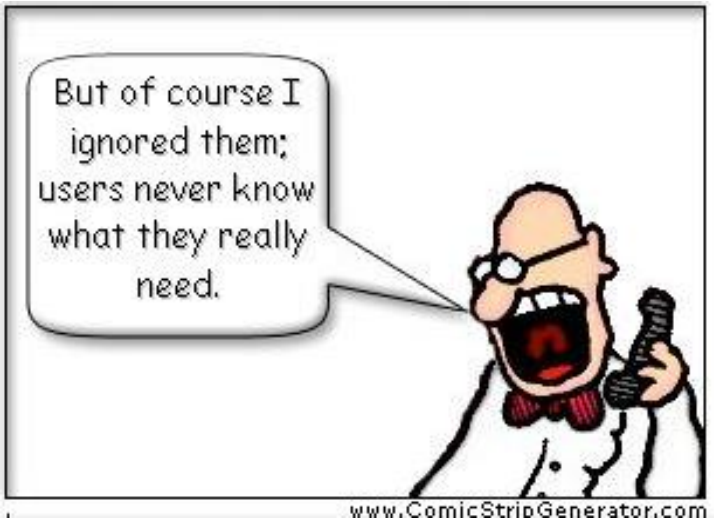
SENG 401

Architectural Evaluation

Table of Contents

- Introduction
- Architecture Tradeoff Analysis Method (ATAM)
- Cost-Benefit Analysis Method (CBAM)
- Other methods
- Conclusion



Introduction

SW Architecture Challenges

- Most of the modern software systems are required to be modifiable and have good performance
- They may also need to be usable, secure, interoperable, portable, and reliable
- Questions for any particular system:
  - What precisely do these quality attributes such as modifiability, usability, security, performance and reliability mean?
  - Can a system be analyzed to determine these desired qualities?
  - How soon can such an analysis occur?
  - **How do you know if software architecture for a system is suitable without having to build the system first?**

→ SW Architecture Evaluation / Assessment!

Problem Types

- Determining whether an architecture satisfies its requirements often involves:
  - Being very explicit about what the requirements (functional & non-functional) are and how they are reflected in the architecture
  - Understanding where one has to make trade-offs between different design alternatives
  - Applying analysis wherever possible to determine the consequences of an architectural choice
  - Mediating between desires of different stakeholders

To achieve these goals an architectural evaluation process is needed

Problem Types

- Comparing two or more systems
  - e.g., what are advantages of solution A vs. B?
  - e.g., is Safari x.x better than Firefox x.x?
- Assessment of various architectural solutions for a pre-built system
  - e.g., is P2P better than client-server for this problem?
  - e.g., is 64-bit encryption better that 128-bit from performance point of view?
- Post release quality assessment
  - e.g., does this system meet its reliability target?

Our focus

SW Architecture Evaluation

Informal / ad-hoc architectural evaluation

- Pros?

- Cons?

SW Architecture Evaluation

- Are there ~~better~~ other methods than ad-hoc evaluation?
- The answer is “YES”:
  - SAAM (Software Architecture Analysis Method)
    - Scenario-based evaluation
  - ATAM (Architecture Tradeoff Analysis Method)
    - Scenario-based evaluation with focus on trade-offs
  - SACAM (Software Architecture Comparison Method)
    - Business goal-driven comparison of architecture alternatives
  - CBAM (Cost-Benefit Analysis Method)
    - Focus on economic aspects
  - etc.

Architecture Tradeoff Analysis Method (ATAM)

ATAM: Overview

- Architecture Tradeoff Analysis Method (ATAM) is a method for evaluating software architectures relative to goals specified by quality attributes.
- ATAM exposes architectural risks that potentially inhibit the achievement of an organization's business goals.
- ATAM not only reveals how well an architecture satisfies particular quality goals, but

  it also provides insight into how

  quality goals can be possibly

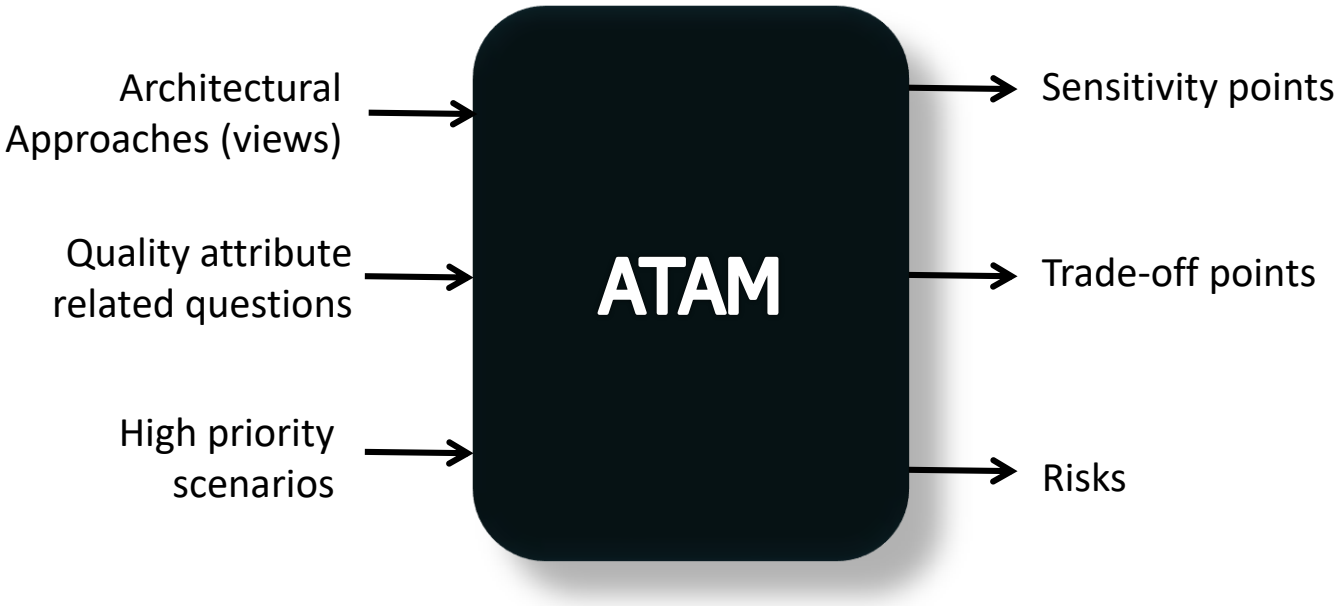
  traded-off against each other.

ATAM: Summary

Goal:

Evaluate whether the design decisions satisfactorily address quality requirements

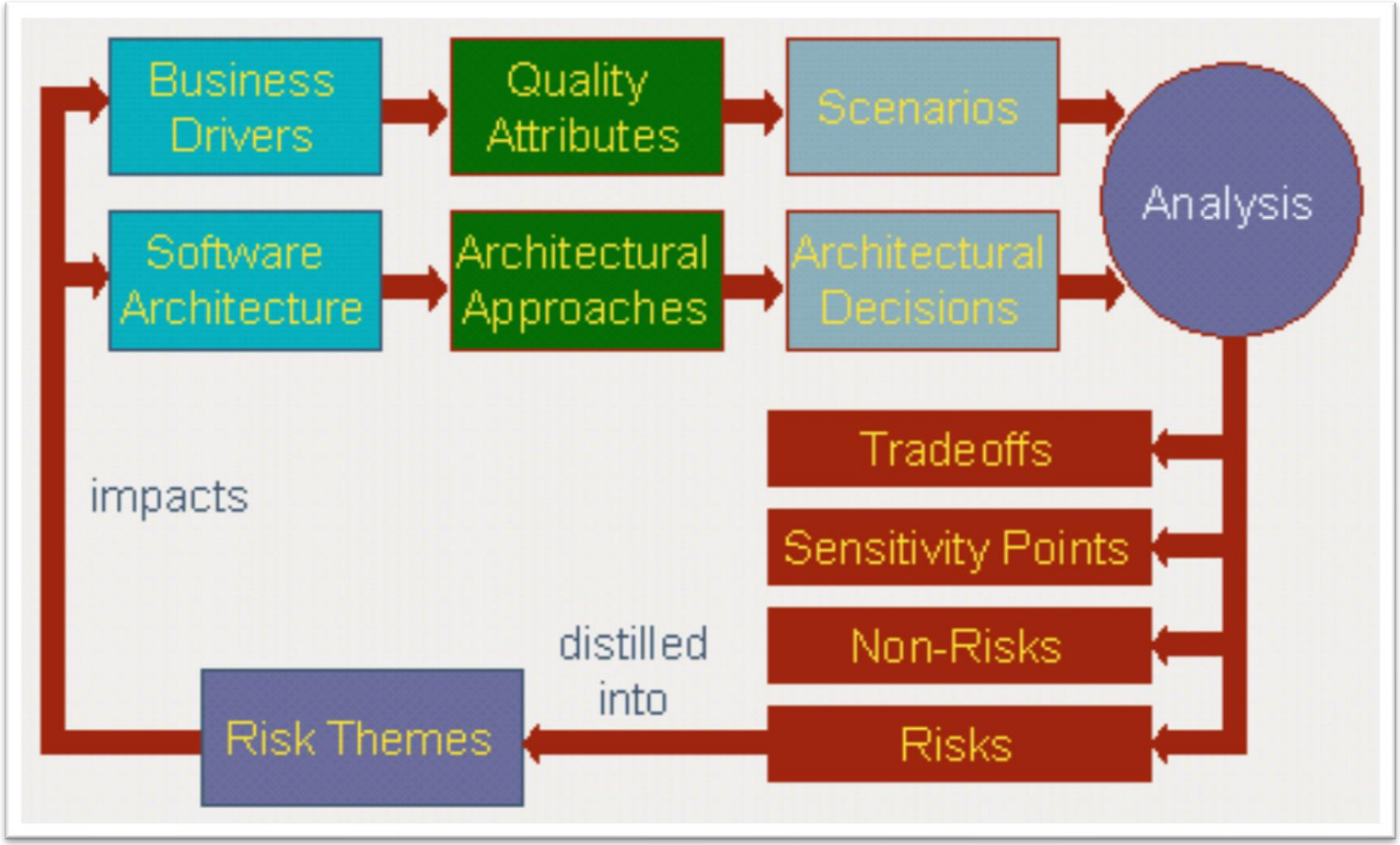
i.e., assess the consequences of architectural decisions in light of quality requirements



ATAM: Overview

- Evaluate if the design decision satisfactorily address the quality requirements.
- **Traceability**: Elicit rationale of design decisions
- Discover **risks**: Alternatives that might create (future) problems in some quality attribute
- Discover **sensitivity points**: Alternatives for which a slight change makes a significant difference in a quality attribute
- Discover **tradeoffs**: Decisions affecting more than one quality attribute

ATAM – Conceptual Flow



ATAM – Conceptual Flow (1)

- Business Drivers are elicited from project's stakeholders.
- They are refined into Scenarios and matched against the Architectural Decisions made in architectural design.
- Analysis of scenarios and architectural decisions results in identification of Risks, Non-risks, Sensitivity Points, and Trade-off Points in the architecture.
- Risks are synthesized into a set of Risk Themes, showing how each one threatens a business driver.

ATAM – Conceptual Flow (2)

- The output of an ATAM is a presentation and/or a written report that includes the major findings of the evaluation:
  - The architectural approaches (or styles) identified/used
  - A "utility tree" – a hierarchic model of the driving (non-functional) architectural requirements
  - The set of scenarios generated and the subset that were mapped onto the architecture
  - A set of quality-attribute specific questions that were applied to the architecture and the responses to these questions
  - A set of identified risks and non-risks
  - A set of sensitivity points and trade-offs

ATAM Methodology: Details

- Motivation
- ATAM Elements
- ATAM Process
- Summary

ATAM: Motivation (1)

Starting Point:

- Software Architecture shall address both functional and non-functional requirements. ← cf. global analysis

Related Terminology:

- Quality attribute = non-functional characteristic of a software system

  → maintainability, dependability, portability, availability, reliability, security, performance, compatibility, usability, concurrency, etc.

