Carnegie Mellon
Software Engineering Institute
Pittsburgh, PA 15213-3890

QUality Assessment of System ARchitectures (QUASAR)

Donald Firesmith
Acquisition Support Program (ASP)

Sponsored by the U.S. Department of Defense

Carnegie Mellon
Software Engineering Institute

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

Carnegie Mellon
Software Engineering Institute

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)

Carnegie Mellon
Software Engineering Institute

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

Carnegie Mellon
Software Engineering Institute

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

Carnegie Mellon
Software Engineering Institute

Why is Architectures Critical?$_2$

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
- …

Carnegie Mellon
Software Engineering Institute

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

Carnegie Mellon
Software Engineering Institute

Why Assess the Architecture?$_2$

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

Carnegie Mellon
Software Engineering Institute

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

Carnegie Mellon
Software Engineering Institute

Assessment Philosophy$_2$

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

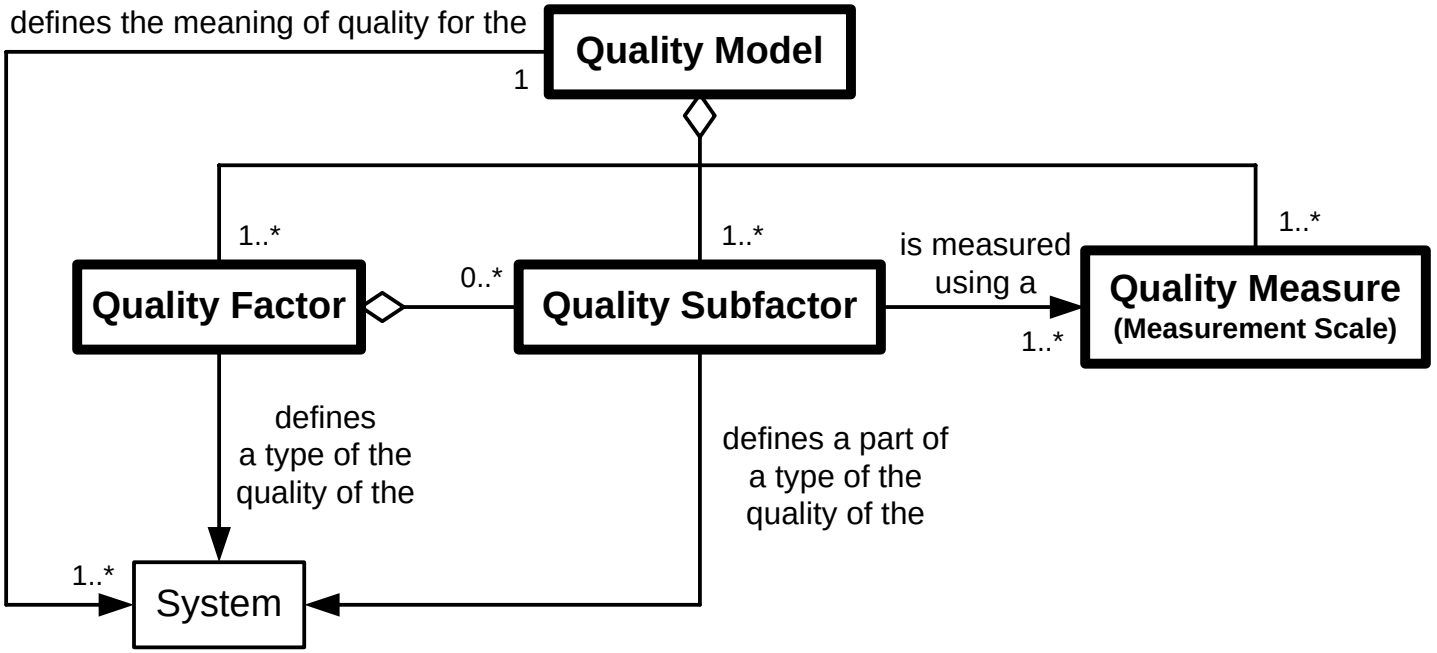
Assessor Responsibilities:

- Prepare for Assessments
- Probe Quality Cases
- Determine and Report Assessment Results

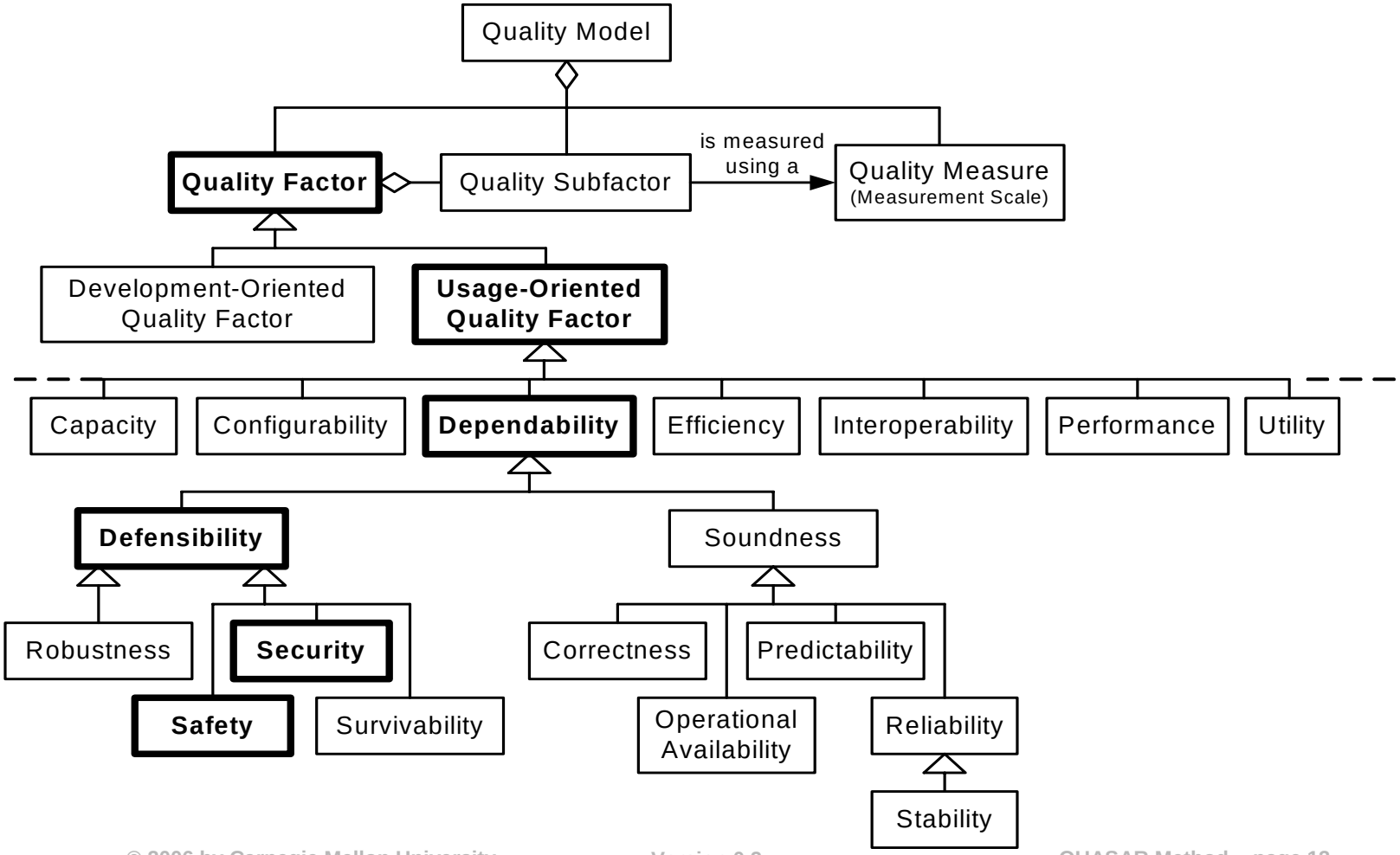
Carnegie Mellon
Software Engineering Institute

Quality Cases – Quality Model

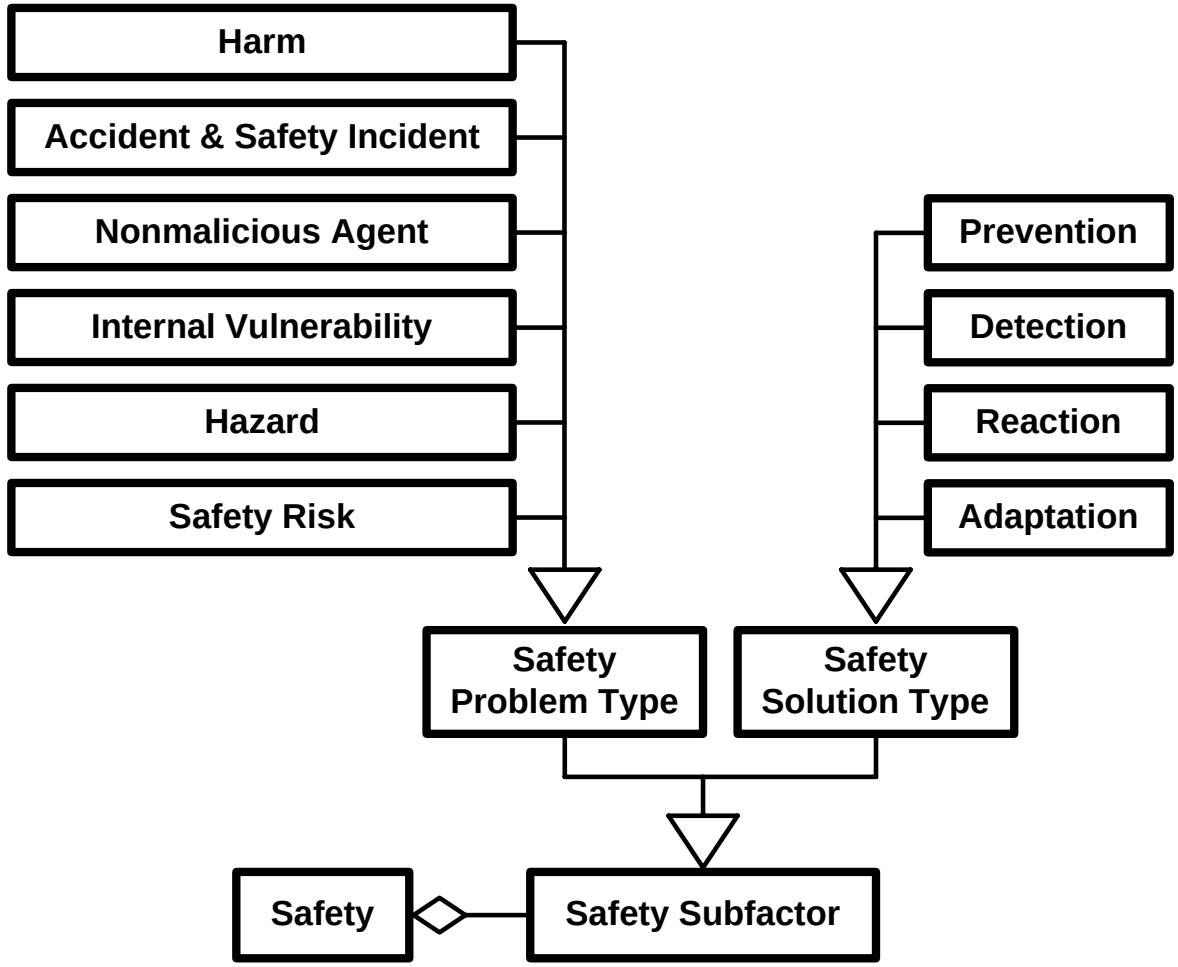
Quality of a system (and system architecture) is defined in terms of a quality model:



