase Principles
SRR Phase Challenges
SRR Phase Context
SRR Phase Overview
  • SRR Preparation Task
  • SRR Meeting Task
  • SRR Follow-Through Task
SRR Roles and Responsibilities
Discussion
SRR Questions for the Attendees

As a requirements engineer, what are your biggest problems with respect to engineering (e.g., identifying, deriving, analyzing, specifying, and managing) system and subsystem requirements that significantly impact the architecture?

As a system architect, what are your biggest problems with respect to the system requirements that significantly impact the architecture?

As a subsystem architect, what are your biggest problems with respect to the derived architecturally significant requirements that have been allocated to your subsystem?
Key Questions for the Attendees

How can you know the architecture is ‘good enough’ if the requirements do not specify exactly how good it has to be?

• How else can the architects:
  - Make engineering trade-offs among the different quality factors?
  - Know when the architecture is done?
• How can the architecture assessors assess the quality of the architecture without having requirements against which to make the assessment?
• How can testers determine success or failure without measurable test completion criteria?
• How can managers know the quality and status of the architecture without measurable indicators?
SRR Phase Objectives

Ensure that the:

• Architecturally significant requirements are properly engineered in time to support the engineering of the subsystem architecture.

• Subsystem architects know how to prepare for and support the coming subsystem architecture quality assessment.
Architecturally Significant Requirements

Architecturally Significant Requirement
Any requirement that has a significant impact on the system architecture

Quality requirements are typically the most important architecturally significant requirements.

Definition
Any requirement that specifies a minimum level of quality
Quality Requirements

Format
The system shall do X with threshold Y under condition(s) Z.

Bad Example(s)
The system shall be highly reliable, robust, safe, secure, stable, etc.

Good Example (Stability)
The system shall ensure that the mean time between the failure of non-critical functionality* causing the failure of critical functionality* is least 5,000 hours of continuous operation under normal operating conditions*.

* Must be properly defined in the project glossary.
SRR Phase Principles

Not all requirements are architecturally significant. Quality requirements should be major drivers of the system architecture. Quality requirements should specify a minimum required amount of some type of quality. Quality requirements should be:

• Unambiguous
• Feasible
• Complete
• Consistent
• Mandatory
• Verifiable
• Validatable
SRR Phase Principles

Quality requirements should be organized according to a quality model that defines quality factors (a.k.a., attributes, “ilities”) and their quality subfactors:

- Availability
- Interoperability
- Performance
  - Jitter, Response Time, Schedulability, and Throughput
- Portability
- Reliability
- Safety
- Security
- Usability
SRR Phase Principles

Different quality factors are important for different subsystems.

• Performance is paramount for some subsystems.
• Security is more important for other subsystems.

Engineering architecturally significant requirements is the responsibility of the requirements team, not the architecture team and not the assessment team.

• Architects and assessors are not qualified to engineer quality requirements.
• Many stakeholders need quality requirements.
• Architecture assessment time is too late to engineer quality requirements.
SRR Phase Challenges

Architects are rarely given/allocated a *complete* set of architecturally significant requirements. These architecturally significant requirements rarely include quality requirements for *all* of the relevant quality factors and subfactors.
SRR Phase Challenges

The quality of the derived and allocated architecturally significant requirements are typically poor:

- Requirements are often ambiguous.
  - “The system shall be safe and secure.”
- Requirements rarely specify thresholds on relevant quality measurement scales.
  - “The system shall have adequate availability.”
- Requirements are often mutually inconsistent.
  - Security vs. usability, performance vs. reliability.
- Many requirements are infeasible (or at least impractical) if taken literally.
  - “The system shall have 99.99999 reliability.”
Requirements are often unstable.

Specialty engineering requirements (e.g., reliability, safety, security) are often documented separately from the functional requirements.

The architecturally significant requirements are often improperly prioritized for implementation.

The subsystem architects often do not understand how to prepare for an architecture assessment.

