CSSE 374: Architectural Analysis

Shawn Bohner

Office: Moench Room F212

Phone: (812) 877-8685

Email: bohner@rose-hulman.edu

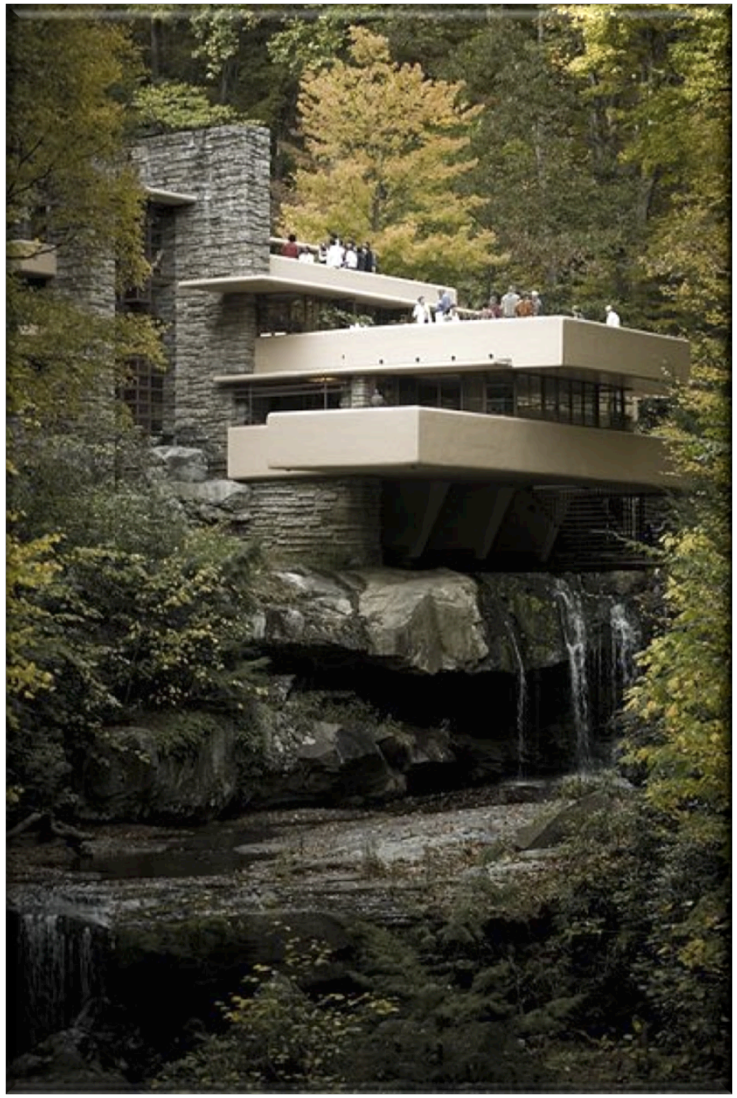
Q1

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Learning Outcomes: Analysis of Design

Analyze and explain the feasibility & soundness of a software design.

- Introduce Architectural Analysis
- Discuss SAAM, an Example Architectural Analysis approach
- Do an exercise… if time

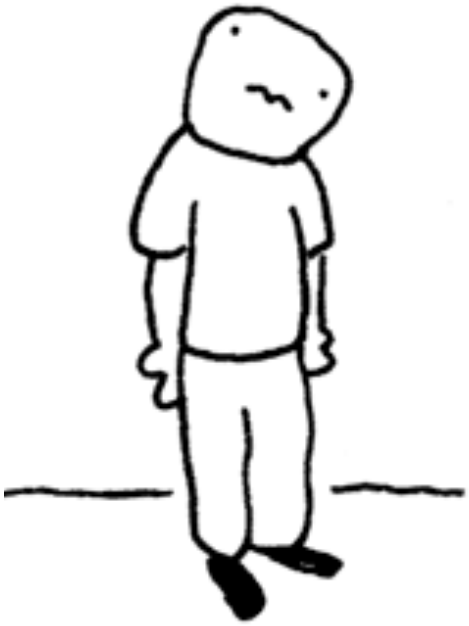


http://en.wikipedia.org/wiki/File:Wrightfallingwater.jpg

Beyond "analyzing architectures," what do you think Architectural Analysis is about?