Quality Cases – Quality Factors



Quality Cases – Quality Subfactors



Carnegie Mellon
Software Engineering Institute

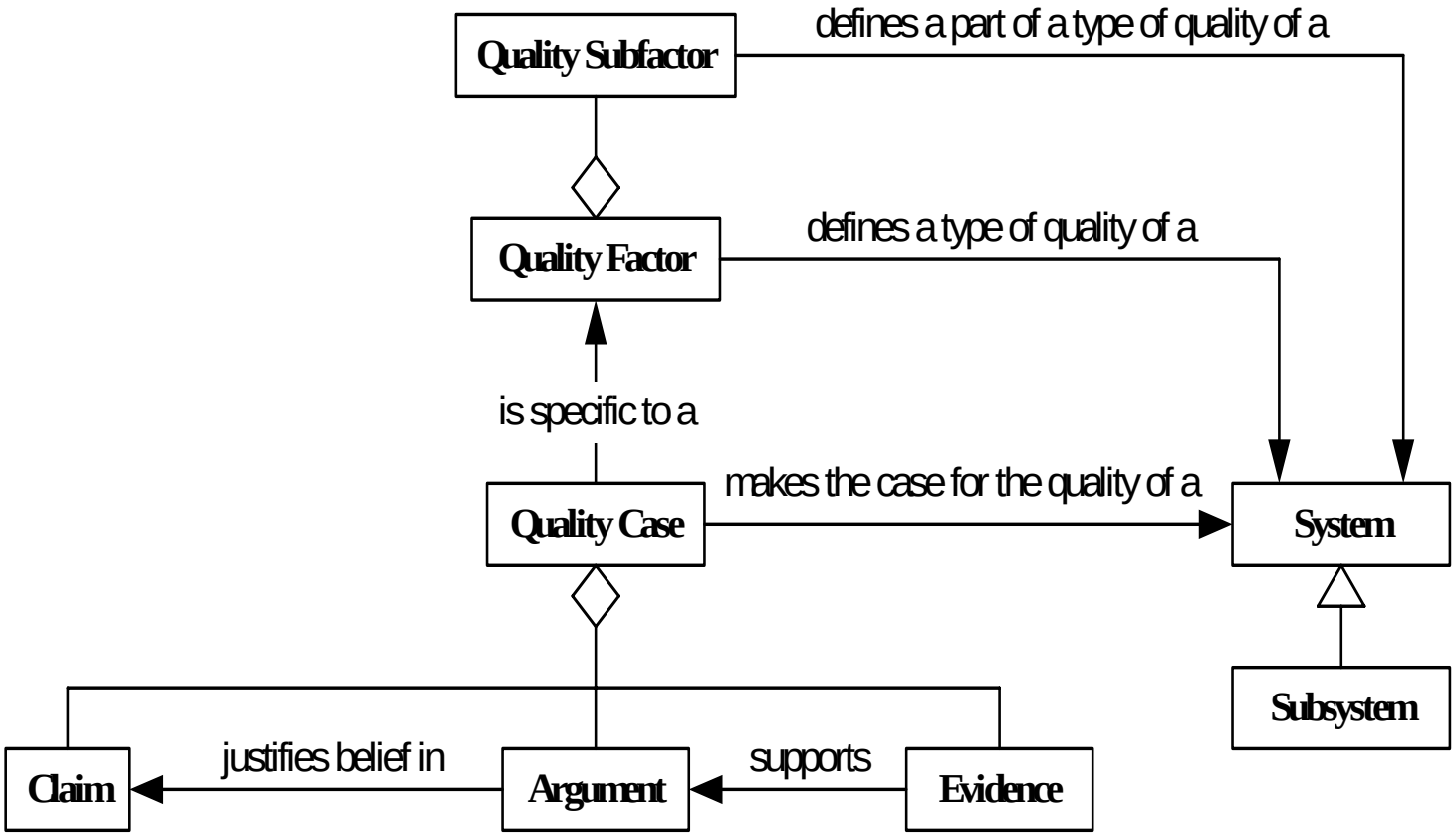
Quality Cases - Components

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

Simplified version of safety case from safety community

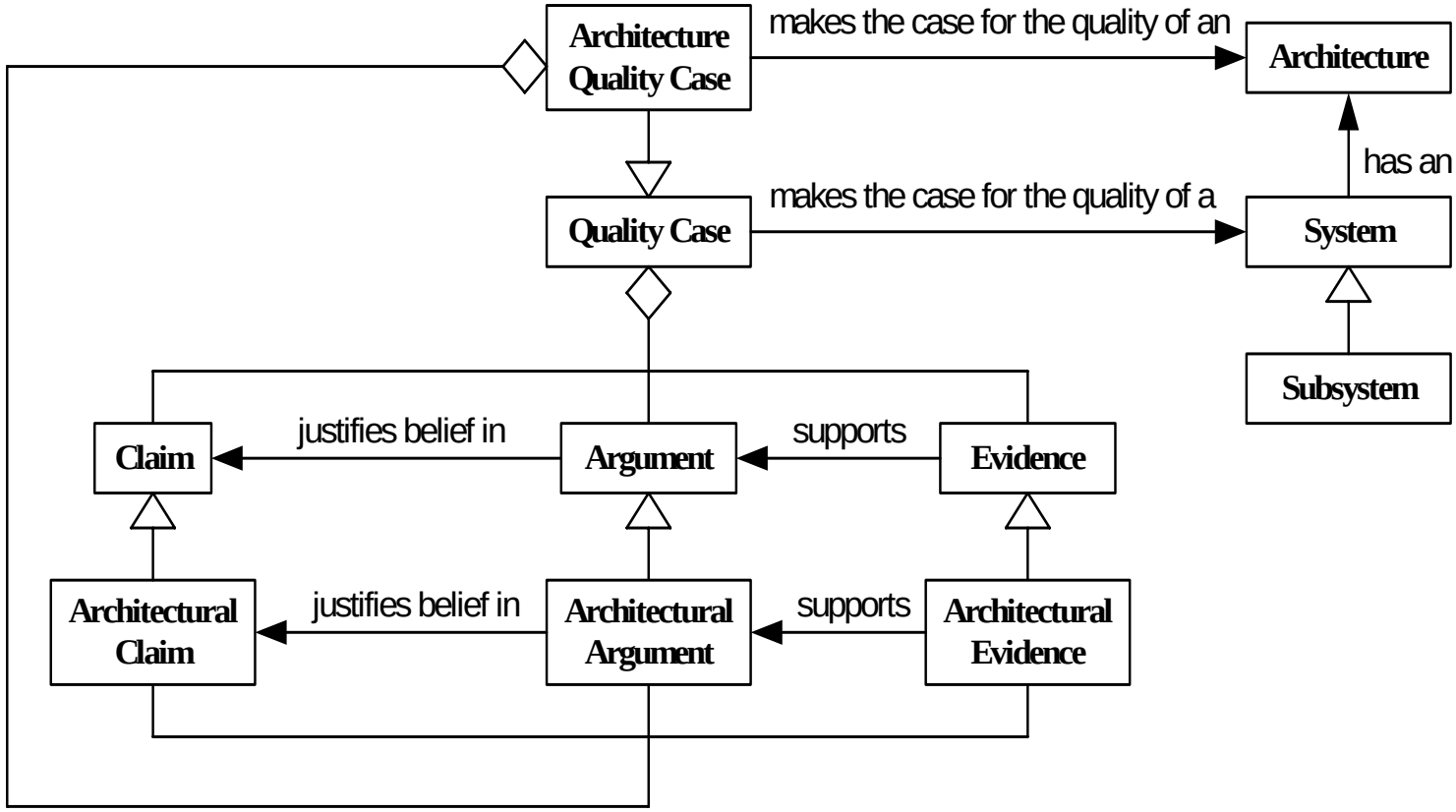
Carnegie Mellon
Software Engineering Institute

Quality Cases - Relationships

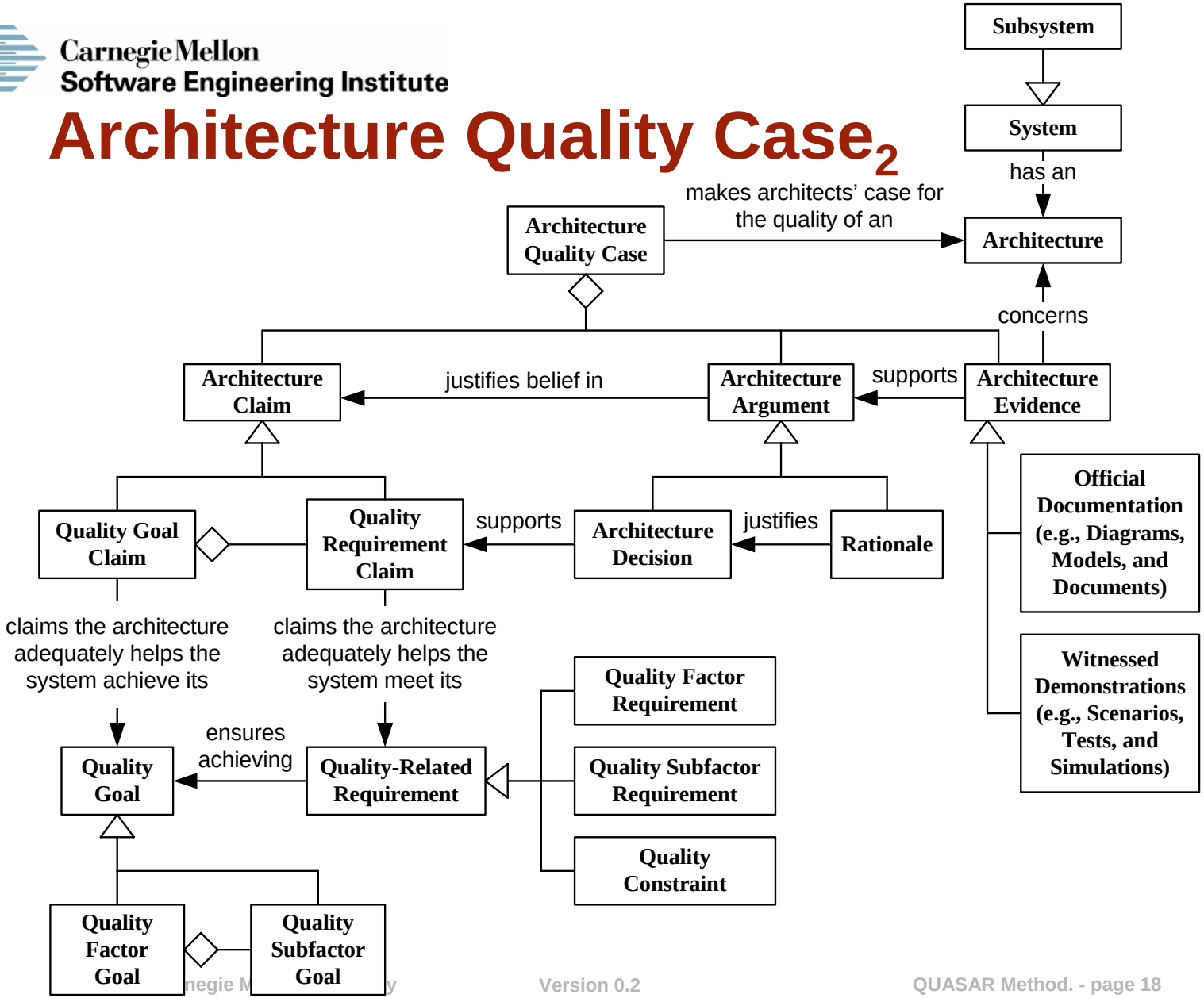


Carnegie Mellon
Software Engineering Institute

Architecture Quality Cases

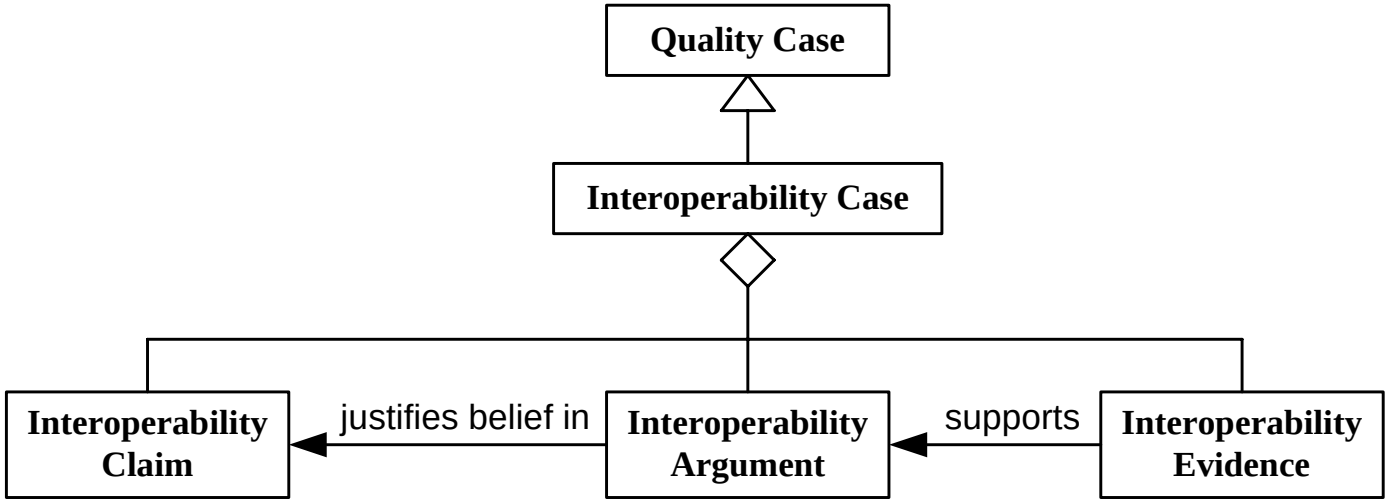


Architecture Quality Case$_2$



Carnegie Mellon
Software Engineering Institute

Interoperability (Quality) Case



Carnegie Mellon
Software Engineering Institute

Quality Case Diagram

Quality Cases contain a large amount of Information.