- Too busy
- Not trained
- No standards exist
- Bias against assessments/audits
SRR Phase Challenges

The subsystem architects do not understand how to give the assessment team the information they need to assess the architecture:

- How good must the architecture be to *sufficiently* supports its derived and allocated quality requirements (i.e., to ‘pass’ the assessment)?
- What architectural decisions did the architects make to support the quality goals and requirements?
- What were the rationales for these decisions?
- What is the official documentation of actual architectural decisions?
  - Not plans and procedures
  - Official program documentation
  - Not hastily produced PowerPoint slides
SRR Phase Context

<table>
<thead>
<tr>
<th>System Architecture Assessment Initiation Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prep.</td>
</tr>
</tbody>
</table>

**Subsystem 1 Architecture Assessment**

<table>
<thead>
<tr>
<th>Subsystem Requirements Review Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prep.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subsystem Architecture Assessment Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prep.</td>
</tr>
</tbody>
</table>

**Subsystem 2 Architecture Assessment**

<table>
<thead>
<tr>
<th>Subsystem Requirements Review Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prep.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subsystem Architecture Assessment Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prep.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subsystem N Architecture Assessment</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Subsystem Requirements Review Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prep.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subsystem Architecture Assessment Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prep.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>System Architecture Assessment Summary Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prep.</td>
</tr>
</tbody>
</table>

Time (not to scale) ——>
SRR Preparation Task

Steps:

• Subsystem Requirements Team provides access to the architecturally significant subsystem requirements as well as a summary of these requirements

• Subsystem Architecture Team provides sample of planned Quality Cases

• Subsystem Assessment Team reviews this information prior to the meeting
SRR Meeting Task

Steps:

• Subsystem Requirements Team presents Summary of the architecturally significant subsystem requirements (organized by quality factor and quality subfactors)

• Subsystem Assessment Team recommends Improvements

• Subsystem Architecture Team presents sample of planned Quality Cases

• Subsystem Assessment Team recommends Improvements

• Assessment Team Manages Action Items
SRR Follow-Through Task

Steps:

• Subsystem Assessment Team presents Outbrief
• Subsystem Assessment Team develops and publishes Meeting Minutes containing recommendations for improving:
  • Architecturally significant subsystem requirements
  • Quality Cases
• Assessment Team tailors and distributes updated Assessment Procedure and Assessment Training Material (for future requirements reviews)
• Assessment Team captures Lessons Learned
• Assessment Team Manages Action Items
SRR Roles and Responsibilities

The subsystem requirements review phase is performed by the following three teams:

- Subsystem Requirements Team
- Subsystem Architecture Team
- Subsystem Assessment Team
Subsystem Requirement Team

Responsibilities:

- Work with specialty engineering teams to engineer the architecturally significant subsystem requirements
- Provide these requirements to the subsystem architecture team in time to drive the subsystem architecture
- Provide the subsystem assessment team with access to these requirements sufficiently prior to the meeting
- Summarize these requirements at the requirements review meeting
- Answer questions from the assessment team (and architecture team)
Subsystem Architecture Team

Responsibilities:

- Develop a proposed representative sample of the architectural information to be presented during the coming subsystem architecture assessment meeting:
  - Architectural decisions and rationale
  - Supporting evidence
- Present this information to the subsystem assessment team
- Ask questions (if necessary) of the:
  - Subsystem requirements team (regarding architecturally significant requirements)
  - Subsystem assessment team (regarding the assessment process and adequacy of proposed sample architectural decisions, rationale, and evidence)
Subsystem Assessment Team

Responsibilities:
• Review supplied information prior to the requirements review meeting
• Ensure that the architecturally significant requirements are adequately engineered to support the subsystem architecture assessment.
• Ensure that the proposed architectural information to will be adequate to support the coming subsystem architecture assessment meeting
• Answer questions from and provide advice to the:
  - Requirements team regarding the architecturally significant requirements
  - Architecture team regarding what will be expected of them during the coming subsystem architecture assessment meeting
Subsystem Assessment Team

Responsibilities:
- Must include members having expertise in:
  - Requirements engineering and quality requirements
  - The system architecture quality assessment method (with all members having been trained in the method)
- Should include members having experience in the subsystem application domain(s) such as avionics, sensors, or weapons
SRR Discussion Questions

What is the two main objectives of the subsystem requirements review?