ATAM: Motivation (2)

- System designs typically have *Trade-offs*
  - Example: *Global Evaluation* section in architectural views

- It is important to make these Trade-offs explicit
  - Possibility for (early) risk identification
  - *Trade-off-points* are spots in the architecture that address several quality attributes concurrently (→ critical)

ATAM: Motivation (3)

Goals of ATAM:

- **To assess the consequences of architectural decisions in light of quality attribute requirements**
- To identify problem/risk areas of *one specific* architecture
- *Not* to make precise predictions of quality attribute values ... but to find correlations between *architectural parameters* and *quality attributes* (→ parameters can later be influenced/changed!)
- **Examples:**
  - Message encryption component: #bits of the key (→ security vs. performance)
  - Module decomposition: cohesion and coupling (→ maintainability vs. performance)

ATAM: Motivation (4)

Benefits of ATAM:

- Can be performed early in the software development life cycle
- Clarifies requirements (functional, non-functional)
- Improved documentation of architecture (→ explicit trade-off points)
- Decision-support at architecture level

- ***Eliminate surprise!***
- ***Better architecture!***

ATAM: Elements (1)

- **Quality Attributes**
  - Performance, Maintainability, Reliability, Security, Testability, etc.
- **Architectural Views**
  - Aspects: structure (module, code), behavior (control-flow, data-flow), execution, physical devices (computers, networks, …), processes, etc.
- **Stakeholders**
  - Developers, Testers, Maintainers, System Administrators, Customers, Users, Architects (!!), etc.
- **Scenarios**

ATAM: Elements (2)

1. Quality Attributes:

- If ATAM is intended for analysis of an architecture with respect to quality attributes, there is a problem with this focus: “We do not understand quality attributes well!”
- What it means to be “open” or “interoperable” or “secure” or “high performance”?
- The meaning changes from system to system, from stakeholder to stakeholder.
- Key idea: **Scenarios & *Trade-off-points***

ATAM: Elements (2)

2. Architectural Views:

- Facilitate discussion of changes (enhancements)
- Selection of Views depends on the Quality Attributes to be analyzed
  - Modifiability → module, static, use cases, data flow
  - Security → module view, data flow
  - Performance → execution, process, physical views
  - Reliability → physical view, reliability block diagram

3. Stakeholders:

- Decide which Quality Attributes are relevant

ATAM: Elements (3)

Scenarios:

- Define *Quality Attributes* more precisely
- A Scenario describes how the software system will be
  - used / abused
  - evolved
  - put under stress / load
  - crashed
  - etc.
- From the point of view of a (specific) *Stakeholder*

ATAM Process: Description

- ATAM process consists of 9 steps separated into 4 groups (phases):
  - Presentation
    - To exchange information
  - Investigation and Analysis
    - To assess key quality attribute requirements
    - As reflected in the architectural approaches
  - Testing
    - To check the results available to date against the needs of all relevant stakeholders
  - Reporting
    - To record and present results of the ATAM

- **Steps:**

1. Present the ATAM
2. Present Business Drivers
3. Present Architecture

Presentation

4. Identify Architectural Approaches
5. Generate Quality Attribute Utility Tree
6. Analyze Architectural Approaches

Investigation & Analysis

7. Brainstorm and Prioritize Scenarios
8. Analyze Architectural Approaches

Testing

9. Present Results

Reporting

ATAM Process: Presentation

1. **Present the ATAM:** The evaluation leader describes the evaluation method to the assembled participants, tries to set their expectations, and answers questions they may have.
2. **Present business drivers:** A project spokesperson (project manager or customer) describes what business goals are motivating the development effort and hence what will be the primary architectural drivers (e.g., high availability or time to market or high security).
3. **Present architecture:** The architect will describe the architecture, focusing on how it addresses the business drivers.

ATAM Process: Investigation & Analysis

4. **Identify architectural approaches:** Architectural approaches are identified by the architect, but are not analyzed.
5. **Generate quality attribute utility tree:** The quality factors that comprise system 'utility' (performance, availability, security, modifiability, usability, etc.) are elicited, specified down to the level of scenarios, annotated with stimuli and responses, and prioritized.
6. **Analyze architectural approaches:** Based upon the high-priority factors identified in Step 5, the architectural approaches that address those factors are elicited and analyzed (e.g., an architectural approach aimed at meeting performance goals will be subjected to a performance analysis). During this step architectural risks, sensitivity points, and trade-off points are identified.

ATAM Process: Testing

7. **Brainstorm and prioritize scenarios:** A larger set of scenarios is elicited from the entire group of stakeholders. This set of scenarios is prioritized via a voting process involving the entire stakeholder group.
8. **Analyze architectural approaches:** This step reiterates the activities of Step 6, but using the highly ranked scenarios from Step 7. Those scenarios are considered to be test cases to confirm the analysis performed thus far. This analysis may uncover additional architectural approaches, risks, sensitivity points, and tradeoff points, which are then documented.

ATAM Process: Reporting

9. **Present results:**

   Based upon the information collected in the ATAM (approaches, scenarios, attribute-specific questions, the utility tree, risks, non-risks, sensitivity points, trade-offs) the ATAM team presents the findings to the assembled stakeholders.

ATAM Output

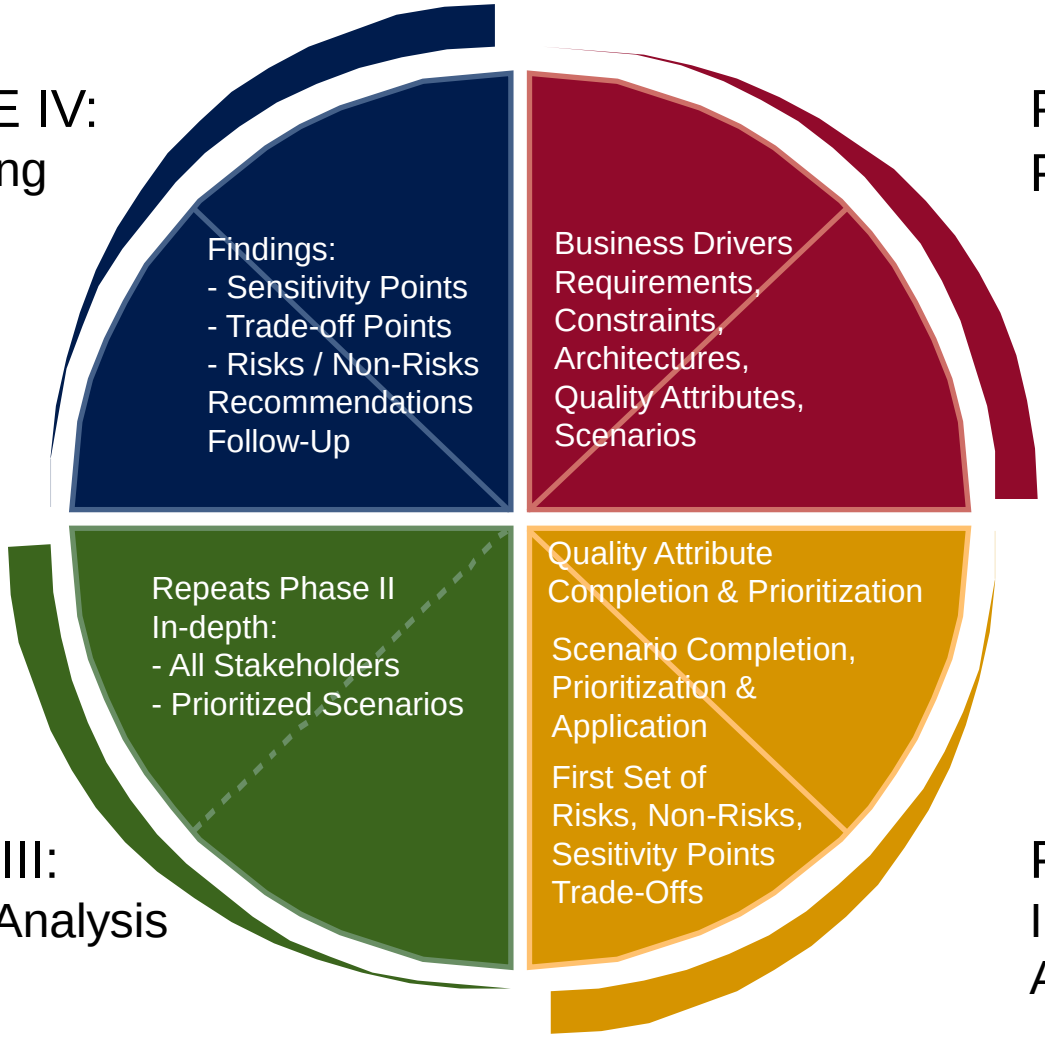
- Precise description of the architecture
- Articulation of the business goals
- Quality requirements in terms of scenarios

+

- Risks
- Trade-offs and sensitivity-points

ATAM Details

ATAM Process: Phases



ATAM: Preparation

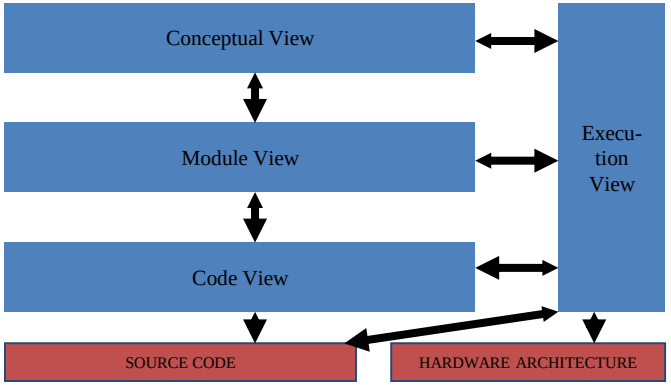
- Planning of ATAM (by ATAM Team):
  - Meeting before actual evaluation starts
  - Goals
    - Identify important quality attributes
    - Collect ideas for initial set of scenarios
    - If not yet available: Sketch / refine architectural approaches and select relevant views
    - Select stakeholders for actual evaluation

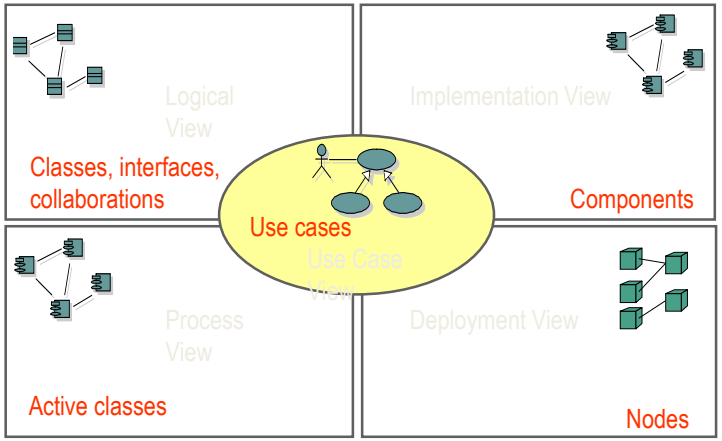
ATAM: Presentation (1)

- Presentation of the Business Drivers
  - Project Spokesman (e.g., Project Manager, Customer)
    - Describes what business goals are motivating the development effort ...
    - … and hence what will be the primary architectural drivers (e.g., high availability or time to market or high security or high reliability, etc.)

ATAM: Presentation (2)

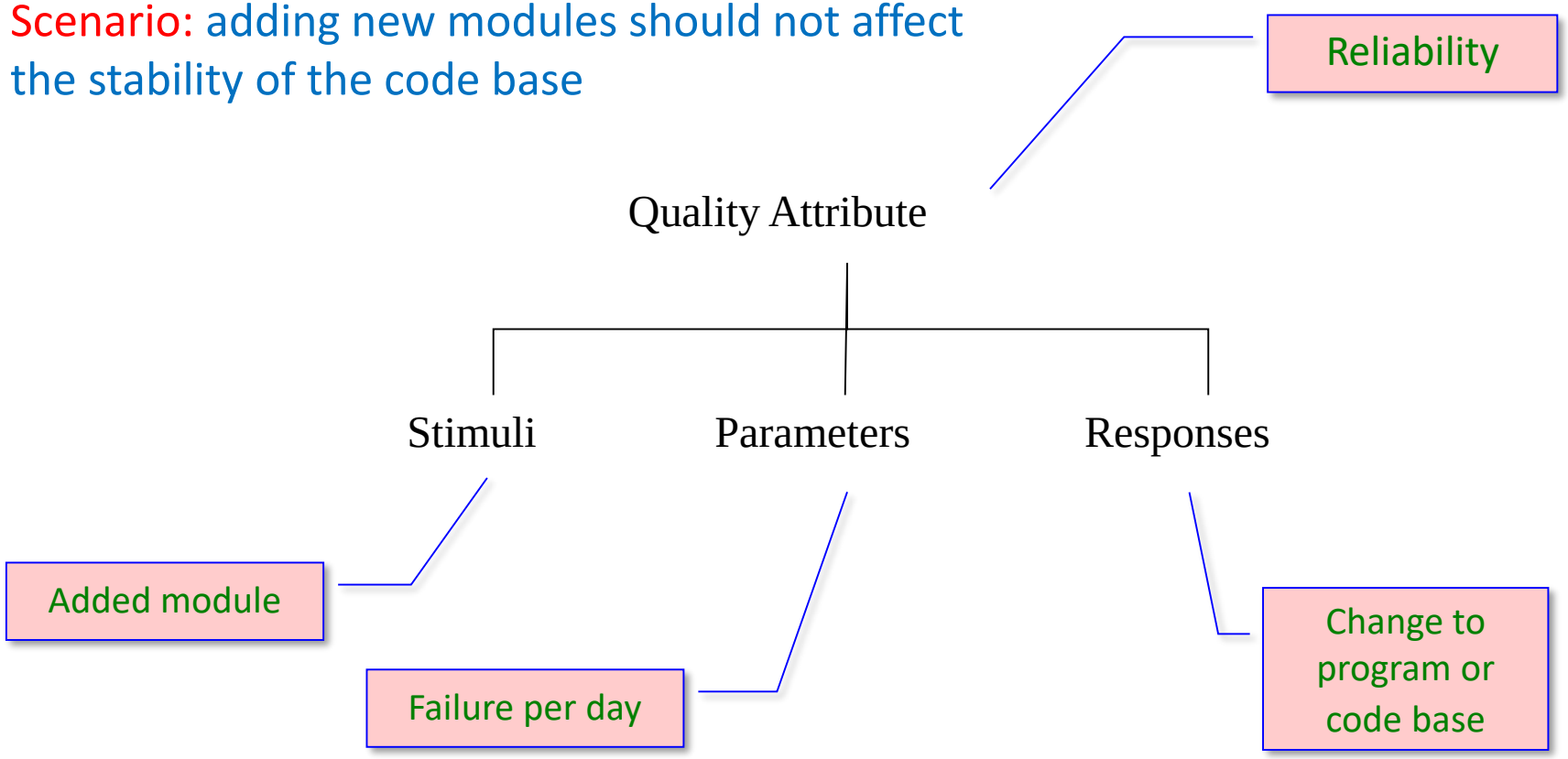
- Presentation of the architecture
- Architect
  - Presents architectural views
  - Uses scenarios for explaining the architecture
- Evaluators
  - Identify architectural styles / patterns
  - Ask questions, add / complete scenarios



