

EVALUATING SOFTWARE ARCHITECTURES

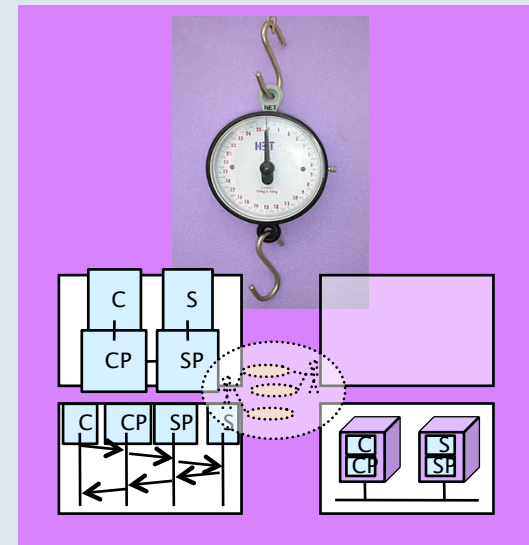
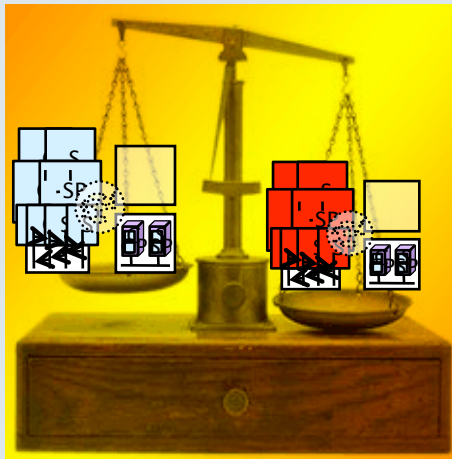
If you haven't analyzed it, don't build it.

M.R.V. Chaudron

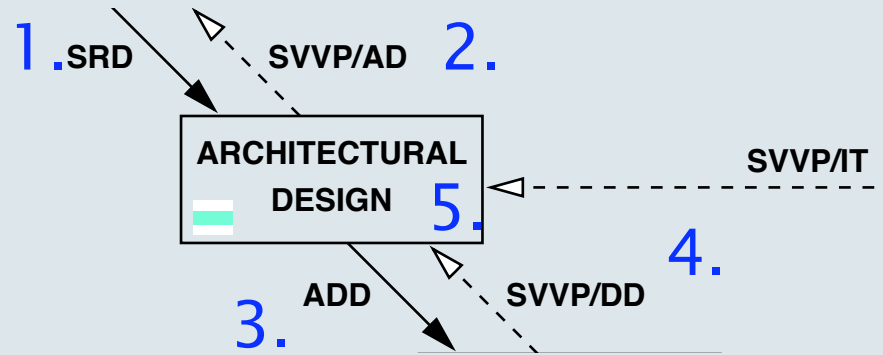
Technische Universiteit Eindhoven

Adapted by Tom Verhoeff for 2II45 in 2009

With slides from Rick Kazman



Topics in Part 2



1. From Req. to Arch.: Doing Design
2. From Arch. to Req.: Doing Evaluation
3. From Arch. to Code: Doing Implementation, code generation, testing infrastructure, code configuration management
4. From Code to Arch.: Monitoring impl. work, Reverse Engineering, Integration
5. Process, Documentation, Tools, Standards

System Quality Attributes (Extra-Func. Req.)

