Architecture Evaluation Using ATAM
Software Engineering II

Rami Bahsoon
School of Computer Science
University of Birmingham, UK
r.bahsoon@cs.bham.ac.uk
Architecture Evaluation

• Evaluation by the designer within the design process.
  – Evaluation by the designer is the “test” part of the “generate-and-test” approach to architecture design.

• Evaluation by peers within the design process.
  – Architectural designs can be peer reviewed, just as code can.
  – A peer review can be carried out at any point of the design process where a candidate architecture, or at least a coherent reviewable part of one, exists.
  – Allocate at least several hours and possibly half a day.

• Analysis by outsiders once the architecture has been prototype, designed.
Evaluation Steps

• Step 1. The reviewers determine a number of quality attribute scenarios to drive the review.
  – Scenarios can be developed by the review team or by additional stakeholders.

• Step 2. The architect presents the portion of the architecture to be evaluated.
  – The reviewers individually ensure that they understand the architecture. Questions at this point are specifically for understanding.

• Step 3. For each scenario, the designer walks through the architecture and explains how the scenario is satisfied.
  – The reviewers ask questions to determine (a) that the scenario is, in fact, satisfied and (b) whether any of the other scenarios being considered will not be satisfied.
Evaluation Steps

Step 4. Potential problems are captured.

- Real problems must either must be fixed or a decision must be explicitly made by the designers and the project manager that they are willing to accept the problems and its probability of occurrence.

- Outsiders
  - Chosen because they possess specialized knowledge or experience, or long experience successfully evaluating architectures.
  - Managers tend to be more inclined to listen to problems uncovered by an outside team.
  - An outside/independent team tends to be used to evaluate complete architectures.
Architecture Stakeholders

• Stakeholders have a vested interest in the architecture performing as advertised
  – Stakeholders include developers, testers, integrators, maintainers, performance engineers, users, builders of systems interacting with the one under consideration, and, possibly, others.
  – Their job is to articulate the specific quality attribute goals that the architecture should meet.
  – Expect to enlist 12 to 15 stakeholders for the evaluation of a large enterprise-critical architecture.
The Architecture Tradeoff Analysis Method (ATAM)

- Been used for over a decade to evaluate software architectures in domains ranging from automotive to financial to defense.
- The purpose is to assess the consequences of architectural decision alternatives in light of quality attribute (QA) requirements.
- QA examples: Availability, Reliability, Security etc.
- Evaluators need not be familiar with the architecture or its business goals, the system need not yet be constructed, and there may be a large number of stakeholders...
ATAM Objectives...

• A method in which the right questions are asked early to:
  – Discover risks - alternatives that might create future problems in some quality attribute
  – Discover non-risks - decisions that promote qualities that help realize business/mission goals
  – Discover sensitivity points - alternatives for which a slight change makes a significant difference in some quality attribute
  – Discover trade-offs - decisions affecting more than one quality attribute
• Example of sensitivity and tradeoff points
  – A backup database positively affects reliability (sensitivity point with respect to reliability). However it negatively affects performance (tradeoff)
Benefits of ATAM Analysis

• There are a number of benefits from performing the ATAM analyses:
  – Clarified quality attribute requirements
  – Improved architecture documentation
  – Documented basis for architectural decisions
  – Identified risks early in the life-cycle
  – Increased communication among stakeholders

The results are improved/refined/evaluated architectures
ATAM Phases

1. Present the ATAM
2. Present business drivers
3. Present architecture
4. Identify architectural approaches
5. Generate quality attribute utility tree
6. Analyze architectural approaches
7. Brainstorm and prioritize scenarios
8. Analyze architectural approaches
9. Present results

Phase I

Phase II
1. Present the ATAM

**Evaluation Team** presents an overview of the ATAM including:

- ATAM steps in brief
- Techniques
  - utility tree generation
  - architecture elicitation and analysis
  - scenario brainstorming/mapping
- Outputs
  - architectural approaches
  - utility tree
  - scenarios
  - risks and “non-risks”
  - sensitivity points and tradeoffs
2. Present Business Drivers