Claims, Arguments, and a large amount of Evidence are typically text.

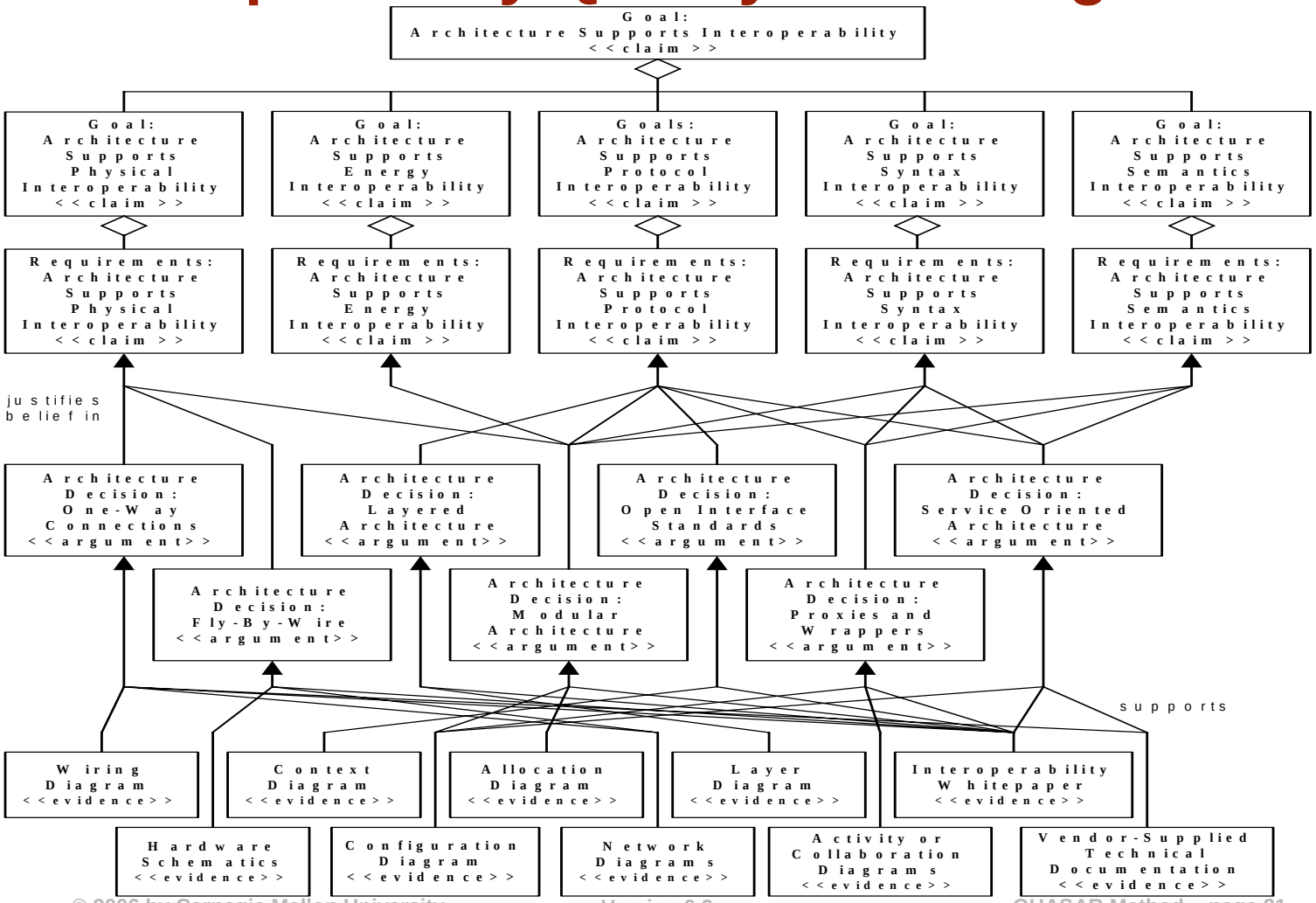
It is easy to get lost in a large, complex, textual quality case.

A quality Case Diagram is a layered UML class diagram that labels and summarizes the parts of a single quality case:

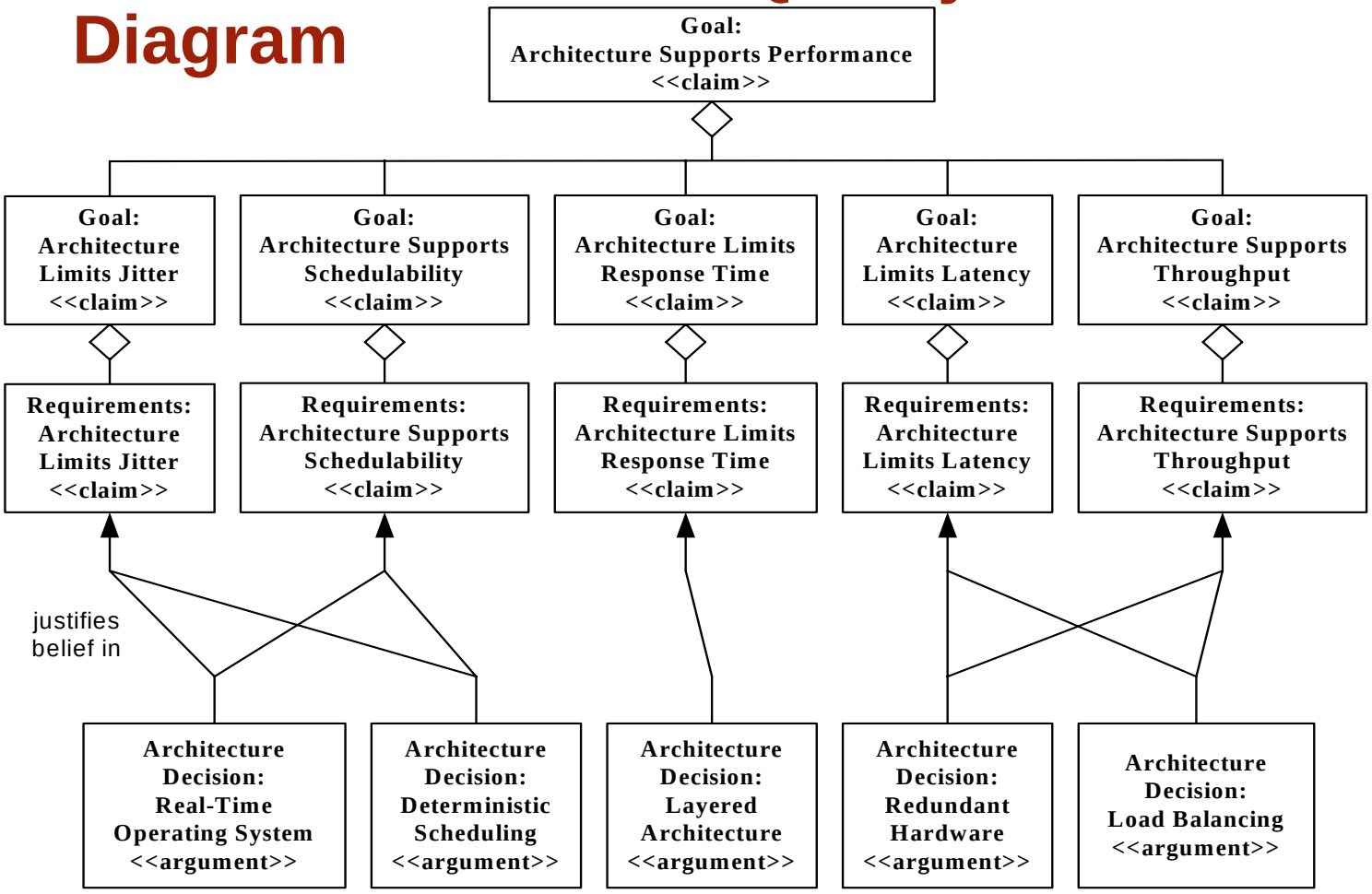
- Claims:
  - Quality Goals
  - Quality Requirements
- Arguments:
  - Architectural Decisions
  - Rationale
- Evidence:
  - Documentation
  - Witnessed Demonstrations

Carnegie Mellon
Software Engineering Institute

Interoperability Quality Case Diagram



Partial Performance Quality Case Diagram



Carnegie Mellon
Software Engineering Institute

QUASAR Assessment Process

Four Phases:

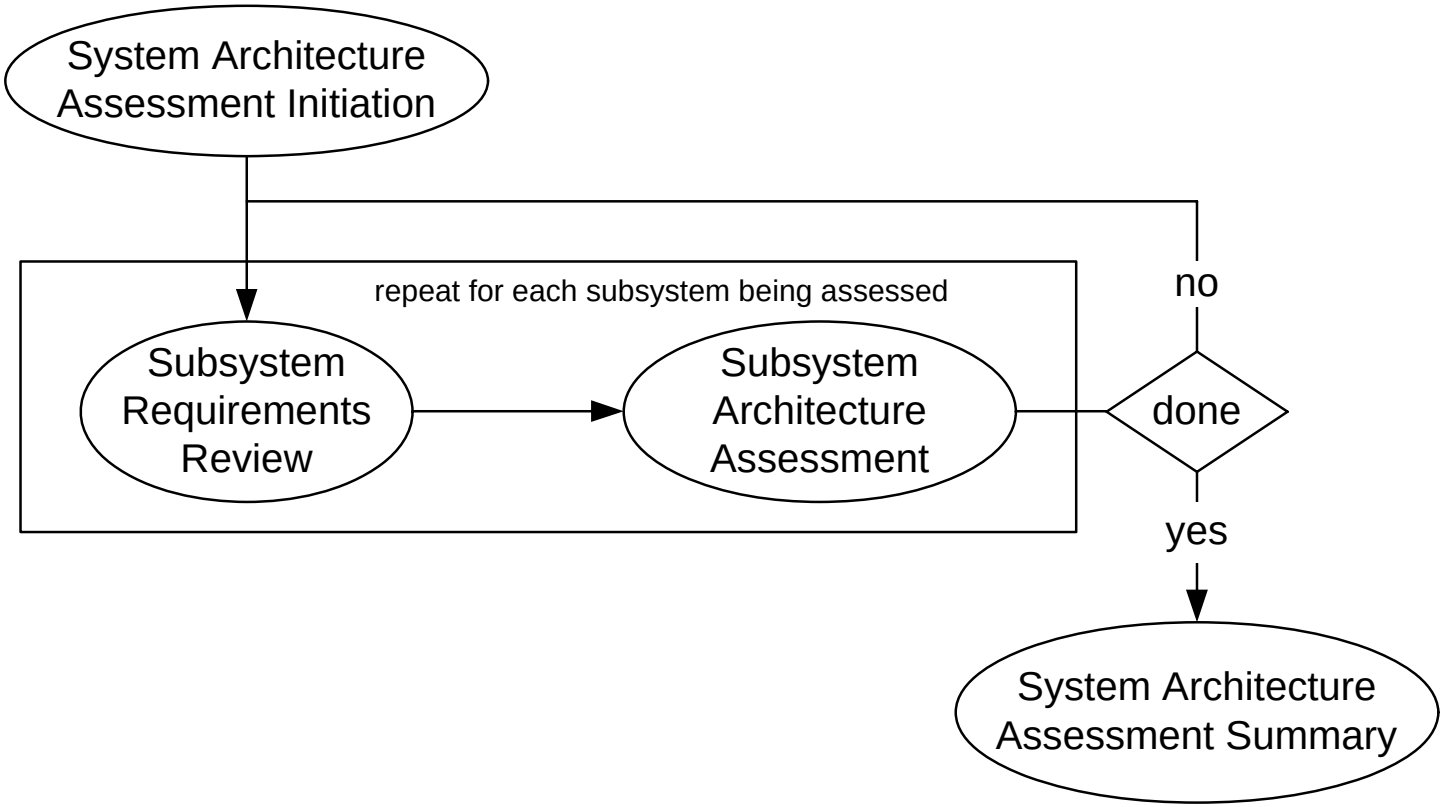
1. System Architecture Assessment Initiation (SAAI)
   For each Subsystem to be assessed:
   2. Subsystem Requirements Review (SRR)
   3. Subsystem Architecture Assessment (SAA)
4. System Architecture Assessment Summary (SAAS)