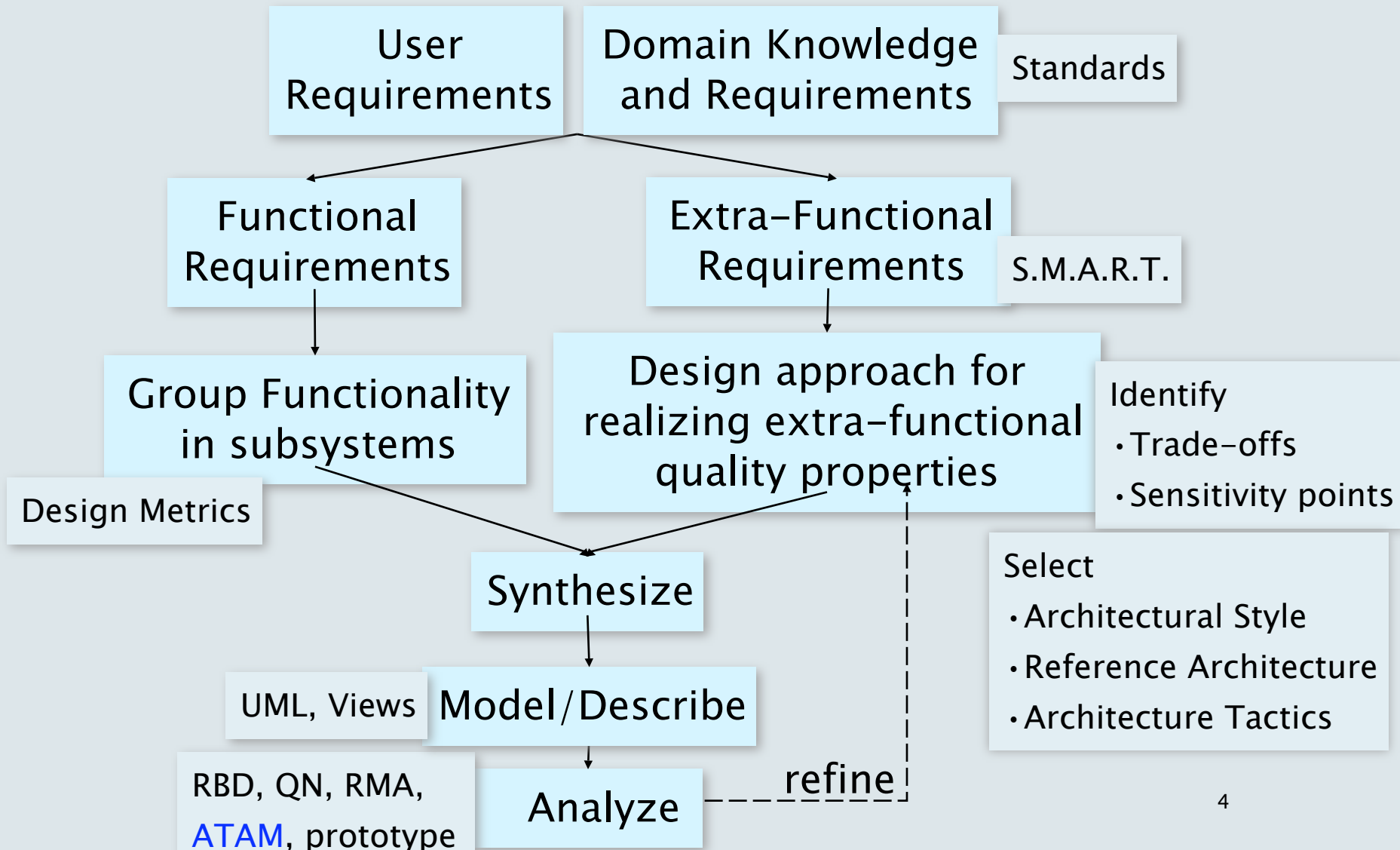
- Time To Market
- Cost and Benefits
- Projected life time
- Targeted Market
- Integration with Legacy System
- Roll back Schedule

Business
Community
view

- Performance
 - Availability
 - Usability
 - Security
- End User's
view

- Maintainability
 - Portability
 - Reusability
 - Testability
- Developer's
view

Design of Software Architecture



Why analyze architecture?

- In the majority of projects, the only model available for measurement is the final implementation

This is far too late and causes excessive costs and risks

- Every design involves **tradeoffs**

A software architecture is the earliest life-cycle artifact that embodies significant design decisions with **high risks**

Heuristic (to avoid common pitfall)

Don't evaluate
what can be done easily.

Do evaluate
what you need to know!

Types of Analysis

Quantitative: How much ...?

- Analysis based on (mathematical) model
- Measurements
 - Feasibility prototypes
 - (Process) models
 - Simulation

