*QU*ality *A*ssessment of *S*ystem *A*rchitectures and their *R*equirements (QUASAR) Version 3.0

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Donald Firesmith
13 February 2008

Topics

Requirements and Architecture Challenges ◄

Underlying Concepts

QUASAR Method

Reasons to use QUASAR

QUASAR – Today and Tomorrow

Requirements and Architecture Challenges$_1$

Requirements and Architecture are the first two Opportunities to make Major Engineering Mistakes.

Architecturally Significant Requirements are typically poorly engineered.

Architecture and associated Architecturally Significant Requirements Affect:

- Project Organization and Staffing (Conway's Law)
- Downstream Design, Implementation, Integration, Testing, and Deployment Decisions

A common project-specific Quality Model is needed to drive the

- Quality Requirements, which drives the
- Quality of the System Architecture, which drives the
- Quality of the System

Requirements and Architecture Challenges$_2$

Architecturally-Significant, Quality-Related Requirements and their associated Architectural Decisions *Drive* the System and Component:

- Ultimate Quality
- Development Schedule
- Development Costs
- Sustainment Costs
- Maintainability and Upgradeability
- Acceptance and Usage by Stakeholders

Requirements and Architecture Challenges$_3$

System Quality is the Union of Relevant Quality Characteristics:

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

Requirements and Architecture Challenges$_4$

It is important to determine *Actual System and Component Requirements and Architecture*:

- Quality
- Maturity and Completeness
- Integrity and Consistency
- Usability

It is important to identify System and Component Requirements and Architectural *Defects* Early:

- Fix Defects Early
- Decrease Development and Maintenance Costs
- Decrease Schedule

Requirements and Architecture Challenges$_5$

It is important to identify (and thereby help Manage) Risks:

- Requirements and Architecture Risks
- System and Project Risks
- Business Risks

It is important to provide Acquirer/Management:

- *Visibility* into
- *Oversight* over

the System and Component Requirements and Architecture

It is important to determine *Compliance*:

- Requirements and Architecture with Contract (Acquirer) Requirements
- Architecture with System and Component (Developer) Requirements

Requirements and Architecture Challenges$_6$

It is important to develop *Consensus* regarding requirements and architecture quality, status, etc.:

- Among Developers (e.g., Requirements, Architecture, and Management Teams)
- Between Acquirer and Developer Organizations

Topics

Requirements and Architecture Challenges

Underlying Concepts:

Quality Model ◄

QUASAR Method

Reasons to use QUASAR

QUASAR – Today and Tomorrow

What is Quality?

Quality

the Degree to which a Work Product (e.g., System, Subsystem, Requirements, Architecture) Exhibits a Desired or Required Amount of Useful or Needed Characteristics and Attributes

Not just lack of defects!

Question:

What Types of Characteristics and Attributes are these?

Answer:

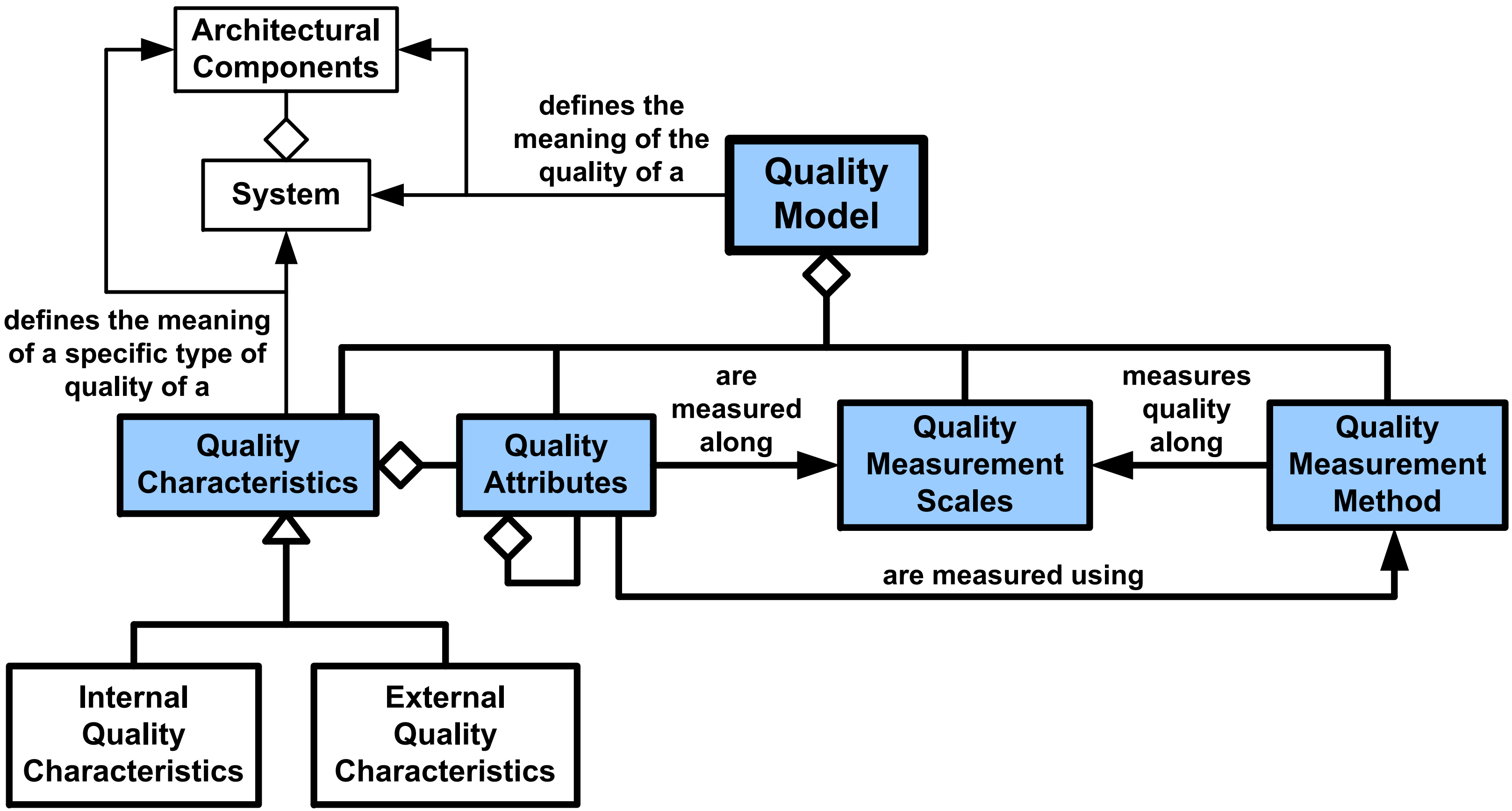
They are the Characteristics defined by the Project Quality Model.

Quality Model$_1$

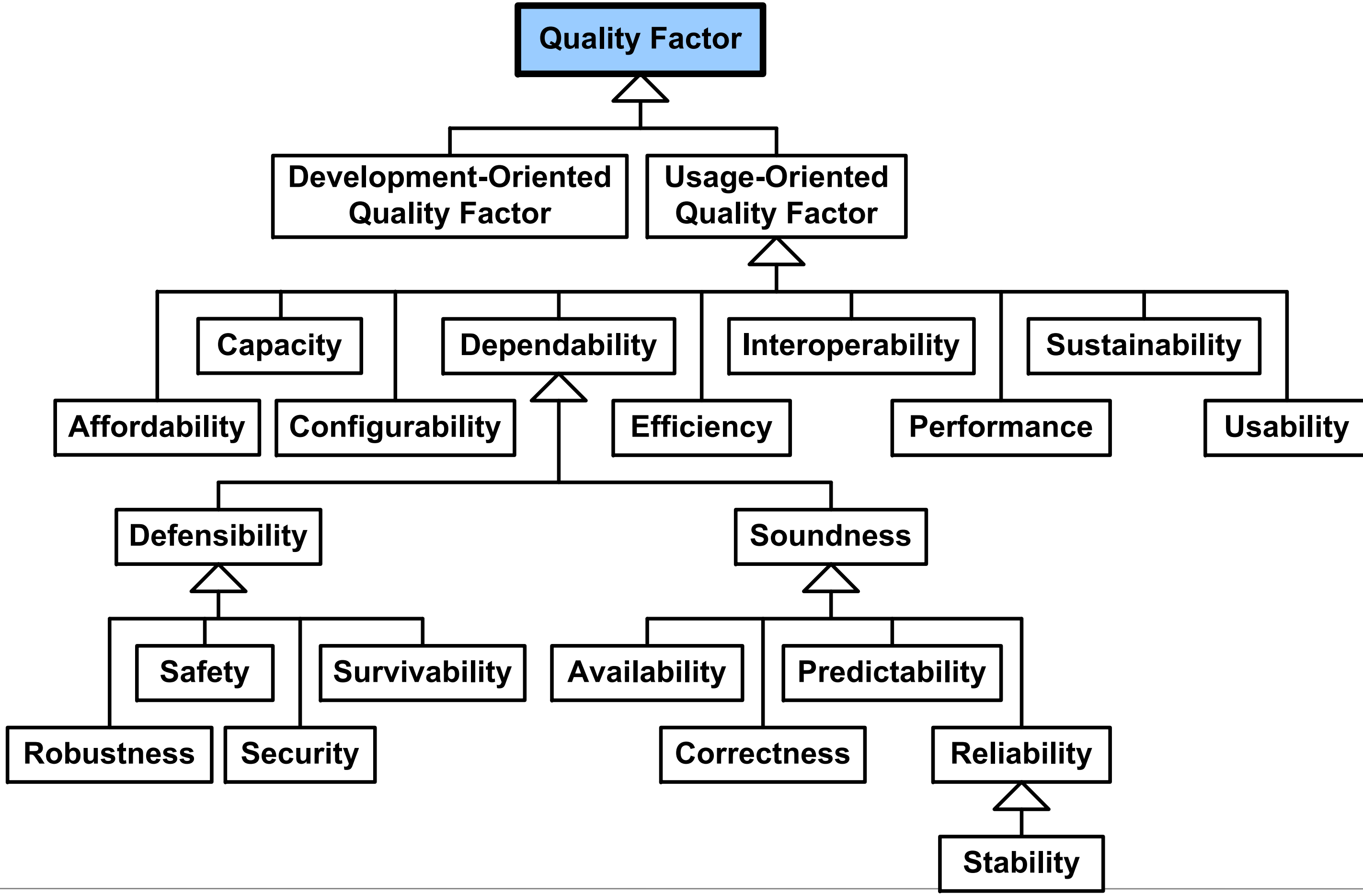
Quality of a Work Product is defined in terms of a **Quality Model**:

- **Quality Characteristics**
  (a.k.a., Quality Factors, the 'ilities')
  (e.g., availability, extensibility, interoperability, maintainability, performance, portability, reliability, safety, security, and usability)
- **Quality Attributes**
  (a.k.a., Quality Subfactors)
  (e.g., the quality attributes of performance are jitter, latency, response time, schedulability, throughput)
- **Quality Measurement Scales**
  (e.g., milliseconds, transactions per second)

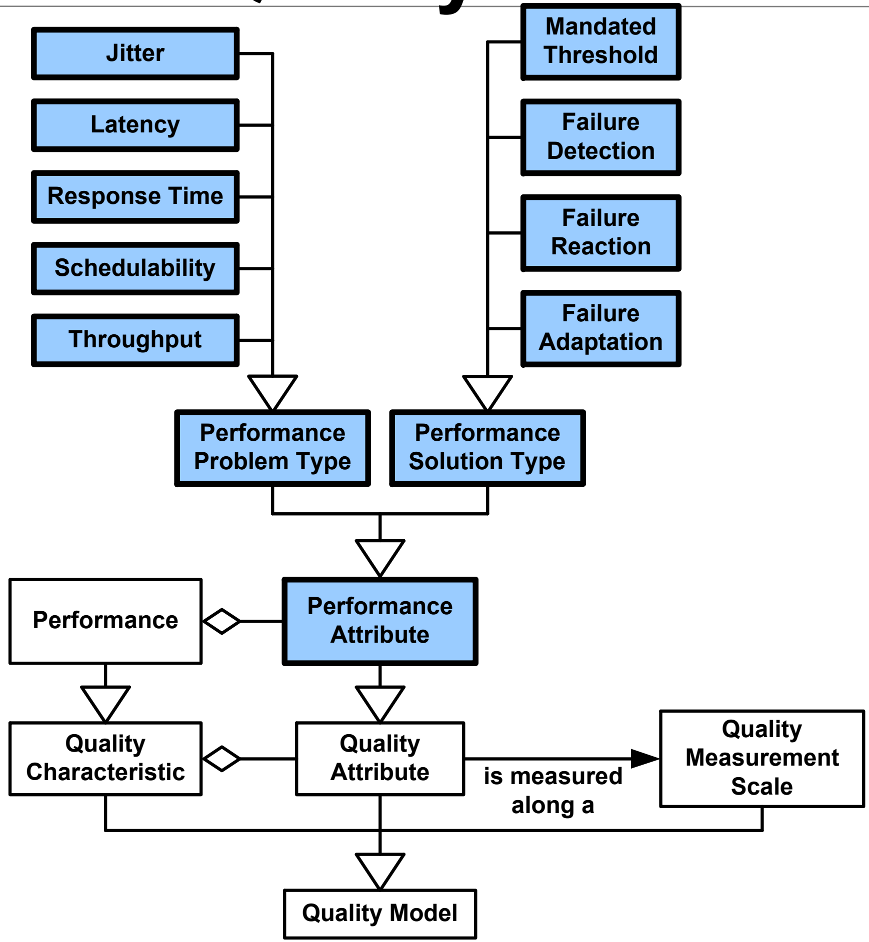
Quality Model_2



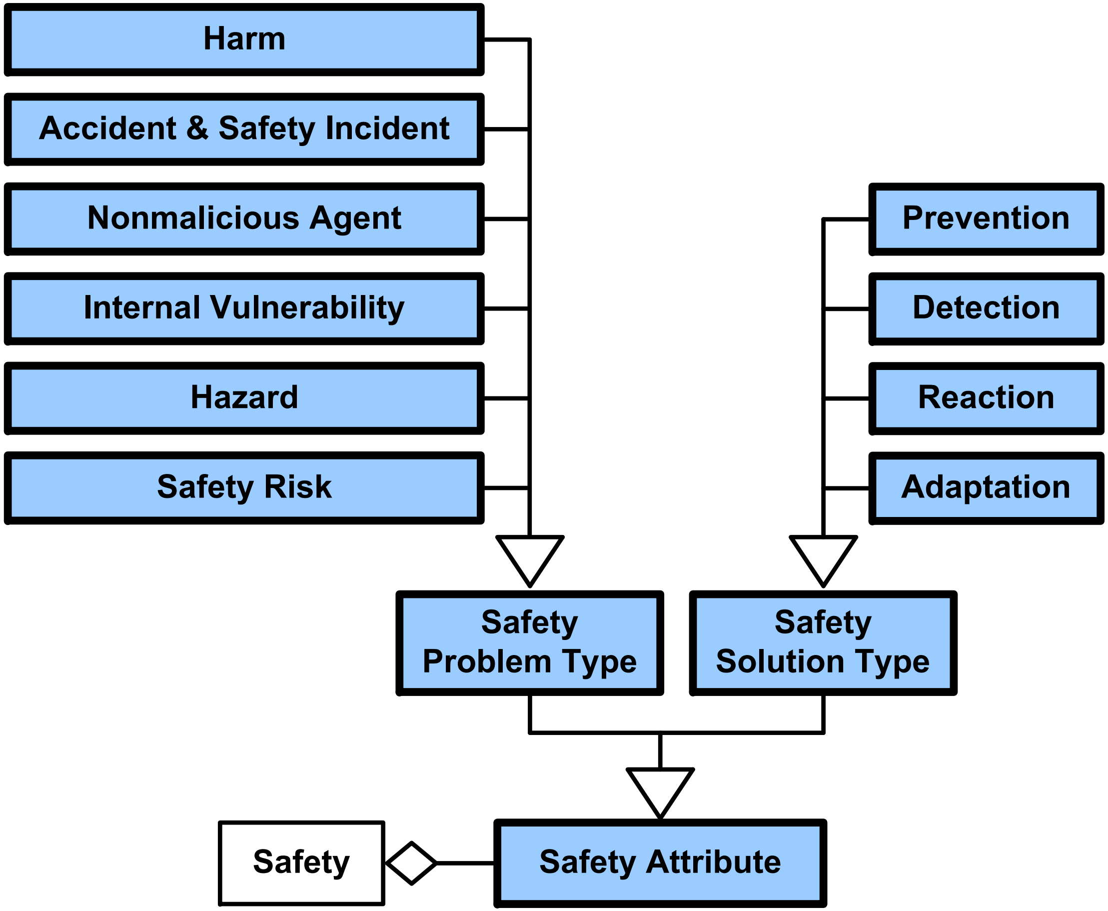
Quality Model – Quality Characteristics



Quality Model – Performance Quality Attributes



Quality Model – Safety Quality Attributes



Topics

Requirements and Architecture Challenges

Underlying Concepts:

Quality Case ◄

QUASAR Method

Reasons to use QUASAR

QUASAR – Today and Tomorrow

Quality Case - Definition

Quality Case

a Cohesive Collection of *Claims*, *Arguments*, and *Evidence* that Makes the Developers' Case that their Work Product(s) have *Sufficient Quality*

Foundational Concept underlying QUASAR

A Generalization and Specialization of Safety Cases from the Safety Community:

More) Can Address any Quality Characteristic and/or Quality Attribute

Less) May be Restricted to only Requirements or Architecture

Useful for:

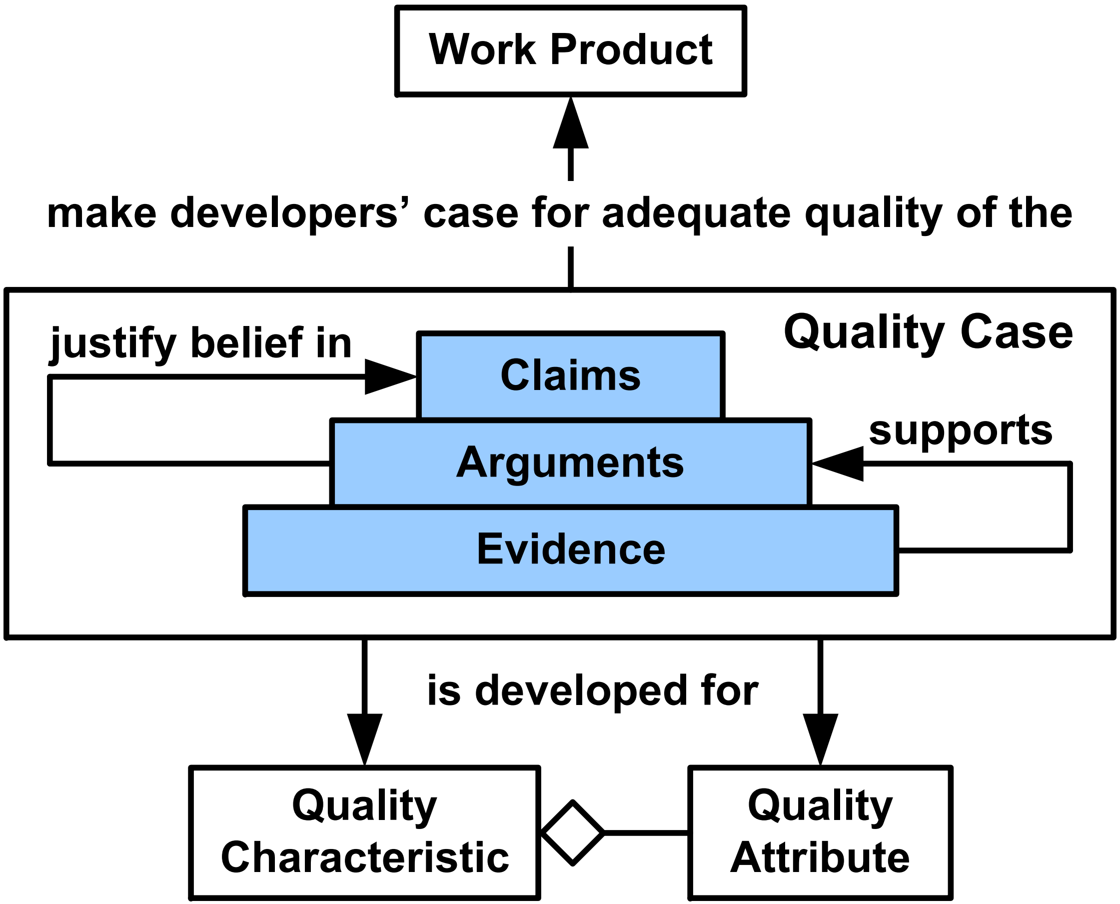
- Assessing Quality
- System Certification and Accreditation (e.g., safety and security)