**ATAM customer** representative describes the **system’s** business drivers including:

- Business context for the system
- High-level functional requirements
- High-level quality attribute requirements
  - architectural drivers: quality attributes that “shape” the architecture
  - critical requirements: quality attributes most central to the system’s success
3. Present Architecture

Architect presents an overview of the architecture including:

- Technical constraints such as an OS, hardware, or middleware prescribed for use
- Other systems with which the system must interact
- Architectural approaches/styles used to address quality attribute requirements

Evaluation team begins probing for and capturing risks.
4. Identify Architectural Approaches

Start to identify places in the architecture that are key for realizing quality attribute goals.

Identify any predominant architectural approaches.

Examples:
• client-server
• 3-tier
• watchdog
• publish-subscribe
• redundant hardware
5. Generate Quality Attribute Utility Tree

Identify, prioritize, and refine the most important quality attribute goals by building a utility tree.

- A utility tree is a top-down vehicle for characterizing the “driving” attribute-specific requirements

- Select the most important quality goals to be the high-level nodes (typically performance, modifiability, security, and availability)

- Scenarios are the leaves of the utility tree

Output: a characterization and a prioritization of specific quality attribute requirements.

High/Medium/Low importance for the success of the system
High/Medium/Low difficulty to achieve (architect’s assessment)
Utility Tree Construction - 1

- **Performance**
  - Data Latency
  - Transaction Throughput

- **Modifiability**
  - New product categories
  - Change COTS

- **Availability**
  - H/W failure
  - COTS S/W failures

- **Security**
  - Data confidentiality
  - Data integrity

- **Utility**

- **Reduce storage latency on customer DB to < 200 ms.**
  - Deliver video in real time

- **Add CORBA middleware in < 20 person-months**
  - Change web user interface in < 4 person-weeks

- **Power outage at site1 requires traffic redirected to site2 in < 3 seconds.**
  - Restart after disk failure in < 5 minutes
  - Network failure detected and recovered in < 1.5 minutes

- **Credit card transactions are secure 99.999% of the time**
  - Customer DB authorization works 99.999% of the time
Utility Tree Construction - 2

- **Performance**
  - Data Latency
  - Transaction Throughput
    - Reduce storage latency on customer DB to < 200 ms.
    - Deliver video in real time

- **Modifiability**
  - New product categories
  - Change COTS
    - Add CORBA middleware in < 20 person-months
    - Change web user interface in < 4 person-weeks

- **Availability**
  - H/W failure
  - COTS S/W failures
    - Power outage at site1 requires traffic redirected to site2 in < 3 seconds.
    - Restart after disk failure in < 5 minutes
    - Network failure detected and recovered in < 1.5 minutes

- **Security**
  - Data confidentiality
  - Data integrity
    - Credit card transactions are secure 99.999% of the time
    - Customer DB authorization works 99.999% of the time
Example Scenarios

Use case scenario
Remote user requests a database report via the Web during peak period and receives it within 5 seconds.

Growth scenario
Add a new data server to reduce latency in scenario 1 to 2.5 seconds within 1 person-week.

Exploratory scenario
Half of the servers go down during normal operation without affecting overall system availability.

=> Scenarios should be as specific as possible.
Stimuli, Environment, Responses

Use Case Scenario
Remote user requests a database report via the Web during peak period and receives it within 5 seconds.

Growth Scenario
Add a new data server to reduce latency in scenario 1 to 2.5 seconds within 1 person-week.

Exploratory Scenario
Half of the servers go down during normal operation without affecting overall system availability.

=> Scenarios should be as specific as possible.
6. Analyze Architectural Approaches

Evaluation Team probes architectural approaches from the point of view of specific quality attributes to identify risks.

- Identify the approaches that pertain to the highest priority quality attribute requirements
- Generate quality-attribute specific questions for highest priority quality attribute requirement
- Ask quality-attribute specific questions
- Identify and record risks and non-risks, sensitivity points and tradeoffs
Scenarios

Scenarios are used to

- Represent stakeholders’ interests
- Understand quality attribute requirements

Scenarios should cover a range of

- Anticipated uses of (use case scenarios),
- Anticipated changes to (growth scenarios), or
- Unanticipated stresses (exploratory scenarios) to the system.