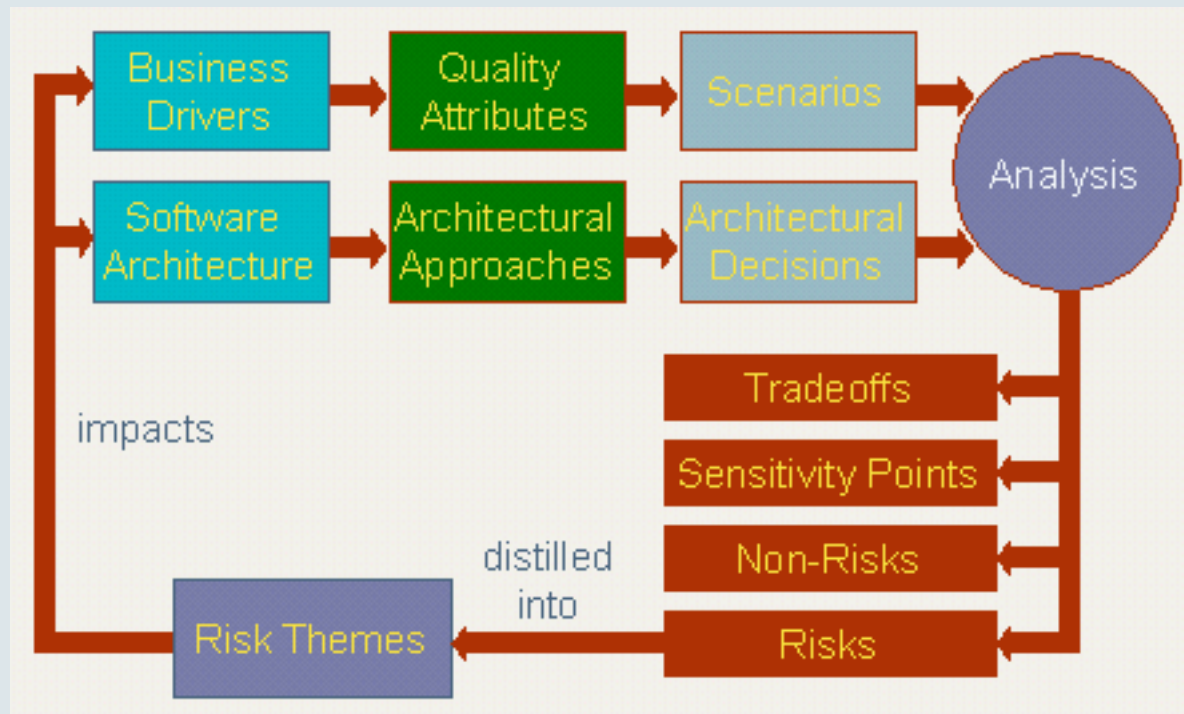
Qualitative: What if ...?

- Architecture Trade-off Analysis Method (ATAM),
- Cost-Benefit Analysis Method (CBAM), ... 7

Architecture Tradeoff Analysis Method (ATAM): Origin

- **Software Engineering Institute (SEI)** at **Carnegie Mellon University (CMU)**
- Comparable to LaQuSo at TU/e
- **Consulting** role for (multi-party) projects
- Need to evaluate architectural designs independent of how they were created
- Evaluate and report in a standardized format

Architecture Tradeoff Analysis Method (ATAM): Overview



ATAM Reference

- “[The Architecture Tradeoff Analysis Method.](#)” Article '98.
Rick Kazman et al. [Ex.: Remote Temperature Sensor System]
- [ATAM: Method for Architecture Evaluation](#),
Rick Kazman, Mark Klein, Paul Clements, August 2000,
Technical Report CMU/SEI-2000-TR-004
[Ch. 9 & Appendix optional. Ex.: Battlefield Control System]
- Chapter 11 from the BCK book [Optional. Ex.: Nightingale]

ATAM Purpose

- Evaluate whether the design decisions satisfactorily address the quality requirements.
- Elicit **rationale** of design decisions (traceability).
- Discover **risks**: decisions 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, in opposite direction.

ATAM Output

- Precise description of the architecture
- Articulation of the business goals
- Quality requirements in terms of Qu. Attr. Scenarios
- Relation between business goals and architecture tactics (the rationale of the design)

+

- Risks
- Sensitivity points
- Tradeoff points



ATAM Qualifications (Reservations)

Result depends on the quality of the specification of the architecture

- garbage in, garbage out

Not an attempt to predict resulting quality attributes

Quality properties that are not easily expressed quantitatively, such as usability, interoperability, ...

ATAM Side-effect

- Improve the architecture documentation.
 - Elicit/make precise a statement of the architecture's driving quality attribute requirements
- Process benefit:
 - Foster stakeholder communication & consensus

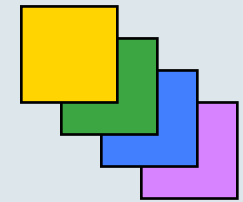


ATAM Preconditions

The 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

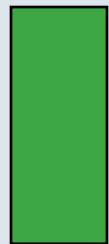
ATAM STEPS



These slides
by Rick Kazman



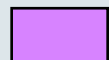
1. Explain the ATAM
2. Present business drivers
3. Present architecture



4. Identify architectural approaches
5. Generate quality attribute utility tree
6. Analyze architectural approaches

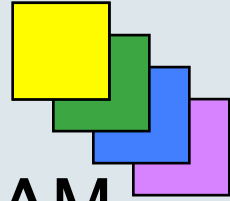


7. Brainstorm and prioritize scenarios
8. Analyze architectural approaches



9. Present results

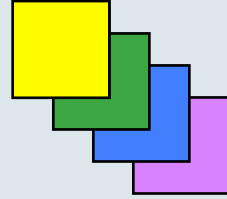
1. PRESENT THE ATAM



Evaluation Team presents an overview of ATAM

- ATAM steps in brief
- Techniques
 - Utility tree generation
 - Architecture elicitation and analysis
 - Scenario brainstorming
- Outputs
 - Architectural approaches
 - Utility tree
 - Scenarios
 - Risks and “non-risks”
 - Sensitivity points and tradeoffs