Each Phase consists of 3 Tasks:

1. Preparation
2. Meeting
3. Follow-Through

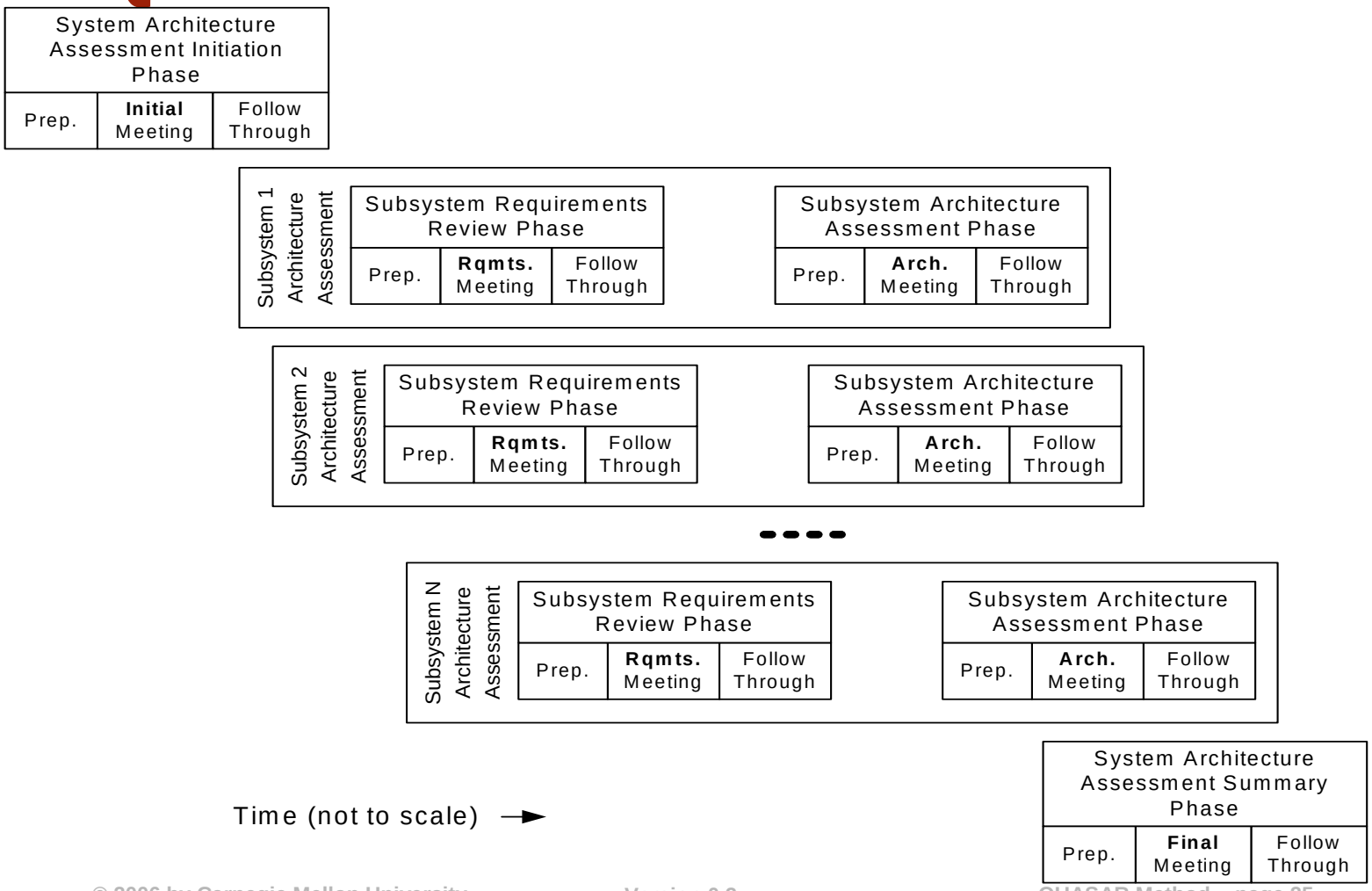
Carnegie Mellon
Software Engineering Institute

QUASAR Phases



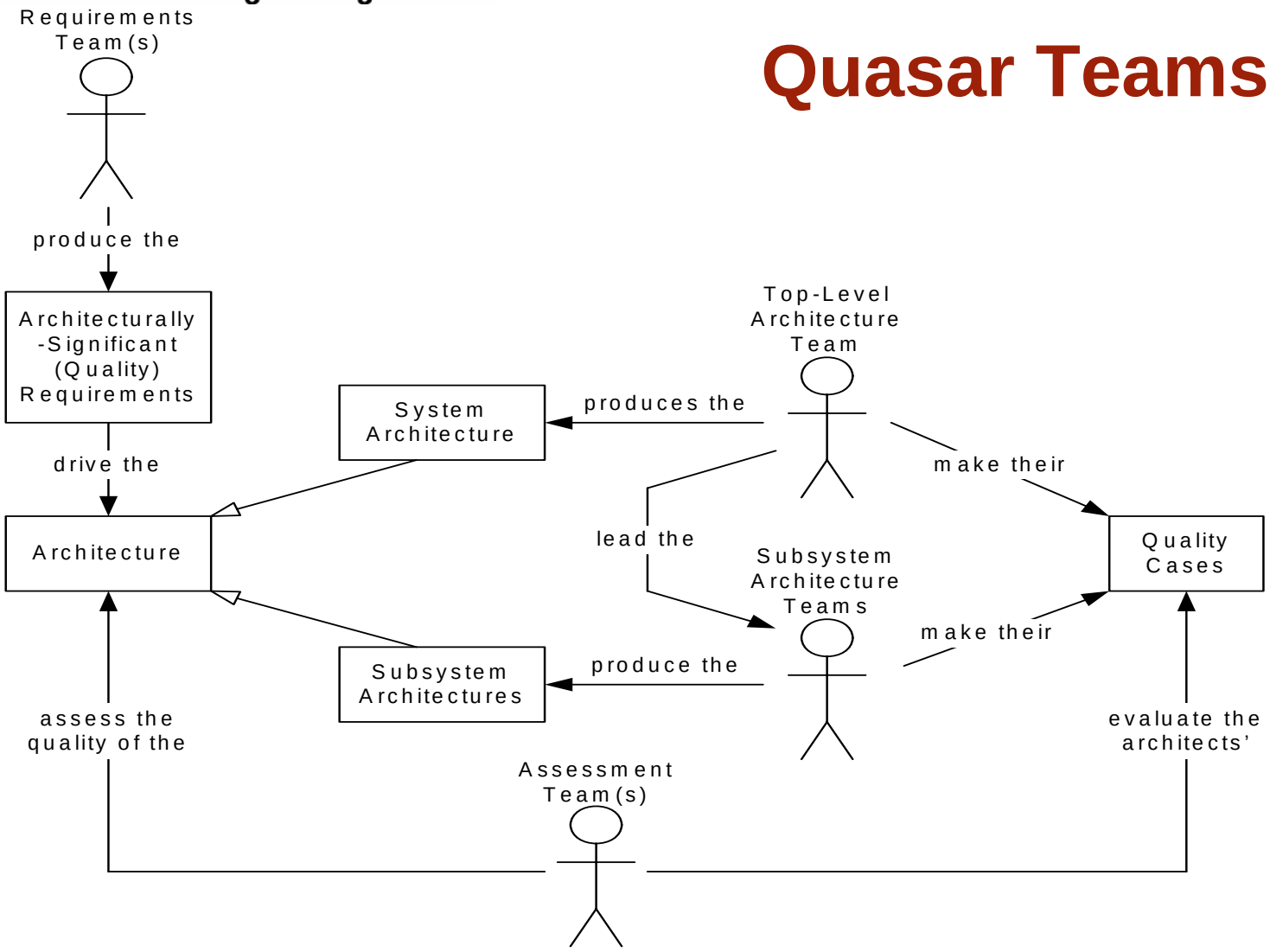
Carnegie Mellon
Software Engineering Institute

QUASAR Phases and Tasks



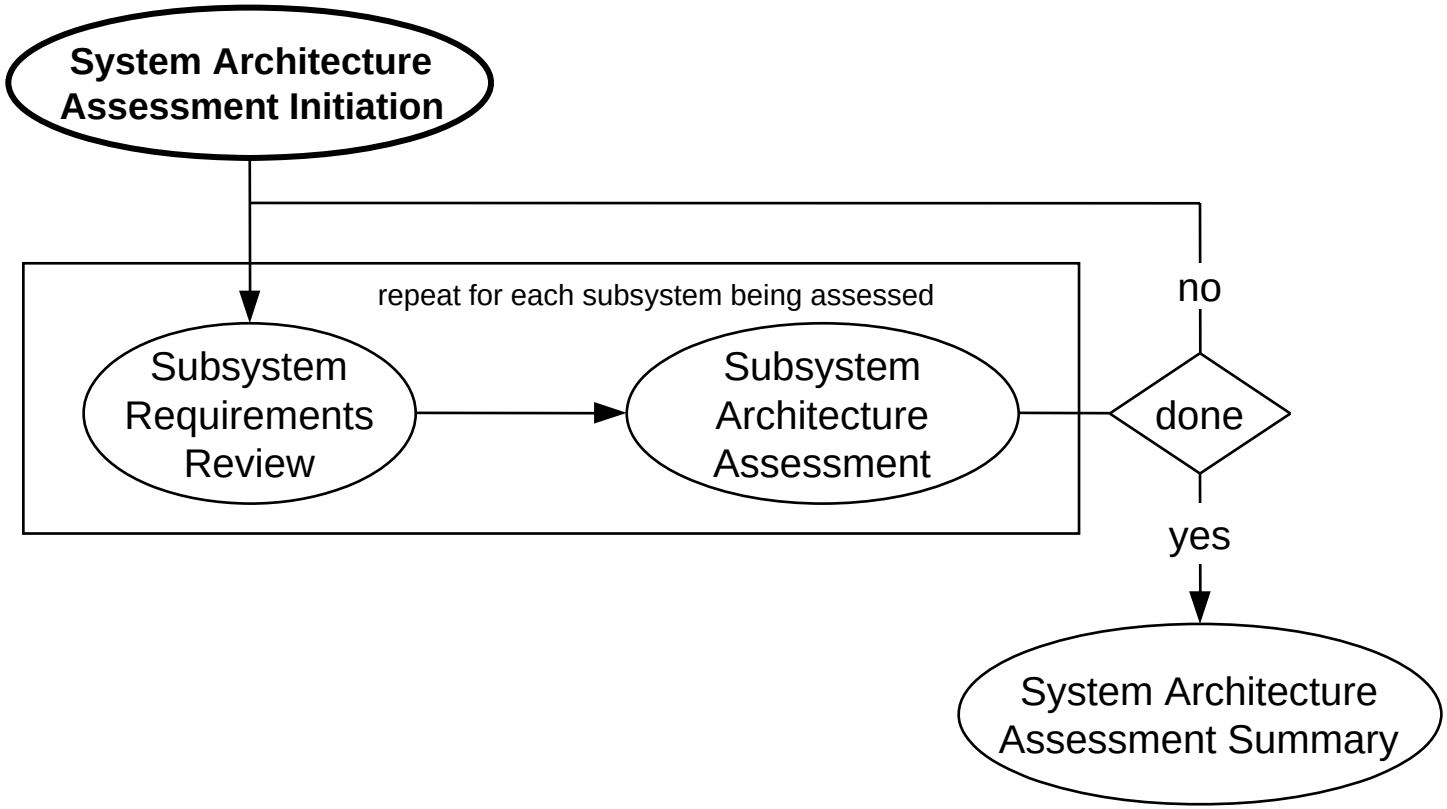
Carnegie Mellon
Software Engineering Institute