Quality Cases – Components$_1$

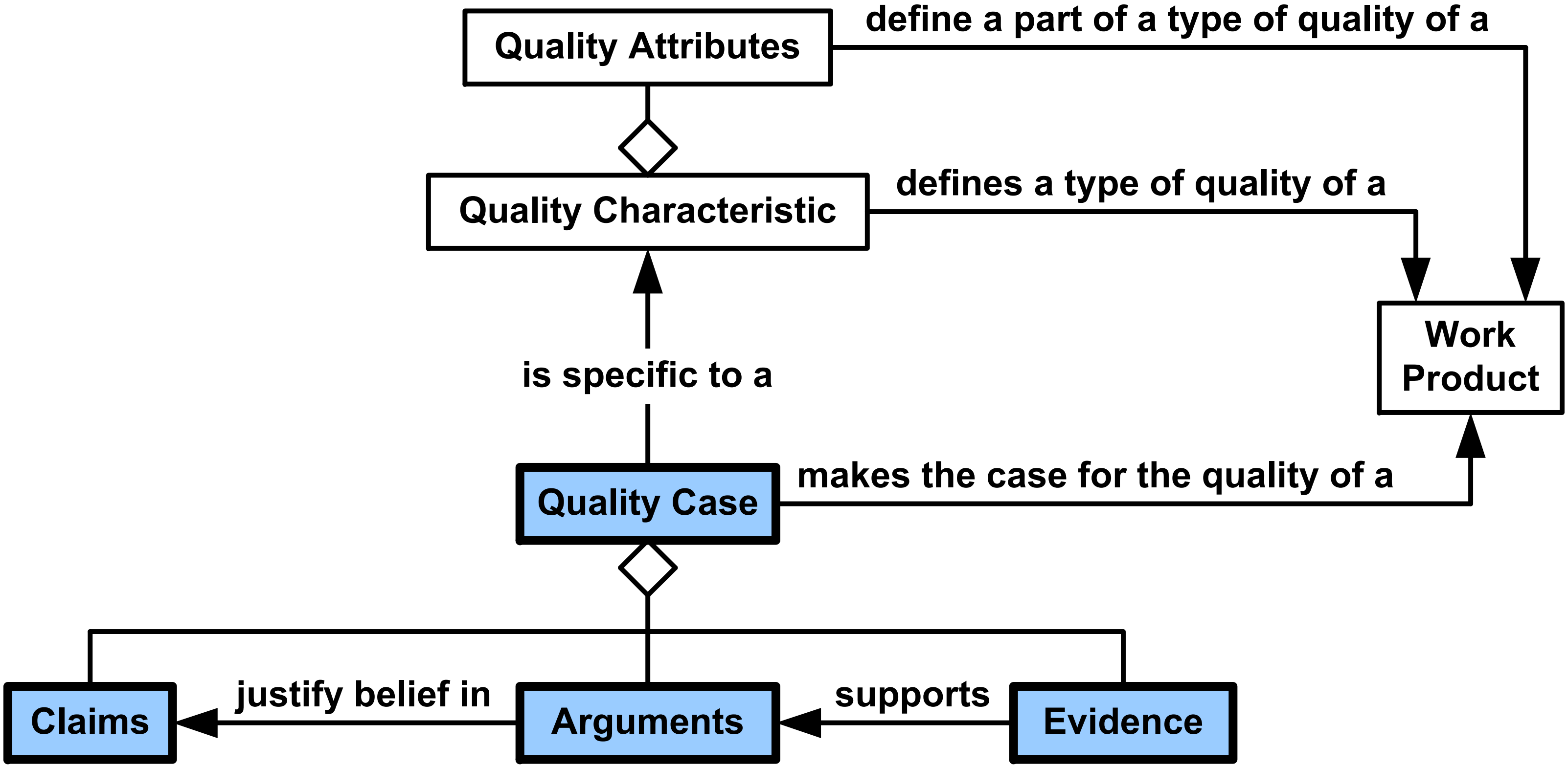
A Quality Case consists of the following types of Components:

1. **Claims**
   Developers' Claims that their Work Products have *Sufficient* Quality, whereby quality is defined in terms of the qualify characteristics and quality attributes defined in the official project quality model
2. **Arguments**
   Clear, Compelling, and Relevant Developer Arguments Justifying the Assessors' Belief in the Developers' Claims
   (e.g., decisions, inventions, trade-offs, analysis and simulation results, assumptions, and associated rationales)
3. **Evidence**
   Adequate Credible Evidence Supporting the Developers' Arguments (e.g., official project diagrams, models, requirements specifications and architecture documents; requirements repositories; analysis and simulation reports; test results; and demonstrations witnessed by the assessors)

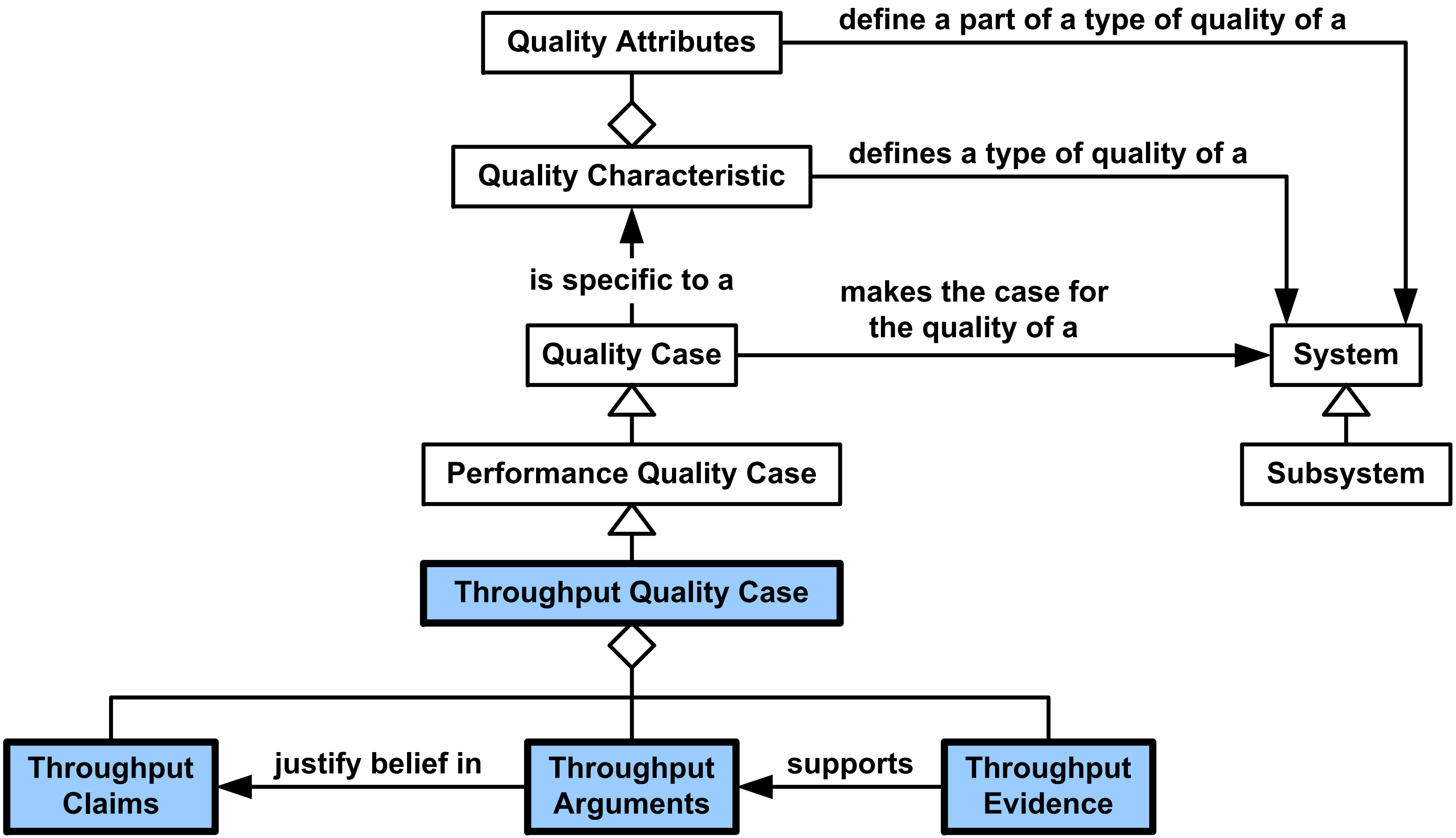
Quality Cases – Components$_2$



Quality Cases – Components$_3$



QUASAR Throughput Quality Case



Specialized QUASAR Quality Cases

QUASAR utilizes the following specialized types of Quality Cases:

1. Requirements Quality Cases
2. Architectural Quality Cases

QUASAR Version 1 only had Architectural Quality Cases.

QUASAR Versions 2 and 3 have Both Types of Quality Cases.

QUASAR *Requirements* Quality Cases$_1$

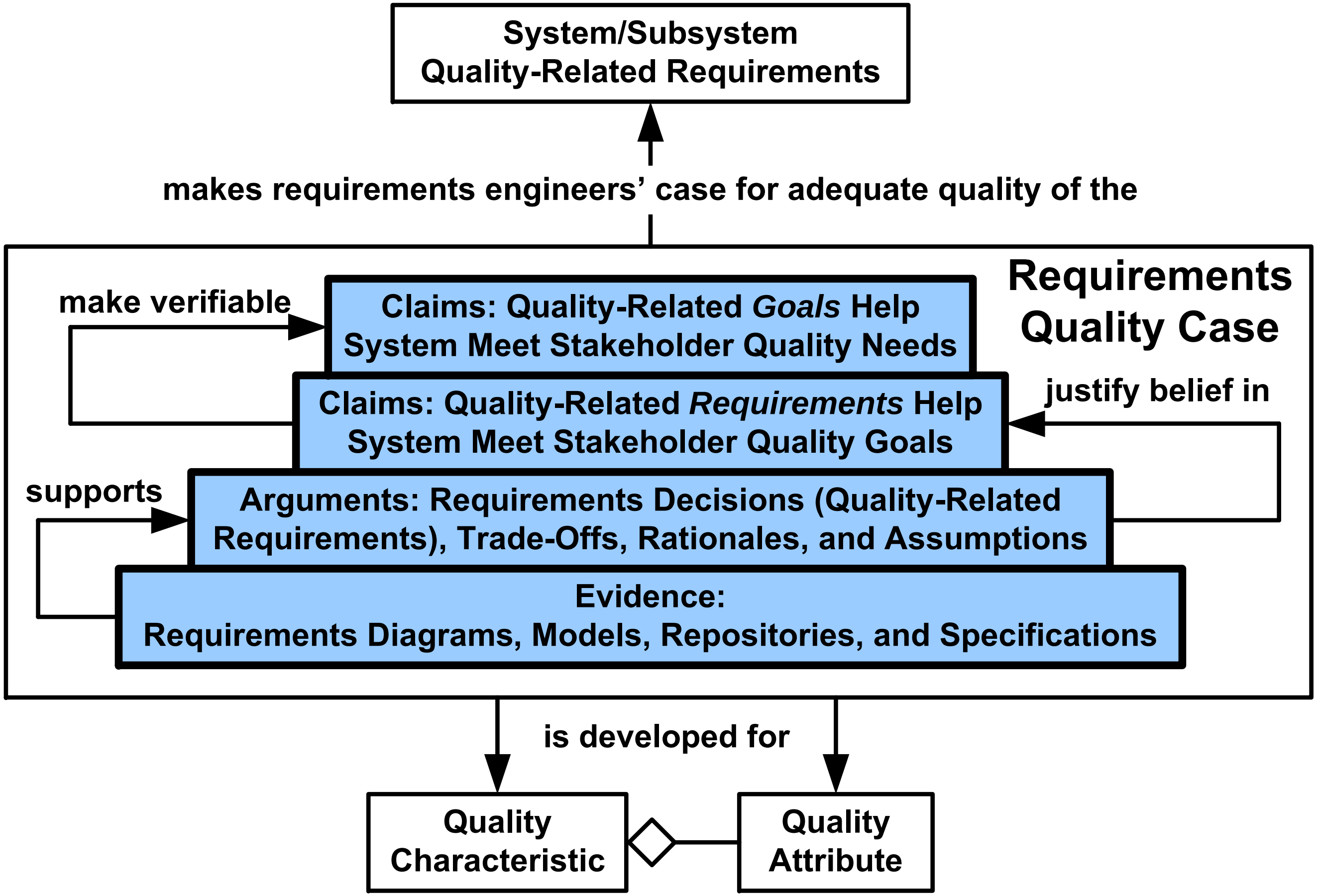
Requirements Quality Case

a Specialized Quality Case that is Limited to the Quality of Architecturally-Significant, *Quality-Related* Requirements

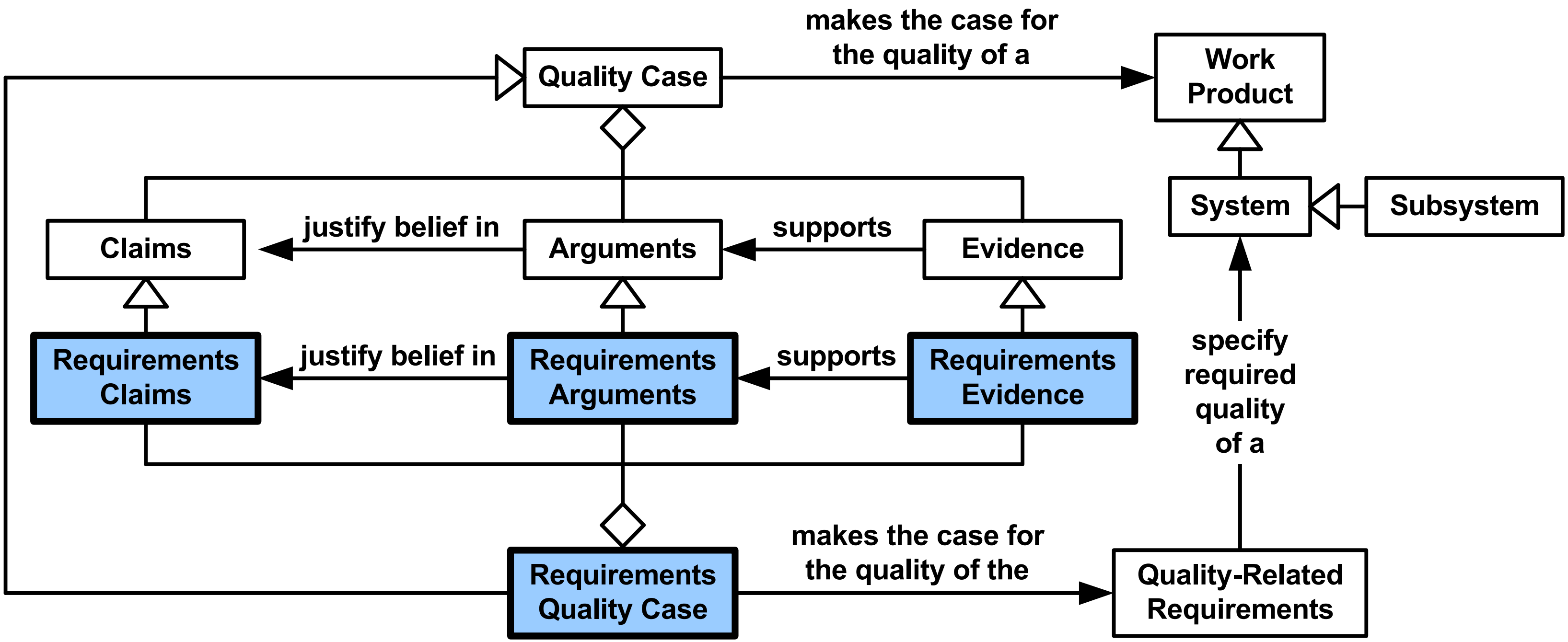
Makes Requirements Team's Case that their Relevant Requirements are:

- Ready to Drive the Engineering of the Architecture:
  - **Sufficient Quality**
    (e.g., are Correct, Complete, Consistent, Mandatory, Unambiguous, Verifiable, Usable, etc.)
  - **Sufficient Quantity**
    (e.g., Sufficient for Current Point in Project Schedule)
- Sufficient on which to base the System/Architecture Quality Assessment

QUASAR Requirements Quality Cases$_2$



QUASAR Requirements Quality Cases$_3$



QUASAR Requirements Quality Cases$_4$

Claims:

- Quality-Related Goals Sufficiently Meet Stakeholder Quality Needs
- Quality-Related Requirements Sufficiently Operationalize Quality Goals and Higher-Level Requirements from which They are Derived

Arguments:

- Requirements Decisions and Inventions
(e.g., Existence and Quality of Quality-Related Requirements)
- Requirements Engineering Trade-Offs, Assumptions, and Rationales

Evidence:

- Requirements Diagrams, Models, and Prototypes/Simulations
- Requirements Repositories and Specification Documents
- Requirements Inspection and Checking Results

Example Requirements Quality Case$_1$

Example Requirements *Reliability* Case **Claims**:

- Subsystem X Requirements Engineers claim that their Subsystem Goals Sufficiently Meet Stakeholder Reliability Needs:
  - “All Stakeholder Reliability Needs Allocated to Subsystem X have been Transformed into Subsystem X Reliability Goals.”
- Subsystem X Requirements Engineers Claim that their Subsystem Reliability Requirements Sufficiently Help the Subsystem Meet its Reliability Goals and higher-level Requirements:
  - “All Subsystem X Reliability Goals for this block/release have been Operationalized into Requirements.”
  - “All Subsystem X Reliability Requirements for this block/release have been Properly Engineered.”

Example Requirements Quality Case$_2$

Example Requirements *Reliability* Case **Arguments**:

- Subsystem X Reliability Requirements are:
  - Stored in the Project Requirements Repository
  - Published in the Subsystem X Requirements Specification
- Subsystem X Reliability Requirements in the Requirements Repository have been annotated as Reliability Requirements using Requirements Metadata.
- The Subsystem X Architects have verified the Feasibility of the Reliability Requirements given available Hardware and Software Technology.
- Appropriate Availability, Reliability, and Security Requirements Trade-Offs have been made.
- The Subsystem X Reliability Requirements have been Checked against a Checklist of Necessary Quality Characteristics (e.g., Correctness, Completeness, Consistency, Necessity, Unambiguous, Verifiability, and Usability).

Example Requirements Quality Case$_3$

Example Requirements *Reliability* Case **Evidence**:

- **Requirements Traceability Matrix** showing Allocation of Reliability Requirements from Parent Subsystem A to Derived Reliability Requirements in Subsystem X
- Project **Requirements Repository** with Subsystem X Reliability Requirements identified
- **Reliability Section** in Subsystem X **Requirements Specification** Document
- **Reliability Subsection** of Subsystem X **Requirements Inspection Report**

Requirements Quality Case Challenges$_1$

Most Requirements Engineers are not trained in the Proper Engineering of Non-Functional Requirements (e.g., Quality Requirements).

Vague Unverifiable Goals are often Mistaken as Requirements.

Stakeholders often do *Not* know where to set Thresholds on Quality Measurement Scales.

Requirements Repository is Huge and Complex.

Only Small Subset of the Requirements is Relevant to any specific Quality Characteristic or Quality Attribute for any specific Subsystem.

Tracing Quality Requirements is more Difficult than Tracing Functional Requirements.

