Software Architecture Documentation

Supporting Evaluation

A Comprehensive Guide to Preparing Architecture Documentation
for ATAM, CBAM, and Other Evaluation Methods

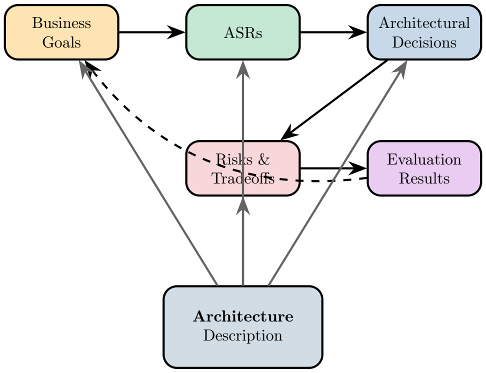
Architecture Documentation Series

Based on SEI Evaluation Methods and Industry Best Practices

December 9, 2025

Abstract

Architecture evaluation is a critical practice for assessing whether a software architecture will meet its intended quality goals before significant development investment. This comprehensive guide examines how architecture documentation supports formal evaluation methods such as ATAM (Architecture Tradeoff Analysis Method), CBAM (Cost Benefit Analysis Method), and ARID (Active Reviews for Intermediate Designs). The document provides detailed guidance on documenting business goals, architecturally significant requirements, quality attribute scenarios, architectural decisions, and risk/sensitivity analysis. Whether preparing for a formal evaluation or conducting internal architecture reviews, this guide ensures your documentation provides the analysis artifacts evaluators need.



Evaluation Traceability Flow

Contents

1 Introduction 3
1.1 Purpose of This Guide . . . 3
1.2 What is Architecture Evaluation? . . . 3
1.3 Evaluation Methods Overview . . . 3
1.4 The Evaluation Traceability Chain . . . 4

2 Business Goals and Drivers 4
2.1 Importance of Business Goals . . . 4
2.2 Business Goal Categories . . . 4
2.3 Documenting Business Goals . . . 5
2.4 Example Business Goals . . . 5

3 Architecturally Significant Requirements 6
3.1 What Makes a Requirement Architecturally Significant? . . . 6
3.2 ASR Sources . . . 6
3.3 ASR Documentation Template . . . 7
3.4 ASR Prioritization . . . 7

4 Quality Attribute Scenarios 8
4.1 What is a Quality Attribute Scenario? . . . 8
4.2 Scenario Structure . . . 8
4.3 Scenario Examples by Quality Attribute . . . 9
4.4 Scenario Catalog Template . . . 9

5 Architectural Decisions 10
5.1 Decision Documentation for Evaluation . . . 10
5.2 Example Decision Documentation . . . 12
5.3 Decision Traceability Matrix . . . 12

6 Risk and Sensitivity Analysis 12
6.1 ATAM Analysis Outputs . . . 12
6.2 Risk Documentation . . . 13
6.3 Example Risks . . . 13
6.4 Sensitivity Points . . . 14
6.5 Tradeoff Points . . . 14

7 Views for Evaluation 15
7.1 View-to-ASR Mapping . . . 15
7.2 View Completeness for Evaluation . . . 16

8 Review Question Sets 16
8.1 Questions for Business Managers . . . 17
8.2 Questions for Architects . . . 18
8.3 Questions for Evaluation Team . . . 19

9 Evaluation Readiness Assessment **19**
9.1 Readiness Checklist 20
9.2 Readiness Assessment Matrix 20

10 Appendix A: ATAM Overview **21**
10.1 ATAM Phases 21
10.2 ATAM Outputs 22

11 Appendix B: Quality Attribute Scenario Template **22**

12 Appendix C: Glossary **22**

13 Appendix D: References **23**

1 Introduction

1.1 Purpose of This Guide

This guide examines how architecture documentation supports formal and informal architecture evaluation. Effective evaluation support ensures that:

- Business goals and their architectural implications are clearly documented
- Architecturally significant requirements are identified, prioritized, and testable
- Architectural decisions and their rationale are captured
- Risks, sensitivities, and tradeoffs are identified and analyzed
- Evaluators can navigate from business goals through technical decisions to implications
- Evaluation results inform architectural refinement and business decisions

1.2 What is Architecture Evaluation?

Definition

Architecture Evaluation: A systematic examination of a software architecture to determine whether it will satisfy its quality attribute requirements, to identify risks and potential problems, and to assess tradeoffs among competing quality attributes.