Quasar Teams



Carnegie Mellon
Software Engineering Institute

System Architecture Assessment Initiation (SAAI) Phase



Carnegie Mellon
Software Engineering Institute

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

Carnegie Mellon
Software Engineering Institute

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

Carnegie Mellon
Software Engineering Institute

SAAI Phase Principles

Need to Develop Consensus between Assessors and Assesses

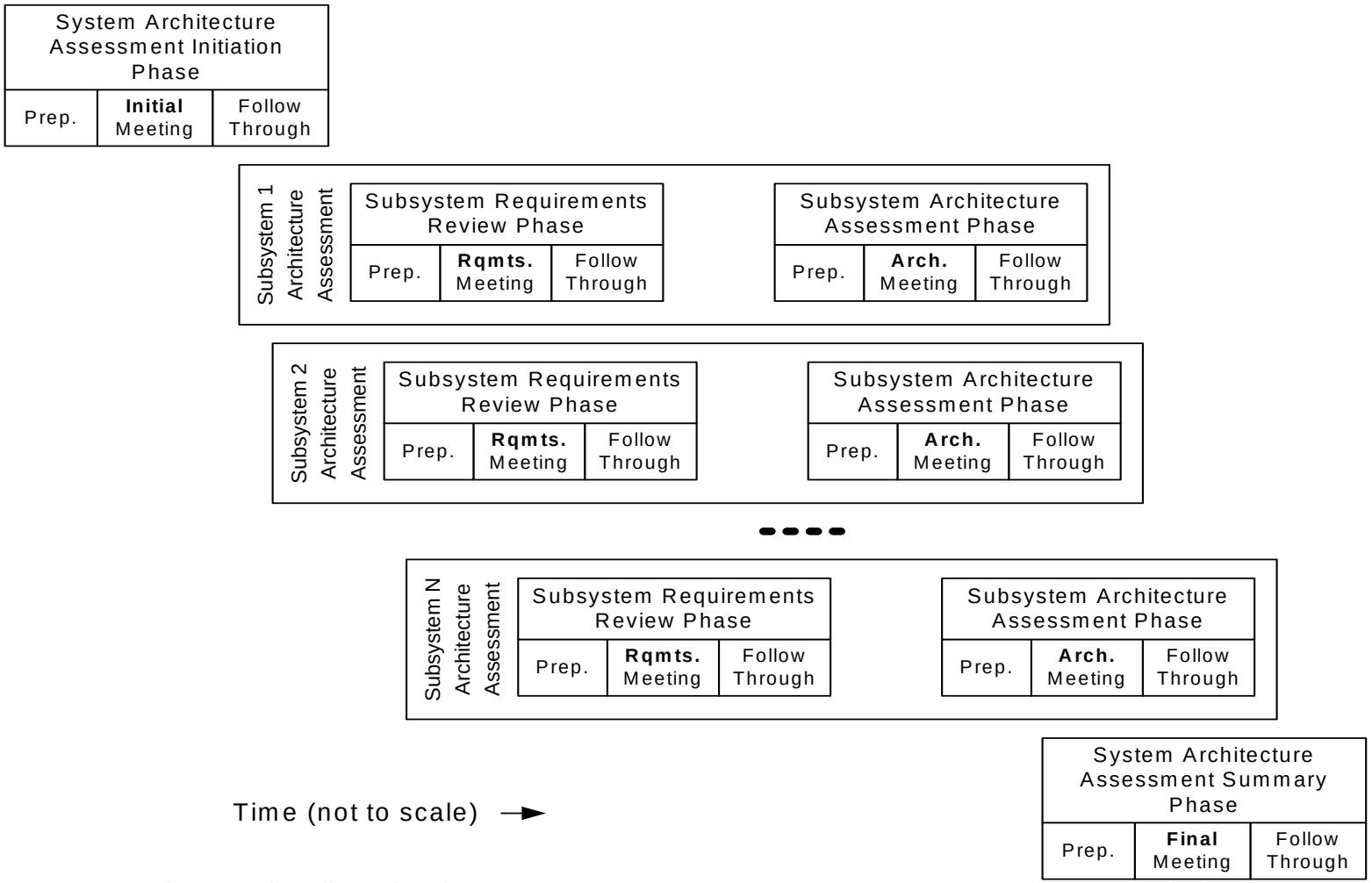
Need to Tailor Process to meet specific Needs of the Overall Assessment

Scope of Assessment should match Project Needs and Resources

Subsystem Assessments must be scheduled to ensure required Resources

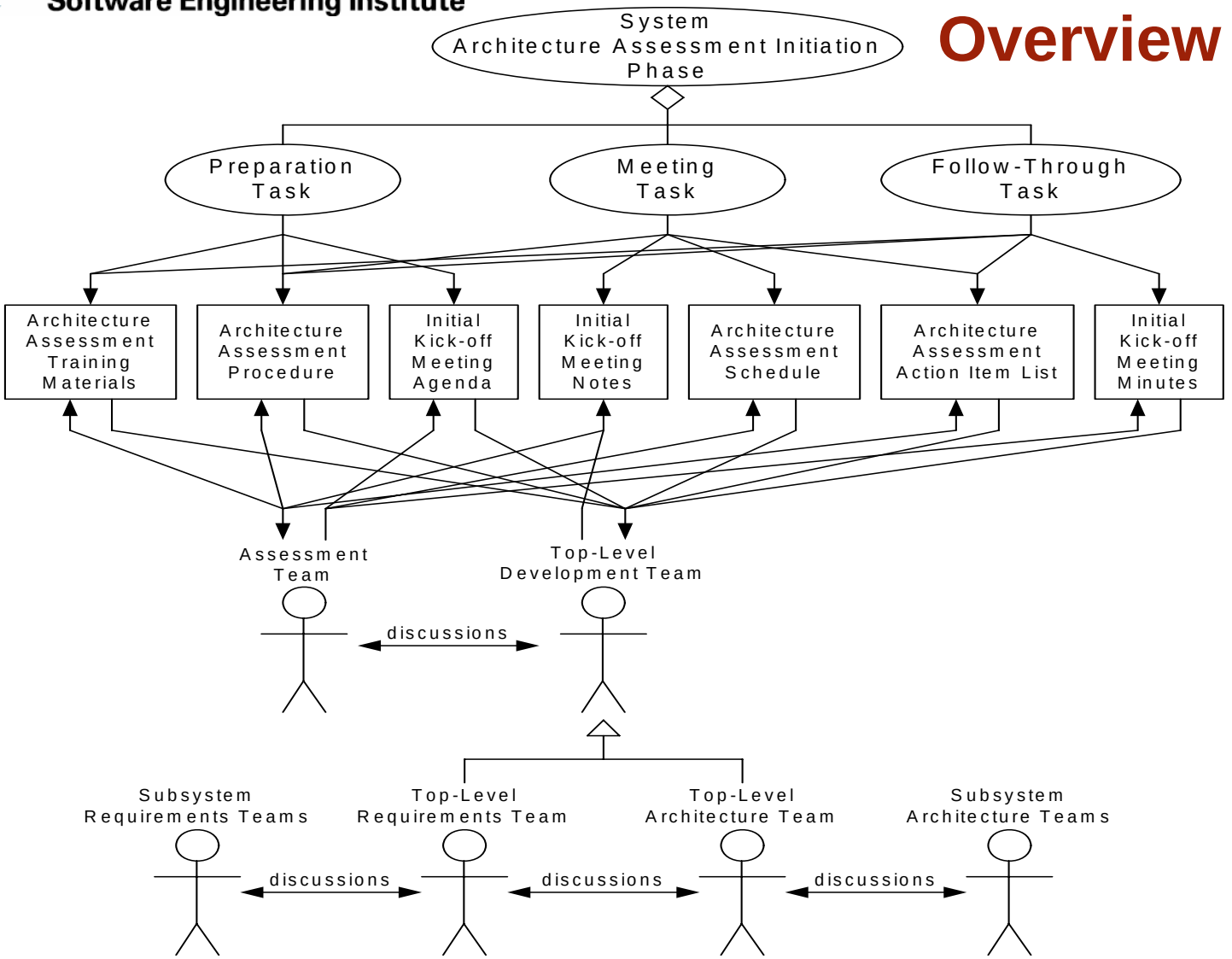
Carnegie Mellon
Software Engineering Institute

SAAI Phase Context



Carnegie Mellon
Software Engineering Institute

SAAI Phase Overview



Carnegie Mellon
Software Engineering Institute

SAAI Preparation Task

Steps:

1. Staff Assessment Team
2. Train Assessment Team
3. Assessment Team Identifies the Top-Level Development Team (Top-Level Requirements Engineers & Architects)
4. Assessment Team Trains Top-Level Development Team
5. Teams Collaborate to Organize Meeting (attendees, time, location, agenda)

Carnegie Mellon
Software Engineering Institute

SAAI Meeting Task

Steps:

1. Teams Collaborate to Determine Assessment Scope:
   - Architecturally Significant Requirements
   - Subsystems
   - Assessment Resources
2. Teams Collaborate to Develop Initial Assessment Schedule
3. Teams Collaborate to Tailor Assessment Process
4. Assessment Team Manages Action Items

Carnegie Mellon
Software Engineering Institute

SAAI Follow-Through Task

Steps:

1. Assessment Team develops and presents Meeting Outbrief
2. Assessment Team develops, reviews, and distributes Meeting Minutes
3. Assessment Team tailors and distributes:
   - Assessment Procedure
   - Assessment Training Material
4. Assessment Team distributes Assessment Schedule
5. Teams obtain Needed Resources
6. Assessment Team captures Lessons Learned
7. Assessment Team Manages Action Items

Carnegie Mellon
Software Engineering Institute

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

Carnegie Mellon
Software Engineering Institute

Assessment Team Membership

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

Carnegie Mellon
Software Engineering Institute

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

Carnegie Mellon
Software Engineering Institute

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

Carnegie Mellon
Software Engineering Institute

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

Carnegie Mellon
Software Engineering Institute

SAAI Discussion

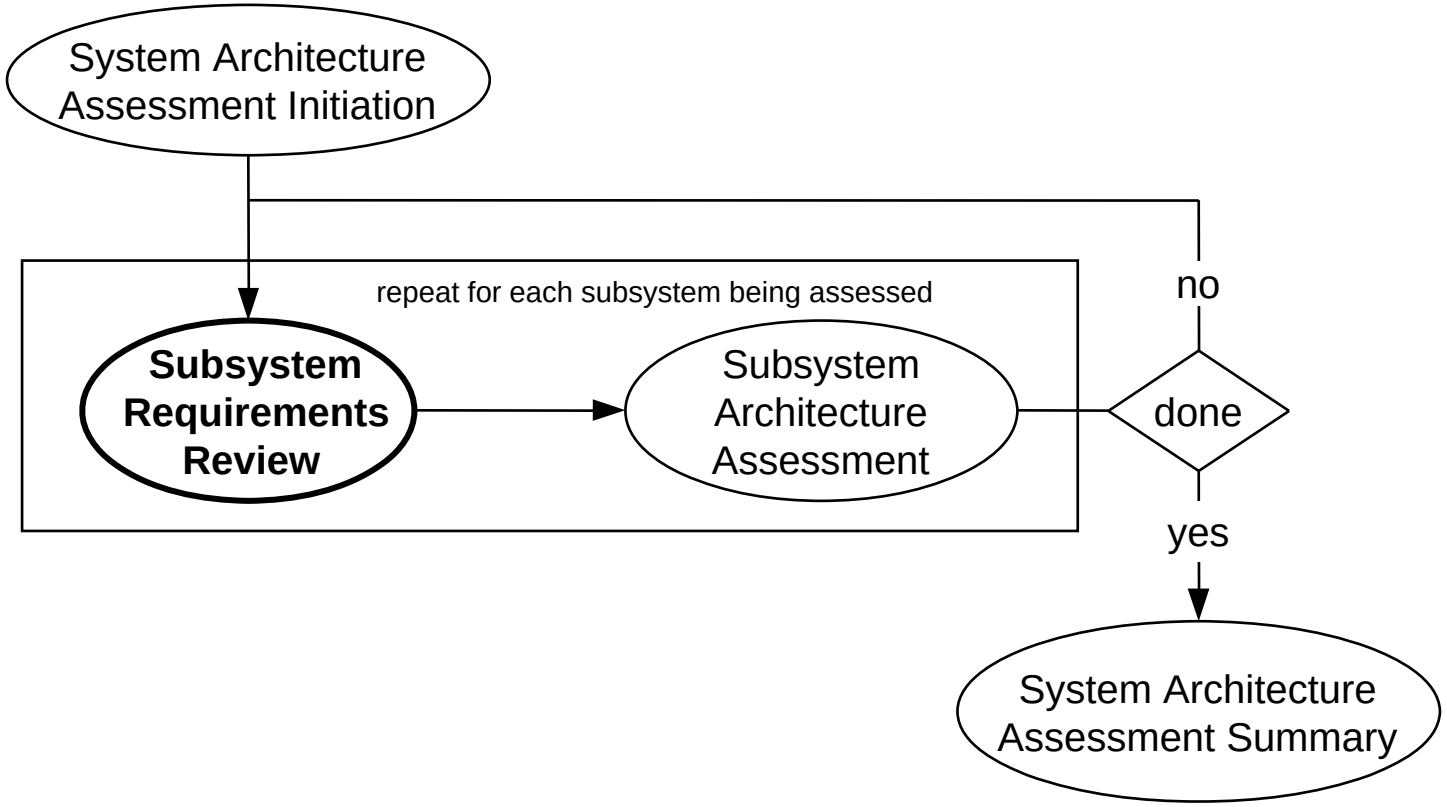
What is the main objectives of the system architecture assessment initiation phase?