Requirements Quality Case Challenges$_2$

Hard to know that:

- Arguments *Sufficiently* Justify Belief in Claims
- Evidence *Sufficiently* Supports Arguments
- Degree of Confidence

Need practical way to Communicate, Summarize, and Act as Index to the Quality Case Essentials.

QUASAR *Architectural* Quality Cases

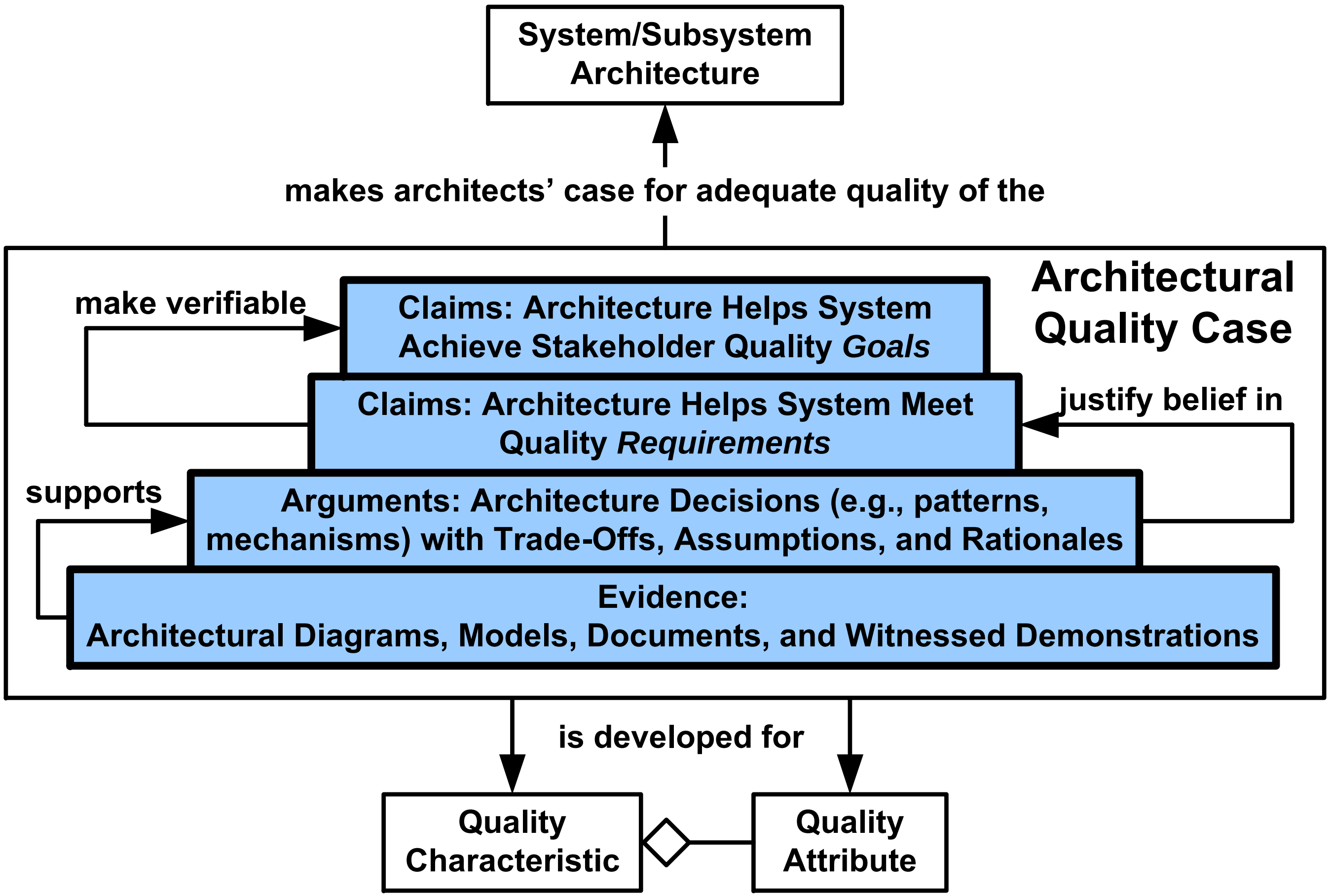
Architecture Quality Case

a Specialized Quality Case that is Limited to the Quality of the System/Subsystem Architectures

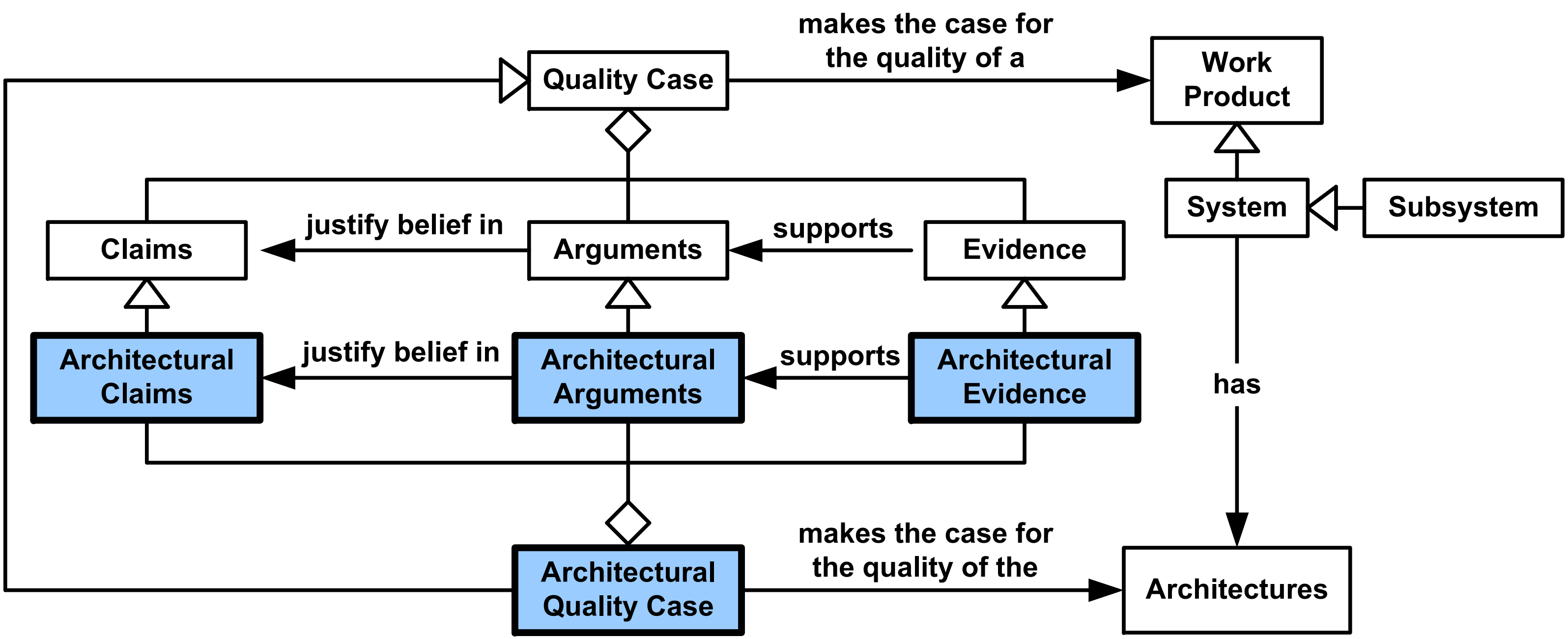
Make Architectures Team's Case that their Architecture(s) are:

- Ready to Drive the Engineering of the Design, Implementation, Integration, and Testing:
  - **Sufficient Quality**
    (e.g., Adequately Support the System's or Subsystem's Ability to meet its Quality-Related Requirements)
  - **Sufficient Quantity**
    (e.g., Sufficient for Current Point in Project Schedule)

QUASAR Architectural Quality Cases$_2$



QUASAR Architectural Quality Case



QUASAR Architectural Quality Cases$_4$

Claims:

- Architectures Sufficiently Supports System/Subsystem’s Ability to Meet All Quality Goals and Quality Requirements

Arguments:

- Architectural Decisions (e.g., Architectural Mechanisms, Patterns, and Styles as well as Choice of OTS Components)
- Architectural Engineering Trade-Offs, Assumptions, and Rationales

Evidence:

- Architectural Diagrams, Models (Static and Dynamic), and Prototypes
- Architecture Documents and Architectural Whitepapers
- Properly Documented Architectural Simulation and Test Results
- Properly Witnessed Demonstrations

Example Architectural Quality Case$_1$

Example **Protocol Interoperability** Case

Goal Claims:

- Subsystem X Architects Claim that their Subsystem Architecture Sufficiently Supports its following derived Protocol Interoperability *Goals*:
  - "Subsystem X will correctly use the interface protocols of all relevant external systems."
  - "Subsystem X will use open interface standards (i.e., industry standard protocols) when communicating with all external systems."

Example Architectural Quality Case$_2$

Example *Protocol Interoperability* Architecture Case **Claims**:

- Subsystem X Architects Claim that their Subsystem Architecture Sufficiently Supports its following derived Protocol Interoperability *Requirements*:
  - “The subsystem shall use open interface standards (i.e., industry standard protocols) when communicating with external systems across all key interfaces identified in document X.”
  - “The subsystem shall use the Ethernet over RS-232 for communication across interface X with external system Y.”
  - “The subsystem shall use HTTPS for communicating securely when performing function X across interface Y with external system Z.”

Example Architectural Quality $Case_3$

Example *Protocol Interoperability* Architecture Case **Arguments:**

- **Layered Architecture**
  The subsystem uses a layered architecture.
  Rationale: Interface layer supports interoperability.
- **Modular Architecture**
  The subsystem architecture is highly modular.
  Rationale: Architecture includes modules (proxies) for interoperability.
- **Wrappers and Proxies**
  The subsystem architecture includes proxies that wrap the interfaces to external subsystems.
  Rationale: Proxies localize and wrap external interfaces.
- **Service Oriented Architecture (SOA)**
  The subsystem service oriented architecture uses XML, SOAP, and UDDI to publish and provide web services over the Internet to external client systems.
  Rationale: Standard languages and protocols support interoperability between heterogeneous systems.

Example Architectural Quality Case$_4$

Example *Protocol Interoperability* Architecture Case **Evidence:**

- **Context Diagram**
  (shows external interfaces requiring protocols)
- **Architectural Class Diagram**
  (shows modularity and location of proxies and web services)
- **Allocation Diagram**
  (shows allocation of software modules to hardware - modularity)
- **Layer Diagram**
  (shows architectural layers)
- **Activity/Collaboration Diagrams**
  (show proxies, wrappers, and the source and use of services)
- **Interoperability Whitepaper**
- **Vendor-Supplied Technical Documentation**
  (show COTS product support for SOA and associated protocols)

QUASAR Quality Case Responsibilities

Requirements Engineers and Architects' Responsibilities:

- Prepare Quality Cases
- Provide Preparation Materials (including Presentation Materials and Quality Cases) to Assessors Prior to Assessment Meetings
- Present Quality Cases (Make their Case to the Assessors)
- Answer Assessors' Questions

Assessor Responsibilities:

- Prepare for Assessments
- *Actively* Probe Quality Cases
- Develop Consensus regarding Assessment Results
- Determine and Report Assessment Results:
  - Present Outbriefs
  - Publish Reports

Architectural Quality Case Challenges

Huge and Complex System Architectures

Only Small Subset of the Architecture (i.e., not view but focus area) is Relevant to any one Quality Factor or Quality Attribute.

Quality Cases still Contain a Large Amount of Information.

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

Easy to get Lost in Real-World Quality Cases

Architectual Quality Case Challenges

Hard to know that:

- Arguments *Sufficiently* Justify Belief in Claims
- Evidence *Sufficiently* Supports Arguments
- Degree of Confidence

Need practical way to Communicate, Summarize, and Act as Index to the Quality Case Essentials

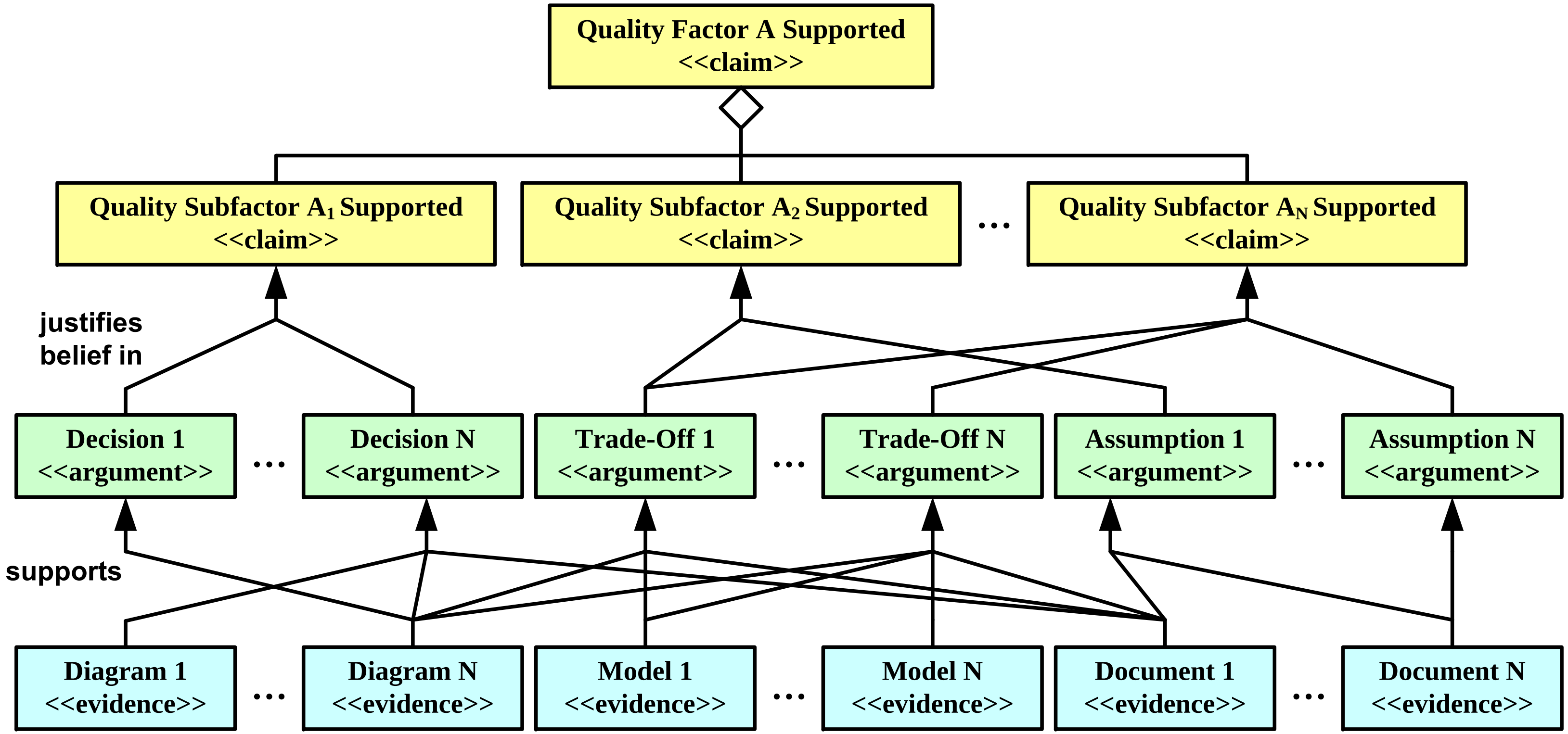
Quality Case Diagram

A *Layered* UML Class Diagram that Labels and Summarizes the parts of a *Single* Quality Case:

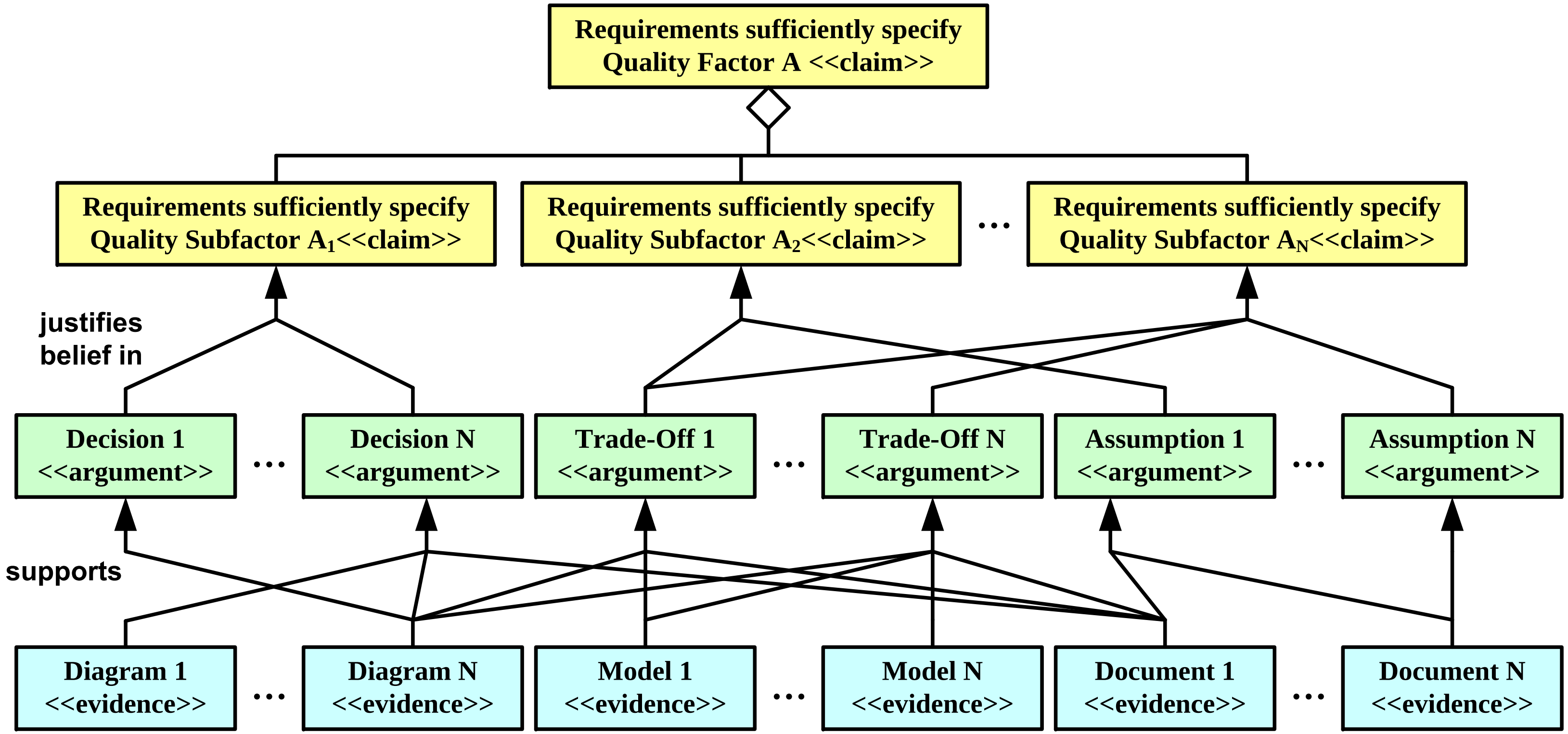
- Classes:
  - Claims
  - Arguments
  - Evidence
- Relationships Among Them:
  - Aggregation Relationships Between Claims
  - “Justifies Belief In” Associations from Arguments to Claims
  - “Supports” Associations from Evidence to Arguments