Architecture evaluation answers critical questions:

- Will this architecture meet its quality goals?
- What are the risks in this architectural approach?
- What tradeoffs have been made and are they appropriate?
- Where are the sensitive points that require careful attention?
- Is the architecture suitable for the business context?

1.3 Evaluation Methods Overview

Table 1: Architecture Evaluation Methods

Method	**Purpose**	**Key Characteristics**
ATAM	Identify risks, sensitivities, tradeoffs	Scenario-based; stakeholder-driven; quality attribute focus
CBAM	Cost-benefit analysis of architectural decisions	Economic analysis; ROI focus; builds on ATAM
ARID	Evaluate intermediate designs	Lightweight; design-focused; early feedback
SAAM	Analyze modifiability	Scenario-based; change-focused; predecessor to ATAM
ALMA	Analyze modifiability	Detailed change analysis; maintenance focus

Lightweight Methods	Quick risk identification	Reduced formality; faster execution; focused scope

1.4 The Evaluation Traceability Chain

Effective evaluation requires clear traceability from business goals to technical implementation:

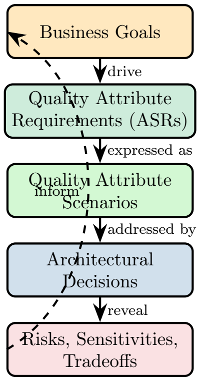

Business Goals
drive
Quality Attribute Requirements (ASRs)
expressed as
Quality Attribute Scenarios
inform
addressed by
Architectural Decisions
reveal
Risks, Sensitivities, Tradeoffs

Figure 1: Evaluation Traceability Chain

2 Business Goals and Drivers

2.1 Importance of Business Goals

Architecture exists to serve business purposes. Evaluation must assess whether the architecture supports these purposes effectively.

Key Point

Business Goals Drive Architecture:
Every architectural decision should trace back to one or more business goals. If a decision cannot be justified by business value, its necessity should be questioned.

2.2 Business Goal Categories

Table 2: Business Goal Categories

Category	**Description**	**Examples**
Revenue/Growth	Generate or increase revenue	New market entry; increased sales; premium features
Cost Reduction	Reduce operational or development costs	Automation; efficiency; reduced maintenance
Time to Market	Deliver capabilities faster	Rapid deployment; competitive response
Quality/Reliability	Ensure system dependability	High availability; data integrity; trust

Category	Description	Examples
Compliance	Meet regulatory requirements	GDPR; HIPAA; PCI-DSS; SOX
Competitive Advantage	Differentiate from competitors	Unique features; better performance; innovation
Risk Mitigation	Reduce business risks	Disaster recovery; security; vendor independence
Customer Satisfaction	Improve user experience	Usability; responsiveness; features
Operational Excellence	Improve operations	Scalability; maintainability; observability
Strategic Alignment	Support corporate strategy	Platform standardization; acquisition support

2.3 Documenting Business Goals

Business Goal Template

Goal ID: [BG-XXX]
Goal Statement: [Clear, measurable statement of the business objective]
Category: [Revenue / Cost / Time / Quality / Compliance / etc.]
Priority: [Critical / High / Medium / Low]
Stakeholder: [Who owns or cares about this goal]
Success Criteria: [How we know the goal is achieved]
Timeframe: [When must this goal be achieved]
Derived Quality Attributes:
natters for this goal
natters for this goal
Related ASRs: [References to specific ASRs derived from this goal]
Constraints: [Business constraints affecting architectural choices]