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



Carnegie Mellon
Software Engineering Institute

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

Carnegie Mellon
Software Engineering Institute

SRR Questions for the Attendees$_1$

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?

Carnegie Mellon
Software Engineering Institute

Key Questions for the Attendees$_2$

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?

Carnegie Mellon
Software Engineering Institute

SRR Phase Objectives

Ensure that the:

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

2. Subsystem architects know how to prepare for and support the coming subsystem architecture quality assessment.

Carnegie Mellon
Software Engineering Institute

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

Carnegie Mellon
Software Engineering Institute

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.

Carnegie Mellon
Software Engineering Institute

SRR Phase Principles$_1$

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

Carnegie Mellon
Software Engineering Institute

SRR Phase Principles$_2$

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

Carnegie Mellon
Software Engineering Institute

SRR Phase Principles$_3$

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.

Carnegie Mellon
Software Engineering Institute

SRR Phase Challenges$_1$

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.

Carnegie Mellon
Software Engineering Institute

SRR Phase Challenges$_2$

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."

Carnegie Mellon
Software Engineering Institute

SRR Phase Challenges$_3$

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

Carnegie Mellon
Software Engineering Institute

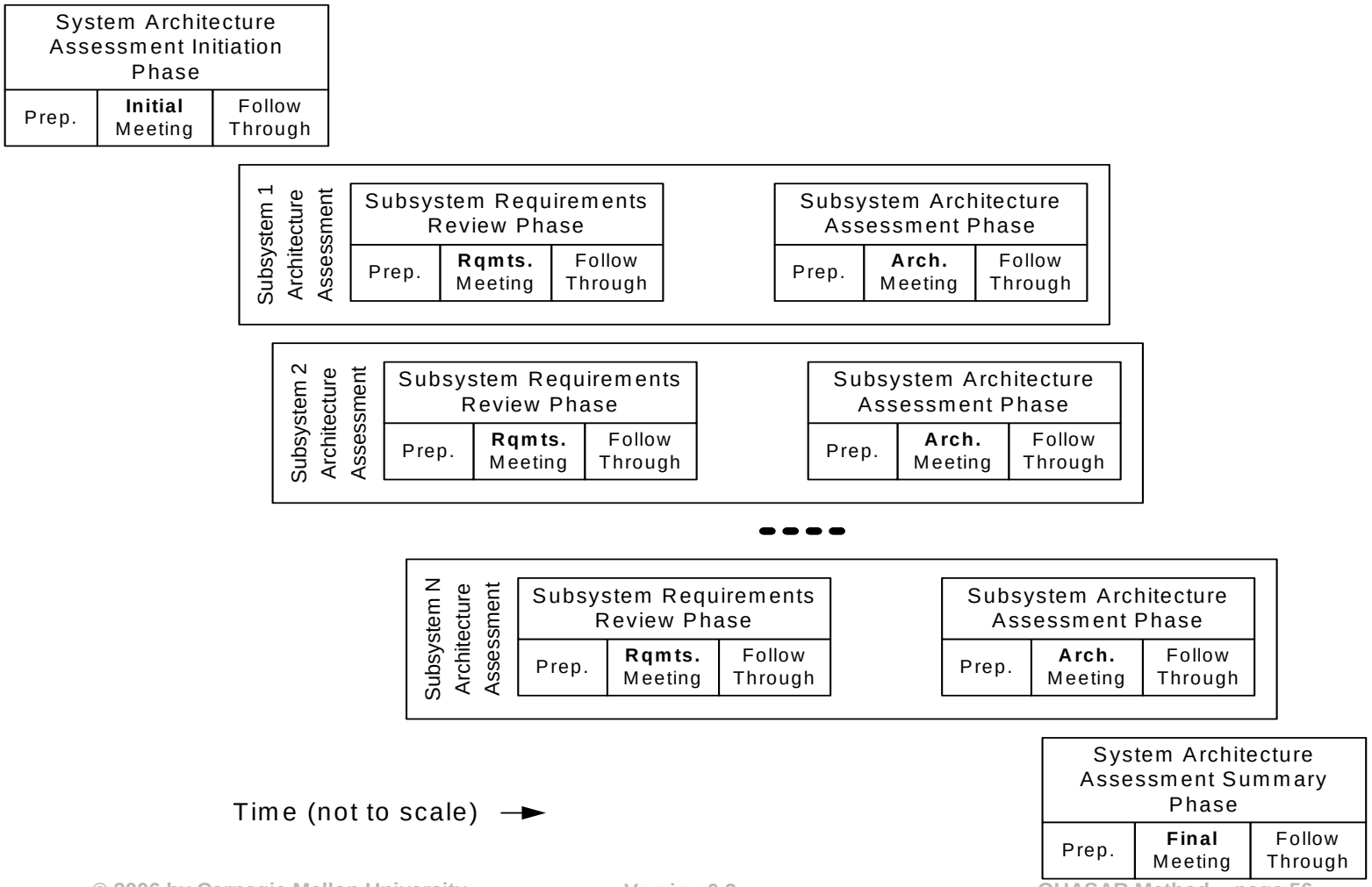
SRR Phase Challenges$_4$

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

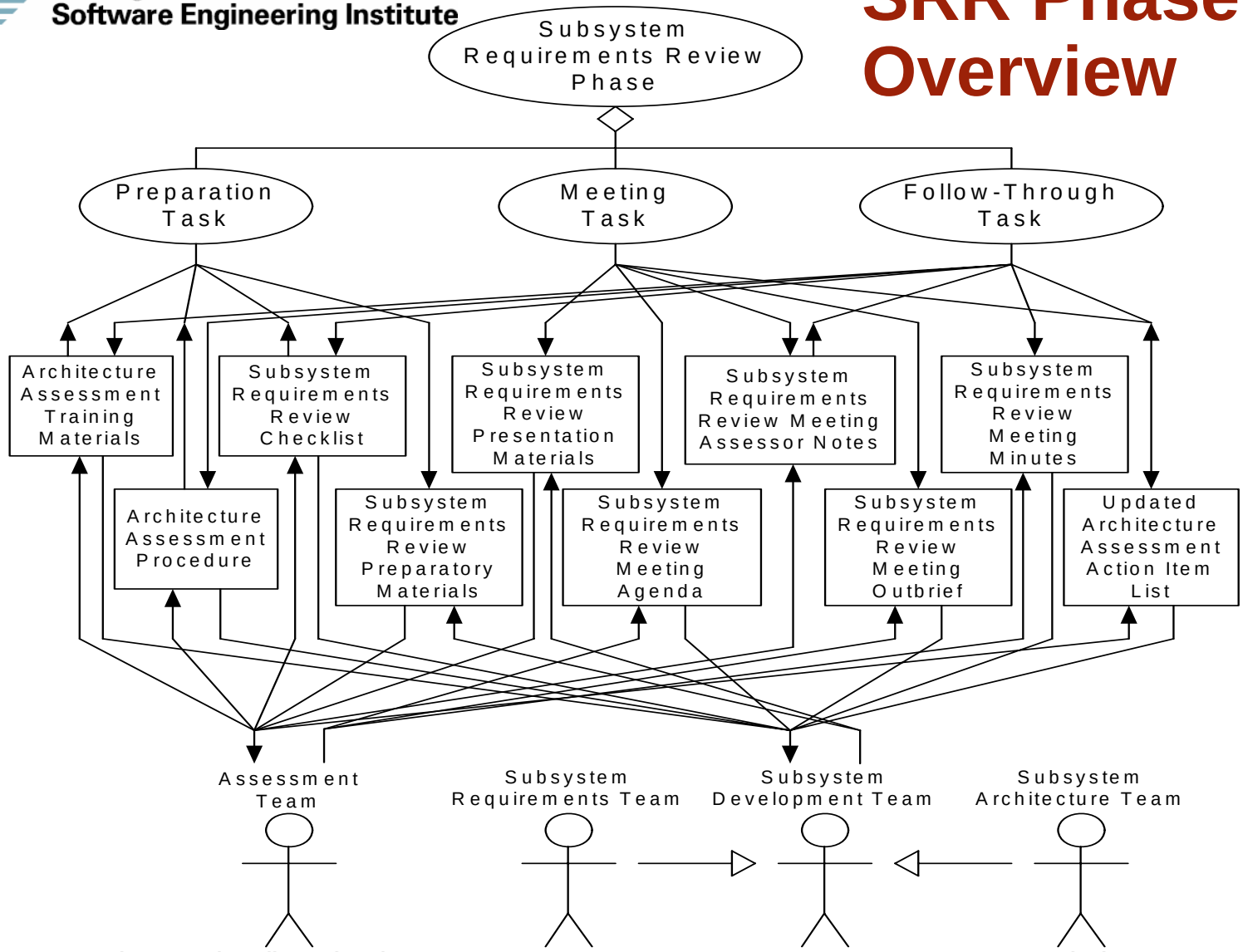
- How good must the architecture be to *sufficiently* supports its derived and allocated quality requirements (i.e., to 'pass' the assessment)?
- What architectural decisions did the architects make to support the quality goals and requirements?
- What were the rationales for these decisions?
- What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

Carnegie Mellon
Software Engineering Institute

SRR Phase Context



SRR Phase Overview



Carnegie Mellon
Software Engineering Institute

SRR Preparation Task

Steps:

1. Subsystem Requirements Team provides access to the architecturally significant subsystem requirements as well as a summary of these requirements
2. Subsystem Architecture Team provides sample of planned Quality Cases
3. Subsystem Assessment Team reviews this information prior to the meeting

Carnegie Mellon
Software Engineering Institute

SRR Meeting Task

Steps:

1. Subsystem Requirements Team presents Summary of the architecturally significant subsystem requirements (organized by quality factor and quality subfactors)
2. Subsystem Assessment Team recommends Improvements
3. Subsystem Architecture Team presents sample of planned Quality Cases
4. Subsystem Assessment Team recommends Improvements
5. Assessment Team Manages Action Items

Carnegie Mellon
Software Engineering Institute

SRR Follow-Through Task

Steps:

1. Subsystem Assessment Team presents Outbrief
2. Subsystem Assessment Team develops and publishes Meeting Minutes containing recommendations for improving:
   - Architecturally significant subsystem requirements
   - Quality Cases
3. Assessment Team tailors and distributes updated Assessment Procedure and Assessment Training Material (for future requirements reviews)
4. Assessment Team captures Lessons Learned
5. Assessment Team Manages Action Items

Carnegie Mellon
Software Engineering Institute

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

Carnegie Mellon
Software Engineering Institute

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)

Carnegie Mellon
Software Engineering Institute

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

Carnegie Mellon
Software Engineering Institute

Subsystem Assessment Team$_1$

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

Carnegie Mellon
Software Engineering Institute

Subsystem Assessment Team$_2$

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

Carnegie Mellon
Software Engineering Institute

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?