Acts as an Index and Guide to the Quality Case

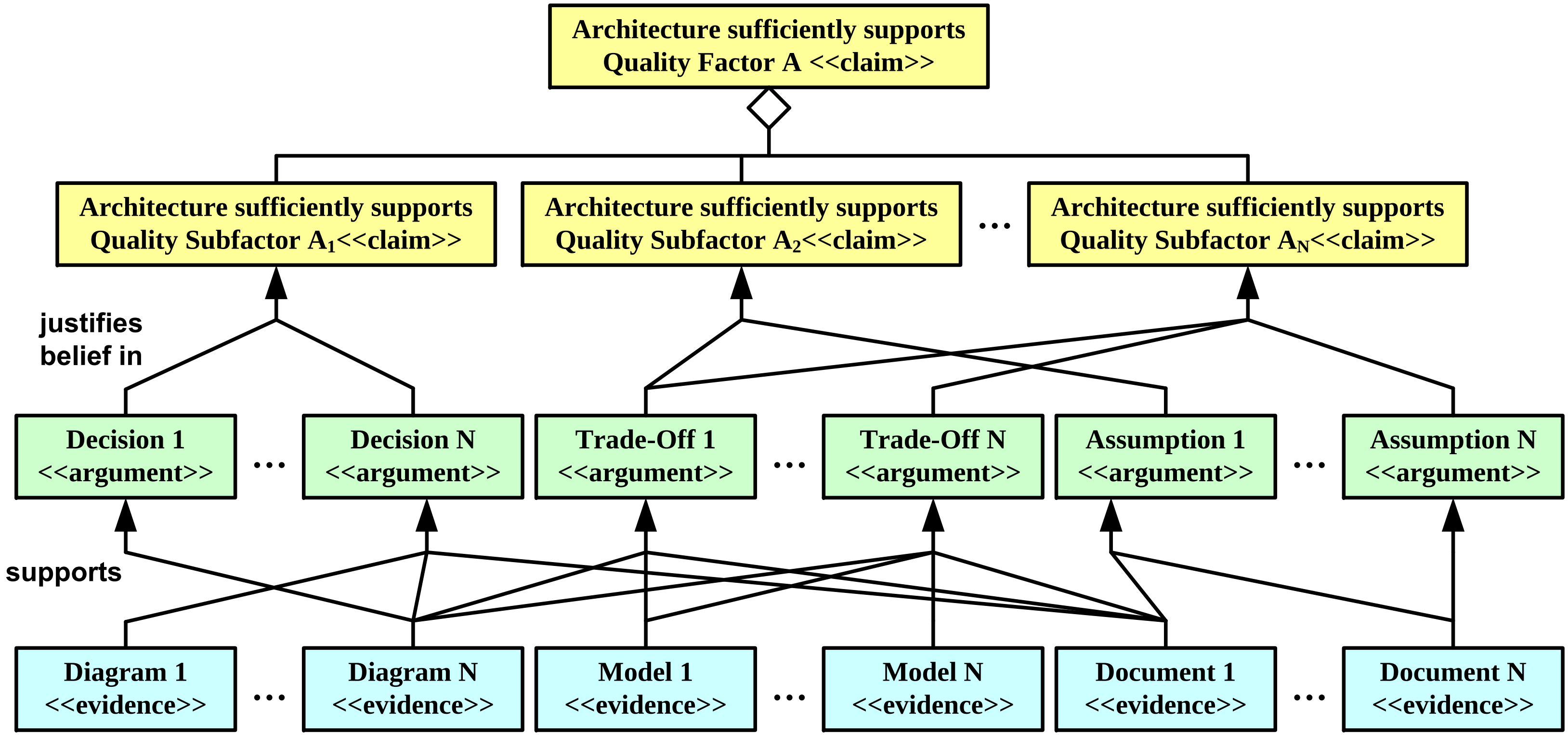
Quality Case Diagram Notation



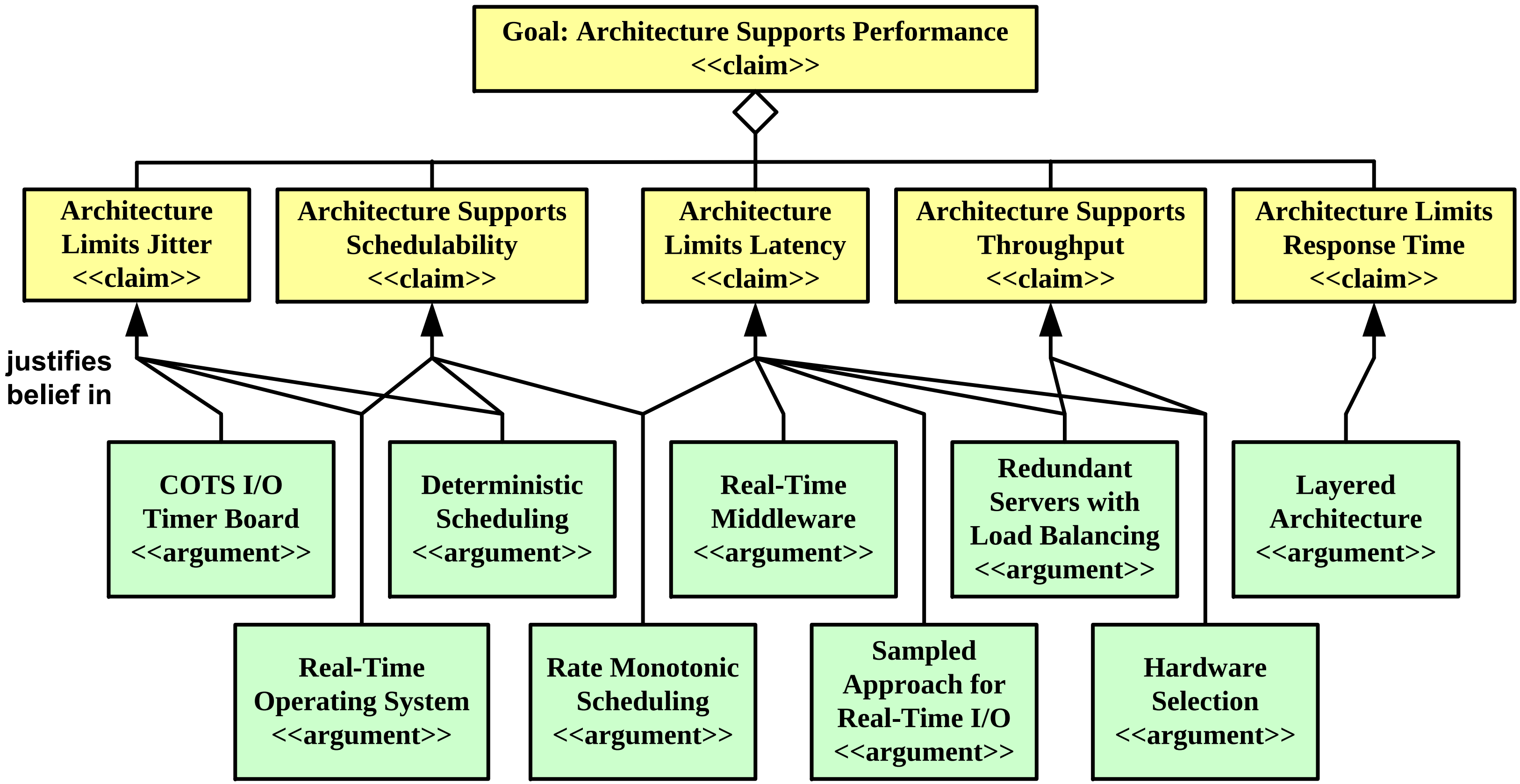
Generic Requirements Quality Case



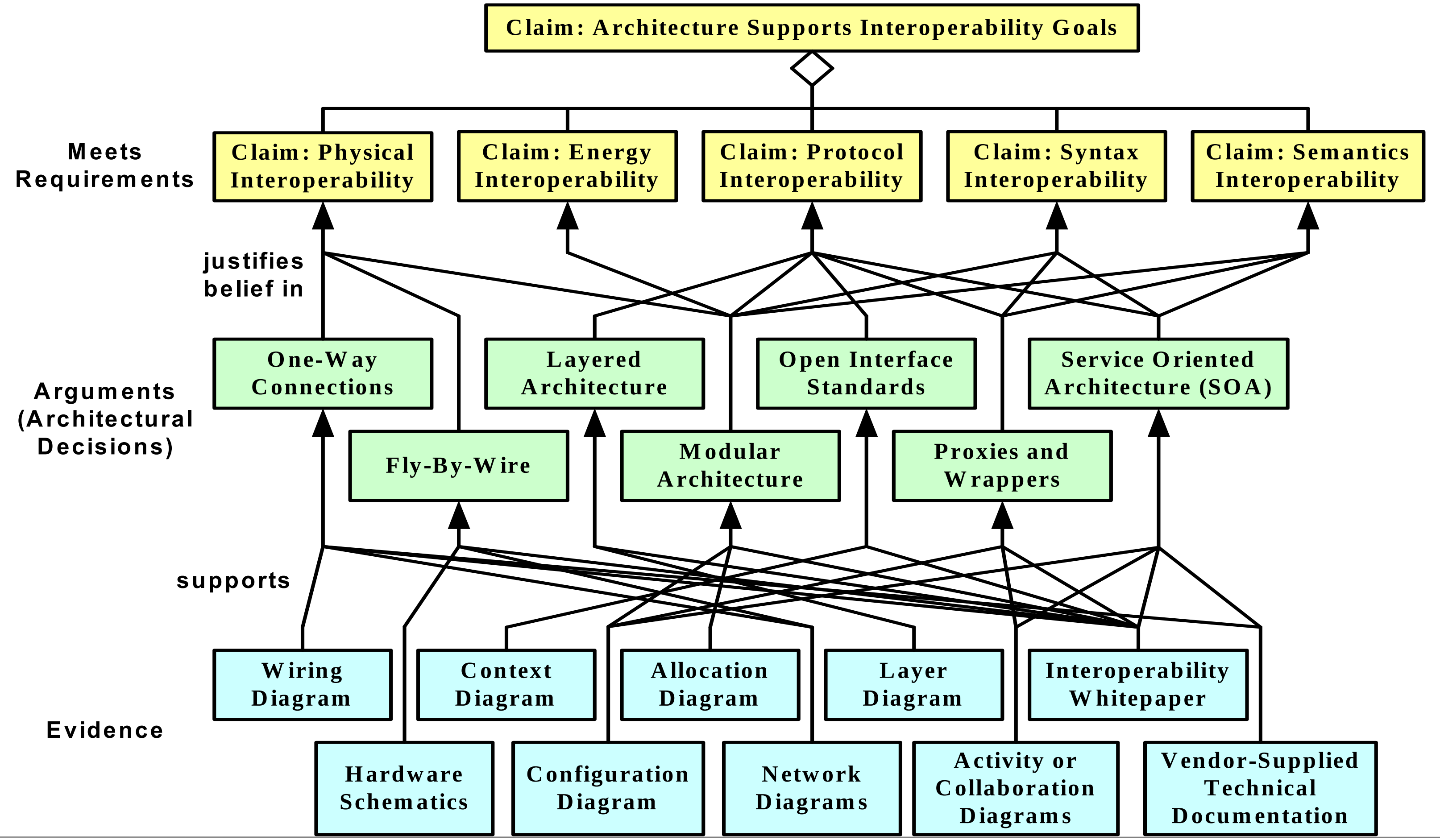
Generic Architecture Quality Case



Example *Partial* Architectural Performance Case Diagram



Architectural Interoperability Case Diagram



Topics

Requirements and Architecture Challenges

Underlying Concepts:

- **System ◄**

QUASAR Method

Reasons to use QUASAR

QUASAR – Today and Tomorrow

What is a System?

System

a Major, Cohesive, Executable, and Integrated Set of *Architectural Elements* that Collaborate to Provide the Capability to Perform one or more related *Missions*

Systems are Decomposed into Architectural Components (e.g., Subsystems):

- Data
- Documentation
- Hardware
- Software
- Manual Procedures
- Personnel (e.g., Roles such as Operators and Administrators)
- Equipment, Facilities, Materials, and Tools

Systems Imply

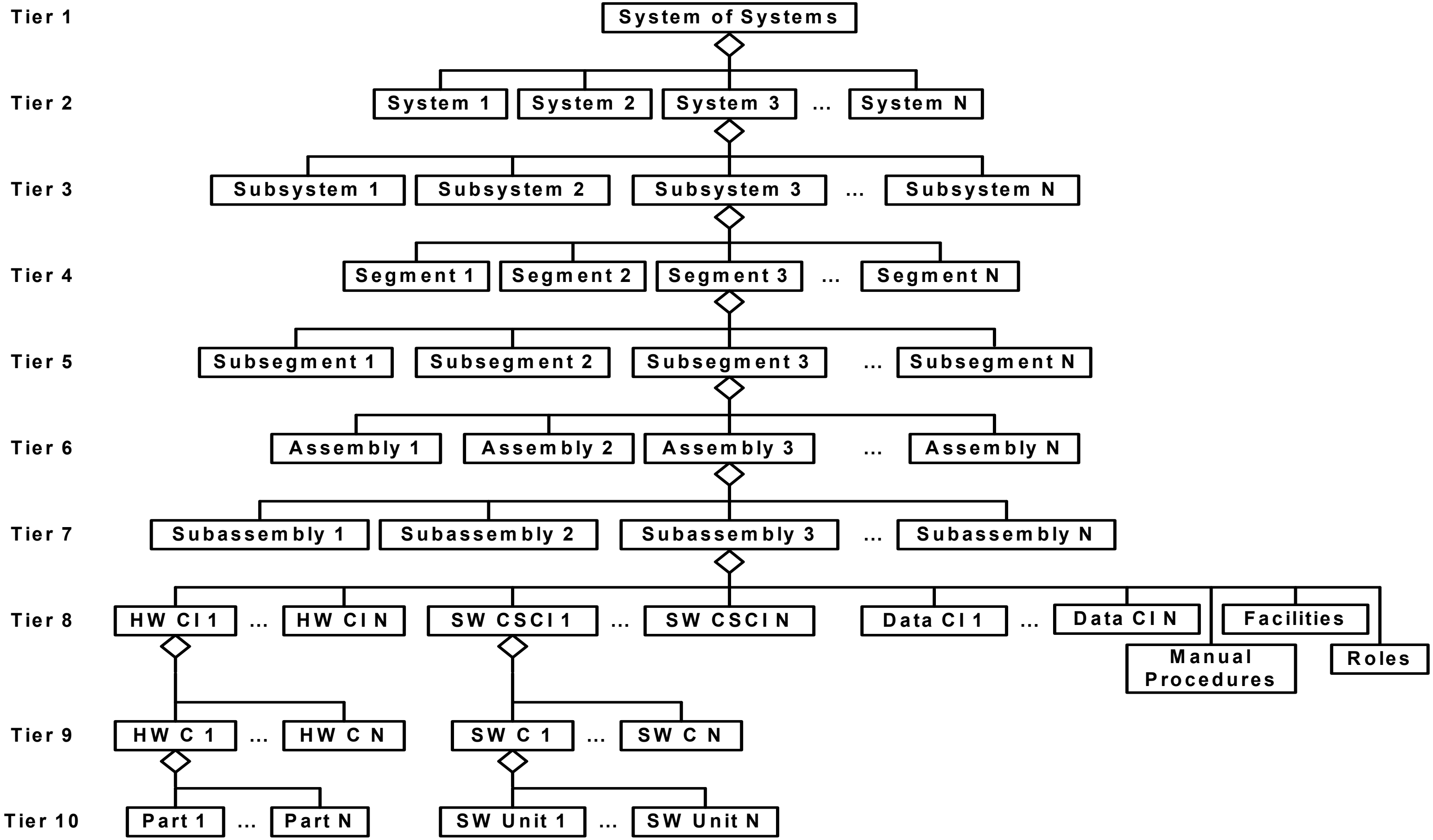
Multiple Static and Dynamic Logical and Physical “Structures” that exist at Multiple ‘Tiers’ in the System:

- Static Functional Decomposition Logical Structure
- Static Subsystem Decomposition Physical Structure
- Hardware, Software, and Data Structures
- Allocation Structure (Software and Data to Hardware)
- Network Structure
- Concurrency (Process) Structure

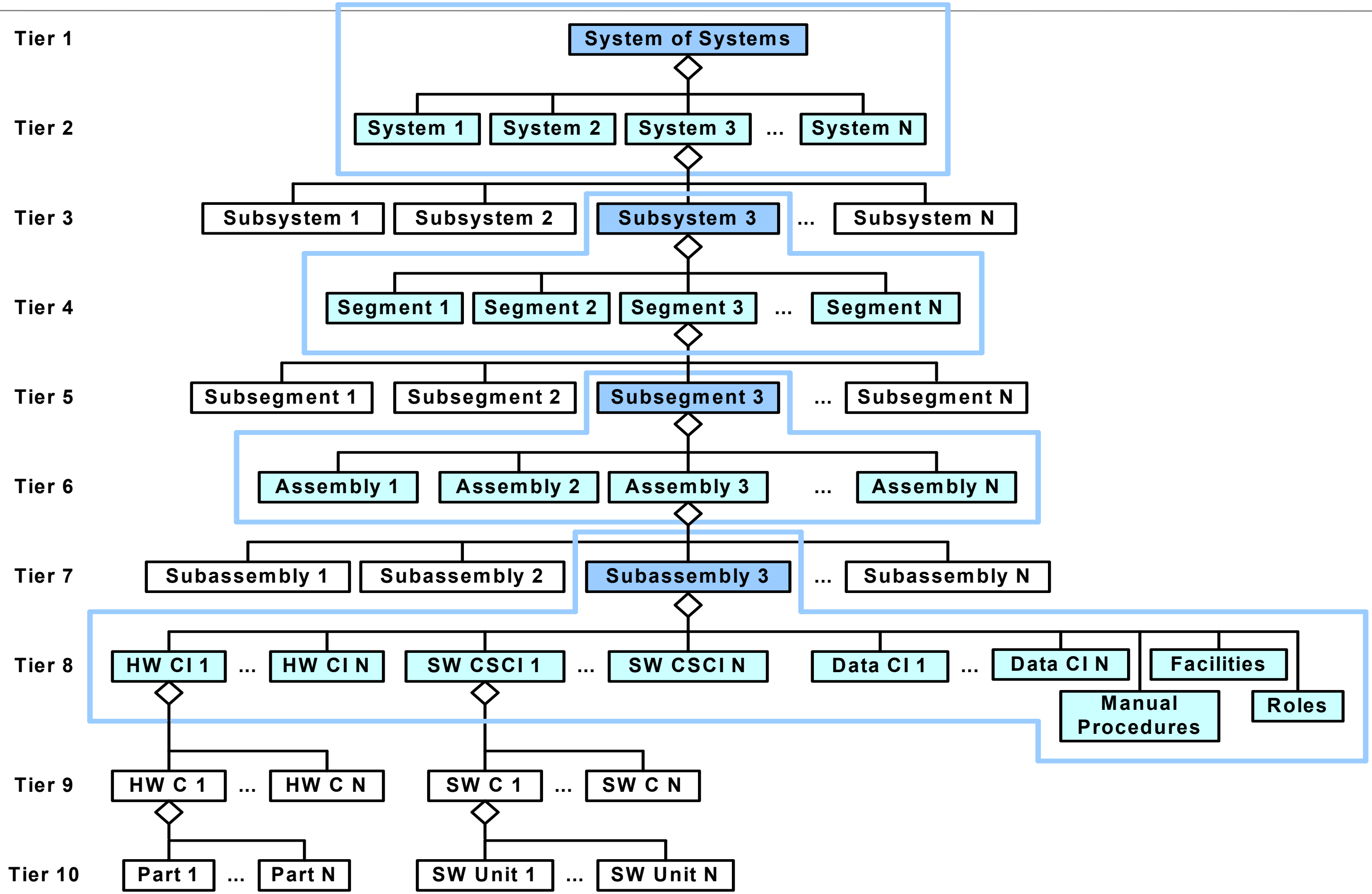
Multiple Specialty Engineering Focus Areas (e.g., Performance, Reliability, Safety, and Security)

Requirements are Derived and Allocated to Lower-Level Architectural Elements

Example System
(Static Physical System Decomposition View Only!)