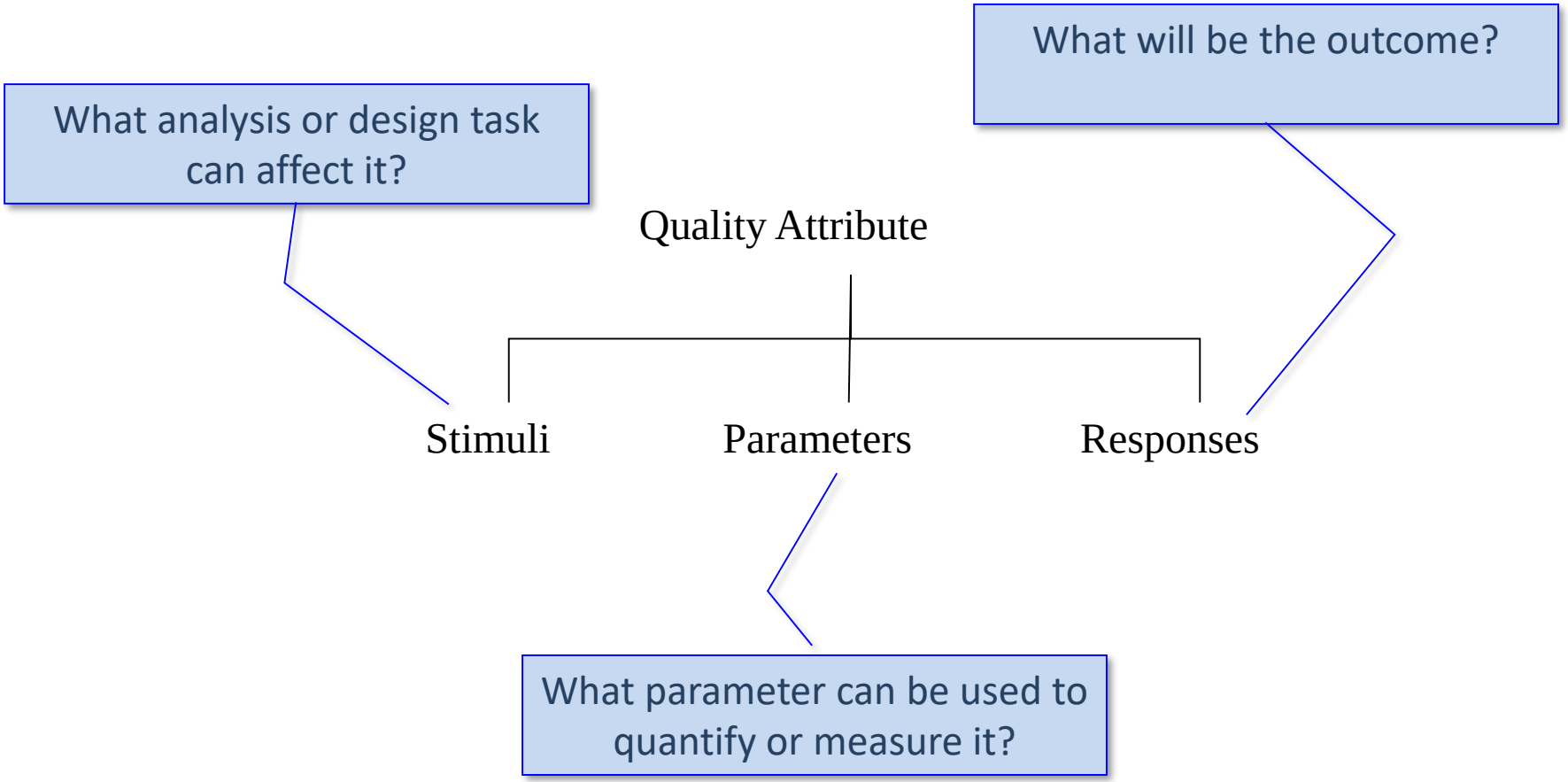


- Example:

Scenario: adding new modules should not affect the stability of the code base



- General structure:



Scenario: Example

- **Scenario:**

  “Switch to backup server should be within 30 seconds of observing a shutdown failure.”

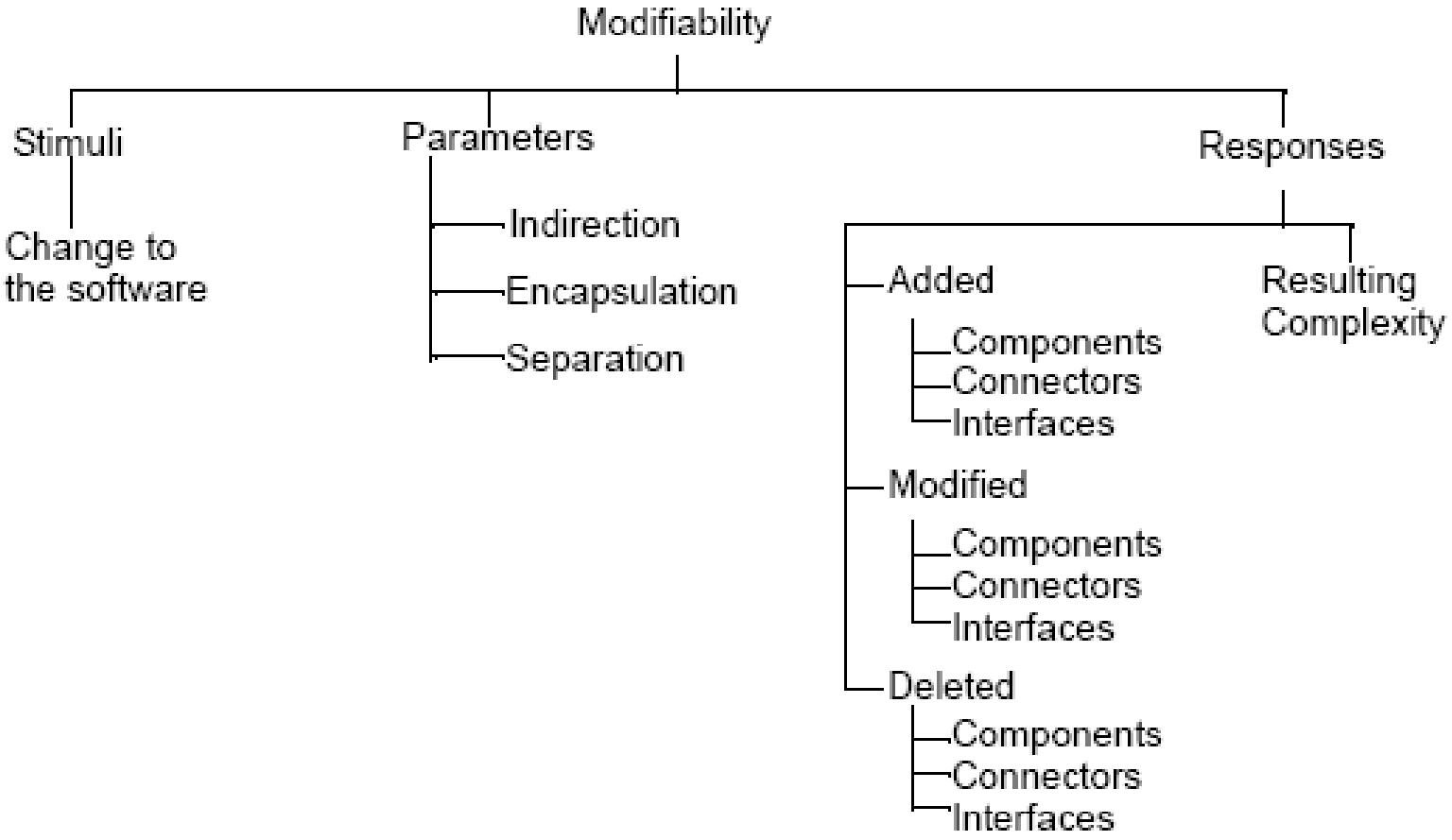
- Stimuli: shutdown failure
- Response: Switch to backup server within 30 seconds
- Quality attribute: Availability

ATAM: Typical Quality Attributes

- ***Modifiability:***
  - Analyses identify affected elements of the architecture and the potential effort needed for changes / modifications
- ***Performance:***
  - Development and application of performance models, e.g., queuing or scheduling models
- ***Security:***
  - Identification of potential weak spots in the case of the realization of threats
- ***Reliability / Availability:***
  - Development of reliability or availability models, e.g. Markov models (based on assumptions on failure rates, MTBF, etc.)

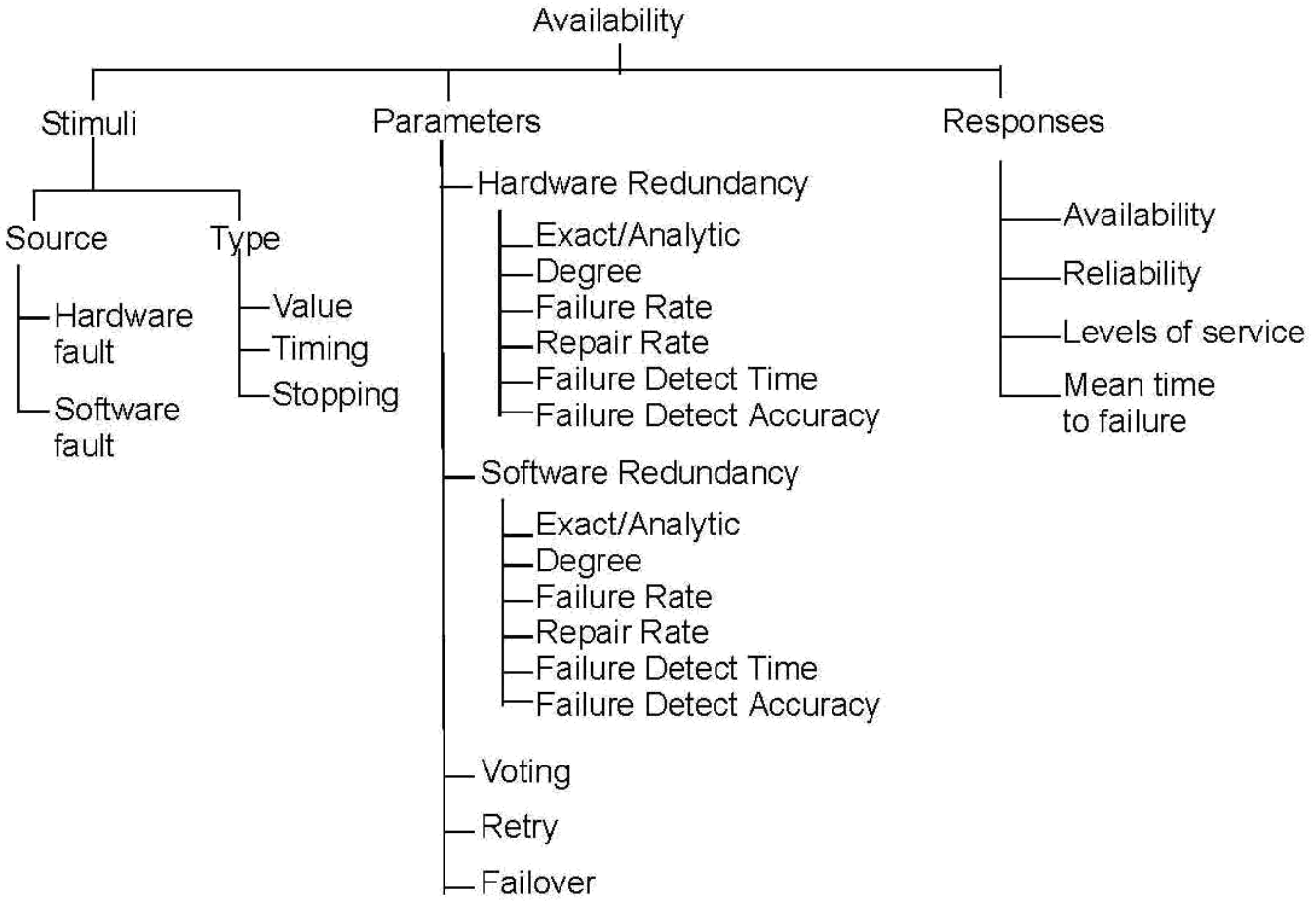
ATAM – Quality Attribute Characteristics (2)