2.4 Example Business Goals

Table 3: Example Business Goals

ID	Goal Statement	Priority	Derived Quality Attributes
BG-001	Support 100K concurrent users during peak events	H Critical	Performance, Scalability, Availability
BG-002	Enable new feature deployment in under 4 hours	H Critical	Deployability, Modifiability, Testability
BG-003	Achieve PCI-DSS Level 1 compliance	H Critical	Security, Auditability

BG-004	Reduce infrastructure costs by 30%	M High	Resource Efficiency, Elasticity
BG-005	Enter European market within 12 months	M High	Portability, Internationalization
BG-006	Integrate with 3 major ERP systems	M High	Interoperability, Integrability

3 Architecturally Significant Requirements

3.1 What Makes a Requirement Architecturally Significant?

Definition

An **Architecturally Significant Requirement (ASR)** is a requirement that has a profound effect on the architecture—that is, the architecture might be dramatically different in the absence of such a requirement.

Requirements are architecturally significant when they:

- Have a measurable effect on system structure
- Constrain design decisions significantly
- Affect multiple components or layers
- Are difficult or expensive to change later
- Require specific architectural tactics or patterns
- Have high business value and high technical risk

3.2 ASR Sources

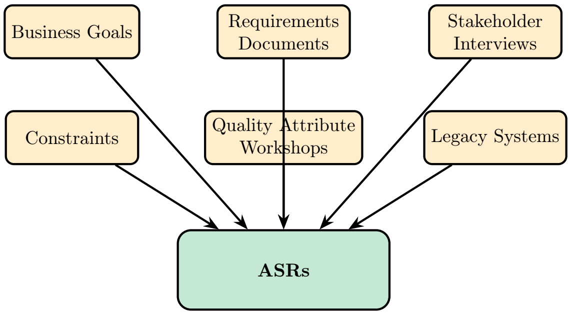


Figure 2: Sources of Architecturally Significant Requirements

3.3 ASR Documentation Template

ASR Documentation Template

ASR ID: [ASR-XXX]

Title: [Brief descriptive title]

Quality Attribute: [Performance / Availability / Security / Modifiability / etc.]

Source: [Business goal, stakeholder, requirement document, etc.]

Stimulus: [What triggers this requirement]

Environment: [Context in which the stimulus occurs]

Response: [How the system should respond]

Response Measure: [Quantifiable measure of success]

Priority: [Critical / High / Medium / Low]

Architectural Impact:

- Affected components: [List of impacted elements]
- Required tactics: [Architectural tactics needed]
- Tradeoffs: [Other qualities that may be affected]

Verification Method: [How this ASR will be verified]

Related Business Goals: [BG-XXX, BG-YYY]

Related Decisions: [ADR-XXX, ADR-YYY]

3.4 ASR Prioritization

Table 4: ASR Prioritization Matrix

ASR	Business Value	Architectural Impact	Priority Rationale
ASR-001	High	High	Critical: Core business requirement with major structural impact
ASR-002	High	Medium	High: Important business need, moderate architectural changes
ASR-003	Medium	High	High: Significant architectural impact requiring early attention
ASR-004	Medium	Medium	Medium: Balance of value and impact
ASR-005	Low	High	Review: High impact but lower value—question necessity
ASR-006	High	Low	Medium: Important but architecturally accommodated

4 Quality Attribute Scenarios

4.1 What is a Quality Attribute Scenario?

Definition

A **Quality Attribute Scenario** is a specific, testable statement of a quality attribute requirement. It consists of six parts: source, stimulus, artifact, environment, response, and response measure.

Quality attribute scenarios make abstract quality requirements concrete and testable.