Example QUASAR Scope – Four Assessments



Topics

Requirements and Architecture Challenges

Underlying Concepts:

System Architecture ◄

QUASAR Method

Reasons to use QUASAR

QUASAR – Today and Tomorrow

What is a System Architecture?[1]

System Architecture

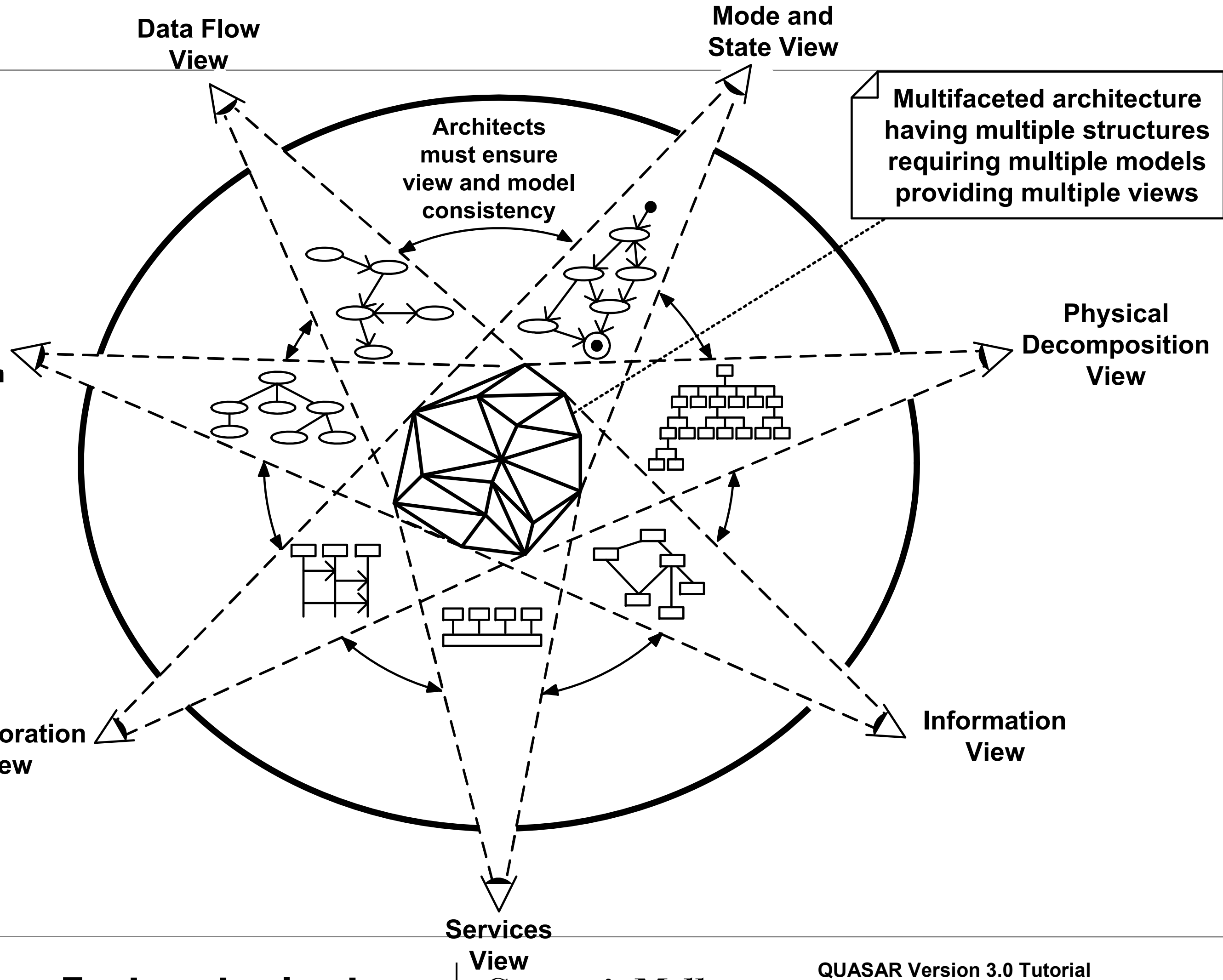
the *Most Important, Pervasive, Top-Level, Strategic Decisions, Inventions, Engineering Trade-Offs, Assumptions,* and associated *Rationales* about *How* a System's Architectural Elements will collaborate to meet the System's Derived and Allocated Requirements

What is a System Architecture?$_2$

System Architecture Includes:

- **The System's Numerous Static and Dynamic, Logical and Physical Structures**
  (i.e., Essential Architectural Elements, their Relationships, their Associated Blackbox Characteristics and Behavior, and how they Collaborate to Support the System's Mission and Requirements)
- **Architectural Decisions, Inventions, and Tradeoffs**
  (e.g., Styles, Patterns, and Mechanisms used to ensure that the System Achieves its Architecturally-Significant Product and Process Requirements (esp. Quality Requirements or 'ilities')
- **Strategic and Pervasive Design-Level Decisions**
  (e.g., using a *Design* Paradigm such as Object-Orientation or Mandated Widespread use of common Design Patterns)
- **Strategic and Pervasive Implementation-Level Decisions**
  (e.g., using a Safe Subset of C++)

Some Example Views of Models of Structures



Architecture vs. Design

Architecture	Design
Pervasive (Multiple Components)	*Local* (Single Components)
Strategic Decisions and Inventions	*Tactical* Decisions and Inventions
Higher-Levels of System	*Lower-Levels* of System
Huge Impact on Quality, Cost, & Schedule	*Small Impact* on Quality, Cost, & Schedule
Drives Design and Integration Testing	*Drives* Implementation and Unit Testing
Driven by Requirements and Higher-Level Architecture	*Driven by* Requirements, Architecture, and Higher-Level Design
Mirrors Top-Level Development Team Organization (Conway's Law)	*No* Impact on Top-Level Development Team Organization

Architectural Documentation Current-State

System Architecture Documents:

- Mostly natural language Text with Visio-like Diagrams (Cartoons)
- Logical (functional) and Physical Architecture

DOD Architecture Framework (DODAF):

- All-Views, Operational Views, Systems Views, and Technical Standards Views for allocating Responsibilities to Systems and Supporting System Interoperability

Models (both static and dynamic; logical and physical):

- Tailored UML becoming *de facto* Industry Standard
- SysML starting to become Popular

Visio Diagrams as Wall Posters

Whitepapers, Reports, and other Specialty-Engineering Documents:

- Performance, Fault Tolerance, Reliability, Safety, Security

What an Architecture is *NOT*

A System Architecture is *Not* an Architectural:

- Plan
- Method
  (architecting procedures and architecture documentation standards)
- Team Organization Chart
  (in spite of Conway's Law)
- Development Schedule

QUASAR assesses Actual Architectures:

- As they Currently Exist (i.e., a Snapshot in Time)
- Not Good Intentions

Topics

Requirements and Architecture Challenges

Underlying Concepts:

System Architect ◄

QUASAR Method

Reasons to use QUASAR

QUASAR – Today and Tomorrow

What is a Systems Architect?$_1$

A *Role* played by a *Systems Engineer*, who is Responsible for:

- *Developing* one or more System or Subsystem Architectures
- *Ensuring* the *Quality* of the System or Subsystem Architectures
- Ensuring the *Integrity* of the System or Subsystem Architectures during Design, Implementation, Manufacture, and Deployment (e.g., Installation and Configuration)
- *Communicating* the System or Subsystem Architectures to their Stakeholders
- *Maintaining* the System or Subsystem Architectures

What is a Systems Architect? 2

A *Role* that*:*

- Requires Significant:
  - Training
  - Experience (Apprenticeship)
  - Mindset:
    - Big Picture
    - Generalist
  - Communications Ability
- Should Probably be a *Job Title*
- But may *Not* be a Job Title

Topics

Requirements and Architecture Challenges

Underlying Concepts:

Architecturally Significant Requirements ◄

QUASAR Method

Reasons to use QUASAR

QUASAR – Today and Tomorrow

Architecturally Significant Requirement

Architecturally Significant Requirements

any Requirement that has a Significant Impact on a System / Subsystem Architecture

Architecturally Significant Requirements typically include:

- Quality Requirements, which specify a Minimum Amount of some Quality Characteristics or Quality Attribute
- Architectural Constraints
- Primary Mission Functional Requirements (Feature Sets)

Quality Requirements are often the:

- Most Important
- Least Well Engineered

Quality Requirements

Format

Conditions – Quality Criteria – Quality Thresholds

Under condition(s) X, the system/subsystem shall exhibit quality criterion Y meeting or exceeding threshold Z.

Bad Example(s)

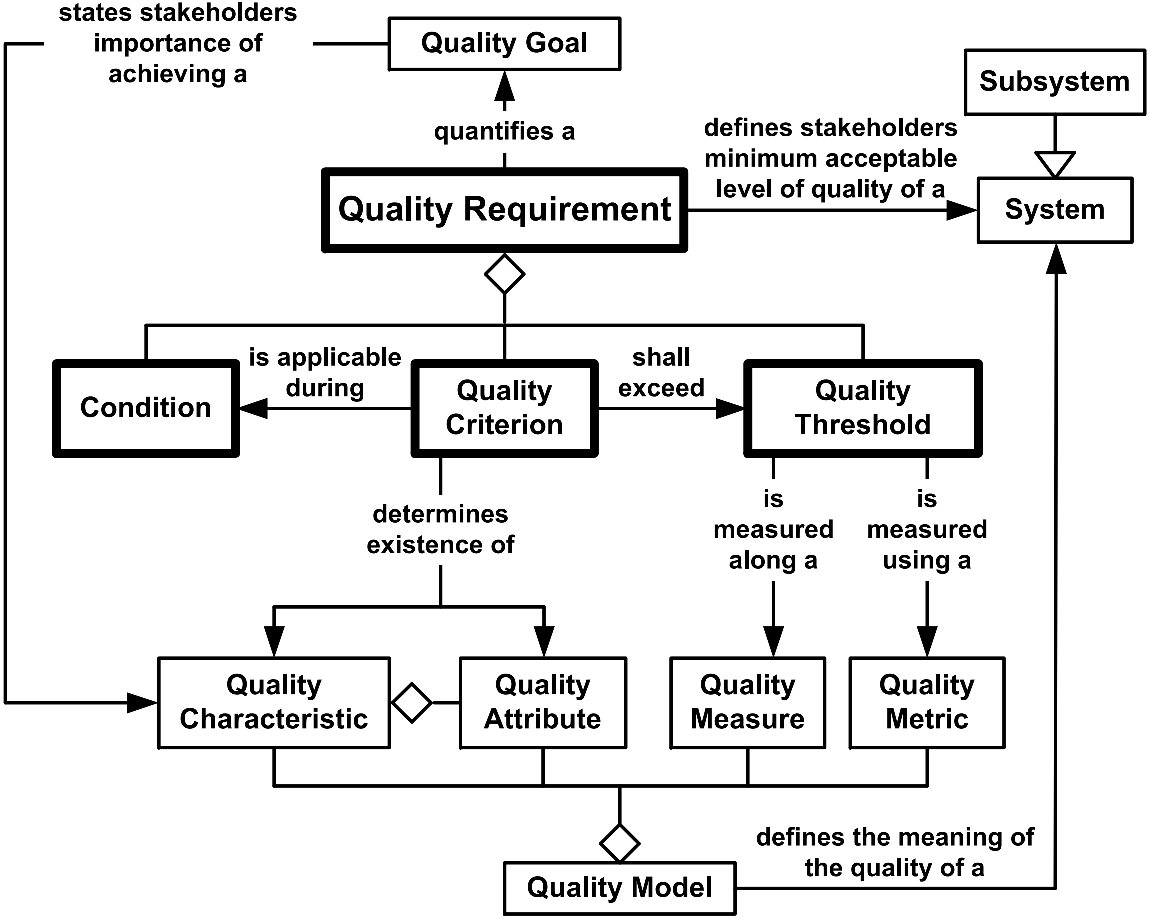
The system shall be highly reliable, robust, safe, secure, stable, etc.

Good Example (Stability)

Under normal operating conditions*, the system shall ensure that the mean time between the failure of mission critical functionality* is at least 5,000 hours of continuous operation.

* Must be Properly defined in the Project Glossary

Quality Requirements – Components



Topics

Requirements and Architecture Challenges

Underlying Concepts

QUASAR Method ◄

Reasons to use QUASAR

QUASAR – Today and Tomorrow

Definition

QUality Assessment of System Architectures and their Requirements

a Well-Documented and Proven Method based on the use of *Quality Cases* for *Independently* Assessing the *Quality* of:

- Software-intensive *System* / Subsystem *Architectures* and the
- *Architecturally Significant Requirements* that Drive Them

QUASAR Versions

Version 1 (July 2006) emphasized the quality assessment of architectures over assessment of architecturally-significant requirements.

Version 2 (February 2007) addresses the quality assessment of *both* architectures *and* their architecturally-significant requirements.

Version 3 (October 2007) simplifies phases and better addresses summary reporting.

QUASAR Philosophy$_1$

Informal *Peer Reviews* are Inadequate:

- Too Informal
- Lack of *Independent* Expert Input
- Requirements and Architecture are too Important

Quality Requirements:

- Most important Architecturally-Significant Requirements
- Largely Drive the System Architecture
- Criteria against which the System Architecture is Assessed

QUASAR Philosophy$_2$

Requirements Engineers (REs) should *Make Case* to Assessors:

- REs *should* know Stakeholder Needs and Goals
- REs *should* know What they Did and Why (Architecturally-Significant Requirements, Rationales, & Assumptions)
- REs *should* Know Where they Documented their Requirements Work Products

Architects should *Make Case* to Assessors:

- Architects *should* know Architecturally-Significant Requirements
- Architects *should* know What they Did and Why (Inventions, Decisions, Rationales, Trade-Offs, and Assumptions)
- Architects *should* know Where Documented their Architectural Work Products

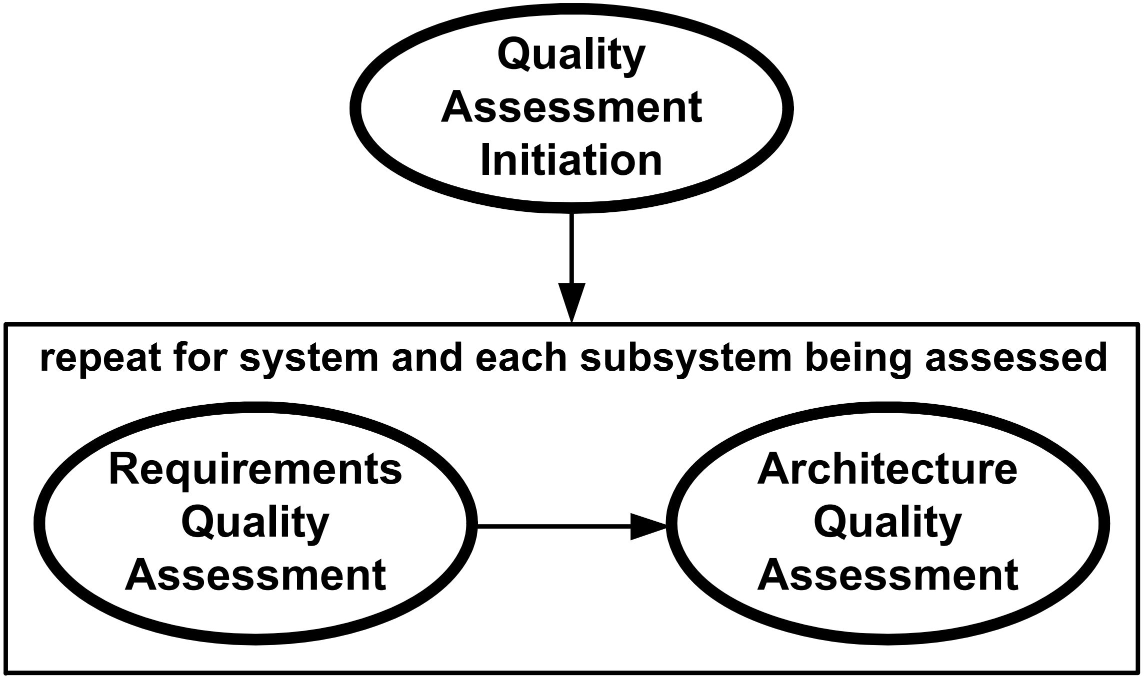
QUASAR Philosophy$_3$

Assessors should *Actively* Probe Quality Cases:

- **Claims Correct and Complete?**
  Do the Claims include *all* relevant Quality Characteristics, Quality Attributes, Quality Goals, and Quality Requirements?
- **Arguments Correct, Complete, Clear, and Compelling?**
  Do the Arguments include *all* relevant Quality Factors, Quality Attributes, Quality Goals, Quality Requirements, Decisions, Inventions, Trade-offs, Assumptions, and Rationales?
- **Arguments Sufficient?**
  Are the Arguments Sufficient to Justify the Claims?
- **Evidence Sufficient?**
  Is the Evidence Sufficient to Support the Arguments?
- **Current Point in the Schedule?**
  Are the Claims, Arguments, and Evidence appropriate for the Current Point in the Schedule?