- Example – Modifiability



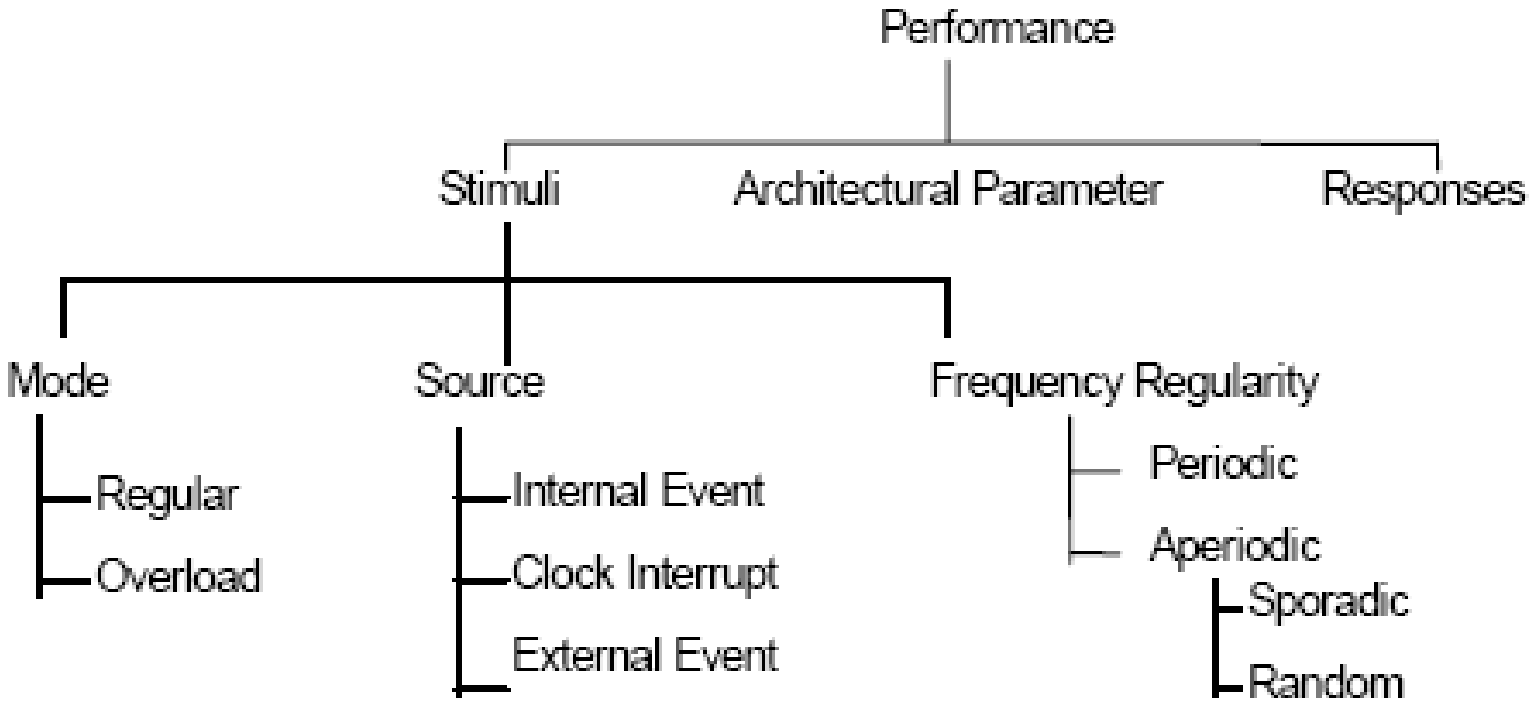
ATAM – Quality Attribute Characteristics (3)

- Example – Availability



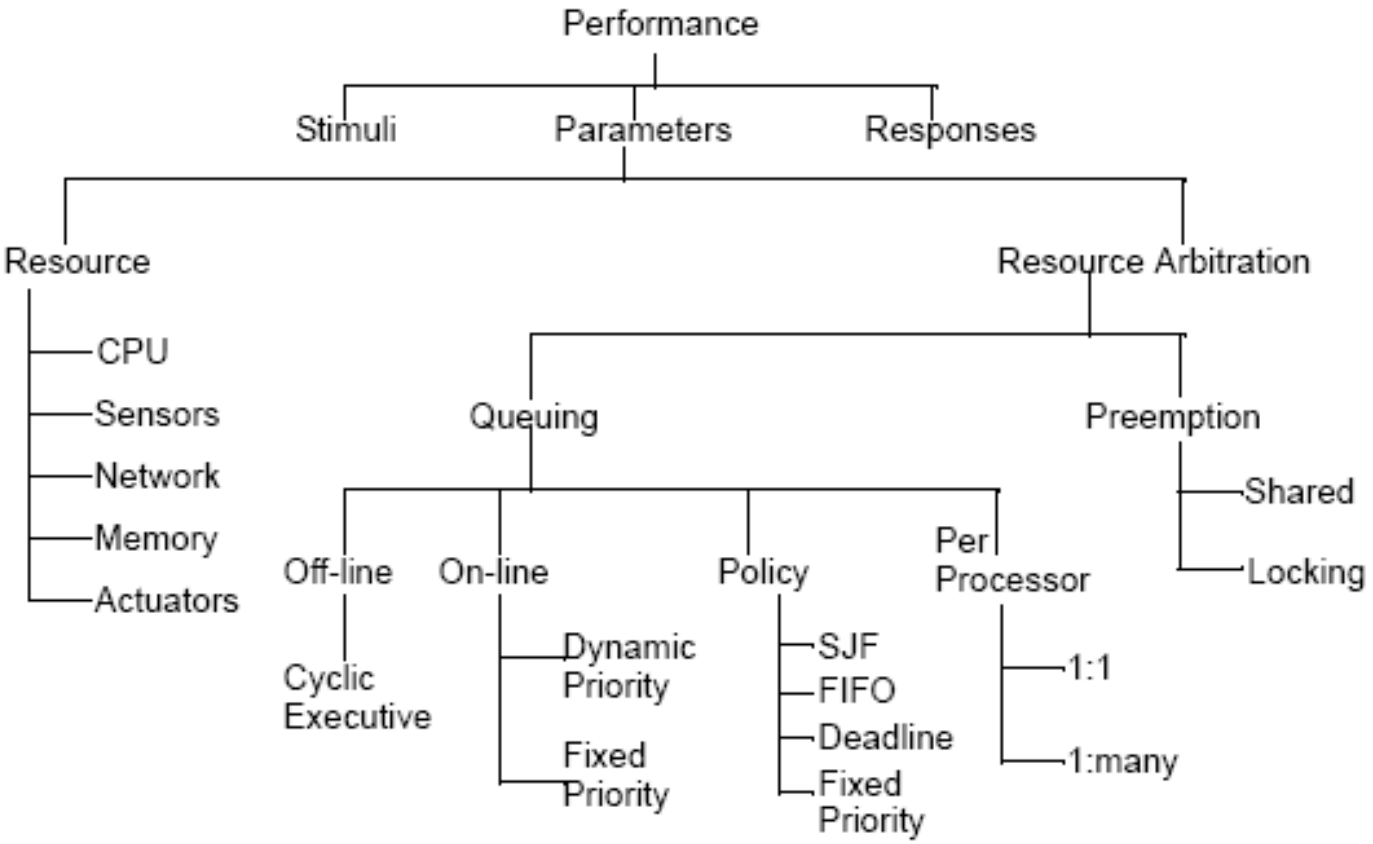
ATAM: Quality Attribute Characteristics (4)

- Example – Performance: Stimuli



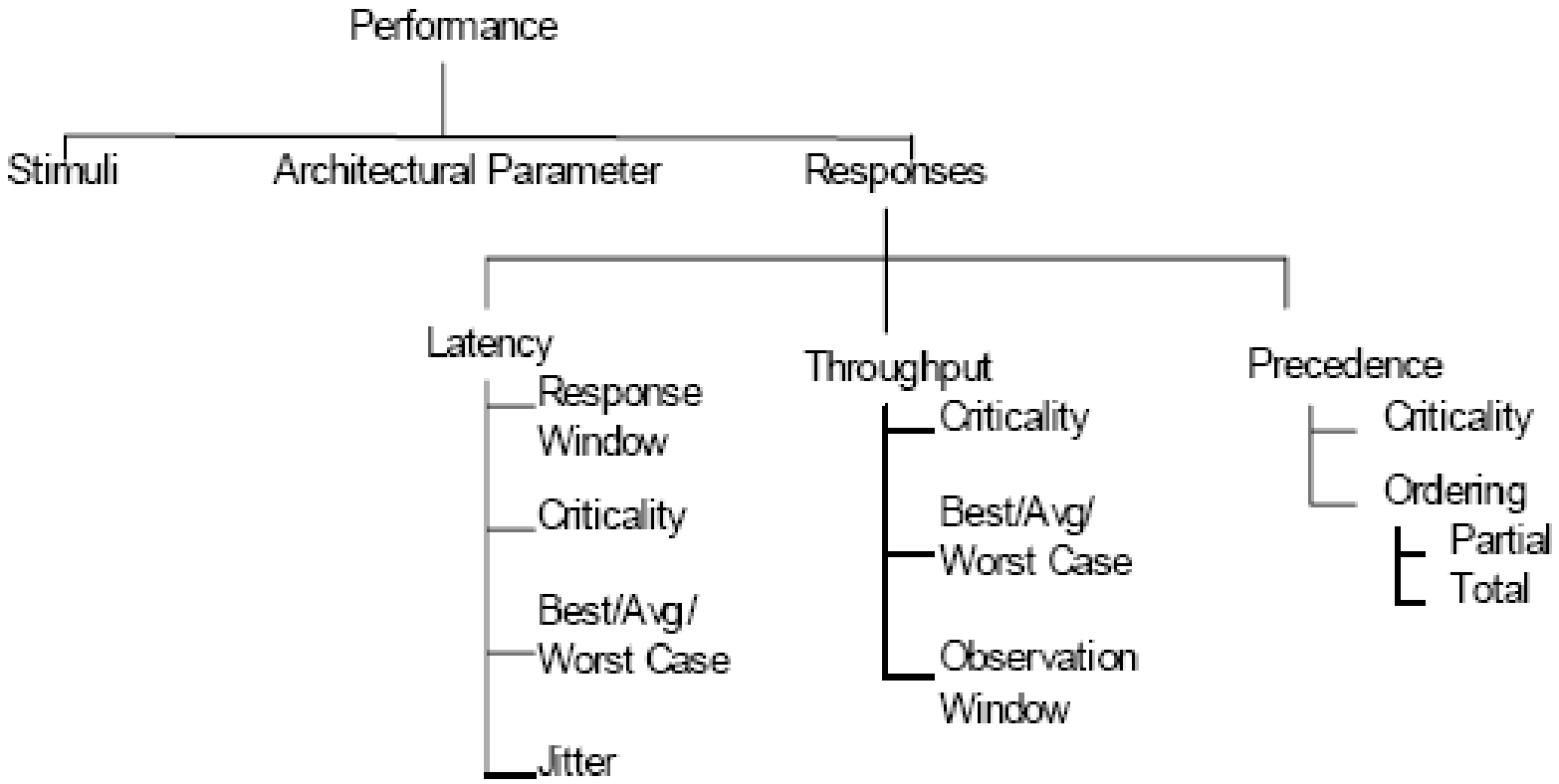
ATAM: Quality Attribute Characteristics (5)

- Example – Performance: Parameters



ATAM: Quality Attribute Characteristics (6)

- Example – Performance: Responses



ATAM: Quality Attribute Related Questions (1)

- Quality attribute questions elicit architectural decisions which bear on quality attribute requirements
- **Example: Modifiability**
  - If this architecture includes layers/facades, are there any places there where the layers/facades are circumvented?
  - If this architecture includes a data repository, how many distinct locations in the architecture have direct knowledge of its data types and layout?
  - If a shared data type changes, how many parts of the architecture are affected?

ATAM: Quality Attribute Related Questions (2)

- **Example: Performance**
  - If there are multiple processes competing for a shared resource, how are priorities assigned to these processes and the process controlling the resource?
  - If there are multiple pipelines of processes/threads, what is the lowest priority for each process/thread in each pipeline?
  - If multiple message streams arrive at a shared message queue, what are the rates and distributions of each stream?
  - Are there any relatively slow communication channels along an important communication path (e.g., a modem)?

ATAM – Quality Attribute Related Questions (3)

- **Example: Availability**
  - If redundancy is used in the architecture, what type of redundancy (analytic, exact, functional) and how is the choice made between redundant components?
  - How are failures identified?
  - Can active as well as passive failures be identified?
  - If redundancy is used in the architecture, how long does it take to switch between instances of a redundant component?