2. PRESENT BUSINESS DRIVERS

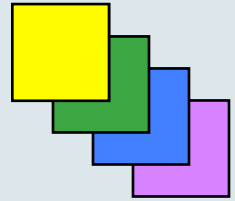


ATAM customer representative describes the system's business drivers including:

- Business context for the system
 - Time to market
- Most important functional requirements
- Most important quality attribute requirements
 - Architectural drivers:
 - Quality attributes that “shape” the architecture
 - Critical requirements:
 - Quality attributes most central to the system's success
 - High availability, high security, ...

Understand the requirements

3. PRESENT ARCHITECTURE



- Architect presents an architecture overview incl:
 - Technical context of the system
 - systems with which the system must interact
 - Technical constraints such as an OS, hardware, or middleware prescribed for use
 - Architectural approaches/styles used to address quality attribute requirements
- Evaluation team begins probing for and capturing risks

Understand the
architecture

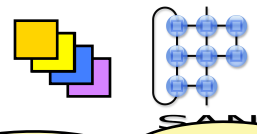
4. IDENTIFY* ARCHITECTURAL APPROACHES

- Start to identify parts of the architecture that are key for realizing quality attribute goals
- Identify any predominant architectural styles, tactics, guidelines & principles
- Examples:
 - 3-tier Client-server
 - Watchdog, Redundant hardware