How often should subsystem requirements reviews be performed?

When should subsystem requirements reviews be performed?

What are the three tasks comprising the subsystem requirements review?

What are the objectives of these three tasks?

What teams are involved?

What are the responsibilities of these teams?
Subsystem Architecture Assessment (SAA)

System Architecture Assessment Initiation

Subsystem Requirements Review

Subsystem Architecture Assessment

repeat for each subsystem being assessed

no

done

yes

System Architecture Assessment Summary
SAA Topics

SAA Questions for the Attendees
SAA Phase Objectives
SAA Phase Principles
SAA Phase Challenges
SAA Phase Context
SAA Phase Overview:
  • SAA Preparation Task
  • SAA Meeting Task
  • SAA Follow-Through Task
SAA Roles and Responsibilities
Discussion
SAA Questions for the Attendees

As a subsystem architect, what are your biggest problems with respect to:

- Engineering the subsystem architecture?
- Ensuring that the subsystem architecture adequately meets its architecturally significant requirements?
- Internally reviewing/evaluating the quality of the subsystem architecture?
- Supporting independent assessments of the quality of your subsystem architecture?

As an independent assessor (e.g., PO of prime contractor, prime contractor of subcontractor), what are your biggest problems with respect to independently assessing the quality of an acquired subsystem’s architecture?
SAA Questions for the Attendees

Is the quality of your architectures being independently assessed?
How are your architectures being assessed?
Who is assessing your architectures?
What do you see as the biggest problems with respect to how your architectures are being assessed?
  • Are your assessors using an effective and efficient process for assessing your architectures?
  • Do you know what is expected of you during the system architecture assessments?
  • Do you develop adequate documentation as a natural part of the architecture process?
  • Is the architecture documentation you develop adequate to support assessments?
SAA Objectives

Assess Quality of Subsystem Architecture in terms of:

- Architectures support for its derived and allocated architecturally significant requirements
- Architectural Quality Cases
SAA Principles

Quality architecture assessments should be organized according to a quality model that defines quality factors (a.k.a., attributes, “ilities”) and their quality subfactors:

- Availability
- Interoperability
- Performance
  - Jitter, Response Time, Schedulability, and Throughput
- Portability
- Reliability
- Safety
- Security
- Usability
SAA Principles

The subsystem architects should know:

- What quality goals and requirements drove the development of their architectures.
- What architectural decisions they made.
- Why they made these decisions.
- Where these decisions are documented.

Because the subsystem architects should already have documented this information as a natural part of their architecting method, little new documentation should be necessary for the subsystem architects to make their cases to the subsystem assessment team.

The subsystem architects are responsible for making their own cases that their architectures adequately support their derived and allocated quality requirements.
SAA Phase Challenges

Architects may not have developed quality cases as a natural part of their architecting process:
  • Architectural documentation typically not organized by quality factors.
  • Quality case evidence is often buried in and scattered throughout massive amounts of architectural documentation.
  • Architectural models (e.g., UML) often do not address support for quality requirements.

Architecture assessments may not be:
  • Mandated by contract or development process
  • Scheduled and funded

Managers feel schedule pressures do not allow time for assessment.
## SAA Context

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Time (not to scale)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prep.</td>
<td>Initial Meeting</td>
<td></td>
</tr>
<tr>
<td>Arch.</td>
<td>Follow Through</td>
<td></td>
</tr>
</tbody>
</table>