ATAM: Scenarios

- Development of scenarios (by core Stakeholders):
- **Scenarios address quality attributes that are of particular interest**
- Scenarios are used to:
  - Represent stakeholders' interests
  - Understand quality attribute requirements
- A good scenario makes clear what the stimulus is and what the measurable response of interest is

STIMULI – (ENVIRONMENT) – RESPONSES → Attribute

ATAM: Types of Scenarios

- **Use case scenarios** (→ intended usage of system)
  - e.g., A remote user requests a database report via the Web during peak period and receives it within 5 seconds
- **Growth scenarios** (→ planned change/modification of system)
  - e.g., Add a new data server during peak hours within a downtime of at most 8 hours.
- **Exploratory scenarios** (→ system's reaction to "stress")
  - e.g., Half of the servers go down during normal operation without affecting overall system availability
- **Exception/Threat scenarios** (→ system's reaction to exceptions/threats)
  - e.g., System should never allow unauthorized user access

ATAM: Scenario Types (1)

- **Type 1: Use Case Scenarios**
  1. The **caching system will be switched to another processor** when its **processor fails**, and will do so **within one second**. (reliability)
  2. User **changes graph layout from horizontal to vertical** and **graph is redrawn in one second**. (performance)
  3. Remote user **requests a database report via the Web** during peak period and **receives it within five seconds**. (performance)
  4. The user wants to examine budgetary and actual data under different fiscal years without re-entering project data. (usability)
  5. A data exception occurs and the system notifies a defined list of recipients by e-mail and displays the offending conditions in red on data screens. (reliability)

  Stimulus – **Response** (Attribute)

ATAM: Scenario Types (2)

- **Type 2: Growth Scenarios**
  1. **Add a new message type** to the system's repertoire **in less than a person-week** of work. (performance)
  2. Add a collaborative planning capability where two planners at different sites collaborate on a plan in less than a person-year of work. (perform.)
  3. Migrate to a new operating system, or a new release of the existing operating system in less than a person-year of work. (perf., rel., security)
  4. Add a new data server to reduce latency in use case Scenario 5 to 2.5 seconds within one person-week. (performance, reliability, security)
  5. Double the size of existing database tables while maintaining 1 second average retrieval time. (performance)

  Stimulus – **Response** (Attribute)

ATAM: Scenario Types (3)

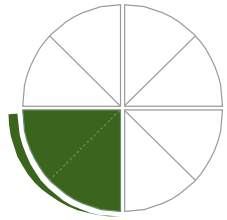
- **Type 3: Exploratory Scenarios**
  1. **Add** a **new 3-D map feature, and a virtual reality interface** for viewing the maps **in less than five person-months of effort**. (performance)
  2. Change the underlying Unix platform to Windows.
  3. Improve the system's availability from 98% to 99.999%. (availability)
  4. Half of the servers go down during normal operation without affecting overall system availability. (availability)
  5. Tenfold increase in the number of bids processed hourly while keeping worst-case response time below 10 seconds. (performance)

  Stimulus – **Response** (Attribute)

ATAM: Scenarios

- Finalization of scenarios (by whole team):
  - Grouping of scenarios
  - Elimination of redundant scenarios
  - Checking whether set of scenarios sufficiently covers all relevant aspects of the system
  - Prioritization of scenarios
  - Typically, 15-20 scenarios will be selected for further analyses

ATAM: Prioritizing Scenarios



- Two mechanisms:

	Utility Trees	Facilitated Brainstorming
Stakeholders	Architects, project leader	All stakeholders
Typical Group size	2 evaluators; 2-3 project personnel	4-5 evaluators; 5-10 project-related personnel
Primary Goals	Elicit, concretize and prioritize the driving quality attribute require-ments. Provide a focus for the remainder of the evaluation.	Foster stakeholder communica-tion to validate quality attribute goals elicited via the utility tree.
Approach	Top-down (general to specific)	Bottom-up (specific to general)
	More focused!	More inclusive!

ATAM: Prioritizing Scenarios

Utility Tree:

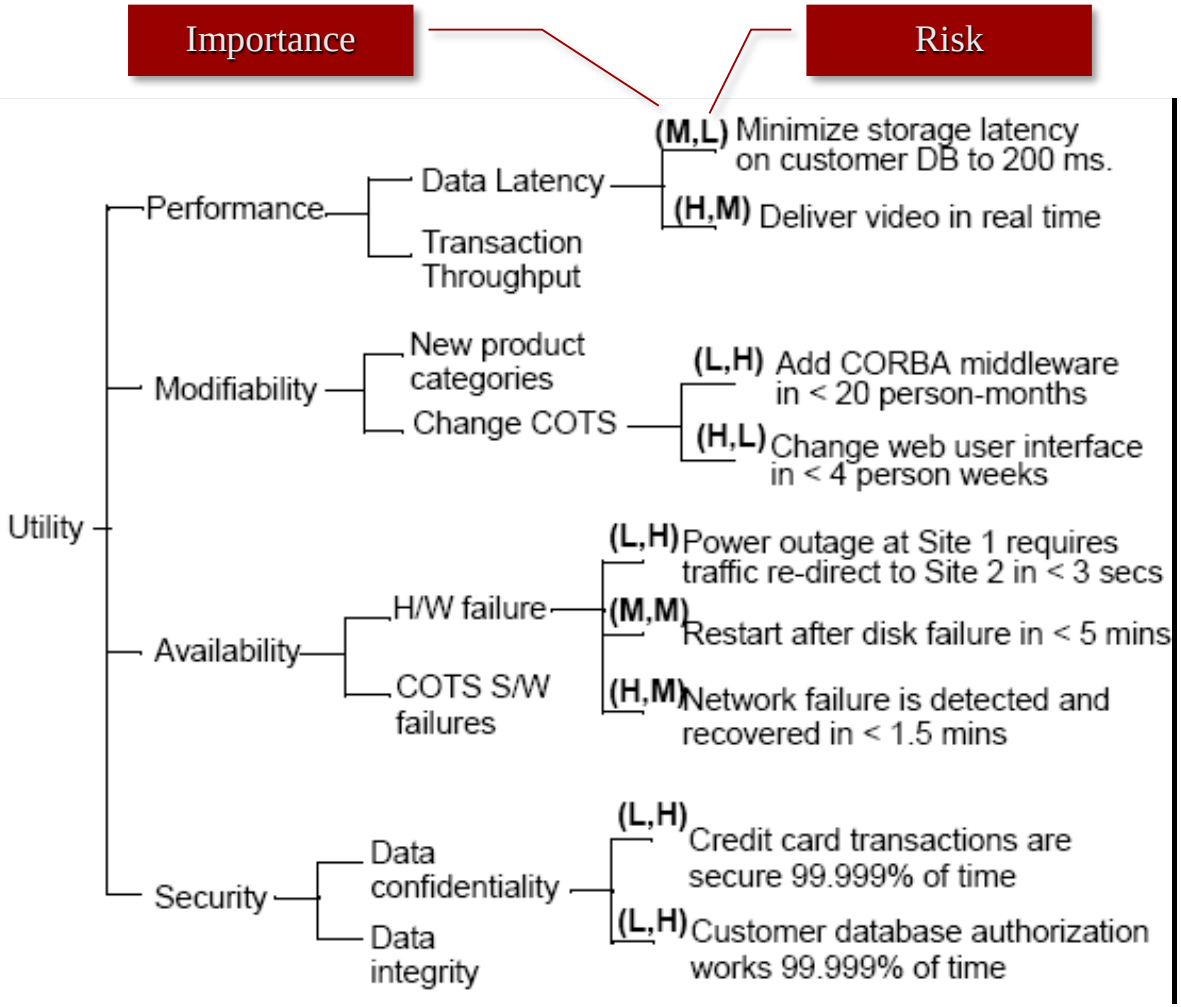
A top-down mechanism for transforming business drivers of a system into concrete quality attribute scenarios.

H = high

M = medium

L = low

(X, Y) = (Importance of factor for success of system, Risk to achieve it)



ATAM: Prioritizing Scenarios

Utility Tree:

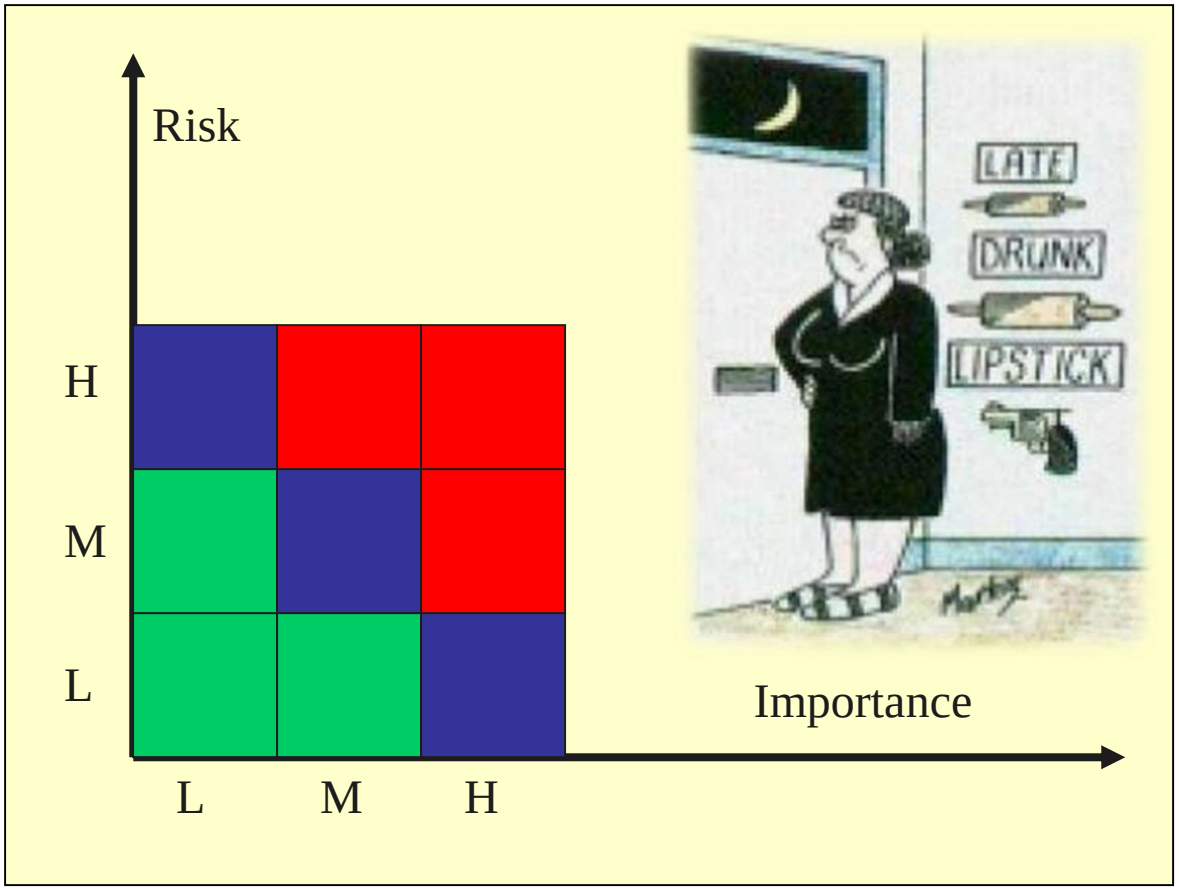
A top-down mechanism for transforming business drivers of a system into concrete quality attribute scenarios.

H = high

M = medium

L = low

(X, Y) = (Importance of factor for success of system, Risk to achieve it)



ATAM: Prioritizing Scenarios

Utility Tree:

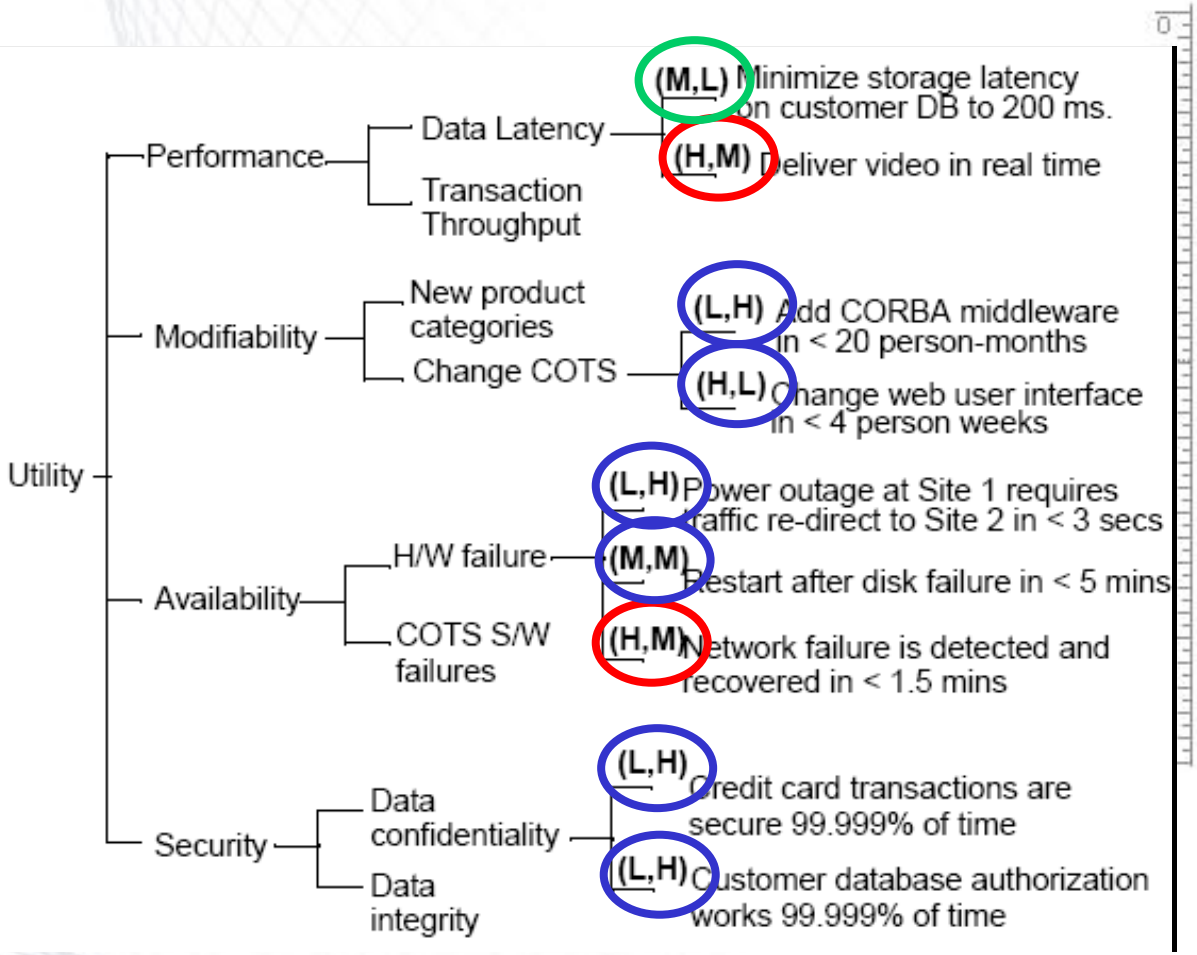
A top-down mechanism for transforming business drivers of a system into concrete quality attribute scenarios.