A good scenario makes clear what the stimulus is that causes it and what responses are of interest.
7. Brainstorm and Prioritize Scenarios

Stakeholders generate scenarios using a facilitated brainstorming process.

- Scenarios at the leaves of the utility tree serve as examples to facilitate the step.
- The new scenarios are added to the utility tree

Each stakeholder is allocated a number of votes
Example Scenarios

Use case scenario
Remote user requests a database report via the Web during peak period and receives it within 5 seconds.

Growth scenario
Add a new data server to reduce latency in scenario 1 to 2.5 seconds within 1 person-week.

Exploratory scenario
Half of the servers go down during normal operation without affecting overall system availability.

=> Scenarios should be as specific as possible.
Conceptual Flow of ATAM

- Business Drivers
- Quality Attributes
- Scenarios
- Architectural Approaches
- Architectural Decisions
- Analysis

impacts → distilled into

Risk Themes
- Tradeoffs
- Sensitivity Points
- Non-Risks
- Risks

R. Bahsoon
When to use ATAM

Academically, the time to use ATAM is right after the architecture has been specified when there is little or no code.

However, in practice, ATAM has been very effective in the following situations:

- Evaluating alternative candidate architectures
- Evaluating existing systems prior to committing to major upgrades
- Deciding between upgrade or replace
The evaluation team documents the relevant architectural decisions and catalogs their risks, non-risks, sensitivity points, and tradeoffs. Examples:

- **Risk**: The frequency of heartbeats affects the time in which the system can detect a failed component. Some assignments will result in unacceptable values of this response.

- **Sensitivity point**: The number of simultaneous database clients will affect the number of transactions that a database can process per second.

- **Tradeoff**: The heartbeat frequency determines the time for detecting a fault. Higher frequency leads to better availability but consumes more processing time and communication bandwidth (potentially reducing performance).
### Example of ATAM Scenario

<table>
<thead>
<tr>
<th>Scenario #: A12</th>
<th>Scenario: Detect and recover from HW failure of main switch.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute(s)</td>
<td>Availability</td>
</tr>
<tr>
<td>Environment</td>
<td>Normal operations</td>
</tr>
<tr>
<td>Stimulus</td>
<td>One of the CPUs fails</td>
</tr>
<tr>
<td>Response</td>
<td>0.999999 availability of switch</td>
</tr>
<tr>
<td>Architectural decisions</td>
<td>Sensitivity</td>
</tr>
<tr>
<td>Backup CPU(s)</td>
<td>S2</td>
</tr>
<tr>
<td>No backup data channel</td>
<td>S3</td>
</tr>
<tr>
<td>Watchdog</td>
<td>S4</td>
</tr>
<tr>
<td>Heartbeat</td>
<td>S5</td>
</tr>
<tr>
<td>Failover routing</td>
<td>S6</td>
</tr>
<tr>
<td>Reasoning</td>
<td>Ensures no common mode failure by using different hardware and operating system (see Risk 8). Worst-case rollover is accomplished in 4 seconds as computing state takes that long at worst. Guaranteed to detect failure within 2 seconds based on rates of heartbeat and watchdog. Watchdog is simple and has proved reliable. Availability requirement might be at risk due to lack of backup data channel ... (see Risk 9)</td>
</tr>
</tbody>
</table>

#### Architecture diagram

```
  Primary CPU (OS1)
  \__________\----------\---------
  |            |            |            |
  | heartbeat  | Switch     | Switch    |
  | (1 sec.)   | CPU (OS1)  | CPU (OS2) |
  |            |            |           |
  \__________\----------\---------
  |            |            |            |
  | Backup     |            |            |
  | CPU with    |            |            |
  | Watchdog   |            |            |
  | (OS2)      |            |            |
```

~ Stress test
References

- https://www.sei.cmu.edu/architecture/tools/evaluate/atum.cfm