4.2 Scenario Structure

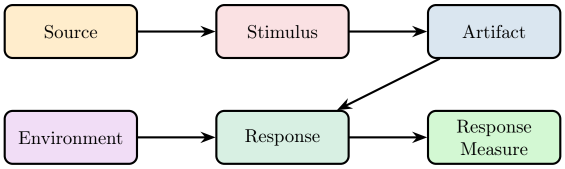


Figure 3: Quality Attribute Scenario Structure

Table 5: Scenario Parts Explained

Part	Description	Example
Source	Origin of the stimulus	User, external system, internal process, attacker
Stimulus	Event that triggers the scenario	Request, failure, attack, change request
Artifact	System or component affected	Whole system, specific service, database
Environment	Context when stimulus occurs	Normal operation, peak load, degraded mode
Response	How system handles stimulus	Process request, recover, resist attack
Response Measure	Quantifiable outcome	Latency, recovery time, successful resistance

4.3 Scenario Examples by Quality Attribute

Performance Scenario

Scenario ID: QAS-PERF-001
Source: External user
Stimulus: Initiates a product search request
Artifact: Search Service
Environment: Normal operation with 10,000 concurrent users
Response: Search results returned to user
Response Measure: Within 200ms at 95th percentile

Availability Scenario

Scenario ID: QAS-AVAIL-001
Source: Internal infrastructure
Stimulus: Primary database server fails
Artifact: Order Processing System
Environment: Normal business hours
Response: System fails over to replica and continues processing
Response Measure: No more than 30 seconds of service interruption; no data loss

Security Scenario

Scenario ID: QAS-SEC-001
Source: External attacker
Stimulus: Attempts SQL injection attack on login form
Artifact: Authentication Service
Environment: Normal operation
Response: Attack detected, blocked, and logged; user session not compromised
Response Measure: 100% of injection attempts blocked; alert generated within 1 second

Modifiability Scenario

Scenario ID: QAS-MOD-001
Source: Product team
Stimulus: Request to add new payment provider integration
Artifact: Payment Service
Environment: Design time
Response: New payment adapter implemented and deployed
Response Measure: Completed by one developer in 3 days; no changes to existing code

4.4 Scenario Catalog Template

Table 6: Quality Attribute Scenario Catalog

ID	**Quality**	**Stimulus**	**Response Measure**	**Priority**
QAS-001	Performance	User search request	<200ms p95	H
QAS-002	Performance	Checkout submission	<1s p99	H
QAS-003	Availability	Database failure	<30s failover	H
QAS-004	Availability	Service crash	Auto-restart <10s	H
QAS-005	Security	SQL injection	100% blocked	H
QAS-006	Security	Brute force login	Lockout after 5 attempts	M
QAS-007	Modifiability	New payment provider	3 person-days	M
QAS-008	Modifiability	UI framework change	2 person-weeks	L
QAS-009	Scalability	10x load increase	Linear scaling	M
QAS-010	Testability	Unit test execution	<5 minutes	M

5 Architectural Decisions

5.1 Decision Documentation for Evaluation

Evaluators need to understand what decisions were made, why they were made, and what alternatives were considered.

Architecture Decision Record (ADR) for Evaluation

Decision ID: [ADR-XXX]
Title: [Short descriptive title]
Status: [Proposed / Accepted / Deprecated / Superseded]
Context: [What is the issue or situation requiring a decision?]
Decision Drivers:

drives this decision
ffecting the decision

Considered Alternatives:
Option 1: [Name]

- Description: [Brief description]
- Pros: [Advantages]
- Cons: [Disadvantages]

Option 2: [Name]

- Description: [Brief description]
- Pros: [Advantages]
- Cons: [Disadvantages]

Decision: [What was decided]
Rationale: [Why this option was chosen]
Consequences:

- Positive: [Benefits of the decision]
- Negative: [Drawbacks or risks introduced]
- Neutral: [Other effects]

Related ASRs: [ASR-XXX, ASR-YYY]
Related Decisions: [ADR-XXX depends on / enables ADR-YYY]

5.2 Example Decision Documentation

Example

ADR-003: Use Event-Driven Architecture for Inter-Service Communication
Context: Services need to communicate state changes. We need loose coupling for independent deployment but also reliable message delivery.
Decision Drivers:

- BG-002: Enable rapid feature deployment
- ASR-003: Services must be independently deployable
- ASR-007: Support audit trail of all state changes