### Subsystem Architecture Assessment

<table>
<thead>
<tr>
<th>Subsystem 1</th>
<th>Subsystem Requirements Review Phase</th>
<th>Subsystem Architecture Assessment Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prep.</td>
<td>Rqmts. Meeting</td>
<td>Arch. Meeting</td>
</tr>
<tr>
<td></td>
<td>Follow Through</td>
<td>Follow Through</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subsystem 2</th>
<th>Subsystem Requirements Review Phase</th>
<th>Subsystem Architecture Assessment Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prep.</td>
<td>Rqmts. Meeting</td>
<td>Arch. Meeting</td>
</tr>
<tr>
<td></td>
<td>Follow Through</td>
<td>Follow Through</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subsystem N</th>
<th>Subsystem Requirements Review Phase</th>
<th>Subsystem Architecture Assessment Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prep.</td>
<td>Rqmts. Meeting</td>
<td>Arch. Meeting</td>
</tr>
<tr>
<td></td>
<td>Follow Through</td>
<td>Follow Through</td>
</tr>
</tbody>
</table>
SAA Phase Overview
SAA Preparation Task

Steps:

• Subsystem Assessment Team Provides Assessment Checklist

• Subsystem Architecture Team Gathers (Generates) and Makes Available *Preparatory* Materials:
  • Subsystem Architecture Overview
  • Updated Quality Requirements
  • Quality Cases including Arguments and Evidence

• Subsystem Architecture Team Gathers (Generates) and Makes Available *Presentation* Materials

• Subsystem Assessment Team:
  • Reads Materials
  • Generates RFIs and RFAs

• Teams Collaborate to Organize Assessment Meeting (Attendees, Time, Location, Agenda, Invitation)
SAA Meeting Task

Steps:

• **Subsystem Architecture Team:**
  • Introduces Subsystem Architecture
    (purpose, location, context, functions)
  • Reviews Architecturally-Significant Requirements
  • Introduces Subsystem Architecture
    (components, relationships, major decisions, trade-offs)
  • Present Quality Cases
    (claims, arguments, and evidence)

• **Subsystem Assessment Team:**
  • Probes Architecture (quality case by quality case)
  • Manages Action Items
SAA Follow-Through Task

Steps:

- Subsystem Assessment Team:
  - Develops Consensus
  - Produces, Reviews, and Presents Meeting Outbrief
  - Produces, Reviews, and Presents Subsystem Assessment Report
  - Manages Action Items
  - Captures Lessons Learned
  - Updates Assessment Method and Training Materials
SAA Roles and Responsibilities

The Subsystem Architecture Assessment Phase is performed by the following teams:
- Subsystem Architecture Team
- Subsystem Assessment Team
Subsystem Architecture Team

Responsibilities:
• Develop the architectural information to be presented during the meeting:
  - Architectural decisions and rationale
  - Supporting evidence
• Present this information to the subsystem assessment team
• Answer probing questions raised by the subsystem assessment team:
Subsystem Assessment Team

Responsibilities:
- Review supplied information prior to the subsystem architecture assessment meeting
- Assess the quality of the subsystem architecture:
  - Actively listen to the quality cases presented by the subsystem architecture team
  - Ask probing questions of Architects
SAA Discussion

Should the quality cases be developed as a:
- Natural part of the architecting process?
- Part of the assessment process?

How does the answer to the previous question affect the amount of time needed to prepare for the assessment meeting?

Which team has the most work to do during each task?

How should the development of the subsystem assessment report be divided up between members of the assessment team?
System Architecture Assessment Summary (SAAS)

System Architecture Assessment Initiation

repeat for each subsystem being assessed

Subsystem Requirements Review → Subsystem Architecture Assessment

no ➔ System Architecture Assessment Summary

done ➔ yes
SAAS Topics

SAAS Questions for the Attendees
SAAS Phase Objectives
SAAS Phase Principles
SAAS Phase Challenges
SAAS Phase Context
SAAS Phase Overview:
  • SAAS Preparation Task
  • SAAS Meeting Task
  • SAAS Follow-Through Task
SAAS Roles and Responsibilities
Discussion
SAAS Questions for the Attendees

How do you summarize the results of subsystem assessments at the system level?
Should the system architecture assessment summary phase be performed:
- Once at the end?
- On an ongoing rolling-wave basis?
SAAS Objectives

Collect previous Subsystem Architecture Assessment Results
Create System Architecture Assessment Summarize Results
Capture Method Lessons Learned
Update Assessment Method and Training Materials
SAAS Principles

All subsystems are not equally important.
All quality factors are not equally important for different subsystems.
It is probably better to concentrate on identifying problem/risk areas so that they can be fixed than to provide an overall summary assessment result.
SAAS Phase Challenges

How should subsystem findings be summarized without ending up comparing apples and oranges?