- Think for 15 seconds…
- Turn to a neighbor and discuss it for a minute

Change is hard… Architectural Analysis just indicates how hard!



When asked "would you rather work for change, or just complain?" 81% of the respondents replied, "Do i have to pick? This is hard."

Recall Architectural Building Blocks

Component – a unit of computation or a data store (either atomic or composite)

Connector – an architectural element that models interactions among components and rules that govern those interactions

Configuration (or topology) – a connected graph (composite) of components and connectors which describe architectural structure

Architectural Analysis

The identification and resolution of the system's non-functional requirements (e.g., security, maintainability) in the context of functional requirements (e.g., calculate trajectory, generate report)

Goals of Architectural Analysis

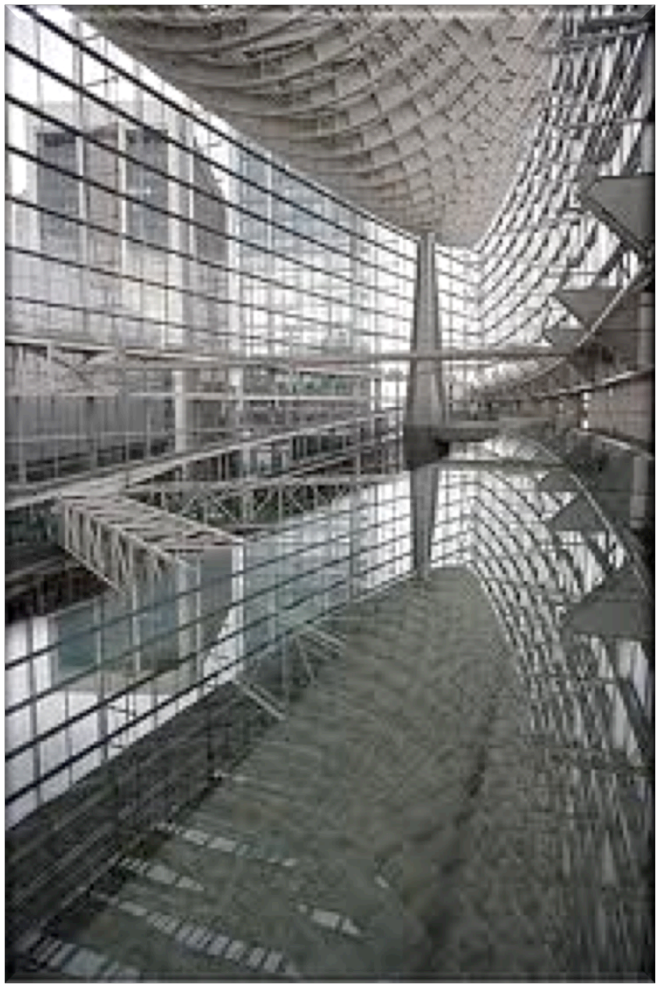
- Identify and resolve non-functional requirements
- Identify variation points
- Identify most probable evolution points
- Hierarchy of Decision Goals
  - Inflexible constraints (e.g., safety or legal)
  - Business goals (they pay the money…)
  - All others

Why do Architectural Analysis?

- Reduce risk of missing something key to the design of the system
- Avoid applying excessive effort to low priority issues
- Help align the software product with the business (or system) goals

When do we Analyze the Architecture?

- Before first iteration, to manage *risk*
- In elaboration and after each design iteration
  - Act as ‘toll-gate’ before starting next phase