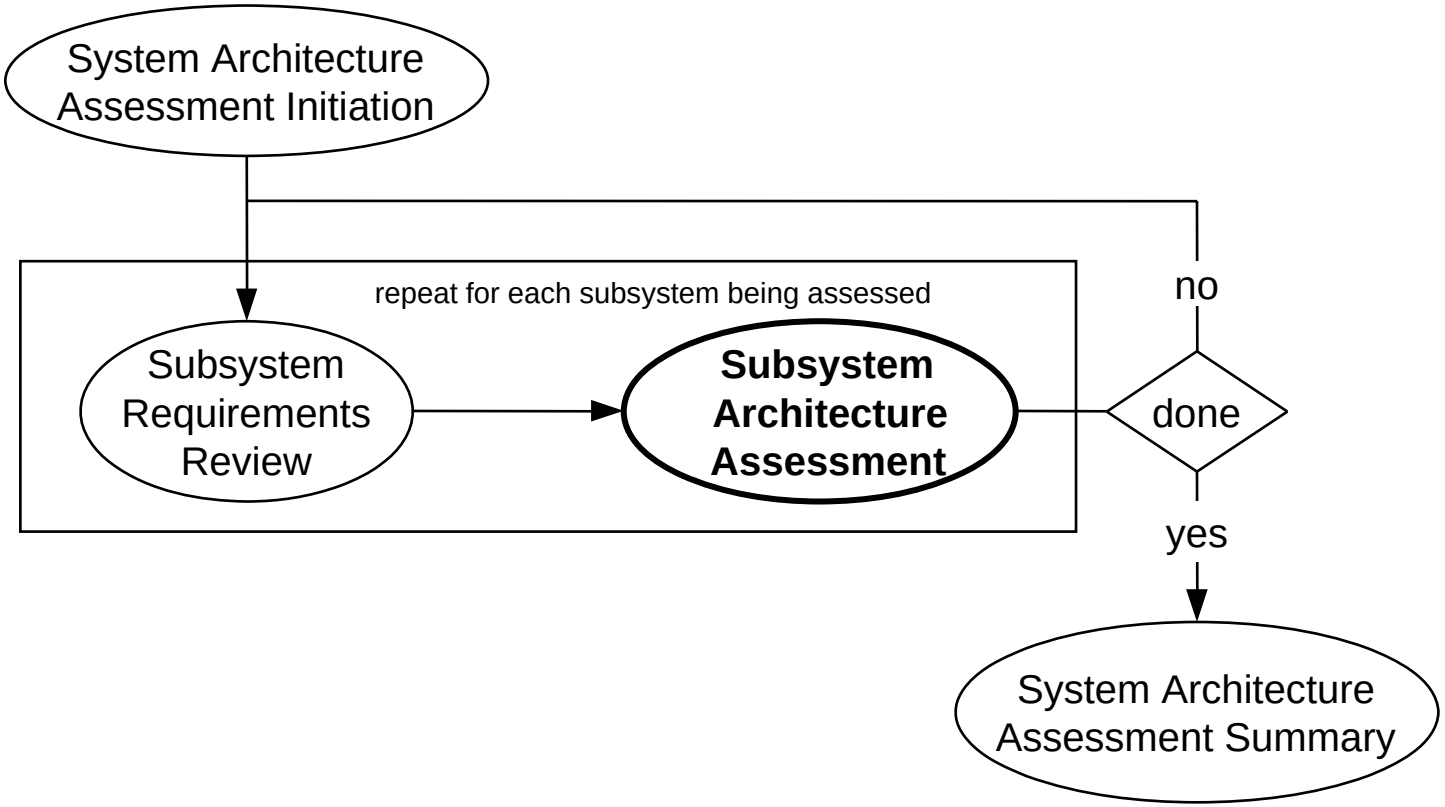
What are the three tasks comprising the subsystem requirements review?

What are the objectives of these three tasks?

What teams are involved?

What are the responsibilities of these teams?

Subsystem Architecture Assessment (SAA)



Carnegie Mellon
Software Engineering Institute

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

Carnegie Mellon
Software Engineering Institute

SAA Questions for the Attendees$_1$

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?

Carnegie Mellon
Software Engineering Institute

SAA Questions for the Attendees$_2$

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?

Carnegie Mellon
Software Engineering Institute

SAA Objectives

Assess Quality of Subsystem Architecture in terms of:

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

Carnegie Mellon
Software Engineering Institute

SAA Principles$_1$

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

Carnegie Mellon
Software Engineering Institute

SAA Principles$_2$

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.

Carnegie Mellon
Software Engineering Institute

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.

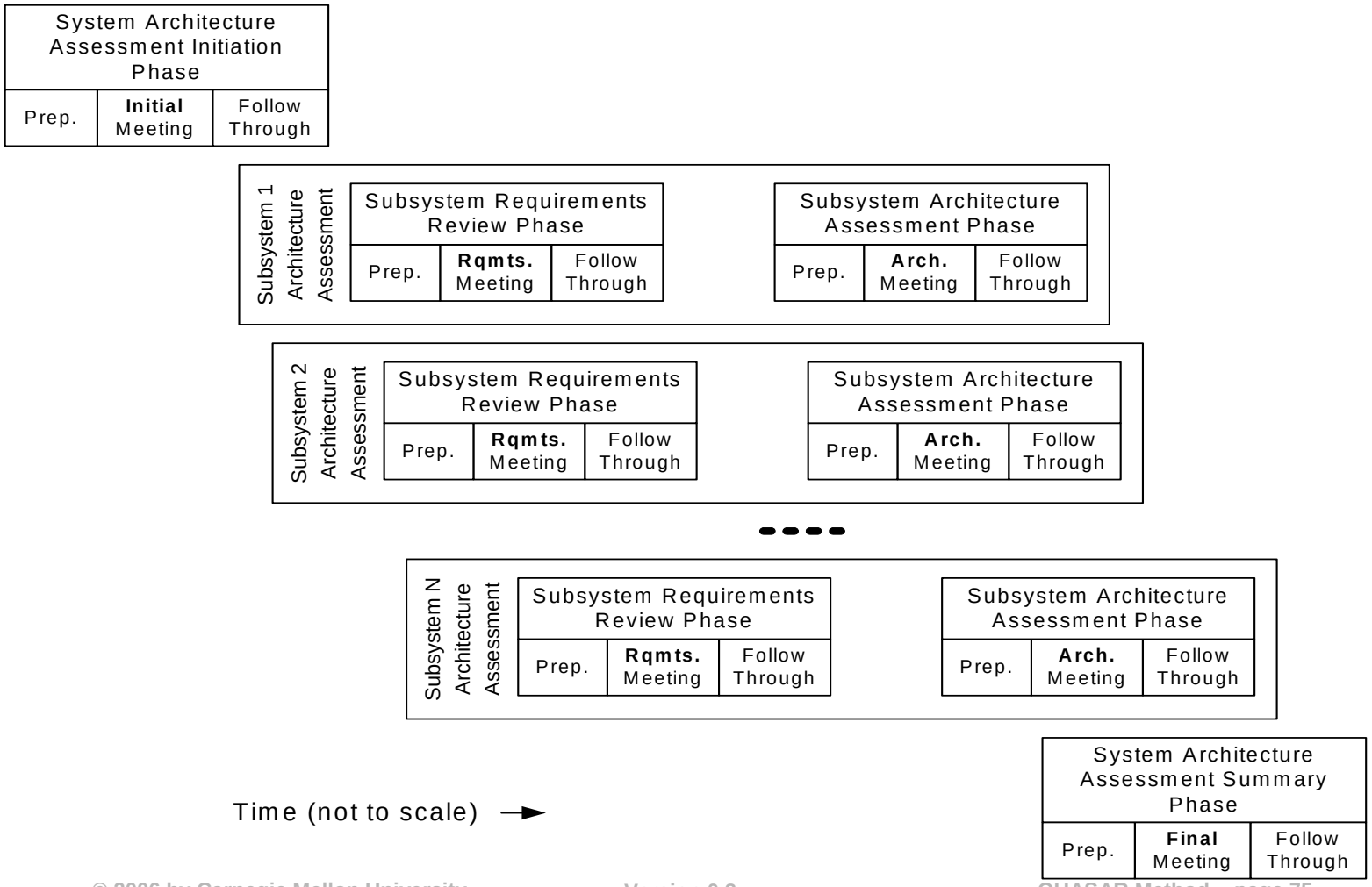
Architecture assessments may not be:

- Mandated by contract or development process
- Scheduled and funded

Managers feel schedule pressures do not allow time for assessment.

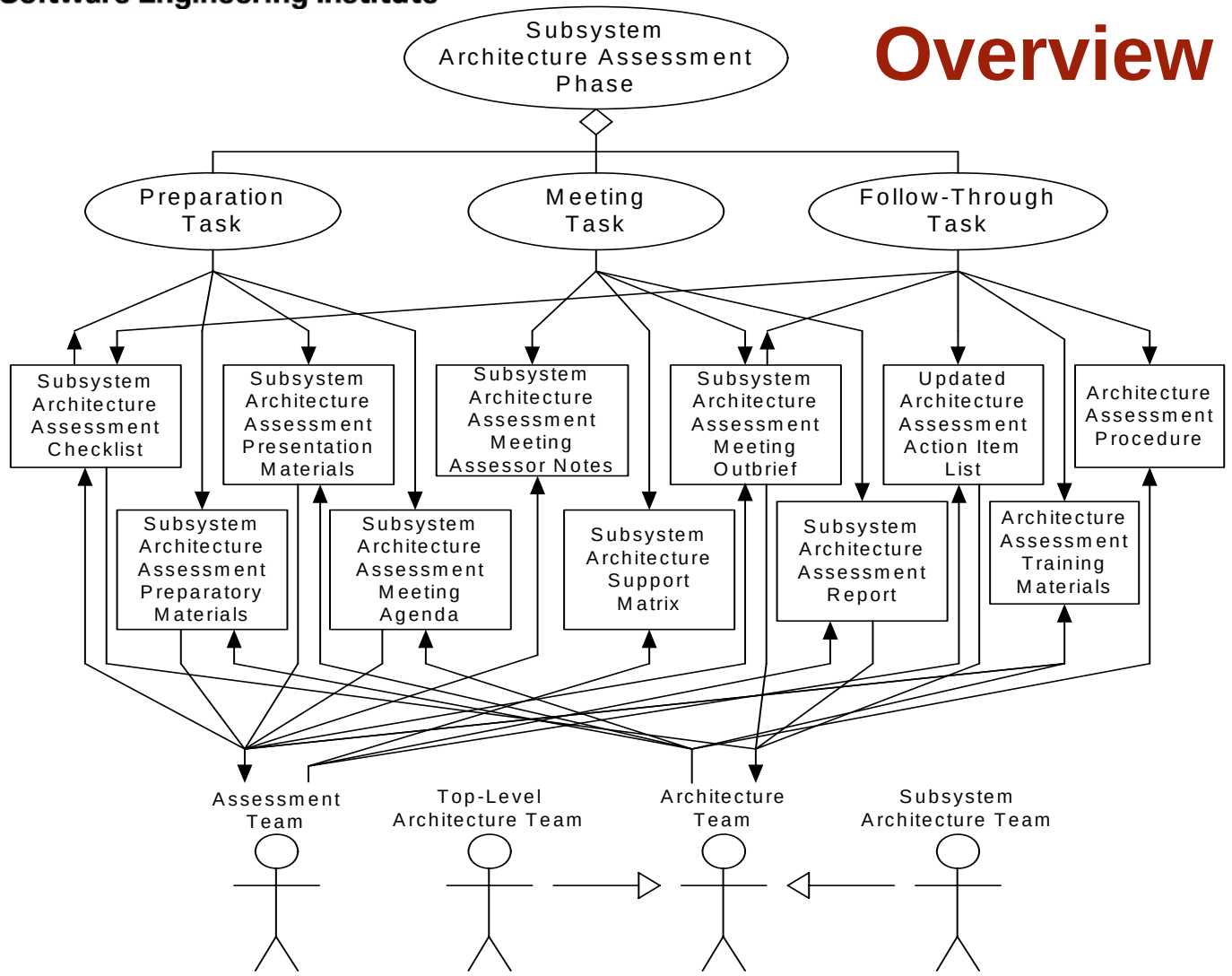
Carnegie Mellon
Software Engineering Institute

SAA Context



Carnegie Mellon
Software Engineering Institute

SAA Phase Overview



Carnegie Mellon
Software Engineering Institute

SAA Preparation Task

Steps:

1. Subsystem Assessment Team Provides Assessment Checklist
2. Subsystem Architecture Team Gathers (Generates) and Makes Available *Preparatory* Materials:
   - Subsystem Architecture Overview
   - Updated Quality Requirements
   - Quality Cases including Arguments and Evidence
3. Subsystem Architecture Team Gathers (Generates) and Makes Available *Presentation* Materials
4. Subsystem Assessment Team:
   - Reads Materials
   - Generates RFIs and RFAs
5. Teams Collaborate to Organize Assessment Meeting (Attendees, Time, Location, Agenda, Invitation)

Carnegie Mellon
Software Engineering Institute

SAA Meeting Task

Steps:

1. 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)
2. Subsystem Assessment Team:
   - Probes Architecture (quality case by quality case)
   - Manages Action Items

Carnegie Mellon
Software Engineering Institute

SAA Follow-Through Task

Steps:

1. 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

Carnegie Mellon
Software Engineering Institute

SAA Roles and Responsibilities

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

- Subsystem Architecture Team
- Subsystem Assessment Team

Carnegie Mellon
Software Engineering Institute

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:

Carnegie Mellon
Software Engineering Institute

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

Carnegie Mellon
Software Engineering Institute

SAA Discussion

Should the quality cases be developed as a:

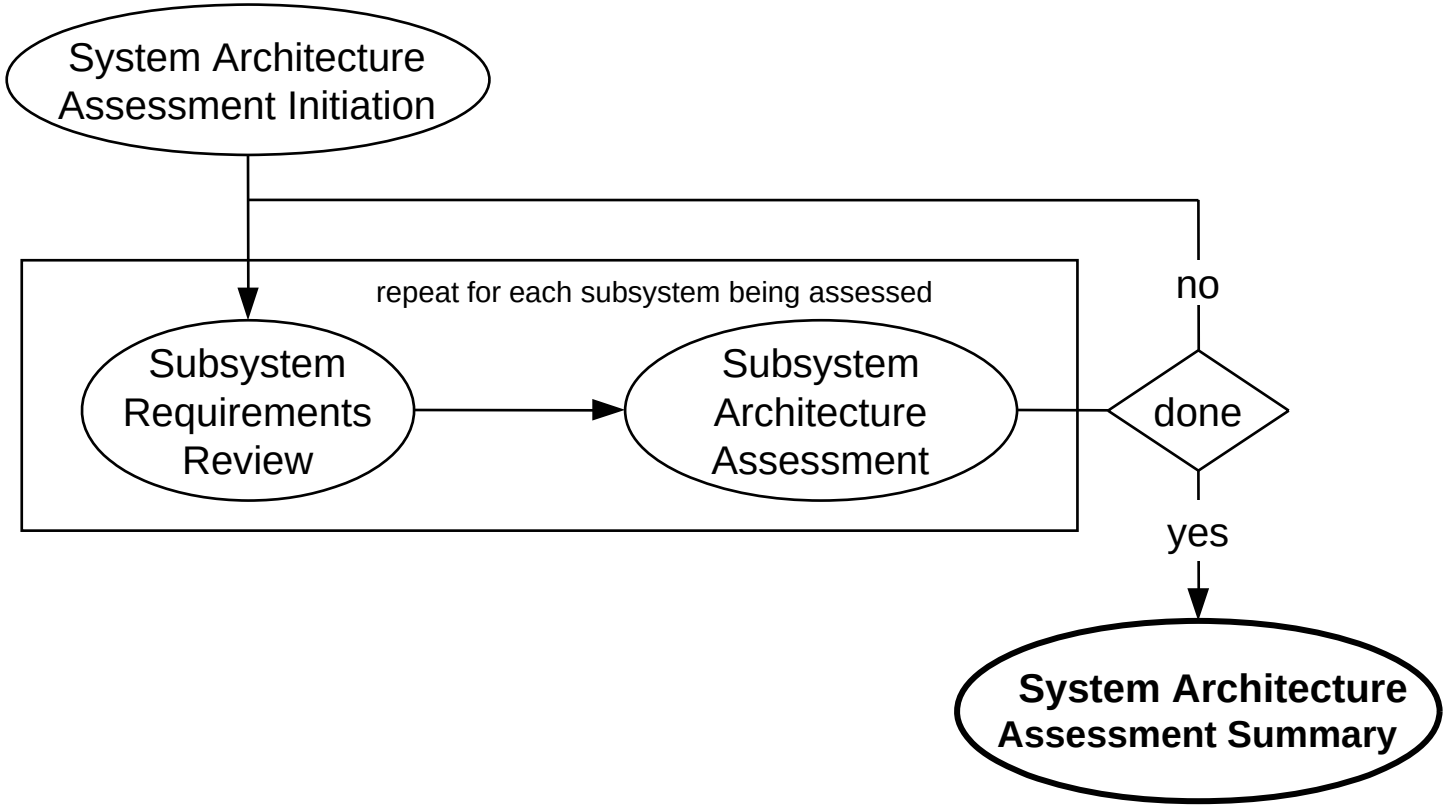
- Natural part of the architecting process?
- Part of the assessment process?

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)



Carnegie Mellon
Software Engineering Institute

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

Carnegie Mellon
Software Engineering Institute

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?

Carnegie Mellon
Software Engineering Institute

SAAS Objectives

Collect previous Subsystem Architecture Assessment Results

Create System Architecture Assessment Summarize Results

Capture Method Lessons Learned

Update Assessment Method and Training Materials

Carnegie Mellon
Software Engineering Institute

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.

Carnegie Mellon
Software Engineering Institute

SAAS Phase Challenges

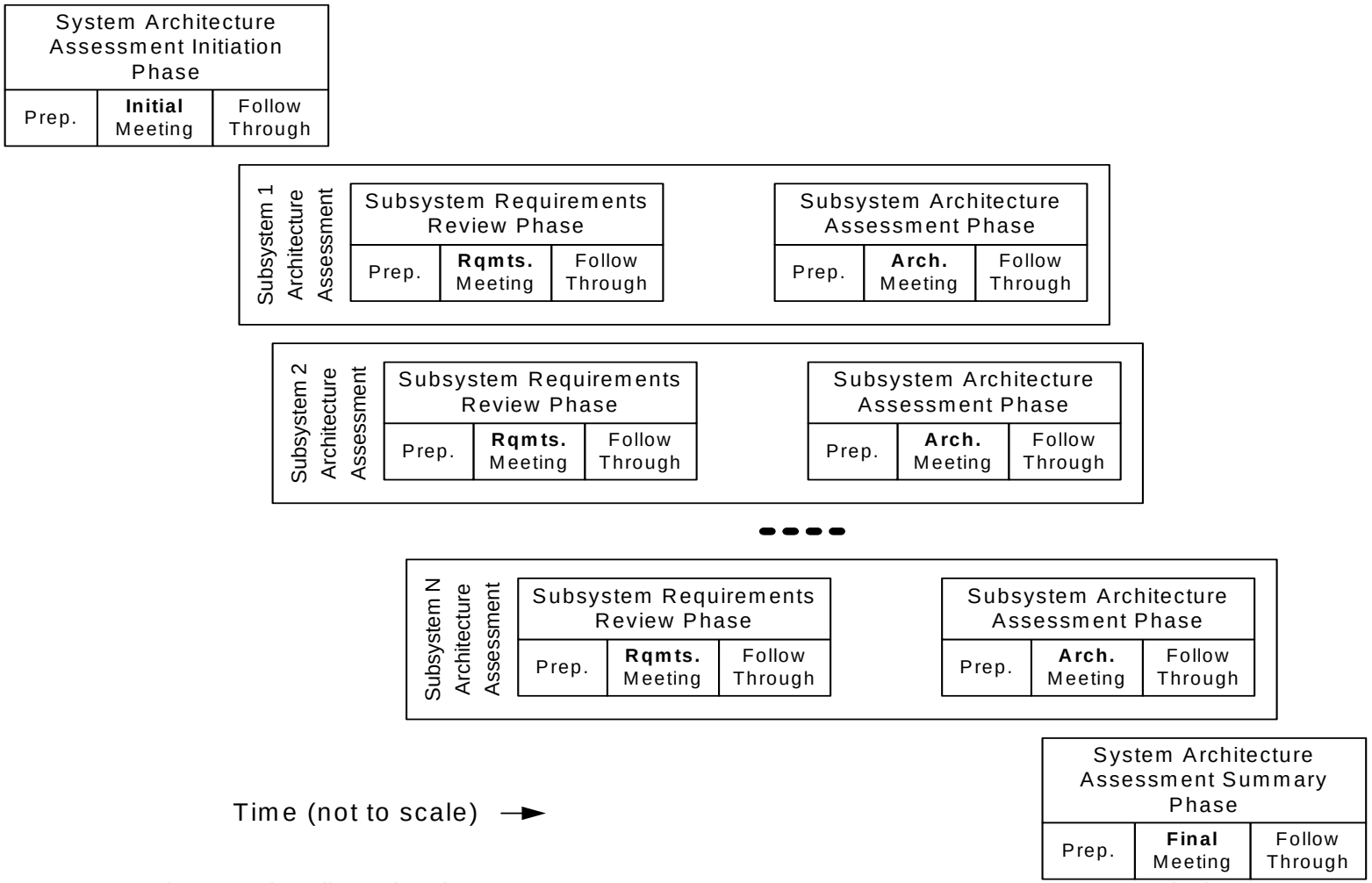
How should subsystem findings be summarized without ending up comparing apples and oranges?

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

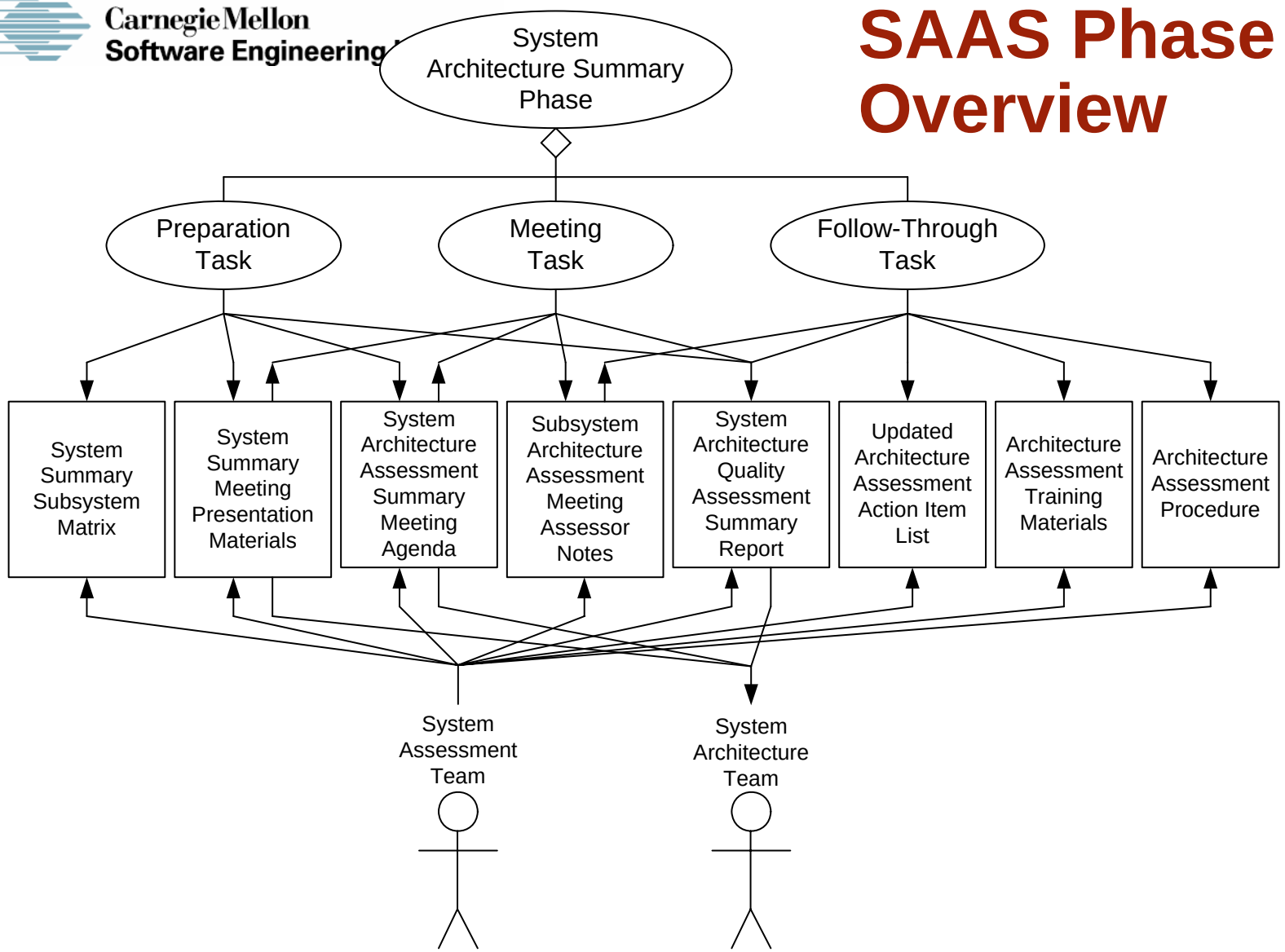
Executive management may demand simplistic single number summary of system architecture.

Carnegie Mellon
Software Engineering Institute

SAAS Context



SAAS Phase Overview



Carnegie Mellon
Software Engineering Institute

SAAS Preparation Task

Steps:

1. 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

Carnegie Mellon
Software Engineering Institute

SAAS Meeting Task

Steps:

1. System Assessment Team:
    - Restates Assessment Objectives
    - Summarizes Assessment Method
    - Summarizes Quality of Subsystem Architectures
    - Summarizes Quality of System Architecture
    - Solicits Feedback
    - Captures Lessons Learned
2. System Architecture Team:
    - Captures Lessons Learned

Carnegie Mellon
Software Engineering Institute

SAAS Follow-Through Task

Steps:

1. System Assessment Team:
   - Updates and Distributes the System Architecture Assessment Summary Report
   - Manages Action Items
   - Updates Assessment Method and Training Materials
2. System Architecture Team:
   - Updates Architecture Method and Training Materials

Carnegie Mellon
Software Engineering Institute

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

Carnegie Mellon
Software Engineering Institute

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?

Carnegie Mellon
Software Engineering Institute

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

Carnegie Mellon
Software Engineering Institute

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

Carnegie Mellon
Software Engineering Institute

Questions?

For more information, contact:

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