QUASAR Method – Three Phases

1. Quality Assessment Initiation (QAI)
2. Requirements Quality Assessment (RQA)
3. Architecture Quality Assessment (AQA)

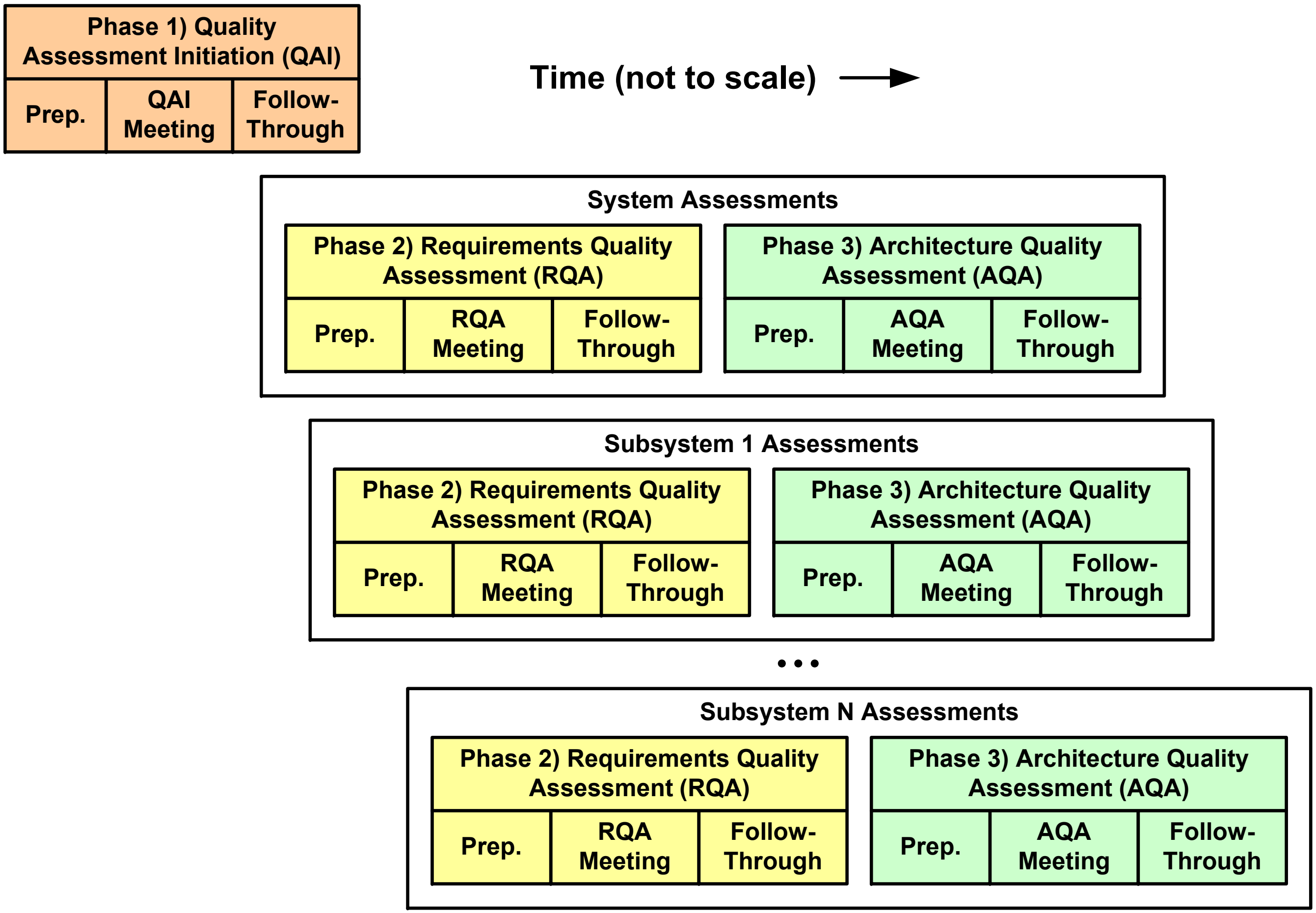


QUASAR Methods – Three Tasks

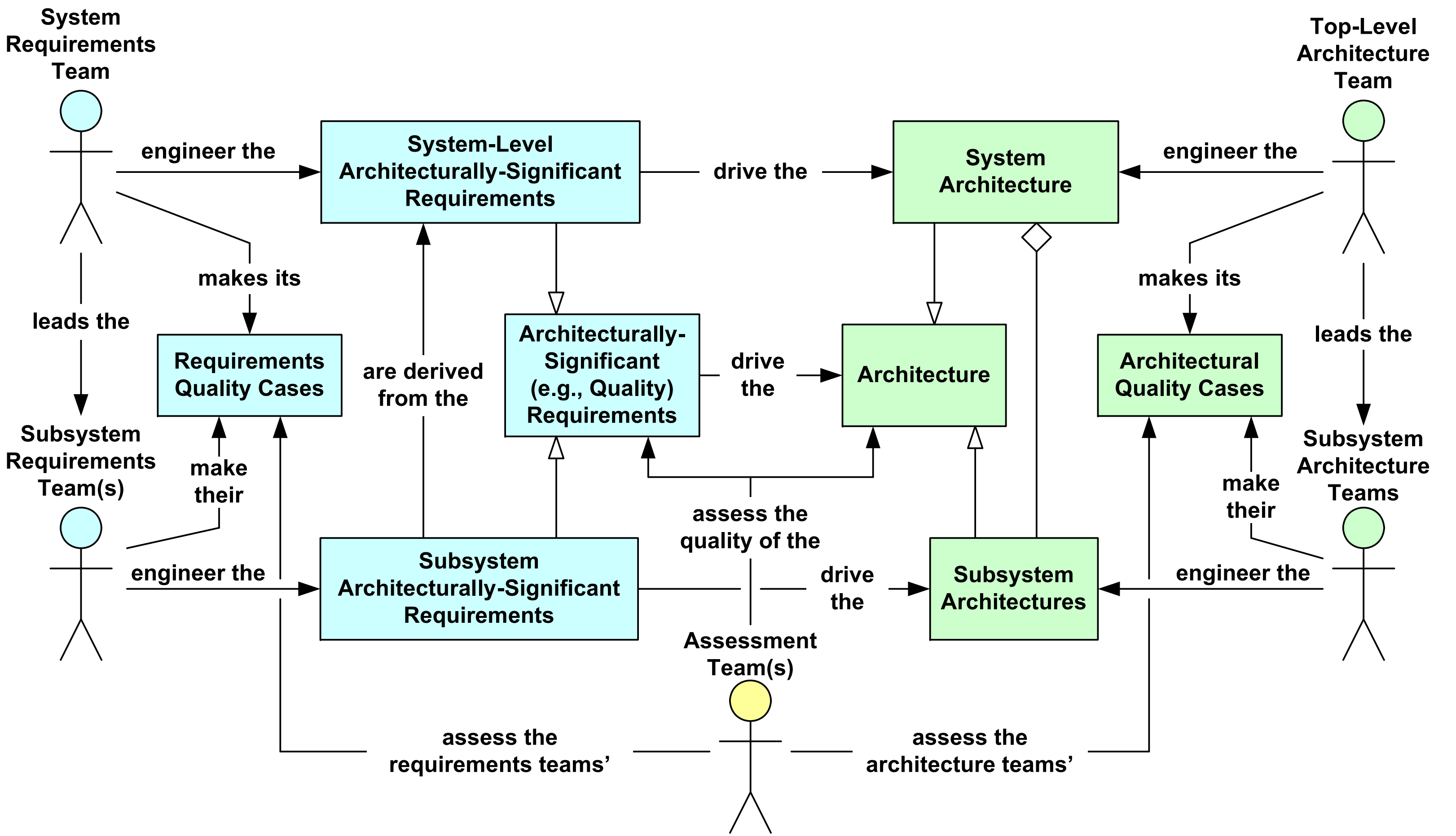
Each Phase consists of 3 similar ***Tasks***:

1. Preparation
2. Meeting
3. Follow-Through

QUASAR Phases and Tasks



Quasar Teams and their Work Products



Topics

Requirements and Architecture Challenges

Underlying Concepts

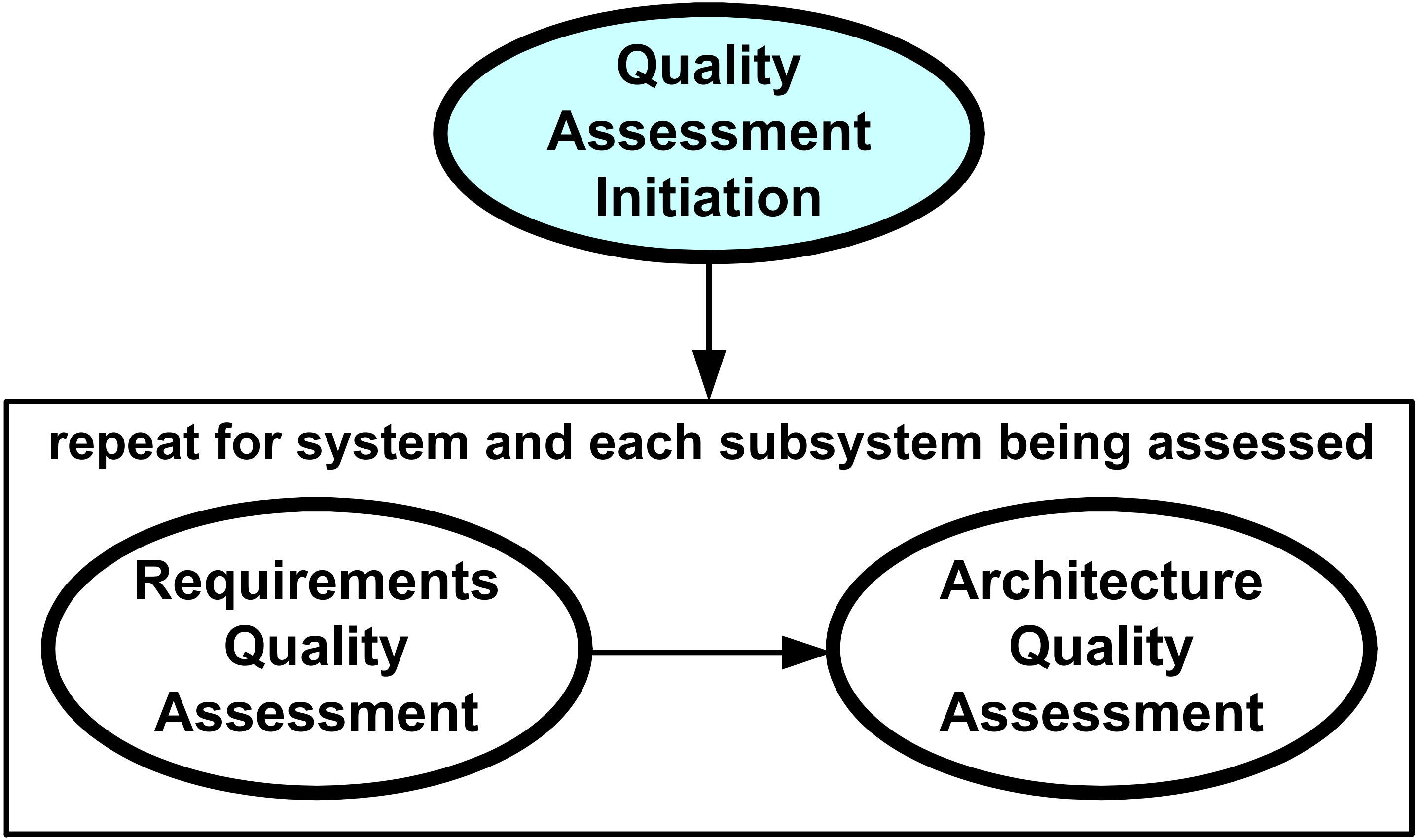
QUASAR Method:

- **Phase 1) Quality Assessment Initiation (QAI) ◄**
- Phase 2) Requirements Quality Assessment (RQA)
- Phase 3) Architecture Quality Assessment (AQA)

Reasons to use QUASAR

QUASAR – Today and Tomorrow

Quality Assessment Initiation (QAI)



Phase 1) QAI – Objectives

Prepare Teams for Requirements and Architecture Assessments

Develop Consensus:

- Scope of Assessments
- Schedule Assessments
- Tailor the Assessment Method and associated Training Materials

Produce and Publish Meeting Outbrief and Minutes

Manage Action Items

Capture Lessons Learned

Tailor/Update QUASAR Method and Training Materials

Phase 1) QAI – Principles

It is Important to:

- **Develop Consensus** among Teams
- **Scope the Assessment** to meet Project-Specific Needs and Resources
- **Tailor the Assessment Method** to meet specific Needs of the Overall Assessment

Subsystem Assessments must be scheduled to **Ensure Availability** of the:

- Requirements and Architecture
- Required Resources (e.g., people and funding)

Phase 1) QAI – Challenges$_1$

Acquirer and Development Organizations may Disagree as to the:

- Need to Independently Perform Quality Assessments
- Relative Importance of Quality Factors, Quality Attributes, and Related Goals and Requirements

It can be Difficult to reach Consensus on the Scope of the Assessments in terms of the:

- Number and Identity of Subsystems to Assess
- Number and Identity of Quality Factors and Quality Attributes
- Tailoring of the QUASAR Method

Phase 1) QAI – Challenges$_2$

Quality Assessments of System and Subsystem Architectures and their Architecturally-Significant Requirements may not have been included in the Project:

- Request for Proposal (FRP)
- Contract
- Budget and Schedule

It is often very Difficult to obtain Commitment of Resources:

- Availability of Requirements Engineers and Systems Architects
- Availability of Assessors with Adequate Experience and Expertise
- Consensus on Schedule
- Budget Funding to Pay for the Assessment

Phase 1) QAI – Preparation Task

1. Management Team staffs Assessment Team
2. Process and Training Teams train Assessment Team
3. Assessment Team identifies:
   - System Requirements Team
   - System Architecture Team
4. Process and Training Teams train System Requirements and Architecture Teams
5. Assessment, Requirements, and Architecture Teams collaborate to Organize QAI Meeting (i.e., Attendees, Time, Location, Agenda)

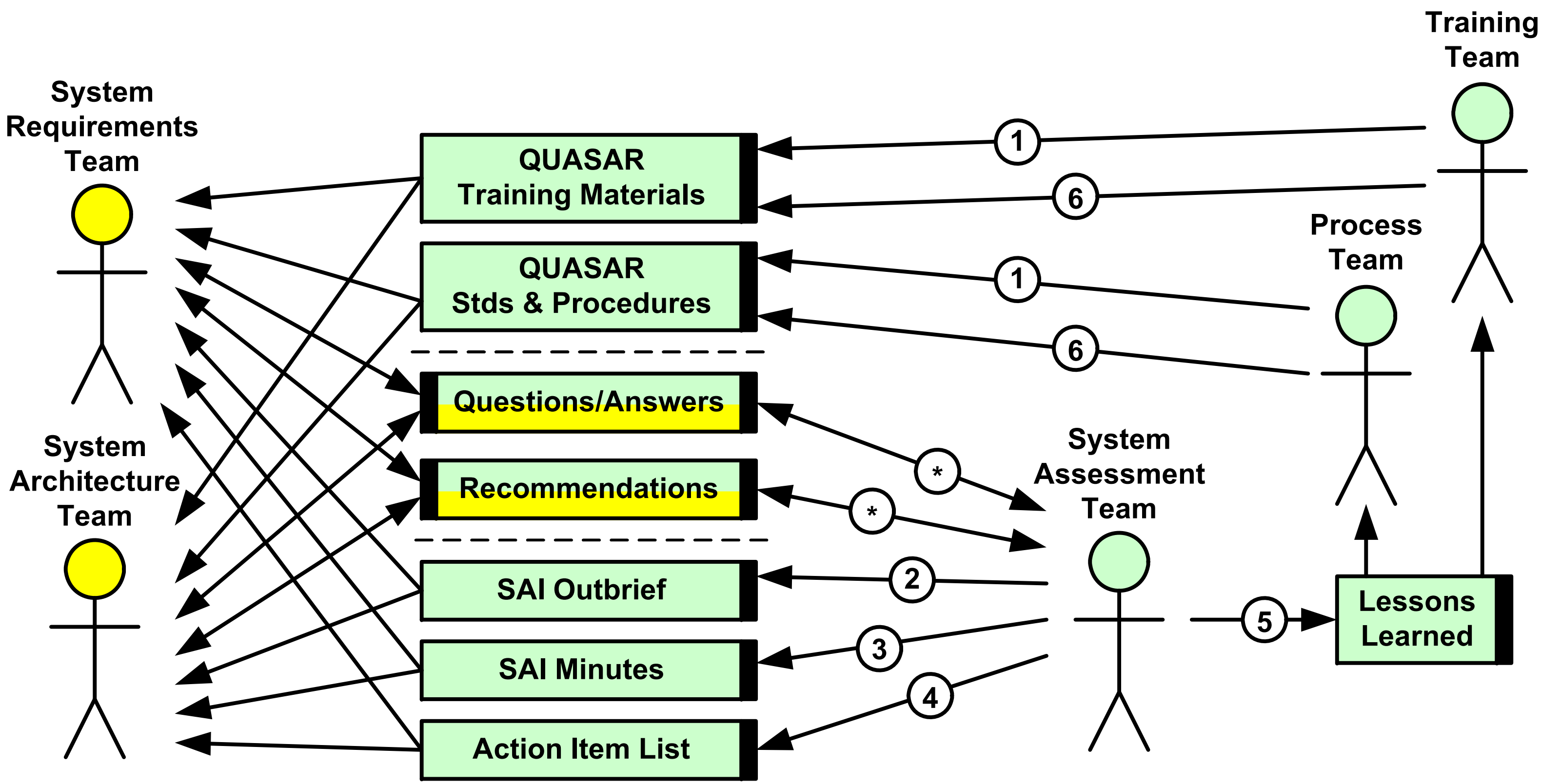
Phase 1) QAI – Meeting Task

1. Assessment, System Requirements, and System Architecture Teams Collaborate to determine Assessment Scope:
   - Subsystems/Architectural Elements/Focus Areas to Assess (Number and Identity)
   - Quality Factors and Quality Attributes underlying Assessment
   - Assessment Resources (e.g., Staffing, Schedule, and Budget)
2. Teams Collaborate to develop Initial Assessment Schedule with regard to System schedule, Subsystem schedule, and associated milestones
3. Teams Collaborate to tailor QUASAR Method
4. Assessment Team captures Action Items

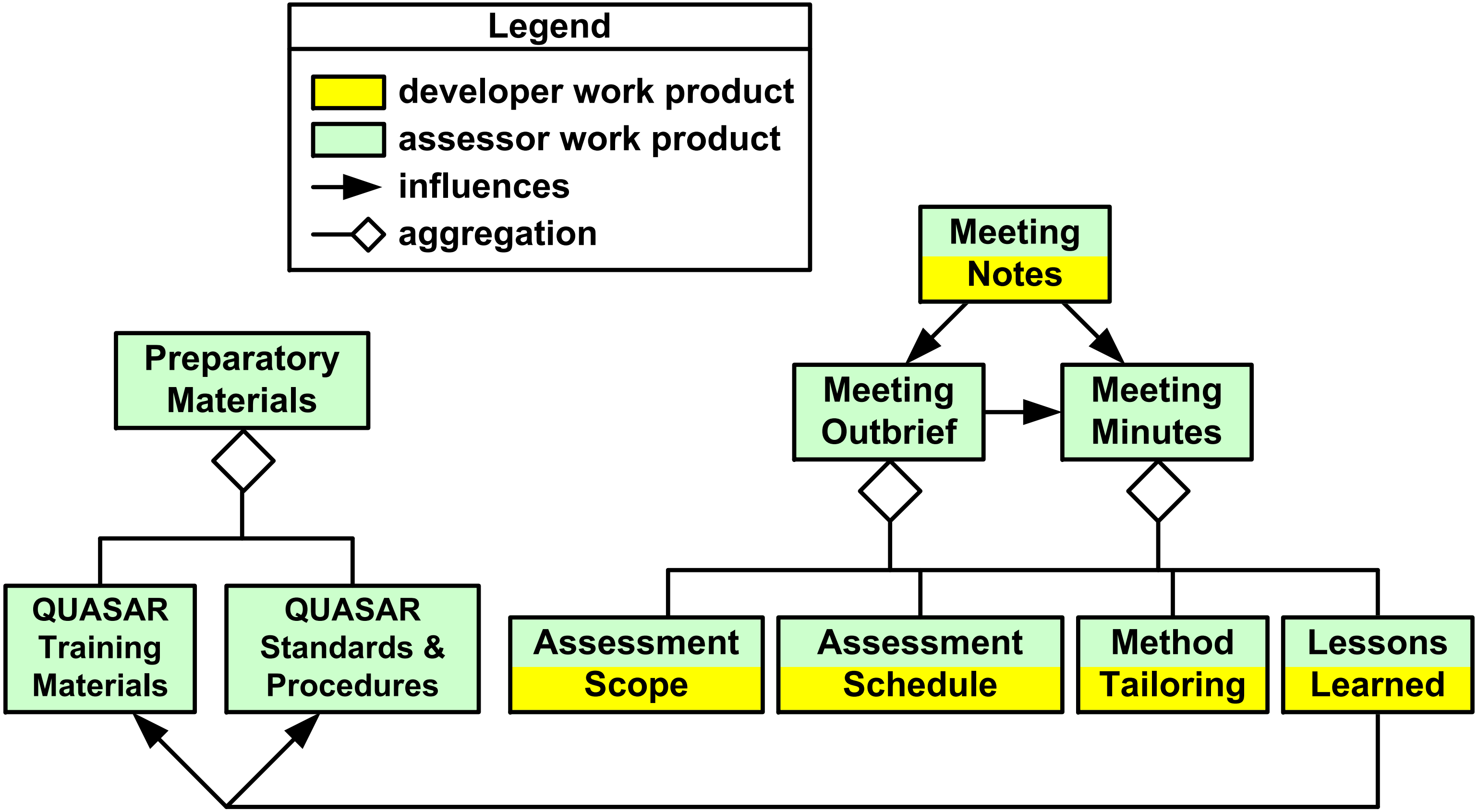
Phase 1) QAI – Follow-Through Task

1. Assessment Team develops and presents Meeting Outbrief
2. Assessment Team develops, reviews, and distributes Meeting Minutes
3. Assessment/Process/Training Teams tailor, internally review, and distribute:
   - QUASAR Procedure, Standards, and Templates
   - QUASAR Training Materials
4. Teams distribute Assessment Schedule
5. Teams obtain Needed Resources
6. Assessment Team Manages Action Items
7. Assessment Team captures Lessons Learned

Phase 1) QAI – Work Product Flow



Phase 1) QAI – Work Products



Phase 1) QAI – Team Memberships$_1$

Quality Assessment Team (Assessors):

- Assessment Team Leader
- Meeting Facilitator
- Acquirer/Customer Liaisons to Developer:
  - Requirements Teams
  - Architecture Teams
- Subject Matter Experts (SMEs) having adequate training and experience in:
  - Application Domains
    (e.g., avionics, sensors, telecommunications, and weapons)
  - Specialty Engineering Groups
    (e.g., reliability, safety, and security)
  - Requirements and Architecture Engineering (including Quality Model)
  - QUASAR
- Scribe
- Acquirer/Customer *Observers*

Phase 1) QAI – Team Memberships$_2$

System Requirements Team (Requirements Engineers):

- Chief System Requirements Engineer
- System Requirements Engineers
- Subsystem Requirements Engineers

System Architecture Team (Architects):

- Chief System Architect
- System Architects
- Subsystem Architects

Developer *Observers*

Phase 1) QAI – Lessons Learned$_1$

Ensure Appropriate Team Memberships (e.g., Authority)