Alternatives Considered:

1. **Synchronous REST:** Simple but creates tight coupling
2. **Event-Driven (Kafka):** Loose coupling, audit trail, complexity
3. **Hybrid:** REST for queries, events for commands

Decision: Event-driven architecture using Apache Kafka for all inter-service state changes.
Rationale: Provides loose coupling, natural audit trail, supports replay for debugging, and enables independent scaling. Complexity is acceptable given team experience.
Consequences:

- (+) Services can be deployed independently
- (+) Built-in audit trail through event log
- (−) Eventual consistency requires careful handling
- (−) Increased infrastructure complexity

5.3 Decision Traceability Matrix

Table 7: Decision Traceability Matrix

Decision	Business Goals	ASRs Addressed	Scenarios Supported
ADR-001	BG-001, BG-004	ASR-001, ASR-009	QAS-001, QAS-009
ADR-002	BG-003	ASR-005, ASR-006	QAS-005, QAS-006
ADR-003	BG-002, BG-003	ASR-003, ASR-007	QAS-007
ADR-004	BG-001, BG-005	ASR-001, ASR-002	QAS-001, QAS-002, QAS-003
ADR-005	BG-004	ASR-008, ASR-009	QAS-009

6 Risk and Sensitivity Analysis

6.1 ATAM Analysis Outputs

The Architecture Tradeoff Analysis Method (ATAM) produces specific analysis outputs:

Table 8: ATAM Analysis Outputs

Output Type	Symbol	Description
Risk	★	Architectural decision that may cause problems if wrong
Non-Risk	–	Decision that is well-understood and acceptable
Sensitivity Point	✪	Property where small changes have large effects on quality
Tradeoff Point	☆	Property affecting multiple quality attributes in different ways

6.2 Risk Documentation

Risk Documentation Template

Risk ID: [RISK-XXX]
Title: [Short descriptive title]
Category: [Technical / Business / Schedule / Resource]
Description: [What could go wrong]
Source: [Related decision, ASR, or scenario]
Probability: [High / Medium / Low]
Impact: [High / Medium / Low]
Priority: [Probability × Impact]
Affected Quality Attributes: [List of affected qualities]
Affected Components: [List of affected architectural elements]
Mitigation Strategy:
ion 1 to reduce risk
ion 2 to reduce risk
Contingency Plan: [What to do if the risk materializes]
Owner: [Who is responsible for monitoring/mitigation]
Status: [Open / Mitigated / Accepted / Closed]

6.3 Example Risks

Table 9: Risk Register Example

ID	Risk Description	Prob.	Impact	Mitigation
RISK-001	Event-driven system may have consistency issues under high load	Med	High	Implement idempotent handlers; add reconciliation jobs
RISK-002	Single cloud provider dependency creates vendor lock-in	Low	High	Abstract cloud services; document migration path

RISK-003	Microservice complexity may slow initial development	High	Med	Start with fewer services; split as needed
RISK-004	Performance targets may not be achievable with chosen database	Med	High	Conduct early performance testing; have backup plan
RISK-005	Team lacks experience with chosen event streaming platform	Med	Med	Training; hire consultant; start with simple patterns

6.4 Sensitivity Points

Table 10: Sensitivity Points

ID	Sensitivity Point	Affected Quality	Implications
SP-001	Database connection pool size	Performance	Too small: timeouts; Too large: resource exhaustion
SP-002	Cache TTL settings	Performance, Consistency	Short: more DB load; Long: stale data
SP-003	Circuit breaker thresholds	Availability, Performance	Sensitive: false positives; Lenient: cascading failures
SP-004	Event partition count	Scalability, Performance	Too few: bottleneck; Too many: overhead
SP-005	Retry attempt limits	Availability, Resource usage	Few: premature failure; Many: resource exhaustion