H = high

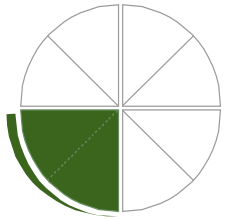
M = medium

L = low

(X, Y) = (Importance of factor for success of system, Risk to achieve it)



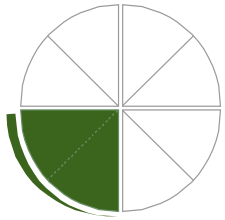
ATAM: Scenarios



Mapping scenarios to the architecture

- The architect applies (mentally) scenarios to the architecture and explains how the architecture will respond / react to it
- These reactions can refer to the development of the system or the execution of the system
- The evaluators record the architect's replies
- **Goal:** *identify Sensitivity Points, Trade-off Points, Risks, Non-Risks*

ATAM: Sensitivity Points

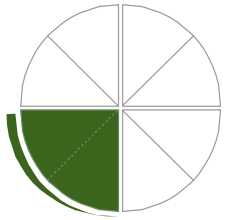


- Definition of ***Sensitivity Point***:
  - A parameter in the architectural description that has positive/negative impact on a specific quality attribute

- **Example:**
  - *Parameter:* encryption level
  - *Quality attribute:* performance
  - *Parameter:* encryption level
  - *Quality attribute:* security

increase encryption level → decrease performance
→ increase security

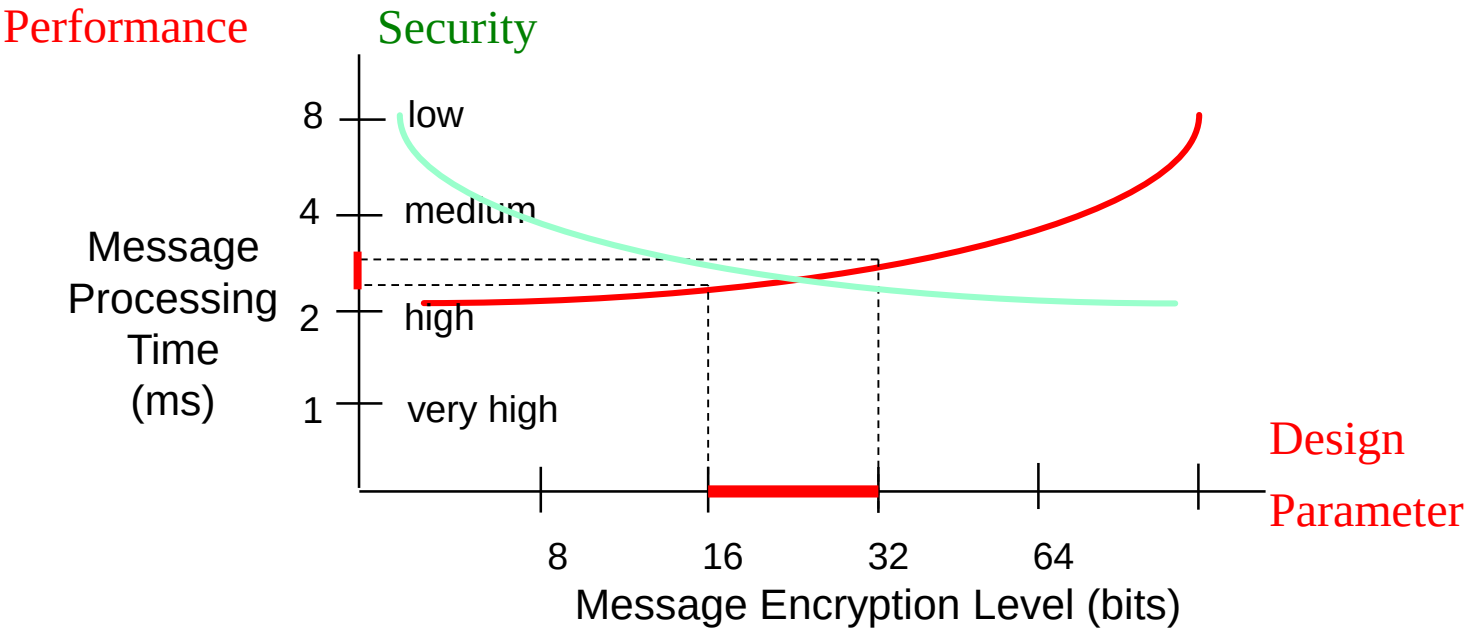
ATAM: Trade-off Points



- Definition of ***Trade-off Point***:
  - A parameter in the architecture that is sensitive to more than one quality attribute in the architecture

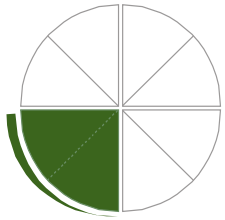
- **Example:**
  - *Parameter:* encryption level
  - *Quality attributes:* performance, security

ATAM – Sensitivity Analysis



This example shows that performance is relatively *insensitive* to the related architectural design parameter (here: Encryption Level)

ATAM: Quality Attribute Questions /1



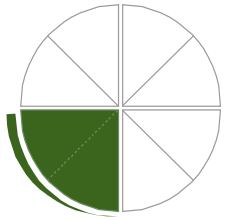
- Identification of trade-off points depends on **Quality Attribute Questions**.
- Quality attribute questions elicit architectural decisions which bear on quality attribute requirements
- **Quality Attribute Questions**:

  Do addressed quality attributes respond to variation of the architecture parameter

  in the *same direction* or

  in the *opposed direction*?

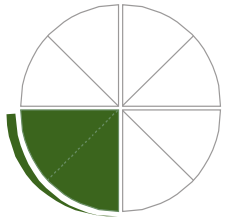
ATAM: Quality Attribute Questions /2



Example:

- What is the sensitivity of the architecture to *Message Encryption Level*?
  - Performance: -
  - Security: +
- Security and Performance are sensitive to the *Message Encryption Level* (in opposite directions)

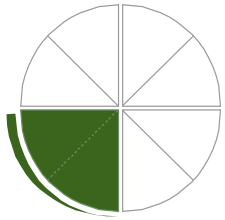
ATAM: Quality Attribute Questions /3



Example:

- What is the sensitivity of the architecture to the # of servers?
  - Performance: +
  - Availability: +
- Availability and Performance are sensitive to the # of servers (but not in opposite directions!)

ATAM: Risks & Non-Risks (1)

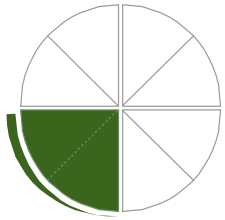


- ***Risks***: potentially problematic architectural decisions that need to be made explicit
- ***Non-risks***: good decisions frequently relying on implicit assumptions

- Both risks and non-risks should be understood and explicitly recorded

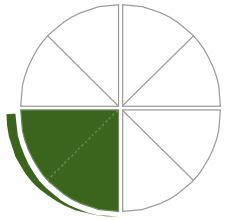
Trade-off points are candidate risks

ATAM: Risks & Non-Risks (2)



- The documenting of risks and non-risks consist of
  - an **architectural decision** (or a decision that has not been made)
  - a **specific quality attribute response** that is being addressed by that decision along with the consequences of the predicted level of the response
  - a **rationale for the positive or negative effect** that decision has on meeting the quality attribute requirement

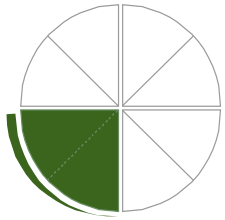
ATAM: Risks & Non-Risks (3)



Example of a Risk:

- "The rules for writing business logic modules in the second tier of your three-tier client-server style architecture are not clearly articulated (→ *decision that has not been made*).
- … This could result in replication of functionality thereby compromising modifiability of the third tier (→ *quality attribute response and its consequences*).
- … Unarticulated rules for writing the business logic can result in unintended and undesired coupling of components (→ *rationale for the negative effect*)."

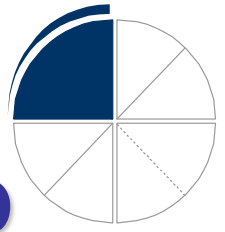
ATAM: Risks & Non-Risks (4)



Example of a Non-Risk:

- "Assuming message arrival rates of once per second, a processing time of less than 30 ms, and the existence of one higher priority process (→ *architectural decisions*),
- … a one-second soft deadline seems reasonable (→ *quality attribute response and its consequences*)
- … since the arrival rate is bounded and the preemptive effects of higher priority processes is known and can be accommodated (→ *rationale*)."

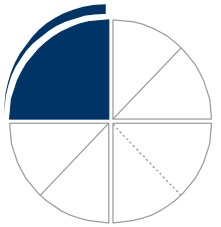
ATAM: Reporting & Follow-Up



Consolidation of results:

- Writing of a report
  - Sensitivity points / trade-off points / risks, etc.
  - Recommendations
- To Do's:
  - Development and evaluation of design alternatives (if necessary)
  - Refinement of analysis models (and application)

ATAM in Practice



- ATAM evaluations are often conducted in two phases:
- Phase 1: The architect describes the quality attribute goals and how the architecture meets these goals
- Phase 2: We determine if a larger group of stakeholders agrees with the goals and the results

ATAM: Realistic Example Use

- The SEI evaluated an engine control system for a major automotive supplier.
- The evaluation resulted in eliciting 52 scenarios, 5 of which were analyzed in detail.
- This resulted in discovering 6 trade-off points and 35 risks.

Section 4
ATAM Case Study

ATAM: Case Study

- The distributed concept mapping and sharing system (CM)
- We want to design the system and evaluate it based on the following quality attributes:

  availability, ***performance*** and ***security***

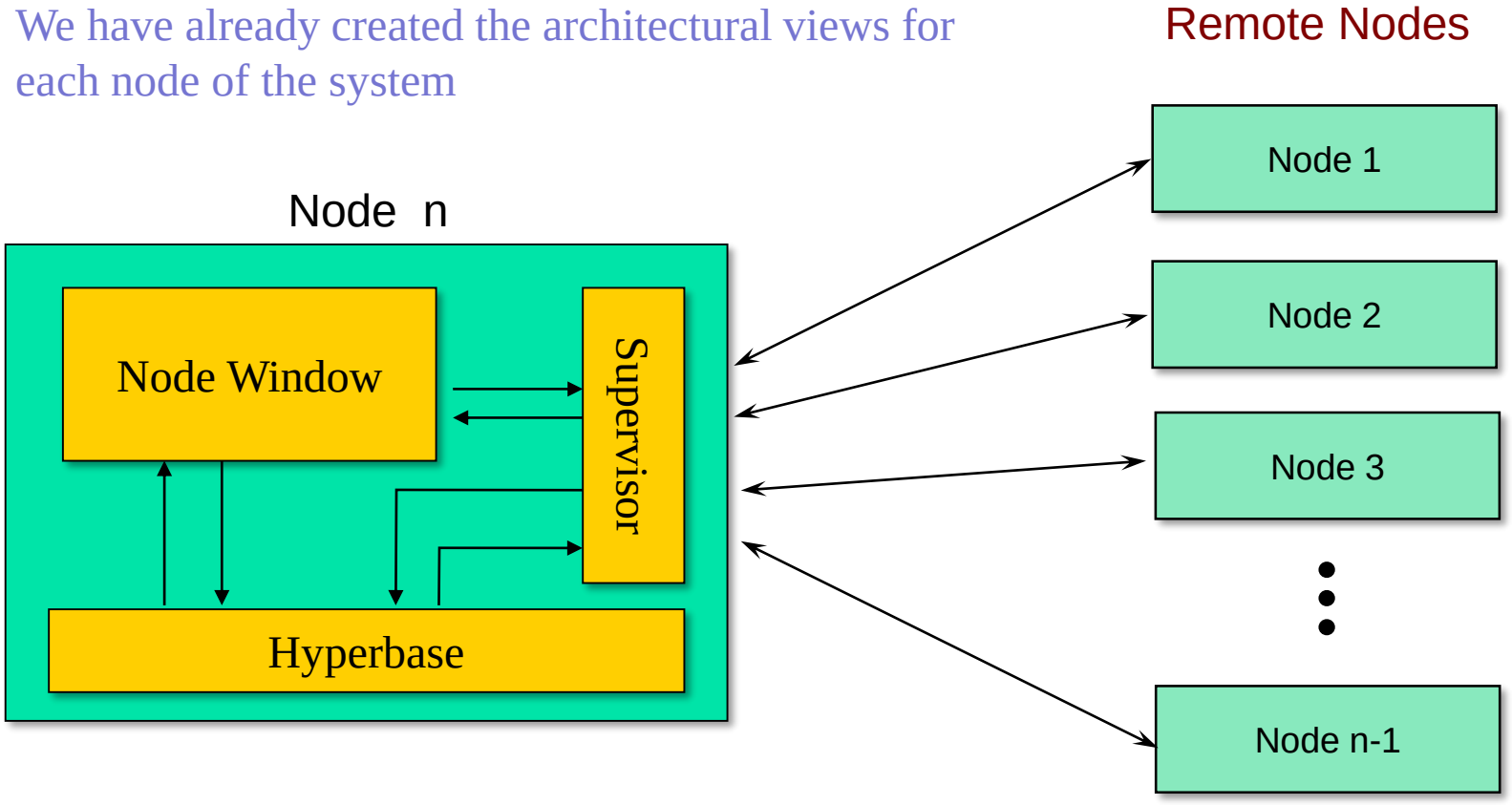
- Other factor of consideration: ***cost***