Understand the
architecture

*approaches are not yet analyzed



5. GENERATE QUALITY

ATTRIBUTE UTILITY TREE

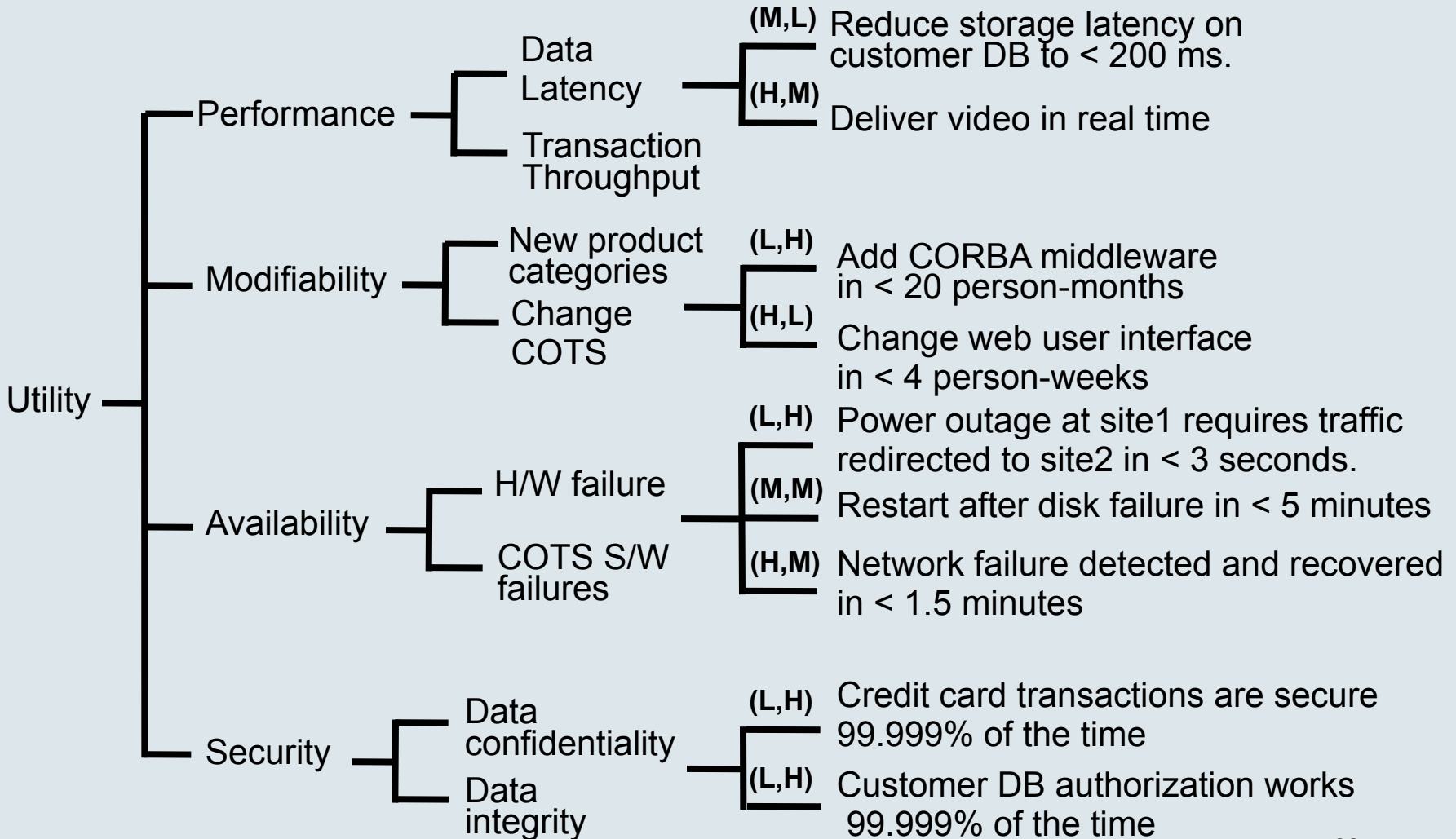
Prioritize requirements

- 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 (e.g. performance, modifiability, security, availability)
 - Scenarios are the leaves of the utility tree
- Output: A characterization and a prioritization of specific quality attribute requirements

EXAMPLE UTILITY TREE

(Importance, Achievability)

H-High, M-Medium, L-Low



SCENARIOS

- Scenarios are used to:
 - Represent stakeholders' interests
 - Understand quality attribute requirements
- Scenarios should cover a range of:
 - Use case scenarios
 - anticipated uses
 - Evolution scenarios
 - anticipated changes; e.g. growth
 - Exploratory scenarios
 - unanticipated stresses to the system

EXAMPLE SCENARIOS

- Use case scenario
A remote user requests a database report via the Web during peak period and receives it within 5 seconds
- Evolution scenario
Add a new data server during peak hours within a downtime of at most 8 hours.
- Exploratory scenario
Half of the servers go down during normal operation without affecting overall system availability

STIMULI–ENVIRONMENT–RESPONSES

‘Formula’ for scenarios

- Use case (performance) 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 during peak hours within a downtime of at most 8 hours.
- Exploratory scenario
Half of the servers go down during normal operation without affecting overall system availability

A good scenario makes clear what the stimulus is and what the measurable response of interest is

General Qual. Attr. Scenario for Scalability

- **Source**: system owner
- **Stimulus**: request to accommodate more concurrent users (**usage parameter**)
- **Artifact**: the system, incl. computing platforms
- **Environment**: normal operation, design/run time
- **Response**: add extra memory/servers (**architectural parameters**)
- **Response measure**: cost of additional hardware, change in performance

Specific Qual. Attr. Scenario for Scalability

- **Source**: system owner
- **Stimulus**: request to accommodate five times more concurrent users over next two years
- **Artifact**: the main server cluster
- **Environment**: normal operation
- **Response**: increase number of servers no more than sixfold, without recompiling the software
- **Response measure**: performance as measured by average number of typical requests processed per minute may not drop more than 10%

SAAM*: Rank Architectures

Summary of suitability



Architecture	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Contention
X	0	0	-	-	-
Y	0	0	+	+	+
Z	-	+	0	+	-

* Software Architecture Analysis Method

6. ANALYZE ARCHITECTURAL APPROACHES

Analyze the Architecture

The evaluation team probes architectural approaches w.r.t. specific quality attributes to identify risks

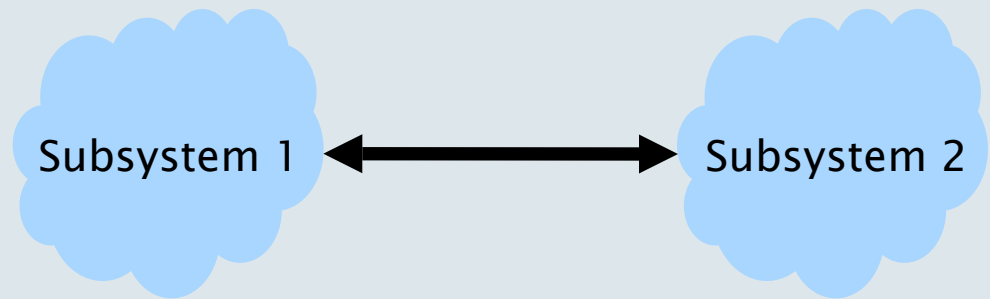
- Identify the approaches that pertain to the highest priority quality attribute requirements
- Generate quality–attribute specific questions for highest priority quality attribute requirement
- Ask quality–attribute specific questions
- Identify and record risks and non–risks, sensitivity points and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

Sensitivity Point


Sensitivity point is a parameter of the architecture to which some quality attribute is highly related.