6.5 Tradeoff Points

Table 11: Tradeoff Points

ID	Tradeoff Point	Qualities Affected	Tradeoff Description
TP-001	Synchronous vs. async communication	Performance ↔ Consistency	Async improves performance but introduces eventual consistency
TP-002	Caching strategy	Performance ↔ Freshness	More caching improves performance but data may be stale

TP-003	Service granularity	Modifiability ↔ Performance	Fine-grained services are more modifiable but have more overhead
TP-004	Encryption level	Security ↔ Performance	Stronger encryption improves security but impacts performance
TP-005	Logging verbosity	Debuggability ↔ Performance	More logging helps debugging but impacts performance

7 Views for Evaluation

7.1 View-to-ASR Mapping

Evaluators need to know which views address which quality attributes:

Table 12: View-to-Quality Attribute Mapping

View Type	**Primary Quality Attributes**	**Evaluation Use**
Module Decomposition	Modifiability, Testability, Reusability	Change impact analysis; team allocation
Uses/Dependencies	Modifiability, Buildability	Build dependencies; change propagation
Layered	Portability, Modifiability	Abstraction analysis; dependency rules
Component-Connector	Performance, Availability, Security	Runtime analysis; communication paths
Deployment	Availability, Performance, Security	Resource allocation; failure analysis
Data Model	Performance, Security, Consistency	Data access patterns; integrity analysis

7.2 View Completeness for Evaluation

View Completeness Checklist for Evaluation

For Each ASR, verify:

- □ At least one view addresses the ASR
- □ View provides sufficient detail for analysis
- □ Relevant architectural tactics are visible
- □ Related decisions are documented

For Each View, verify:

- □ View purpose and stakeholders identified
- □ Notation clearly explained
- □ Models are complete for intended analysis
- □ Assumptions and limitations documented

For Cross-View Analysis:

- □ Correspondences between views documented
- □ Consistency between views verified
- □ Combined analysis is possible

8 Review Question Sets

8.1 Questions for Business Managers

Business Manager Questions

Business Goals:

1. Are the business goals the system must satisfy clearly articulated?
   - Where are business goals documented?
   - Are they stated in measurable terms?
2. Are business goals prioritized?
   - Who determined priorities?
   - What criteria were used for prioritization?
3. Are success criteria defined for each business goal?
   - How will success be measured?
   - What is the timeframe for achievement?

Goal-to-Architecture Traceability:

4. Is there clear mapping between business goals and requirements?
   - Can you trace from each goal to specific requirements?
   - Are requirements prioritized according to business importance?
5. Can you navigate from business goals to technical decisions?
   - Are the connections documented?
   - Is the rationale for decisions clear?
6. Can you trace from technical decisions back to business implications?
   - Do you understand risks to business goals?
   - Are tradeoffs that affect business clearly identified?

Criteria and Evolution:

7. What criteria determine whether architecture supports business goals?
   - Are criteria documented?
   - How will criteria be evaluated?
8. How might the system change over its deployment lifetime?
   - What business changes are anticipated?
   - Is retirement/replacement considered?

8.2 Questions for Architects

Architect Questions

Context and Stakeholders:

1. Is the system context clearly defined?
   - External systems and interfaces identified?
   - Boundaries clearly established?
2. Have stakeholders and their concerns been defined?
   - Is the stakeholder list complete?
   - Are concerns mapped to stakeholders?

Requirements and ASRs:

3. Have requirements, constraints, and standards been defined?
   - Functional requirements documented?
   - Quality requirements documented?
   - Constraints clearly stated?
4. Are ASRs clearly articulated and prioritized?
   - What makes each requirement architecturally significant?
   - How was prioritization determined?
5. Are ASRs clear, unambiguous, and testable?
   - Are quality attribute scenarios defined?
   - Are response measures quantified?

Decisions and Rationale:

6. Are techniques used to achieve ASRs documented?
   - What architectural tactics were applied?
   - What patterns were used?
7. Have alternatives been considered and documented?
   - What options were evaluated?
   - Why were alternatives rejected?
8. Are key decisions identified and located?
   - Where is the decision log?
   - Are decisions traceable to ASRs?
9. Is rationale captured for key decisions?
   - Is reasoning documented?
   - Are tradeoffs explained?