Ensure Adequate Resources (e.g., Staffing, Budget, and Schedule)

Obtain Consensus on:

- Assessment Objectives and Scope
- Definitions (e.g., of Quality Factors, Subfactors, and Cases)

Provide Early Training:

- Method Training
  (QUASAR, Requirements Engineering, and Architecting)
- System/Subsystem Training
  (Requirements and Architecture)

Phase 1) QAI – Lessons Learned$_2$

QUASAR assessments should be Organized according to a Quality Model that defines Quality Characteristics (a.k.a., factors, “ilities’) and their Quality Attributes such as:

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

Topics

Requirements and Architecture Challenges

Underlying Concepts

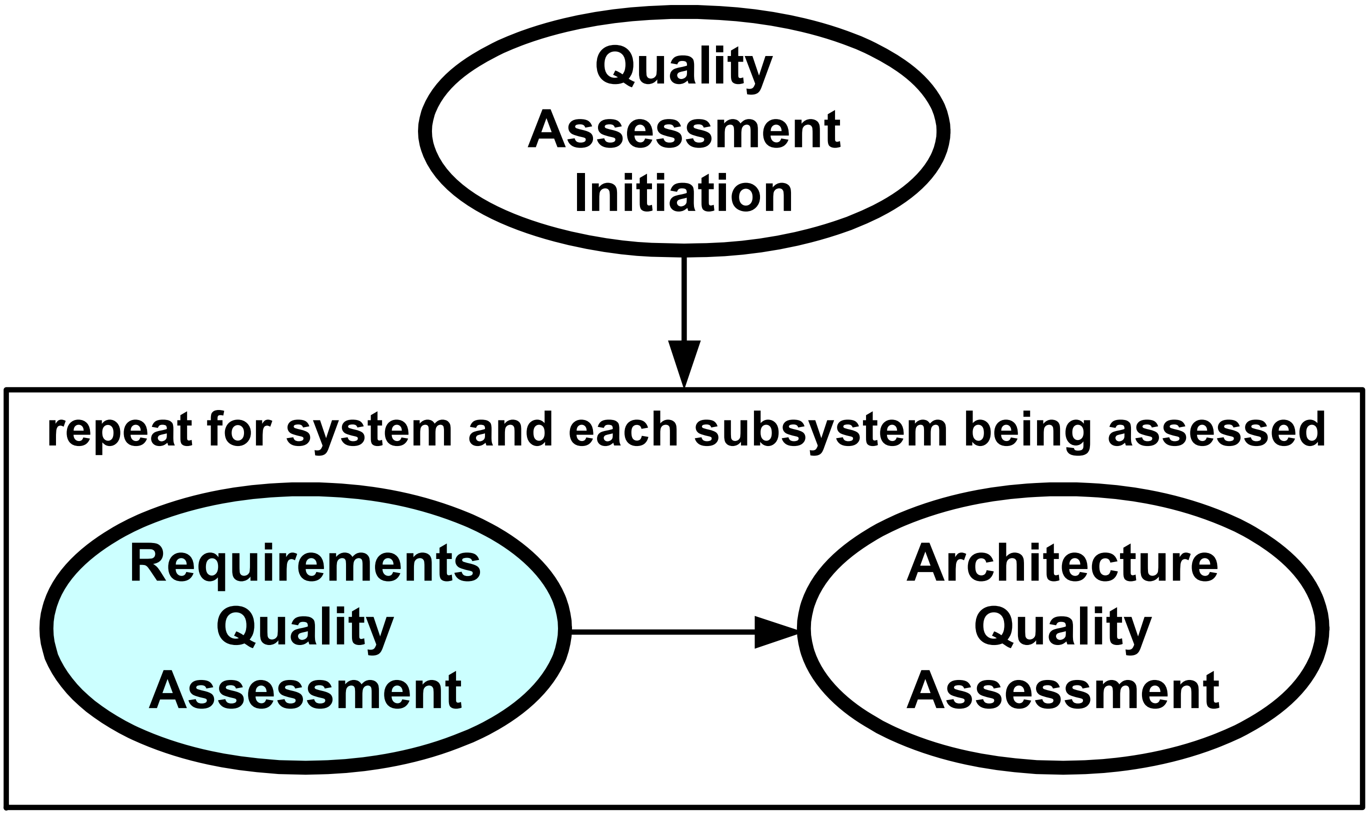
QUASAR Method:

- Phase 1) Quality Assessment Initiation (QAI)
- **Phase 2) Requirements Quality Assessment (RQA) ◄**
- Phase 3) Architecture Quality Assessment (AQA)

Reasons to use QUASAR

QUASAR – Today and Tomorrow

Requirements Quality Assessment (RQA)



Phase 2) ARA – Objectives$_1$

Use Requirements Quality Cases to:

- Independently assess Quality and Maturity of the Architecturally Significant Requirements:
  - Drive the Architecture
  - Form Foundation for Architecture Quality Assessment
- Help Requirements Engineers identify Requirements Defects and Weaknesses so that:
  - Defects and Weaknesses can be Corrected
  - The Architecture (and System) can be Improved

Phase 2) RQA – Objectives$_2$

Use Requirements Quality Cases to:

- Identify Requirements Risks so that they can be Managed
- Provide Visibility into the Status and Maturity of the Requirements
- Increase the Probability of Project Success

Ensure Architecture Team will be Prepared to Support the coming Architecture Quality Assessment.

Capture Lessons Learned.

Update QUASAR Method and associated Training Materials.

Phase 2) RQA – Principles$_1$

Not all Requirements are Architecturally Significant.

Quality-Related Requirements:

- Are typically Major Drivers of the System Architecture.
- Should be Unambiguous, Feasible, Complete, Consistent, Mandatory, Verifiable, Validatable, etc.
- Should *Not Un*necessarily Constrain the Architecture.

Quality Requirements should Specify a Minimum Required Amount of some Quality Factor or Quality Attribute.

Phase 2) RQA – Principles$_2$

Quality Requirements should be Organized according to a Quality Model that defines Quality Characteristics (a.k.a., factors, “ilities’) and their Quality Attributes such as:

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

Phase 2) RQA – Principles_3

Different Quality Factors are Important for Different Components:

- Performance is Paramount for Real-Time Components.
- Security is more Important for other Components.

Engineering Quality Requirements requires Significant Effort and Resources (it *cannot* be accomplished during a short meeting).

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 have Different and Inconsistent Quality Needs.
- Requirements Assessment Time is Too Late to be Engineering Quality Requirements.

Phase 2) RQA – Challenges$_1$

Many Requirements Engineers are not taught how to Engineer Non-functional Requirements including Quality Requirements.

Although popular, Use Case Modeling is not very Effective for Engineering *Quality* Requirements.

Quality Requirements often require the Input from Specialty Engineering Teams (e.g., Reliability, Safety, and Security), who are not often adequately involved during Requirements Engineering.

Quality Goals are often Mistakenly Specified as Quality Requirements.

Architecturally Significant Requirements are typically:

- Incomplete
  (missing important Relevant Quality Characteristics and Attributes)
- Of Poor Quality (lack important characteristics)

Phase 2) RQA – Challenges$_2$

The typical Quality of Derived and Allocated Architecturally Significant Requirements is 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 be available 24/7 every day of the year.”
  - “The system shall have 99.9999 reliability.”

Phase 2) RQA – 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 Requirements Engineers often do *Not* Understand how to Prepare for a Requirements Quality Assessment:

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

Phase 2) RQA – Challenges$_4$

Managers believe Schedule Pressures do *Not* allow Time for Requirements Quality Assessments.

Requirements Engineers may Not Understand how to give the Assessment Team what they need to assess the Requirements:

- Claims
- Arguments including Requirements Decisions, Inventions, Trade-Offs, Assumptions, and Rationales
- What is the proper Evidence?
  - Official program documentation
  - Not requirements engineering plans and procedures
  - Not hastily produced PowerPoint slides

Phase 2) RQA – Preparation Task

Process/Training Team trains the Requirements and Architecture Teams *significantly prior* to the RQA Meeting.

Requirements and Architecture Teams provide Preparatory Materials to the Quality Assessment Team *significantly prior* to the RQA Meeting:

- Summary Presentation Materials
- Requirements Quality Cases
  (including electronic access to evidentiary materials)
- Example of Planned Architectural Quality Case

Quality Assessment Team:

- Reads Preparatory Materials
- Generates RFIs and RFAs

Phase 2) RQA – Meeting Task

1. Requirements Team presents:
   - System Overview
   - Requirements Overview
   - *Requirements* Quality Cases
2. Quality Assessment Team assesses Quality and Maturity of Requirements:
   - Completeness of Quality Cases
   - Quality of Quality Cases
3. Architecture Team presents Example Architectural Quality Case
4. Quality Assessment Team recommends Improvements
5. Quality Assessment Team manages Action Items

Phase 2) RQA – Follow-Through Task

Quality Assessment Team:

1. Develops Consensus Regarding Requirements Quality
2. Produces, Reviews, and Presents Meeting *Outbrief*
3. Produces, Reviews, and Publishes RQA *Report*
4. Updates and publishes the System Quality Assessment Summary Matrix
5. Captures Lessons Learned
6. Manages Action Items

Requirements Team:

Addresses Risks Raised in RQA Report

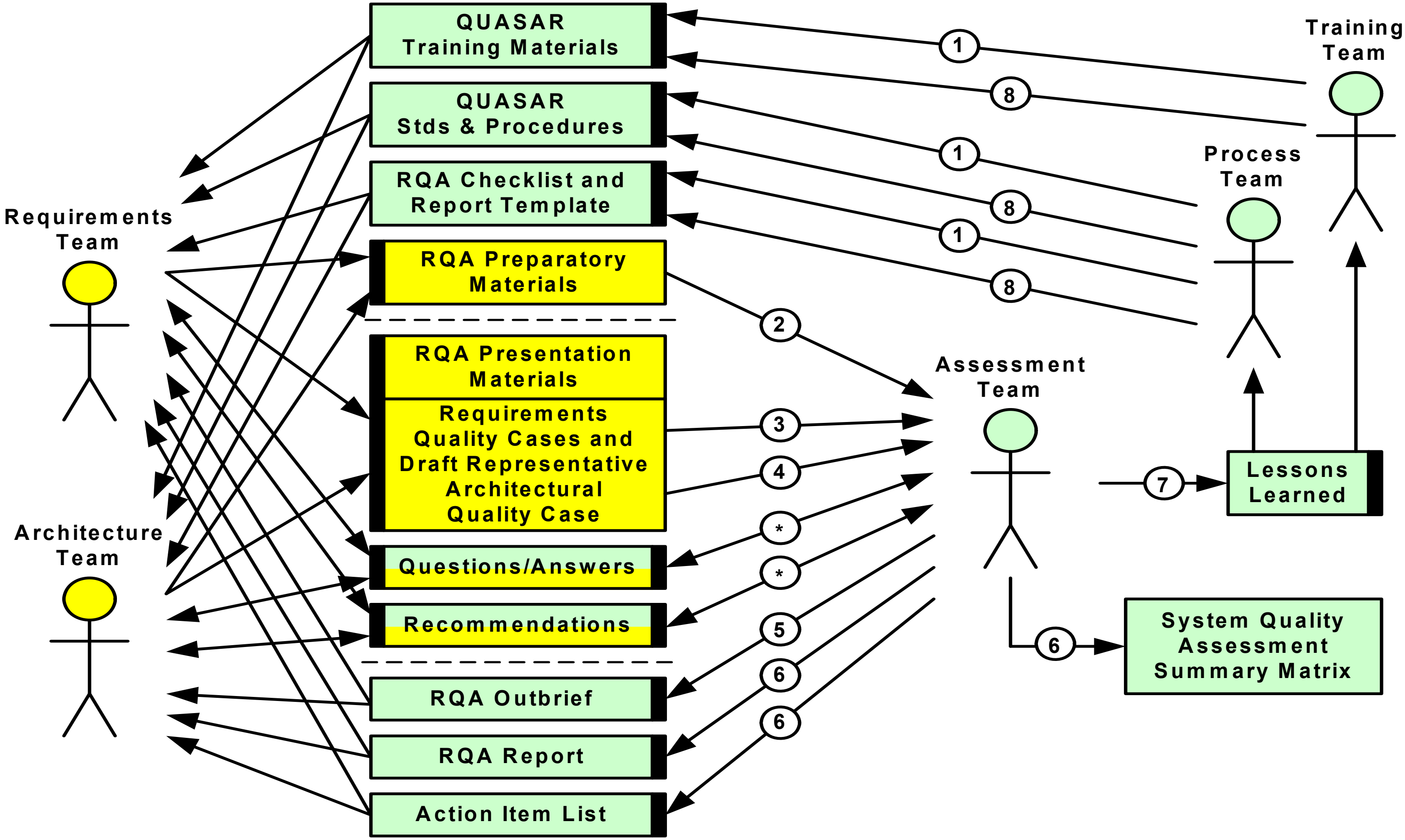
Process Team:

Updates Assessment Method (e.g., Standards and Procedures)

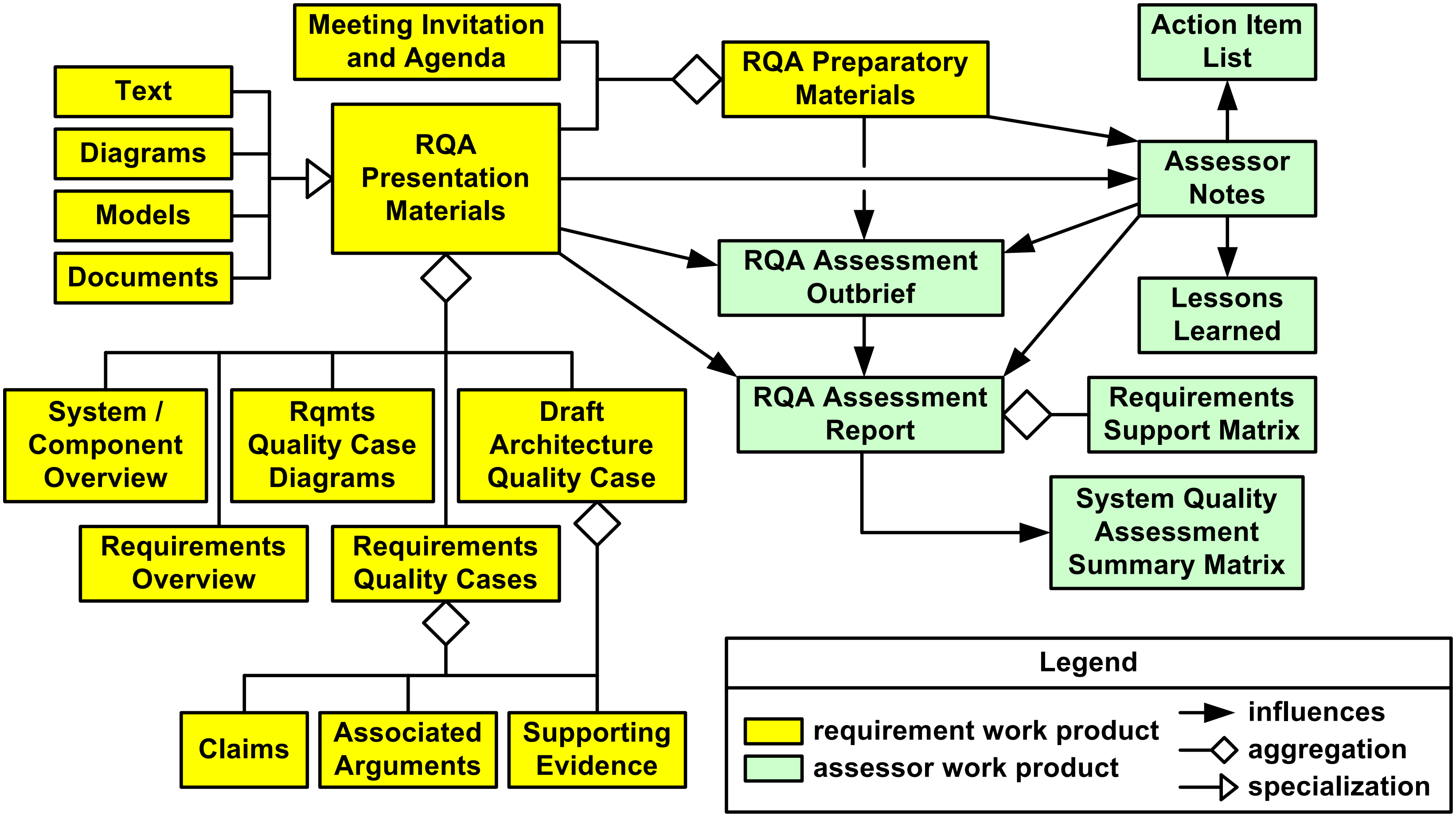
Training Team:

Updates Training Materials (if appropriate)

Phase 2) RQA – Work Product Workflow



Phase 2) RQA – Work Products



System Quality Assessment Summary Matrix

	SYS		AC 1		AC 2		AC 3		AC 4		AC 5			AC n	
	R	A	R	A	R	A	R	A	R	A	R	A		R	A
QF 1															
QF 2															
QF 3			NA	NA					NA						
QF 4															
QF 5															
QF 6									NA						
QF 7			NA	NA											
QF 8															
QF n															

Phase 2) RQA – Checklist$_1$

Are the Claims:

- Based on the project Quality Model?
- Appropriate for the Current Time in the Project Development Cycle?

Are the Arguments:

- Clear (understandable to the assessors)?
- Compelling (sufficient to justify belief in the claims)?
- Relevant (to justify belief in the claims)?

Is the Evidence:

- Credible (official requirements work products under configuration control)?
- Sufficient (to support the arguments)?

Phase 2) RQA – Checklist$_2$

Are the Architecturally Significant Requirements:

- Of sufficient Quality?
- Sufficient to drive Architecture Engineering?
- Sufficient on which to base the Architecture Quality Assessment?

Does the representative draft Architecture Quality Case show that the Architects clearly understand what they need to present at the Architecture Quality Assessment?

Phase 2) RQA –Team Memberships$_1$

Quality Assessment Team (Assessors):

- Assessment Team Leader
- Meeting Facilitator
- Acquirer/Customer Liaisons to Developer:
  - Requirements Teams
  - Architecture Teams
- Subject Matter Experts (SMEs) having adequate training and experience in:
  - Application Domains (e.g., avionics, sensors, telecommunications, and weapons)
  - Specialty Engineering Groups (e.g., reliability, safety, and security)
  - Requirements and Architecture Engineering (including Quality Model)
  - QUASAR
- Scribe
- Acquirer/Customer *Observers*

Phase 2) RQA – Team Memberships$_2$

Requirements Team:

- Requirements Engineers
- Subject Matter Experts (if appropriate):
  - Specialty Engineering Experts
  - Application Domain Experts

Architecture Team:

- Architects
- Subject Matter Experts (if appropriate):
  - Specialty Engineering Experts
  - Application Domain Experts

Phase 2) RQA – Lessons Learned

Select, Define, and Prioritize Quality Factors and Quality Attributes (e.g., as Critical, Important, Desirable, or Relevant).

Concentrate on Quality-related *Requirements*
(i.e., Merely Listing Quality Factors is Not Sufficient).

Architecturally-Significantly Quality Requirements must have certain Properties.

Iterative/Incremental Development implies Iterative/Incremental Requirements Assessments.

Hold Meeting Sufficiently Early for Quality Requirements to Drive the Architecture.