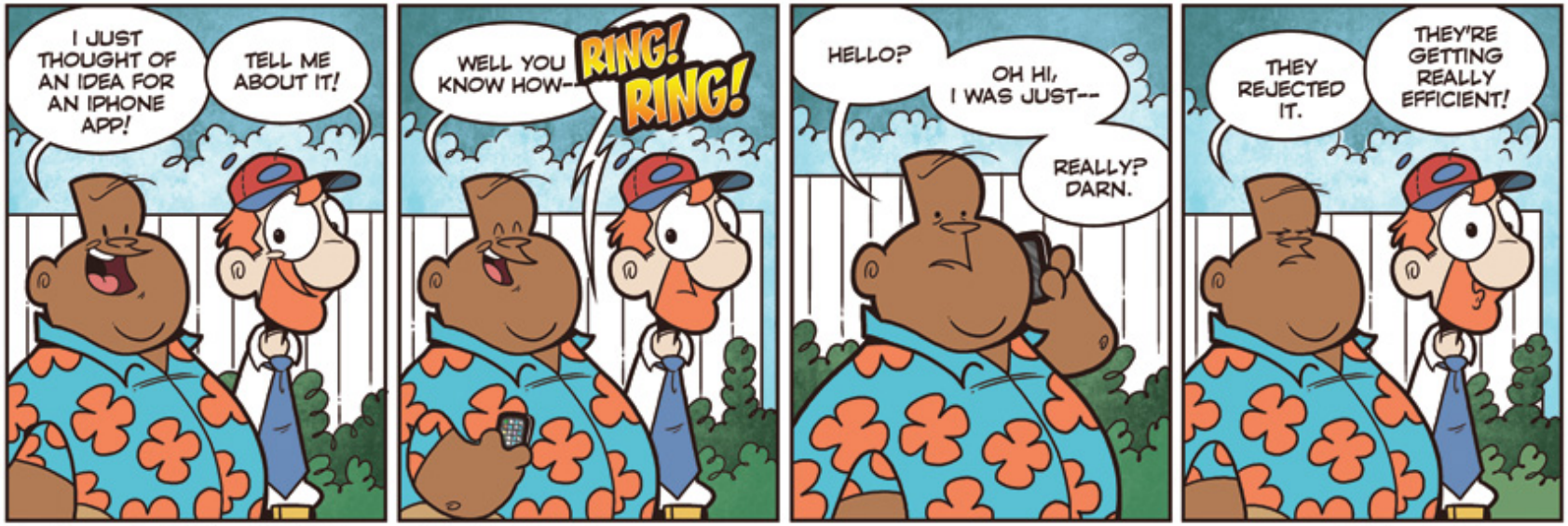
Recall Points of Change from Protected Variation Discussion…

- Key aspect of architectural analysis is determining the *variant* and *invariant* elements of the architecture

- **Variation points:** points of change *in the existing system or requirements*
  - e.g., multiple tax calculators

- **Evolution points**: points of change that *may arise in the future* but not currently present
  - e.g., hand-held POS devices