Analysis Artifacts:

10. Can you describe runtime resources for each operational concern?
    - CPU, memory, network, storage requirements?
    - Resource allocation rationale?
11. Can you describe change impact for modifiability concerns?
    - Size and difficulty of anticipated changes?
    - Components affected by changes?
12. Can you determine views necessary to analyze each ASR?
    - Which views address which ASRs?
    - Are views sufficient for analysis?
13. Do models within views address ASRs adequately?
    - Is there enough information to evaluate ASR satisfaction?
    - Are gaps identified?
14. Are all ASRs addressed by models or correspondences?
    - Is coverage complete?
    - Are any ASRs unaddressed?

8.3 Questions for Evaluation Team

Evaluation Team Questions

Documentation Clarity:

1. Are concepts and notations clearly explained?
   - Is there a glossary of terms?
   - Is there a key for diagrams?
   - Is notation consistent throughout?

Evaluation Scope:

2. Have evaluation scope and objectives been defined?
   - What is being evaluated?
   - What are the evaluation goals?
   - What is out of scope?
3. Is the system context for evaluation clearly defined?
   - Boundaries of evaluation clear?
   - External dependencies identified?
4. Have stakeholders and concerns for evaluation been identified?
   - Who should participate?
   - What concerns will be evaluated?

View Evaluation:

5. For each view, do you understand how to evaluate it?
   - Is the evaluation approach clear?
   - Are analysis techniques applicable?
6. For correspondences across views, do you understand representation?
   - How are correspondences documented?
   - How will you evaluate accuracy and completeness?
7. Are views sufficiently complete for intended analysis?
   - What gaps exist?
   - Can you work around identified gaps?

Analysis Preparation:

8. Is the quality attribute scenario list complete and prioritized?
   - Are scenarios testable?
   - Do scenarios cover key quality concerns?
9. Can you identify architectural approaches for each scenario?
   - Are tactics visible in the architecture?
   - Can you trace scenarios to decisions?
10. Can you identify potential risks, sensitivities, and tradeoffs?
    - Where might problems occur?
    - What are the sensitive parameters?
    - What competing qualities exist?

9 Evaluation Readiness Assessment

9.1 Readiness Checklist

Architecture Evaluation Readiness Checklist

Business Context:

- □ Business goals documented
- □ Goals prioritized
- □ Success criteria defined
- □ Goal-to-requirement traceability exists

ASRs and Scenarios:

- □ ASRs identified and documented
- □ ASRs prioritized by architectural impact
- □ Quality attribute scenarios defined
- □ Scenarios are testable with response measures

Architectural Decisions:

- □ Key decisions identified
- □ Alternatives considered and documented
- □ Rationale captured for each decision
- □ Decision-to-ASR traceability exists

Analysis Artifacts:

- □ Risks identified
- □ Sensitivity points identified
- □ Tradeoff points identified
- □ Preliminary analysis documented

Views and Models:

- □ Views address all ASRs
- □ View notation explained
- □ Models complete for analysis
- □ Cross-view correspondences documented

Documentation Quality:

- □ Glossary of terms provided
- □ Diagram keys provided
- □ Navigation aids in place
- □ Gaps identified with placeholders

9.2 Readiness Assessment Matrix

Table 13: Evaluation Readiness Assessment

Criterion	**Ready**	**Partial**	**Not Ready**	**Notes**
Business goals documented	□	□	□	
Goals prioritized	□	□	□	
ASRs identified	□	□	□	
ASRs prioritized	□	□	□	
Scenarios defined	□	□	□	
Scenarios testable	□	□	□	
Decisions documented	□	□	□	
Alternatives considered	□	□	□	
Rationale captured	□	□	□	
Risks identified	□	□	□	
Views cover ASRs	□	□	□	
Notation explained	□	□	□	
Overall Readiness	□	□	□	

10 Appendix A: ATAM Overview

Key Point

Architecture Tradeoff Analysis Method (ATAM)
ATAM is the most widely used architecture evaluation method, developed by the Software Engineering Institute (SEI). It provides a structured approach for evaluating software architectures against quality attribute requirements.