A system requires

- high performance



Suppose throughput depends on one channel

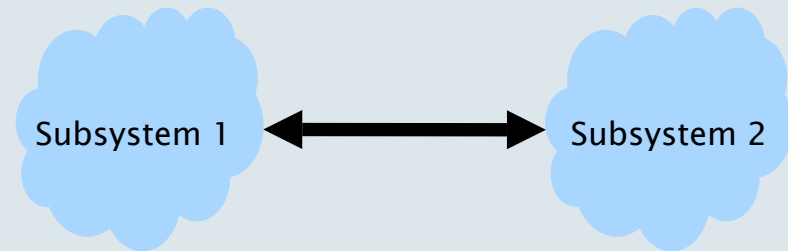
increase channel speed → increase performance 

Trade-off point

A **trade-off point** is a parameter of the architecture that affects multiple quality attributes in opposite directions.

A system requires

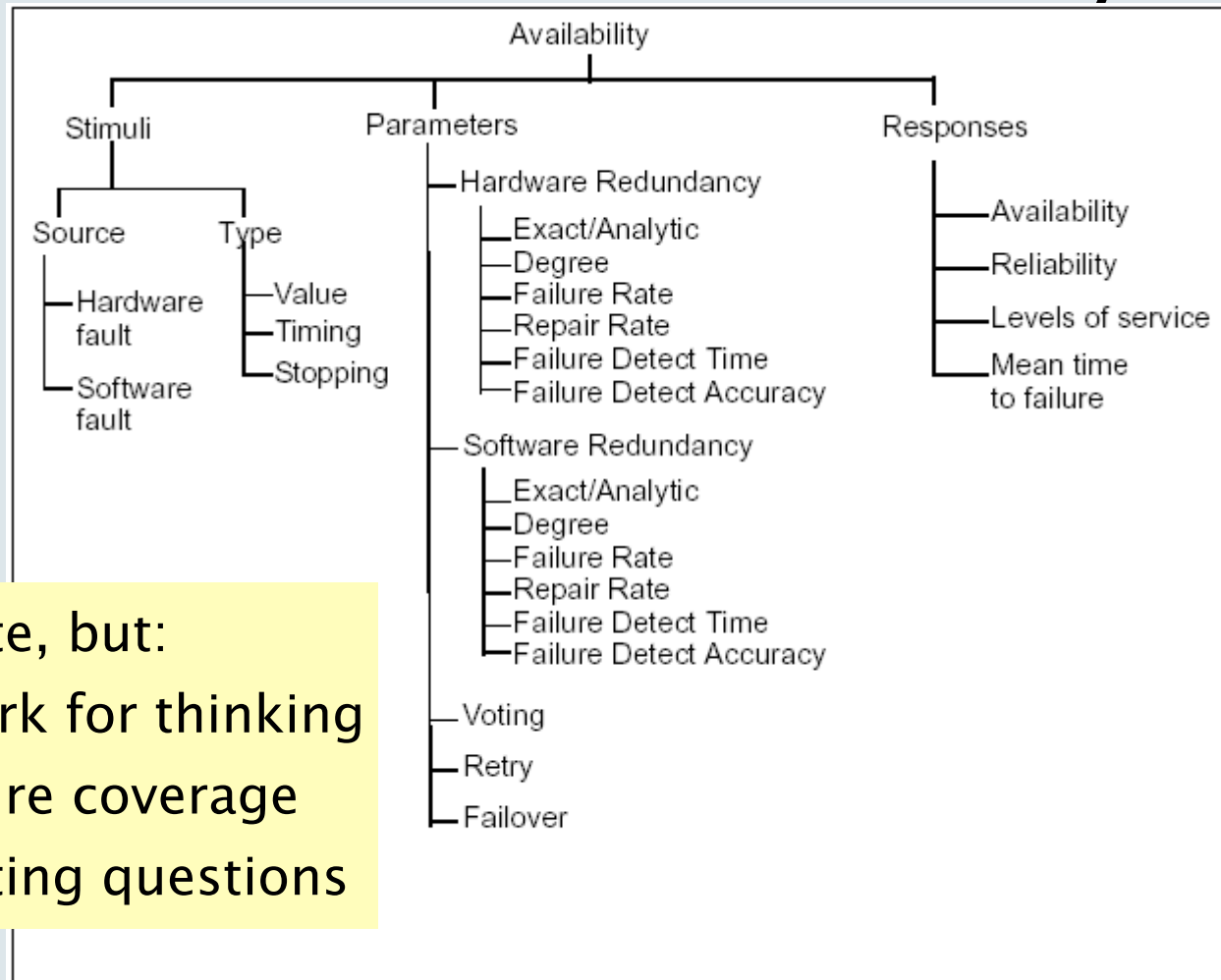
- high performance,
- **high reliability**
- **high security**



QUALITY ATTRIBUTE QUESTIONS

- Quality attribute questions elicit architectural decisions which bear on quality attribute requirements
- Example: Performance
 - How are priorities assigned to processes?
 - What are the message arrival rates?
- Example: Modifiability
 - Are there any places where layers/facades are circumvented ?
 - What components rely on detailed knowledge of message formats?