Topics

Requirements and Architecture Challenges

Underlying Concepts

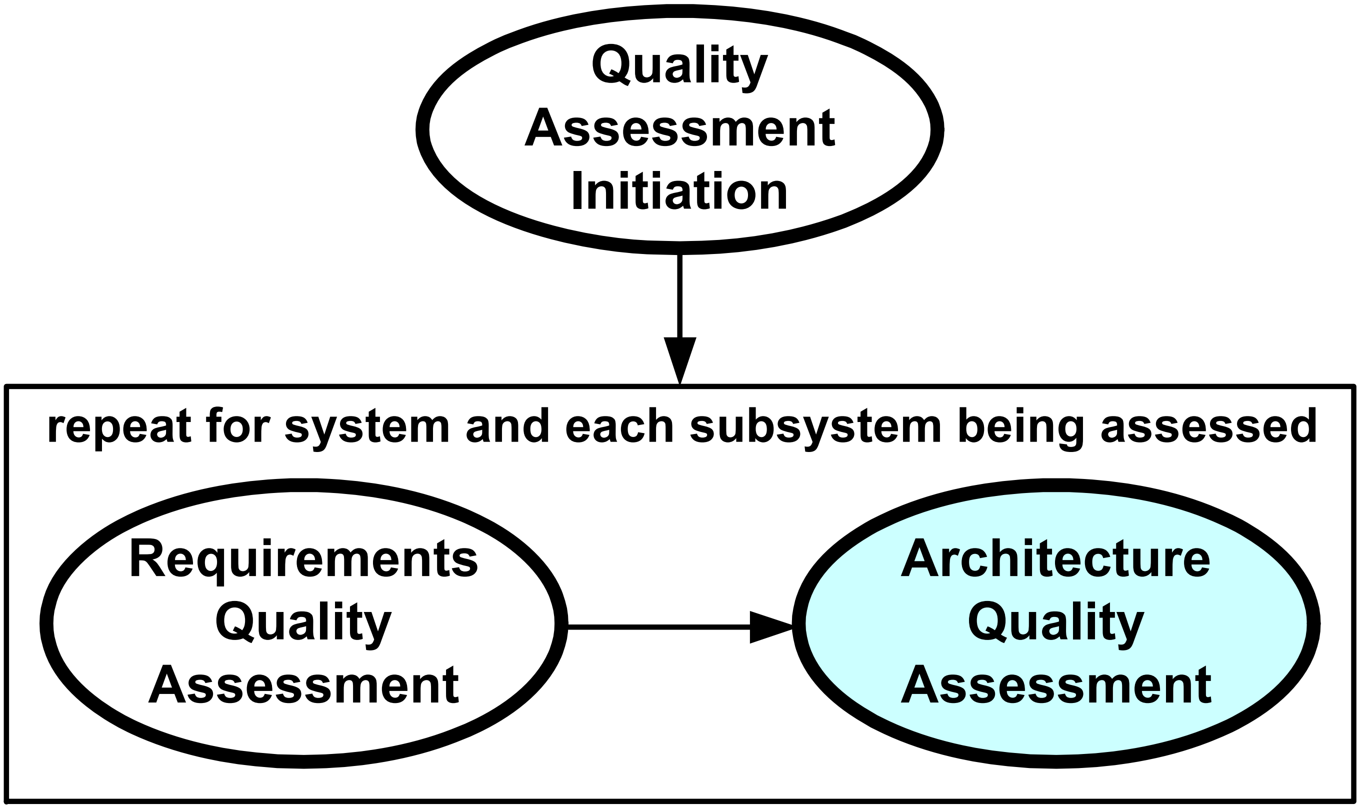
QUASAR Method:

- Phase 1) Quality Assessment Initiation (QAI)
- Phase 2) Requirements Quality Assessment (RQA)
- **Phase 3) Architecture Quality Assessment (AQA) ◄**

Reasons to use QUASAR

QUASAR – Today and Tomorrow

Architecture Quality Assessment (AQA)



Phase 3) AQA – Objectives

Use Architectural Quality Cases to:

- Independently assess Architecture Quality in terms of its Support for its Derived and Allocated Architecturally Significant Requirements
- Help Architects identify Architectural Defects and Weaknesses so that:
  - Defects and Weaknesses can be Corrected
  - The Architecture (and System) can be Improved
- Identify Architectural Risks so that they can be Managed
- Provide Visibility into the Status and Maturity of the Architecture
- Increase the Probability of Project Success

Phase 3) AQA – Principles

The Architects should know:

- The Quality Requirements *driving* the Development of the Architecture.
- What Architectural Decisions they *made* and why they made them.
- Where they *documented* their Architectural Decisions.

The Architects should already have documented this Information as a *Natural* Part of their Architecture Engineering Method.

Little *New* Documentation should be Necessary for the Architects to make their Cases to the Quality Assessment Team.

The Architects are Responsible for making their own Cases that their Architecture Sufficiently Supports its Derived and Allocated Quality Requirements.

Phase 3) AQA – Challenges$_1$

Architects may not have developed Quality Cases as a Natural Part of their Architecture Engineering Method:

- Architectural Documentation are 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 Quality Assessments may not be:

- Mandated by Contract or Development Method
- Scheduled and Funded

Phase 3) AQA – Challenges$_2$

Managers may feel that Schedule Pressures do *not* allow time for Architecture Quality Assessments.

Architects may *not* understand how to prepare for an Architecture Quality Assessment:

- Too Busy
- Not Trained
- No Standards Exist
- Bias against Assessments/Audits

Architecturally-Significant Requirements are Rarely Well Engineered.

Architectural Documentation often varies Widely in Quality and Completeness.

Phase 3) AQA – Challenges$_3$

Architecturally-Significant Requirements (esp. Quality Requirements) are *rarely traced* to the Architectural Elements that collaborate to Implement them.

It is difficult to determine if an Architecture *sufficiently* supports meeting Poorly-Specified Architecturally Significant Requirements.

Phase 3) AQA – Preparation Task

Architecture and Quality Assessment Teams organize the AQA Assessment Meeting.

Training Team provides (at appropriate time):

- QUASAR Training (if not provided prior to RQA assessment)
- AQA Assessment Checklist and Report Template

Architecture Team makes available (min. 2 weeks before meeting):

- Any Updated Quality Requirements
- Architecture Overview
- Quality Case Diagrams
- Architecture Quality Cases (Claims, Arguments, and Evidence)

Quality Assessment Team:

- Reads Preparatory Materials
- Generates RFIs and RFAs

Phase 3) AQA – Meeting Task

Architecture Team:

1. Introduces the Architecture
(e.g., Context and Major Functions)
2. Briefly reviews the Architecturally Significant Requirements
3. Briefly summarizes the Architecture
(e.g., Most Important Architectural Components, Relationships, Decisions, Inventions, Trade-Offs, Assumptions, and Rationales)
4. Individually Presents Architectural Quality Cases
(Quality Case Diagram, Claims, Arguments, and Evidence)

Quality Assessment Team:

1. Probes Architecture (Architectural Quality Case by Quality Case)
2. Manages Action Items

Phase 3) AQA – Follow-Through Task

Quality Assessment Team:

1. Develops Consensus regarding Architecture Quality
2. Produces, reviews, and presents Meeting Outbrief
3. Produces, reviews, and publishes AQA Report
4. Updates and republishes System Quality Assessment Summary Matrix
5. Captures Lessons Learned
6. Manages Action Items

Architecture Team:

Addresses Architectural Defects, Weaknesses, and Risks Raised in AQA Report

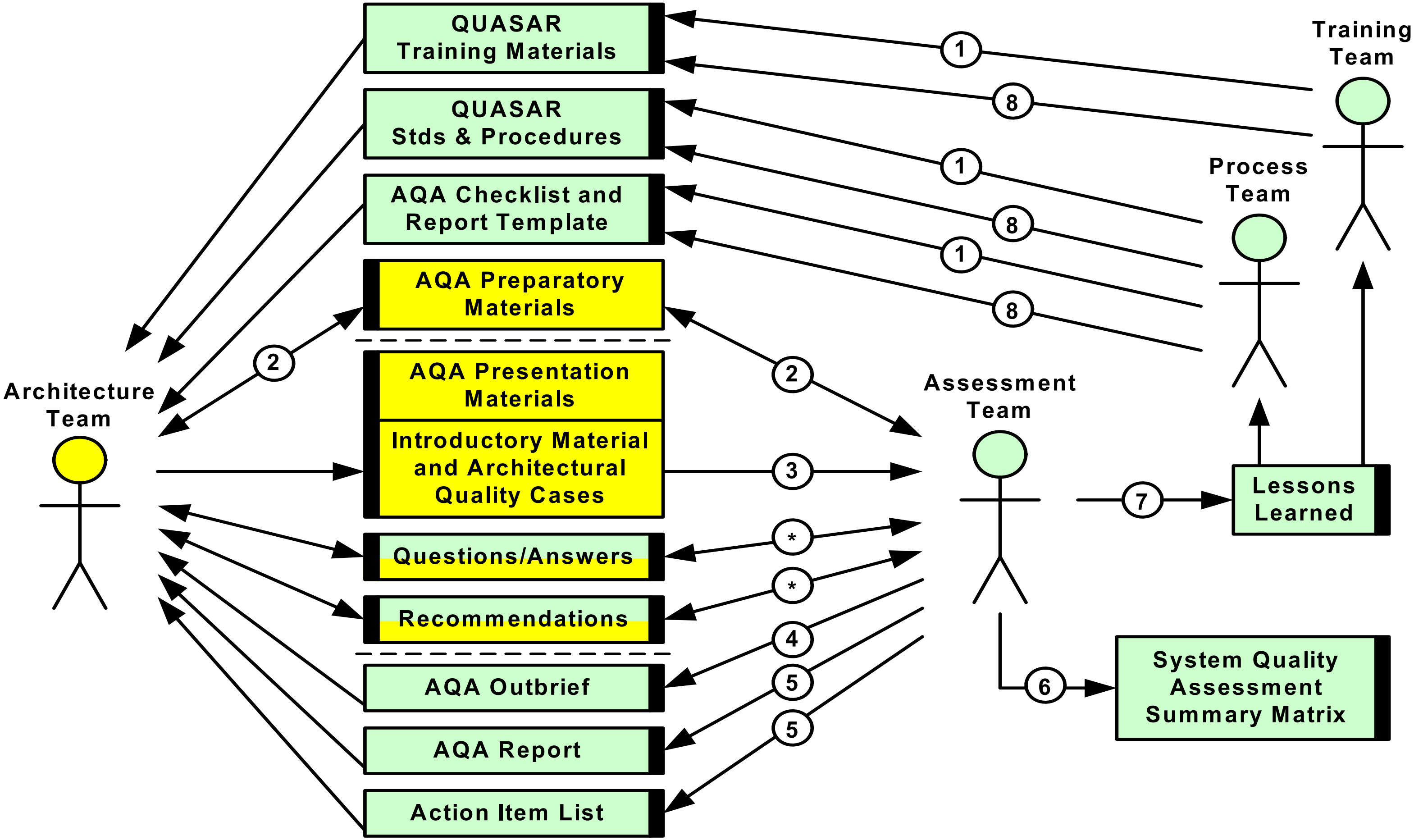
Process Team:

Updates Assessment Method (if appropriate)

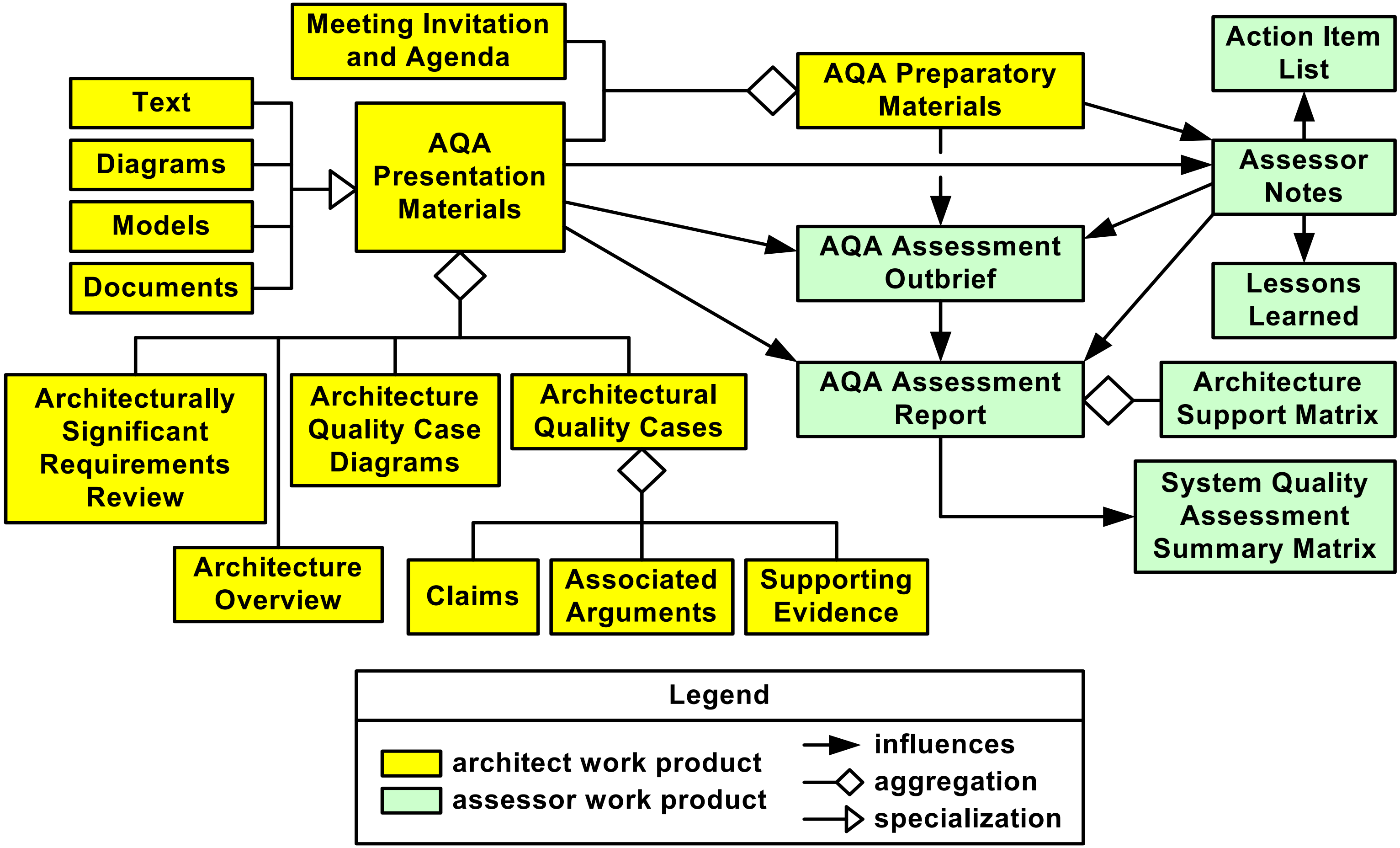
Training Team:

Updates Training Materials (if appropriate)

Phase 3) AQA – Work Product Workflow



Phase 3) AQA – Primary Work Products



Phase 3) AQA – Checklist$_1$

Are the Claims:

- Based on the project Quality Model?
- Appropriate for the Current Time in the Project Development Cycle?

Are the Arguments:

- Clear (understandable to the assessors)?
- Compelling (sufficient to justify belief in the claims)?
- Relevant (to justify belief in the claims)?

Is the Evidence:

- Credible (official architecture work products under configuration control)?
- Sufficient (to support the arguments)?

Phase 3) AQA – Team Memberships[1]

Quality Assessment Team (Assessors):

- Assessment Team Leader
- Meeting Facilitator
- Acquirer/Customer Liaisons to Developer:
  - Requirements Teams
  - Architecture Teams
- Subject Matter Experts (SMEs) having adequate training and experience in:
  - Application Domains
    (e.g., avionics, sensors, telecommunications, and weapons)
  - Specialty Engineering Groups
    (e.g., reliability, safety, and security)
  - Requirements and Architecture Engineering (including Quality Model)
  - QUASAR
- Scribe
- Acquirer/Customer *Observers*

Phase 3) AQA – Team Memberships$_2$

Architecture Team:

- Architects
- Subject Matter Experts (if appropriate):
  - Specialty Engineering Experts
  - Application Domain Experts
- Developer *Observers*

Phase 3) AQA – Lessons Learned$_1$

Iterative, Incremental, Parallel, and Time-boxed Development implies Iterative, Incremental, Parallel, and Time-boxed Architecture Assessments.

Provide Initial Overview of the Architecture:

- Keep Overview Short
- Present Only the Most Important Architectural Decisions, Trade-Offs between Quality Characteristics and Attributes, and Assumptions
- Mount Diagrams on Meeting Room Walls (and Leave Them Up!)
- Highlight Primary Architectural Decisions and Inventions

Focus on Assessing the Existing Architecture.

Avoid a “Trust Me” Mentality.

Phase 3) AQA – Lessons Learned$_2$

Organize Presentation by Quality Cases (i.e., Quality Characteristics and Quality Attributes) and not by Architectural Component.

Architects should present Models of relevant Logical and Physical as well as Static and Dynamic Architectural Structures.

Keep Evidence Presented and Requested within Assessment Scope.

Ensure Availability of Actual Architects.

Architects must have Electronic Access to Evidence to present Existing, Official Documentary Evidence.

Use Scenarios to Probe and Test the Architecture rather than to Introduce the Architecture.

Phase 2) AQA – Lessons Learned$_3$

Take Development Cycle, Project Schedule, and Architectural Maturity into Account.

Emphasize Assessment Results over recommending Architectural Improvements.

Ensure Reasonable Assessment Size and Schedule.

Ensure Adequate Pre-Meeting Preparation.

All Architectural Tiers are not Equal:

- Size, Complexity, Criticality, and Quality Factors/Subfactors
- Apply Different Emphasis at Different Levels of the Hierarchy.

Differentiate Architecture from Design.

Topics

Requirements and Architecture Challenges

Underlying Concepts

QUASAR Method

Reasons to use QUASAR◄

QUASAR – Today and Tomorrow

QUASAR Benefits$_1$

QUASAR ensures Specification of *Architecturally-Significant* Requirements.

QUASAR provides Acquirer Visibility into (and supports oversight of) the Quality of the Requirements and Architecture

QUASAR supports Certification and Accreditation

QUASAR emphasizes using a common project-specific Quality Model:

- Which drives the Quality Requirements
- Which drives the Quality of the System Architecture
- Which drives the Quality of the System

QUASAR Benefits$_2$

QUASAR Supports Process Improvement:

- Solves Major Requirements and Architecture Problems

QUASAR Provides needed Flexibility:

- Any Effective Requirements Engineering and Architecting Methods
- Uses Existing Requirements and Architecture Work Products (i.e., almost no new work products required)
- Any Subsystems based in Need and Risk (i.e., fits any system size, budget, schedule, and tier)
- Any Quality Factors and Quality Attributes

QUASAR Helps:

- *Requirements Engineers* Succeed
- *Architects* Succeed
- *Program* Succeed

Topics

Requirements and Architecture Challenges

Underlying Concepts

QUASAR Method

Reasons to use QUASAR

QUASAR – Today and Tomorrow ◄

QUASAR Today

Used on Largest DoD Acquisition Program ($270,000,000,000 USD)

QUASAR Version 1 Handbook Published
http://www.sei.cmu.edu/publications/documents/06.reports/06hb001.html

Provided as SEI Service by Acquisition Support Program (ASP)

Tutorials at Conferences

Articles in Journals

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 (Version 1)
- Enable Readers to start using QUASAR

Description:

- *Very* Complete
- Too Comprehensive to be Good First Introduction

QUASAR Tomorrow – Technical Plans

Quality Factors across Multiple Subsystems:

- Multiple Cross-Cutting Structures and Models
- Multiple Subsystems Collaborate to Achieve Quality Requirements

Development of Catalog of Quality Factor-Specific Architectural Styles, Patterns, and Mechanisms to use as Standardized Quality Case Arguments

Improve Objective Determination of “Sufficient Quality”

Expand Quality Cases Beyond Requirements and Architecture

QUASAR Tomorrow - Productization

More Conference Tutorials and Classes

Expanded QUASAR Training Materials

More QUASAR Articles

Use and Validation on more Projects

QUASAR Book

How the SEI Can Help You

QUASAR is Ready for Use *Now*.

QUASAR Handbook and Training Materials can be downloaded from SEI Website.

The SEI Acquisition Support Program (ASP) offers QUASAR as a Service:

- Consulting and Training
- Facilitation of QUASAR Assessments
- Recommended RFP and Contract Language

Questions?

For more information, contact:

Donald Firesmith

Acquisition Support Program

Software Engineering Institute

dgf@sei.cmu.edu

Software Engineering Institute | Carnegie Mellon