10.1 ATAM Phases

Table 14: ATAM Phases and Activities

Phase	**Activities**	**Documentation Needed**
Phase 0: Partnership	Define scope; identify stakeholders; plan evaluation	Stakeholder list; evaluation scope; schedule
Phase 1: Evaluation	Present business drivers; present architecture; identify approaches; analyze scenarios	Business goals; architecture overview; architectural approaches; utility tree

Phase 2: Evaluation	Brainstorm scenarios; prioritize scenarios; analyze scenarios; present results	Refined scenarios; analysis results; risks, sensitivities, tradeoffs
Phase 3: Follow-up	Document results; track findings	Final report; action items

10.2 ATAM Outputs

- **Architectural approaches:** Key design strategies identified
- **Quality attribute utility tree:** Hierarchical view of quality requirements
- **Prioritized scenarios:** Scenarios ranked by importance
- **Risk themes:** Patterns of risks across the architecture
- **Sensitivity points:** Parameters with significant impact
- **Tradeoff points:** Competing quality attribute effects
- **Risks and non-risks:** Evaluated architectural decisions

11 Appendix B: Quality Attribute Scenario Template

Quality Attribute Scenario Template

Scenario ID: [QAS-XXX]
Quality Attribute: [e.g., Performance, Security, Availability]
Business Goal: [BG-XXX - Related business goal]
ASR: [ASR-XXX - Related ASR]
Source of Stimulus: [Who or what generates the stimulus]
Stimulus: [The condition or event]
Artifact: [What part of system is affected]
Environment: [Operating conditions]
Response: [How system should behave]
Response Measure: [Quantifiable measure]
Priority: [H/M/L based on business importance and architectural impact]
Architectural Approach: [How architecture addresses this scenario]
Analysis Notes: [Preliminary analysis, risks, concerns]

12 Appendix C: Glossary

Architectural Approach
A collection of architectural decisions that address one or more quality attributes

Architecturally Significant Requirement (ASR)
A requirement that has measurable impact on system architecture

ATAM Architecture Tradeoff Analysis Method; structured approach for evaluating architectures

Business Goal A statement of organizational objective the system must support

CBAM Cost Benefit Analysis Method; economic analysis of architectural decisions

Quality Attribute
A measurable property of a system (e.g., performance, security)

Quality Attribute Scenario
A testable specification of a quality requirement

Risk An architectural decision that may lead to undesirable consequences

Sensitivity Point
An architectural parameter where small changes have large effects

Tradeoff Point An architectural decision affecting multiple qualities differently

Utility Tree A hierarchical representation of quality attribute requirements

13 Appendix D: References

1. Clements, P., Kazman, R., & Klein, M. (2002). *Evaluating Software Architectures: Methods and Case Studies.* Addison-Wesley.
2. Bass, L., Clements, P., & Kazman, R. (2021). *Software Architecture in Practice* (4th ed.). Addison-Wesley.
3. Kazman, R., Klein, M., & Clements, P. (2000). *ATAM: Method for Architecture Evaluation.* CMU/SEI-2000-TR-004.
4. Kazman, R., Asundi, J., & Klein, M. (2001). *Quantifying the Costs and Benefits of Architectural Decisions.* ICSE 2001.
5. Clements, P., et al. (2010). *Documenting Software Architectures: Views and Beyond* (2nd ed.). Addison-Wesley.
6. Rozanski, N., & Woods, E. (2011). *Software Systems Architecture* (2nd ed.). Addison-Wesley.
7. ISO/IEC/IEEE 42030:2019. *Software, systems and enterprise—Architecture evaluation framework.*
8. Barbacci, M., et al. (2003). *Quality Attribute Workshops (QAWs), Third Edition.* CMU/SEI-2003-TR-016.