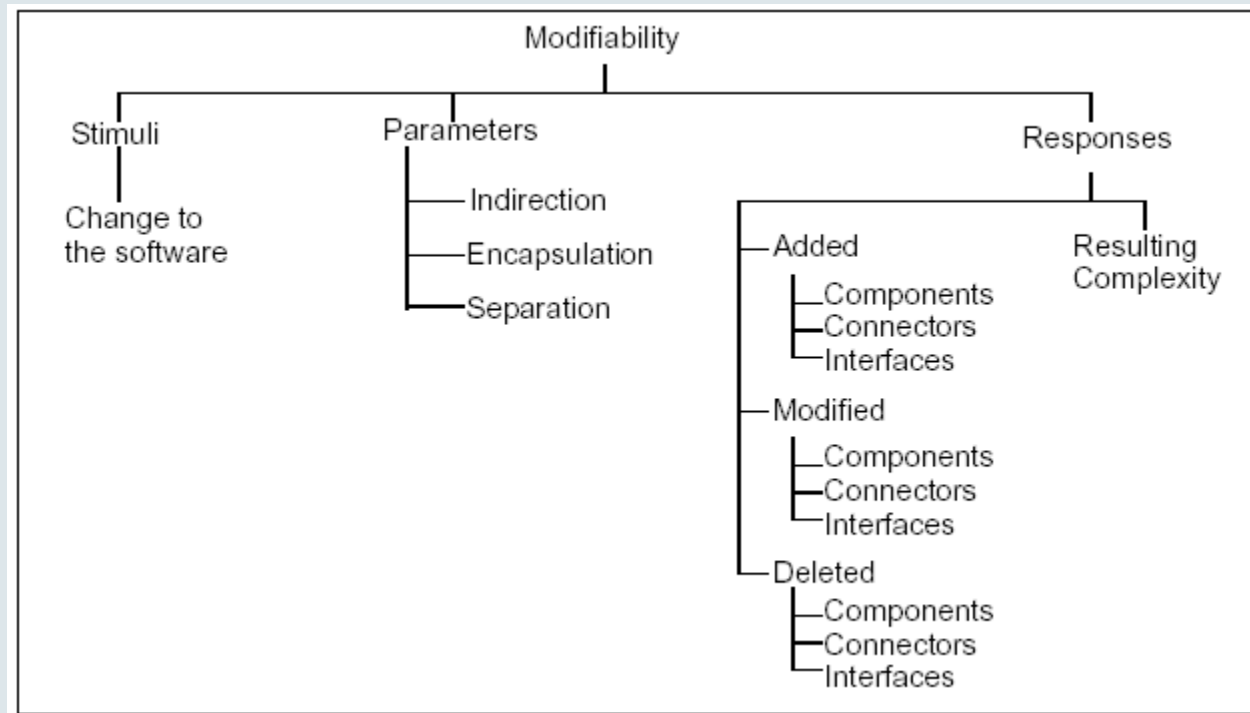
Characterization of Availability



Not Complete, but:

- a framework for thinking
- helps ensure coverage
- helps eliciting questions

Characterization of Modifiability



RISKS and NON-RISKS

- **Risks** are potentially important architectural decisions that may cause problems
- **Non-risks** are good decisions frequently relying on implicit assumptions
- Risk and non-risk constituents
 - Architectural decision
 - Quality attribute requirements
 - Rationale
- **Sensitivity points are candidate risks**

EXAMPLE RISK and NON-RISK

- **Example risk**

Rules for writing business logic modules in the second tier of your 3-tier style are not clearly articulated.

This could result in compromising modifiability.

- A quality concerns is not addressed / has not been analyzed
- Some tactics interact in an unknown manner

- **Example 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, a 1 second soft deadline seems reasonable

- Risk may also occur in the **project organization**.

Engineer X has never heard of requirement Y.