- Average Subsystem Architecture Quality
- Worst Subsystem Architecture Quality
- Union of Subsystem Architecture Qualities

Executive management may demand simplistic single number summary of system architecture.
### SAAS Context

#### System Architecture Assessment Initiation Phase

<table>
<thead>
<tr>
<th>Prep.</th>
<th>Initial Meeting</th>
<th>Follow Through</th>
</tr>
</thead>
</table>

#### Subsystem Architecture Assessment Phase

<table>
<thead>
<tr>
<th>Subsystem</th>
<th>Subsystem Requirements Review Phase</th>
<th>Subsystem Architecture Assessment Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subsystem 1</td>
<td>Prep.</td>
<td>Rqmts. Meeting</td>
</tr>
<tr>
<td>Subsystem 2</td>
<td>Prep.</td>
<td>Rqmts. Meeting</td>
</tr>
<tr>
<td>Subsystem N</td>
<td>Prep.</td>
<td>Rqmts. Meeting</td>
</tr>
</tbody>
</table>

#### System Architecture Assessment Summary Phase

<table>
<thead>
<tr>
<th>Prep.</th>
<th>Final Meeting</th>
<th>Follow Through</th>
</tr>
</thead>
</table>

---

Time (not to scale) ➡️
SAAS Phase Overview
**SAAS Preparation Task**

**Steps:**
- **System Assessment Team:**
  - Collects Subsystem Architecture Assessment Results
  - Summarizes Subsystem Architecture Assessment Results
    - Develops Subsystem Architecture Support Matrix
  - Identifies Primary Stakeholders
- Produces, Reviews, and Distributes:
  - System Architecture Quality Assessment Summary Report
  - Preparatory Materials
  - Meeting Agenda
- Organizes Meeting
SAAS Meeting Task

Steps:

• System Assessment Team:
  • Restates Assessment Objectives
  • Summarizes Assessment Method
  • Summarizes Quality of Subsystem Architectures
  • Summarizes Quality of System Architecture
  • Solicits Feedback
  • Captures Lessons Learned

• System Architecture Team:
  • Captures Lessons Learned
SAAS Follow-Through Task

Steps:

• System Assessment Team:
  • Updates and Distributes the System Architecture Assessment Summary Report
  • Manages Action Items
  • Updates Assessment Method and Training Materials
• System Architecture Team:
  • Updates Architecture Method and Training Materials
SAAS Responsibilities

System Assessment Team:
• Develop and Present System-Level Architecture Assessment Summary Results
• Capture Lessons Learned
• Update Assessment Method and Training Materials

System Architecture Team:
• Validate Assessment Results
• Capture Lessons Learned
• Update Architecture Method and Training Materials

Management Team:
• Manage Architectural Risks
SAAS Discussion

For a given quality factor, what is the best way to summarize the quality of the system architecture in terms of the quality of the architecture of the main subsystems?

• Average subsystem quality?
• Worst subsystem quality?
• Keep separate by listing individually?

What is the best way to summarize across all quality factors?

• Average value?
• Worst value?
• Keep separate by listing individually?
QUASAR Today and Tomorrow

Today:
• In-use on massive DoD Program
• Handbook published
• Provided as SEI Service

Future Plans:
• More Conference Tutorials
• QUASAR Training Materials and Classes
• QUASAR Articles
• Use and Validation on more Programs
• QUASAR Book
QUASAR Handbook

Intended Audiences:
• Acquisition Personnel
• Developers (Architects and Requirements Engineers)
• Subject Matter Experts (domain, specialty engineering)
• Consultants
• Trainers

Objectives:
• Completely Document the QUASAR method
• Enable Readers to start using QUASAR

Description:
• Very Complete
• Too comprehensive to be good first introduction
Contact Information

For more information, contact:

Donald Firesmith
Acquisition Support Program
Software Engineering Institute
dgf@sei.cmu.edu