After consulting with stakeholders

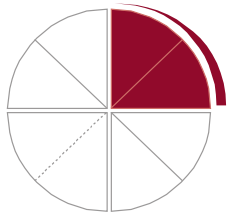
Other possible attributes: ***modifiability***, ***integrity***, etc.

Functional View of System

We have already created the architectural views for each node of the system



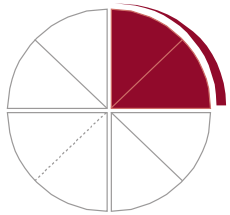
Availability Scenarios



- **Availability** related scenarios:
  - **SC1:** A node suffers a software failure and is rebooted.
  - **SC2:** A node suffers a power outage and is rebooted.
  - **SC3:** A node suffers a power supply failure and is repaired.

  - **Contextual Parameter:** node
  - **Metrics:** uptime/(uptime + downtime)

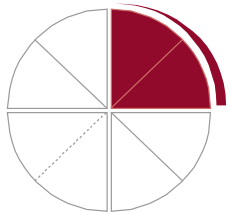
- **Availability** Requirements
  - AR1: unavailability (downtime) of nodes must be less than 60 minutes per year.



Performance Scenarios

- **Performance** related scenarios:
  - **SC4:** Each external node sends a request and receives the requested information in reasonable time.
  - **SC5:** Each external node receives periodic updates for the requested information at a specified rate.
  - **Contextual Parameter:** node
  - **Metrics:** wait time, periodic update time
- **Performance** Requirements
  - PR1: Each external node must receive the response *within F seconds* of sending a request. E.g. *F=10* seconds.
  - PR2: Given that an external node *i* has requested a periodic update every *T(i)* seconds, it must receive an update *on the average every T(i) seconds*. E.g. *T(i) = 120* seconds.
  - PR3: The interval between receipt of consecutive periodic updates must be *not more than 2T(i) seconds*

Security Scenarios



- **Security** related scenarios:
  - **SC6:** A threat object between the serving and requesting node modifies data received by the requesting node.
  - **SC7:** A serving node is spoofed to misrepresent queries received by the requesting nodes.
  - **Contextual Parameter:** node
  - **Metrics:** corruption probability
- **Security** Requirements
  - SR1: < 0.001 probability of served data corrupted before they arrive at the requesting node.

What to Do Next?

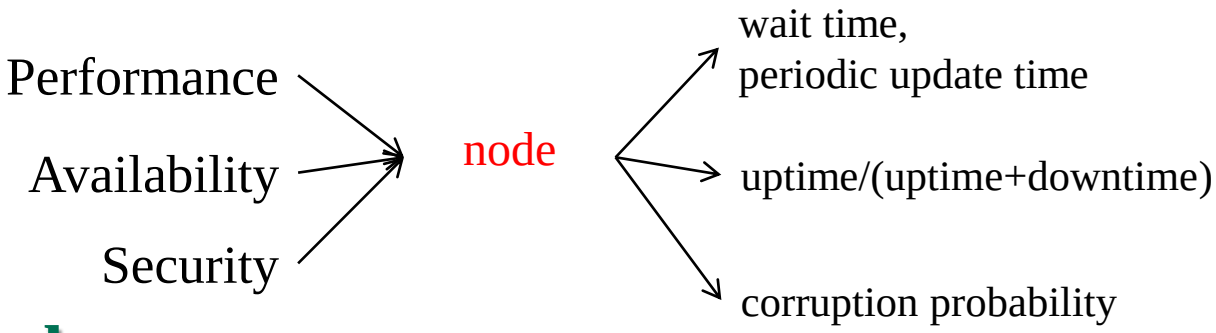
- Sort scenarios based on their priority, for each scenario ask:
  - Is this scenario important for success of system? (H, M, L)
  - Will there be serious problem if it is not addressed? (H, M, L)
- Check your architectural views and explain whether the higher priority scenarios have been addressed there or not.

- **Example: Availability Scenarios**
  - Can your design handle soft/hardware failures? **(SC1)**
  - Can your design handle power supply failures? **(SC3)**
  - If yes, give the reason → record as "non-risk"
  - If no, propose a remedy → record as "risk", identify new issue, devise strategy to handle it.

→ e.g. Add redundancy

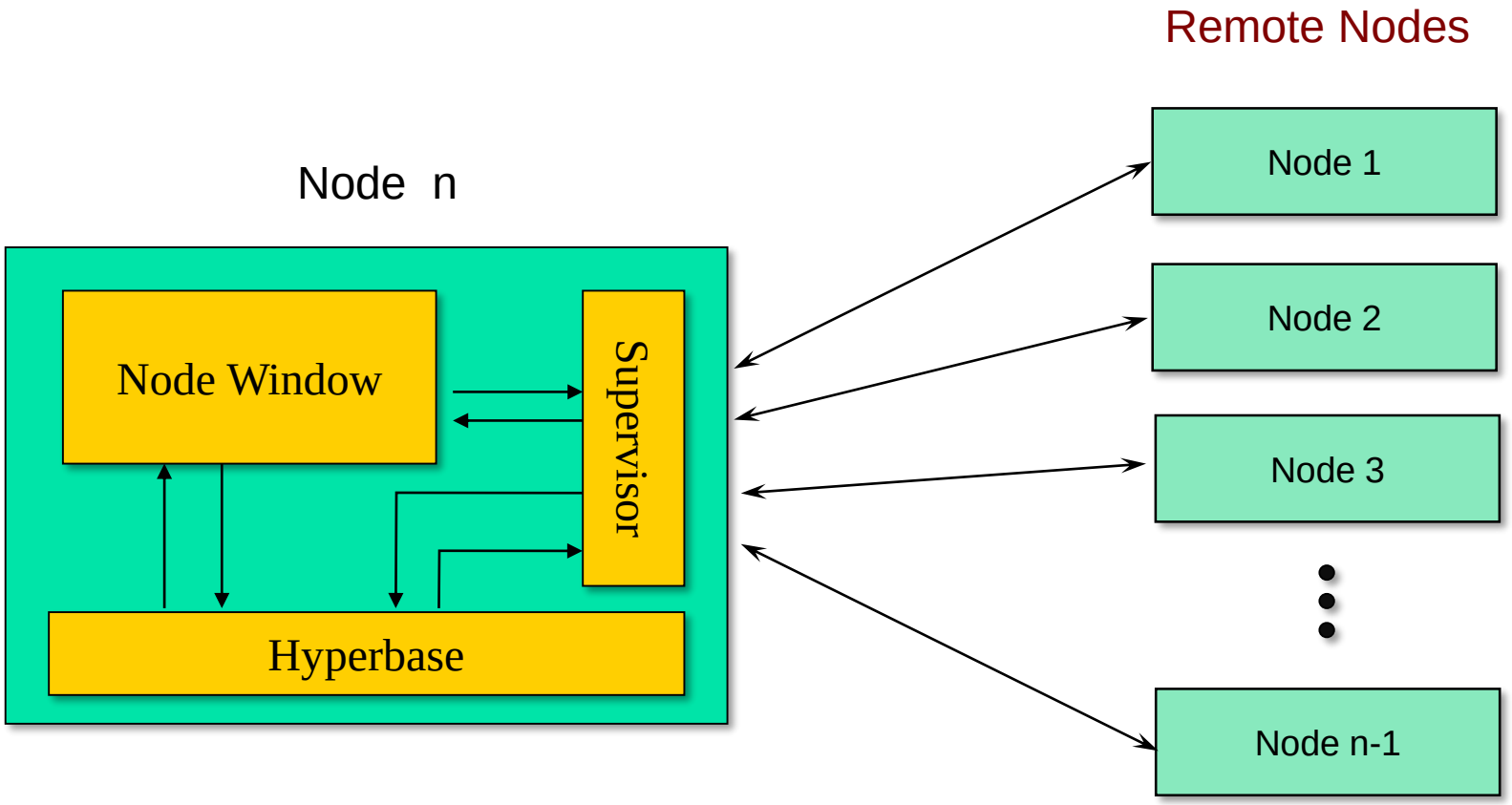
What to Do Next?

- You may find strategies to handle individual quality attributes or a subset of them operating on a "trade-off" contextual parameter (e.g. node)



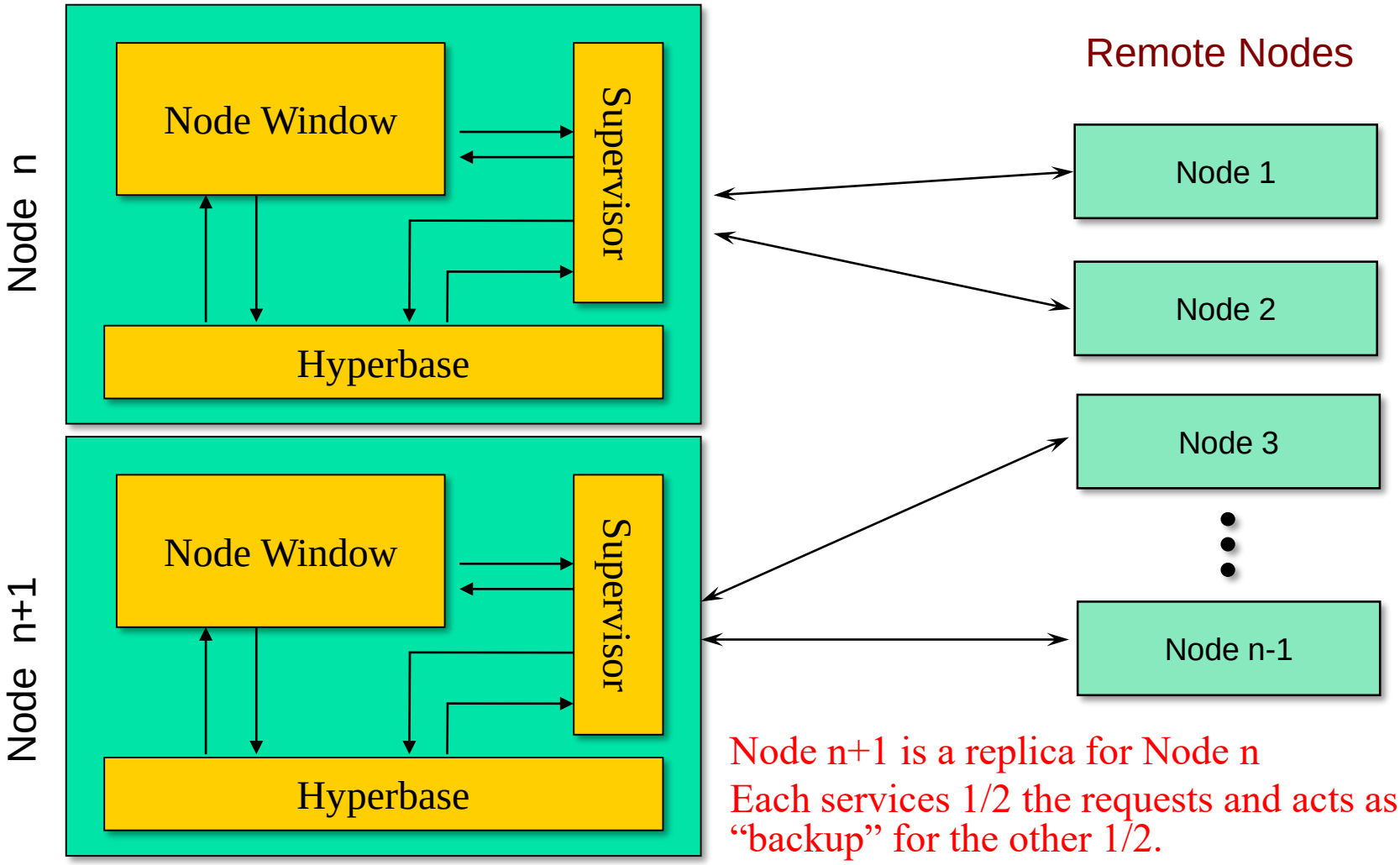
- **Example:**
  - Strategies that can handle availability and performance regarding the trade-off parameter "node": add ***node redundancy*** (to improve availability); add ***cache*** (to boost performance)