Efficient Analysis



Used by permission. http://notinventedhe.re/on/2009-11-23

Common Steps in Architectural Analysis

1. Identify and analyze non-functional requirements (AKA architectural drivers or factors) that impact architecture

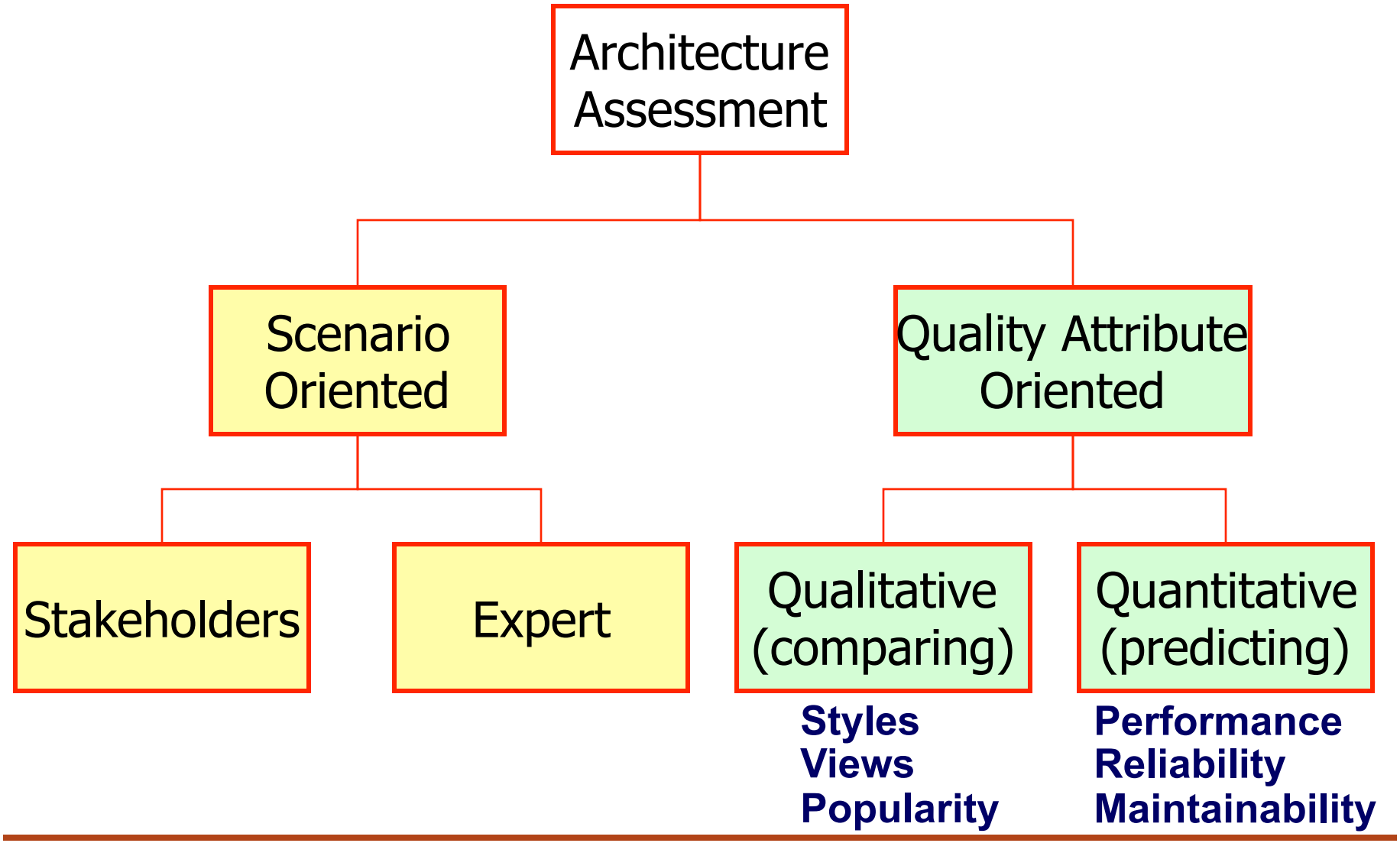
2. Evaluate alternative designs and create solutions to resolve impacts (AKA architectural decisions)
   - Formulate "quality scenarios" that define measurable/observable architectural factors

Balance between Priorities & Future Proofing

- *Future Proofing* can lead to over-engineering for changes that are unlikely to occur
  - Exception: Prudent Future Proofing like Year 2000
- Priorities drive under-engineering
  - Getting it done over getting it done right

The art of the architect is knowing what battles are worth fighting – where it's worth investing in designs that provide protection against evolutionary change.