7. BRAINSTORM AND PRIORITIZE SCENARIOS

Stakeholders generate scenarios using a facilitated brainstorming process

- Examples are used to facilitate the step
- The new scenarios are added to the leaves of the utility tree

Essentially a process step:

- include a larger group of stakeholders
- extend consensus (esp. on priorities)
- extend confidence in completeness of scenario's

8. ANALYZE ARCHITECTURAL APPROACHES

- Identify the architectural approaches impacted by the scenarios generated in the previous step
- This step continues the analysis started in step 6 using the new scenarios
- Continue identifying risks and non-risks
- Continue annotating architectural information

Essentially a process step:

- include a larger group of stakeholders
- extend consensus
- extend confidence in completeness of scenario's

9. PRESENT RESULTS

- Recapitulate steps of the ATAM
- Present ATAM outputs
 - Architectural approaches
 - Utility tree
 - Scenarios
 - Risks and “non-risks”
 - Sensitivity points and tradeoffs

ATAM BENEFITS

- ATAM ‘buys’ time to think about an architecture while development processes are often under time–pressure
- Identification of **risks** early in the life–cycle
- Focuses on features that are essential for the stakeholders and not on technical details
- Improved architecture documentation
 - Forces stakeholders to:
 - think about qualitative requirements
 - prioritize qualitative requirements
- Documented basis for architectural decisions

The results are improved architectures

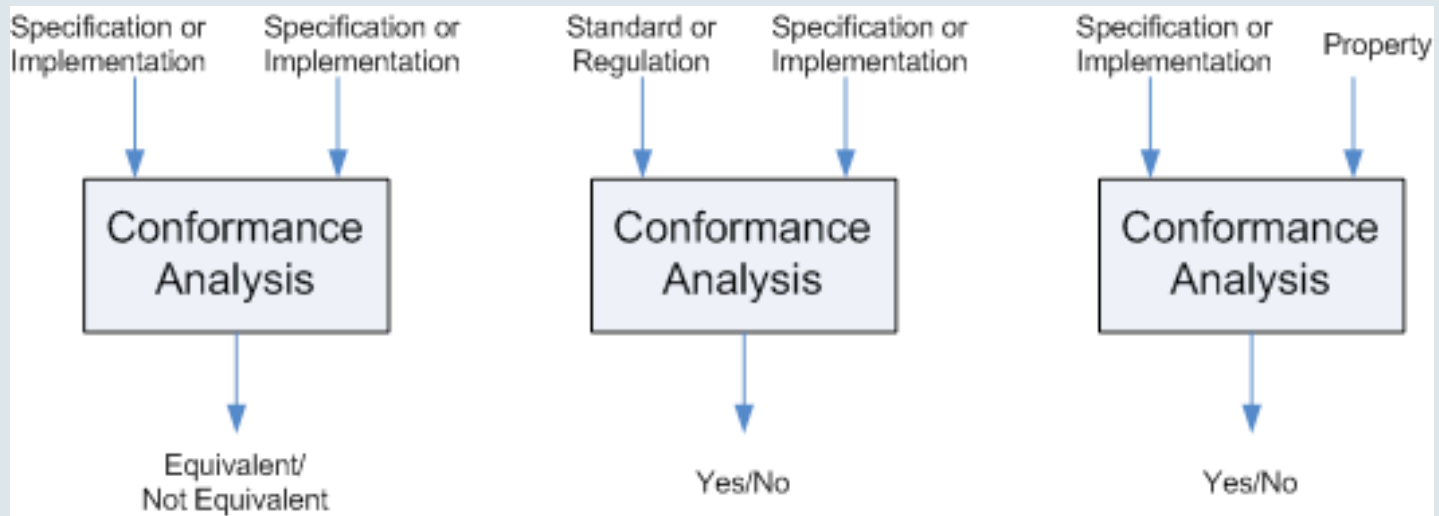
ATAM – Cost/Benefit

- Cost
 - 1 – 2 weeks of time for 8 – 10 highly paid people, 2 days for another 10–12 people (for full formal process!)
 - Delays project start
 - Forces development of architecture up front
- Benefit
 - Financial – saves money
 - Forces preparation / documentation / understanding
 - Captures rationale
 - Catch architectural errors before built
 - Make sure architecture meets scenarios
 - More general, flexible architecture
 - Reduces risk

ATAM WEAKNESSES

- Subjective judgement
that depends on the experience of the participants
- No guidelines for definition of useful change cases
- Risk: check–list thinking

LaQuSo Software Product Certification Model



- Consistency, Functional, Behavioral, Quality, Compliance
- Certification Criteria: Formality, Uniformity, Conformance
- 6 Product Areas: Context Description, User Requirements, **High-level Design**, Detailed Design, Implementation, Tests
- Results in an Achievement Level

Relationship between design & evaluation

- Why (not) use the same method for design and evaluation?
- Should not only design to pass the evaluation; evaluation is often limited
- If evaluation includes important considerations, then these should also have played a role in design
- Evaluation is part of design, not an add-on