Option 1: Client-Server



Initial system configuration

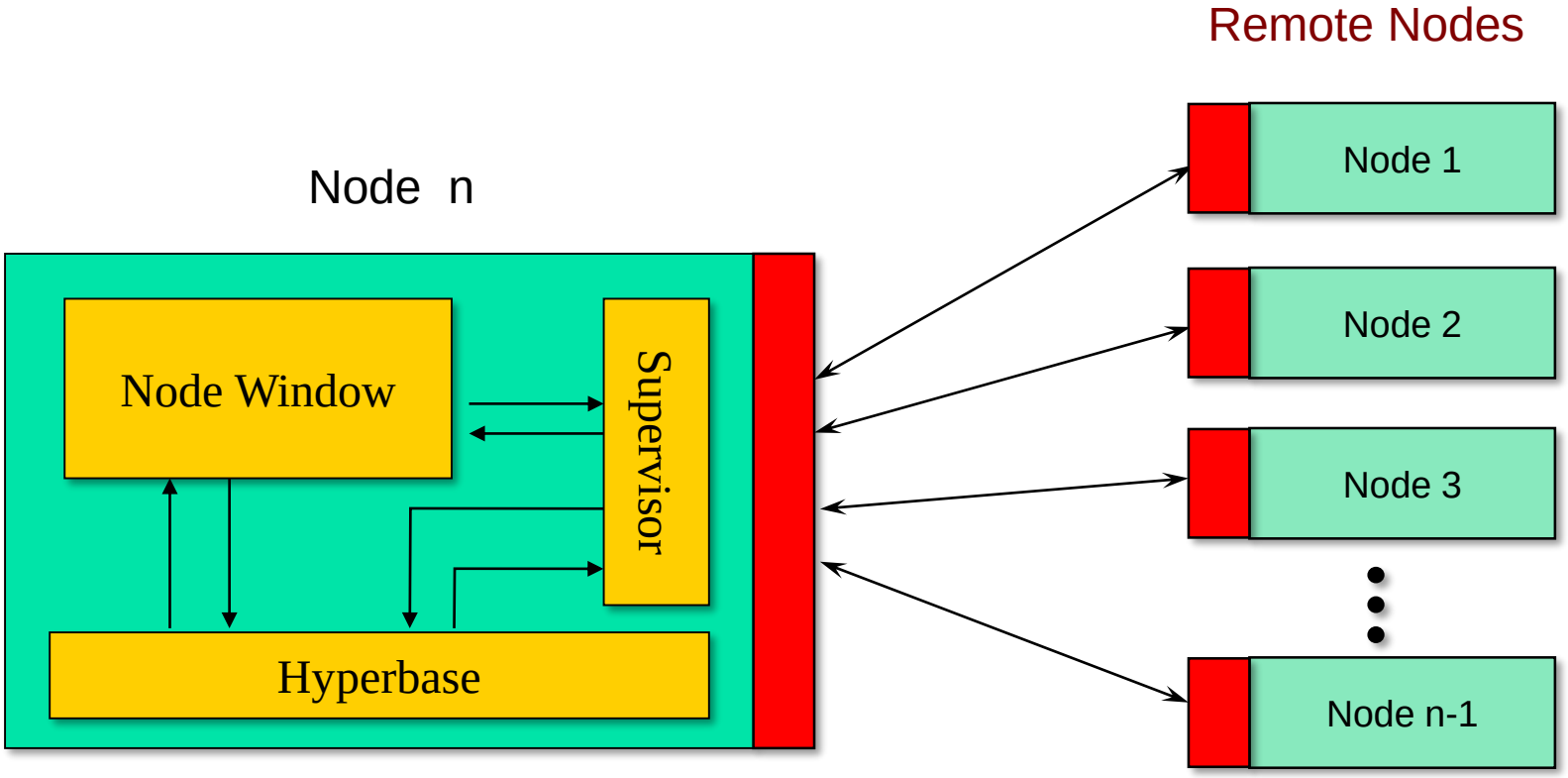
Option 2: Multiple-Servers



Node n+1 is a replica for Node n

Each services 1/2 the requests and acts as "backup" for the other 1/2.

Option 3: Clients + Cache



Nodes are equipped with a cache that stores requests and updates
(Note: push and pull protocols to grab data must be modified)

Analysis

- Perform analysis for the quality attributes: performance, availability, etc.

- If possible, conduct an experiment

  (e.g. using a prototype)

- If not, perform a comparative analysis

Performance Assumptions

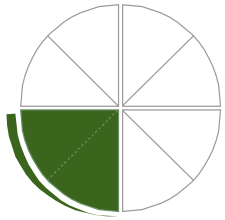
- To do analysis, assumptions *must* be made, e.g.:
  - Relatively infrequent requests
  - Requests are not dropped
  - No message priorities
  - Network communication time between client and server (C_net = 1200 ms)
  - Server latency = task computation (C_fnc = 120 ms)

Performance Analysis

Option 2 has the best performance

	Roundtrip req. max	*Roundtrip req. min*
Option 1	41,120 ms	5,100 ms
Option 2	20,560 ms	2,550 ms
Option 3	42,000 ms	5,200 ms

Availability Assumptions



- For each option we assume a set of failure rates and repair rates:
  - Node failure rates: between 0 and 24 failures per year
  - Repair rates: 1/2 day (service call), 10 minutes (reboot)

Availability of Option 1

1/2 Day Repair

Failures/year	Availability	Hours down/year
8	0.98916	94.959
16	0.97855	187.882
24	0.96817	278.833

10 Minute Repair

Failures/year	Availability	Hours down/year
8	0.99985	1.33313
16	0.9997	2.66586
24	0.99954	3.9982

None can meet availability requirement!

Availability of Option 2

1/2 Day Repair

Failures/year	Availability	Hours down/year
8	0.99977	2.0585
16	0.99908	8.0556
24	0.99798	17.7327

10 Minute Repair

Failures/year	Availability	Hours down/year
8	1.0	0.00040576
16	1.0	0.00162255
24	1.0	0.0036496

½ day repair cannot meet availability requirement!

Availability of Option 3

(1 Min Cache)

None can meet availability requirement!

1/2 Day Repair

Failures/year	Availability	Hours down/year
8	0.98917	94.828
16	0.97858	187.621
24	0.96821	278.446

10 Minute Repair

Failures/year	Availability	Hours down/year
8	0.99986	1.21194
16	0.99972	2.4235
24	0.99959	3.6347

Availability of Option 3

(5 Min Cache)

Almost all cannot meet availability requirement!

1/2 Day Repair

Failures/year	Availability	Hours down/year
8	0.98923	94.304
16	0.9787	186.586
24	0.96839	276.91

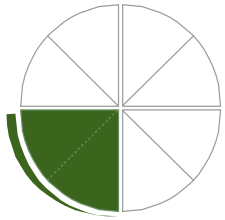
10 Minute Repair

Failures/year	Availability	Hours down/year
8	0.9999	0.88875
16	0.9998	1.77724
24	0.9997	2.66545

Initial Results

- Option 1: poor performance and availability; least expensive.
- Option 2: best availability, at extra cost; best performance (when both nodes running); performance of Option 1 otherwise.
- Option 3: slightly better availability than Option 1; slightly worse performance than Option 1, slightly greater cost than Option 1; lower cost than Option 2.

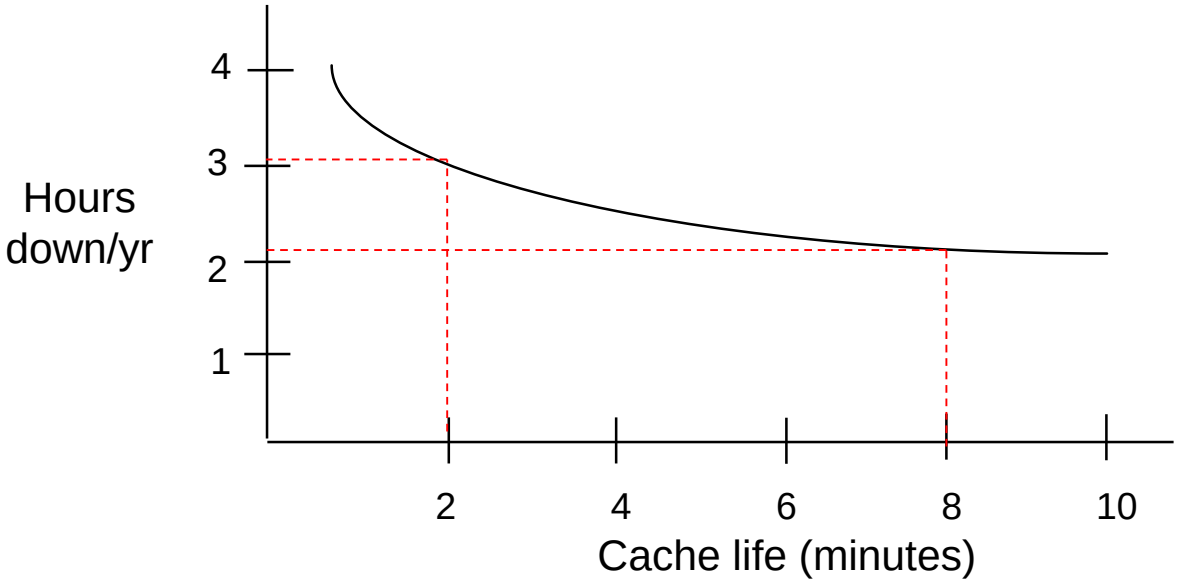
Sensitivity Analysis /1



- **Question:** What is the *sensitivity* of attributes to various architectural parameters?

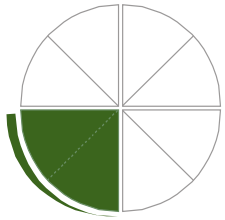
- e.g. Given that Option 3 has some desirable characteristics, we might ask if we can improve availability by improving the cache.
- To answer this, we may want to calculate the architectural sensitivity of availability to cache life.

Sensitivity Analysis /2



This shows that availability is relatively *insensitive* to parameter " cache life"
Given this, very little better availability can be achieved if cache life time is increased.

Sensitivity Analysis /3



- What is the sensitivity of the architecture to number of serving nodes?

- Performance: +
- Availability: +
- Cost: ++

- *Availability* and *Performance* are sensitive to number of serving nodes.

Final Summary

- No design completely meets all requirements ← potential risk
- ***Option 2*** has a lower level of service when one server fails
- In ***Option 2*** increasing the number of server nodes from 1 to 2 improves availability and performance, but at rather higher cost
- ***Option 3*** does not offer that much over ***Option 1*** in terms of performance and availability

- Final recommendation: ***Option 1*** seems to be the better option among the three

- Note: this decision does not reflect security related scenarios and we should do similarly for "security" vs. availability and performance

Benefits

- Clarifies quality attribute requirements
  - Through prioritization and estimation
- Improves documentation
  - Through explicit enumeration of evaluation criteria, rankings, and rationale for decisions
- Identifies risk areas
  - Points of sensitivity, instability
- Improves communication
  - Highlights were stakeholders differ about relative priorities
  - Causes stakeholders to come to some consensus

Observations

- Inexpensive way to find problems
  - Compared with “patch and pray”
  - May be done early in life cycle
- Goal is to:
  - Make informed design choices, informed tradeoffs
  - Analyze systems with respect to *multiple* attributes at the level of their architecture
- Numbers are approximate
  - Initially the numbers are ballpark estimates.
  - Variable resolution analysis--probe deeper when needed through: benchmarking, prototyping
  - Numbers get refined over time but the analytic framework persists.

ATAM: Summary (1)

- ATAM helps when
  - Buying time to think about an architecture while development processes are often under time-pressure
  - Identification of risks early in the life-cycle
  - Understanding the interactions of the requirements (i.e. trade-off and sensitivity points).
- ATAM Forces stakeholders to:
  - Think about quality requirements
  - Prioritize quality requirements
- Results in improved architectures

ATAM: Summary (2)

- ATAM relies critically on:
  - Appropriate preparation by the customer
  - Clearly-articulated quality attribute requirements
  - Active stakeholder participation
  - Active participation by the architect
  - Evaluator familiarity with architectural styles and analytic models
  - Subjective judgment that depends on the experience of the participants

ATAM: Summary (3)

- ATAM can be applied early in the software development process ← positive advantage
- ATAM process typically takes three days and the involvement of 10-20 people – evaluators, architects, customers and other system stakeholders.
- ATAM has been used in several organizations and has seen benefits including:
  - clarified quality attribute requirements
  - improved architecture documentation
  - documented basis for architectural decisions
  - identified risks early in the life-cycle
  - increased communication among stakeholders