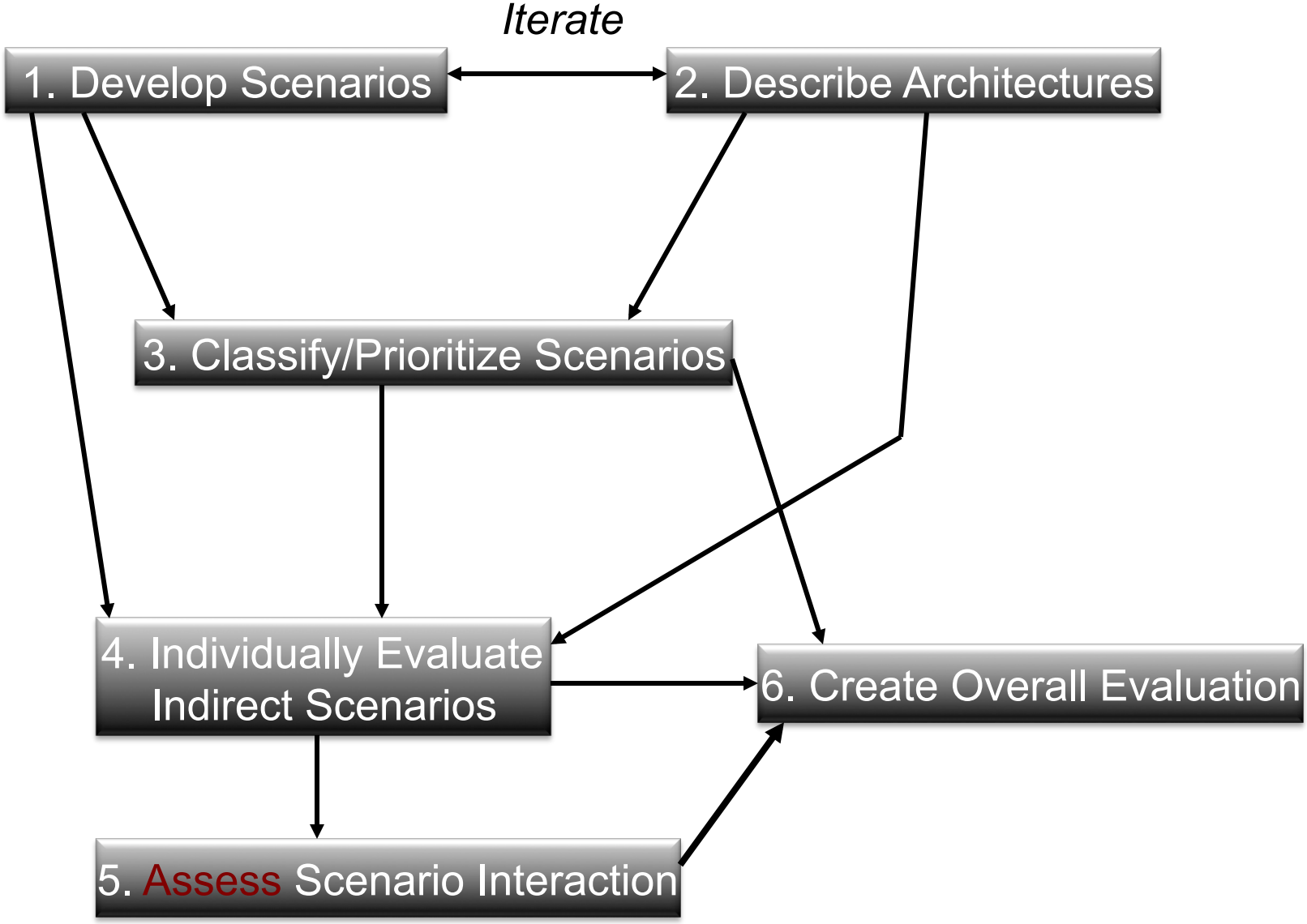
Architecture Analysis/Assessment



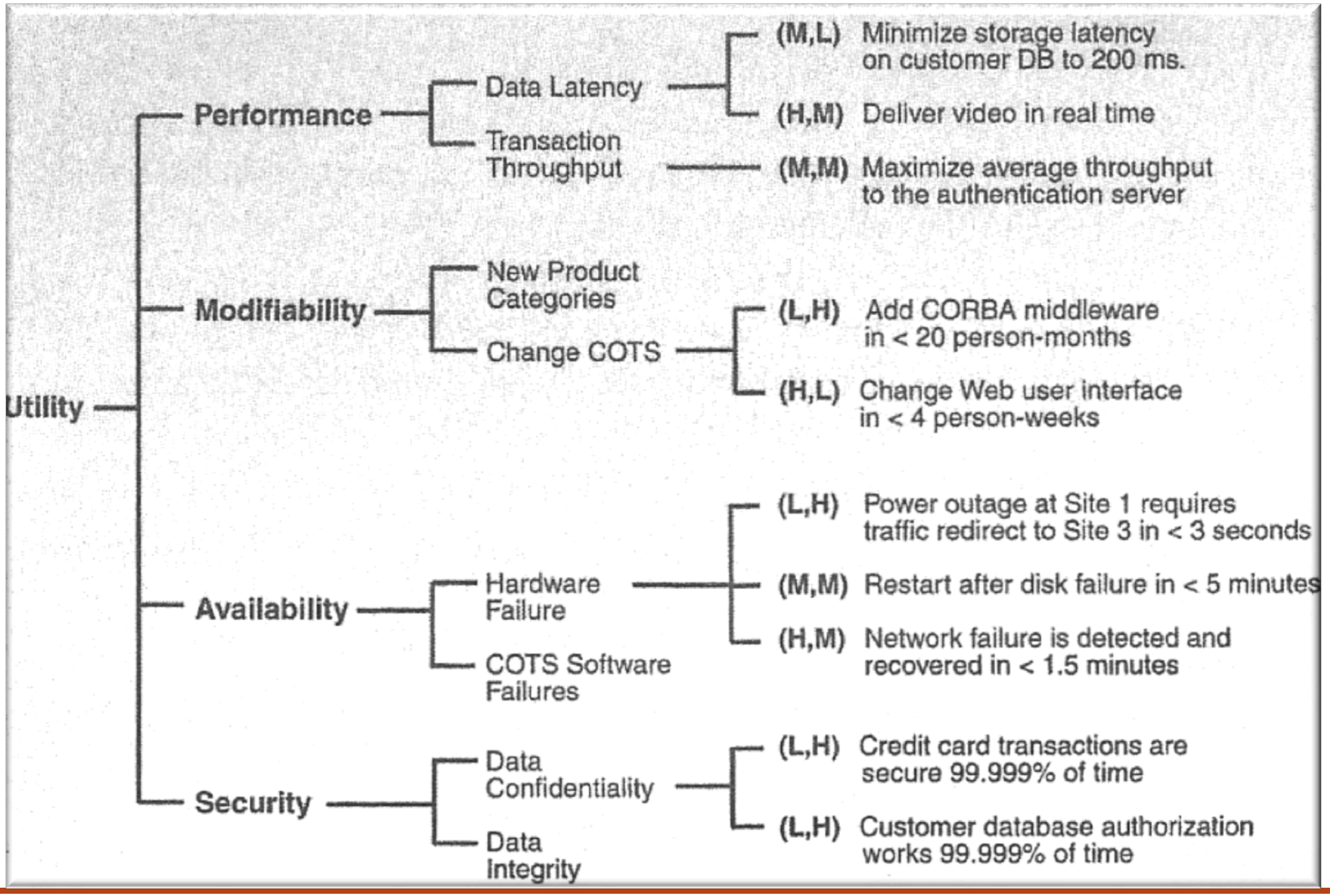
Scenario-Oriented

- Toll-gate Approach (i.e. *after* architectural design)
- Assemble all stakeholders for a meeting
  - End users, customers, operators, implementers, etc.
- Each stakeholder group defines their key scenarios
- Scenarios are Merged into Scenario Set (< 20)
- Scenarios are Discussed & Conflicts Resolved
  - If conflicts remain, architecture design is rejected, otherwise development proceeds
- Example: SAAM *(Kazman et al. 1994)*

SAAM–SW Architecture Analysis Method



Factors: Quality Attribute Utility Tree

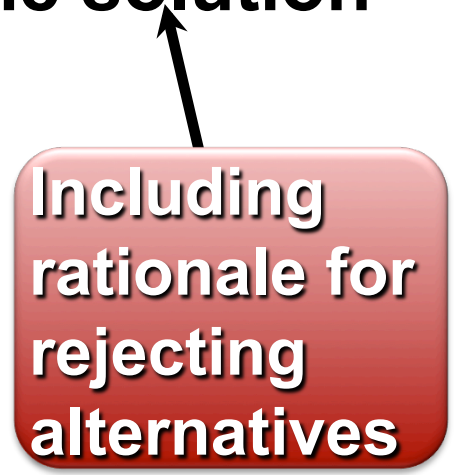


Note: High-Medium-Low for (Priority, Risk)

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Technical Memos: Documenting Decisions

- Summarize the *issue*
- List the relevant *architectural factors*
- Describe the chosen *solution*
- Give the *motivation* for choosing the solution
- Note any *unresolved issues*
- Identify *alternatives considered*



Separation of Concerns & Localizing Impact

Architectural factors often *cross-cutting*

- Modularize/encapsulate into separate components
  - E.g., persistence service/façade, layered arch.
- Use Decorators (special containers for additional features)
- Use post-compilers or aspect-oriented techniques
- Architecture description languages (ADLs)

Key Themes in Architectural Analysis

- Architectural concerns especially related to non-functional requirements
- Architectural analysis deals with variance and invariance of elements in the software architecture
- Architectural concerns involve system-level, large-scale, and broad problems that involve fundamental design decisions
- Architectural analysis is about understanding the interdependencies and tradeoffs in design decisions
- Architectural analysis is about the generation and evaluation of alternative solutions

Thinking Ahead Exercise

- Break up into your teams
- Consider how your junior project product will change over time.

Please brainstorm and list some key variation points and evolution points relevant to your design.

Design Studio Calendar

	Monday	Tuesday	Thursday
8th week		Team 2.4	Today Team 2.1
9th week	Team 2.2	Team 2.3	Team 2.5
10th week	Team 2.4	Team 2.1	Course Wrap-up

Homework and Milestone Reminders

- Read Chapters 34 and 35
- Milestone 5 – Final Junior Project System and Design
  - Draft due by 11:59pm on Friday, February 11th, 2011
  - Final due by 11:59pm on Friday, February 18th, 2011
- Team 2.2 – Rovio doing the Design Review

Factor Table in Supplementary Spec.

Factor	Measures and quality scenarios	Variability (current flexibility and future evolution)	Impact of factor (and its variability) on stakeholders, architecture and other factors	Priority for Success	Difficulty or Risk
Reliability—Recoverability					
Recovery from remote service failure	When a remote service fails, reestablish connectivity with it within 1 minute of its detected re-availability, under normal store load in a production environment.	current flexibility - our SME says local client-side simplified services are acceptable (and desirable) until reconnection is possible. evolution - within 2 years, some retailers may be willing to pay for full local replication of remote services (such as the tax calculator). Probability? High.	High impact on the large-scale design. Retailers really dislike it when remote services fail, as it prevents them from using a POS to make sales.	H	M
Recovery from remote product database failure	as above	current flexibility - our SME says local client-side use of cached "most common" product info is acceptable (and desirable) until reconnection is possible. evolution - within 3 years, client-side mass storage and replication solutions will be cheap and effective, allowing permanent complete replication and thus local usage. Probability? High.	as above	H	M

1. Develop Scenarios

- Quality Requirements – “Use Cases”, growth (changes), exploratory (stress)
- Views of all stakeholders: users, developers, customers ...
- Better understanding of requirements & interactions
- Documentation
- Stakeholder Buy–in and shared understanding
- Requirements Traceability
- A single scenario may be viewed differently from the perspective of different stakeholders

Typically, one starts with a large set and then merge, refine, or eliminate from this set.

2. Candidate Architecture

- May require multiple views:
  - Runtime – performance availability
  - Source – developmental such as modifications and portability
  - Uses – relationships between modules (system sub– and super –sets)
- ”What ifs” – want to be able to see where changing one aspect of an architecture affects the qualities of the architecture along other dimensions

3. Classification: Direct vs. Indirect

- Direct – supported by the architecture
- Indirect – requires change to the architecture
- Question: Is direct always better than indirect?
  - An architecture that supports something directly is generally ”better” than ones that need to be changed to support it
  - Different ”levels ”of direct and indirect
  - Direct scenarios are useful in their own right:
    - Create /elicit representation
    - Help in understanding dynamics of the architecture (implementation details such as scheduling, data movement, security)
  - Serve as starting points for analysis of dynamic properties such as performance

4. Scenario Evaluations

- Scenario must be evaluated in terms of normal completeness, consistency, ambiguity, and cost effectiveness
- How expensive are changes to the architecture?
- What components are affected by a change?
- Result is summary table with direct and indirect costs

5. Scenario Interaction

- One component involved in multiple interactions?
  - The scenarios are of the same class
    - Architecture exhibits high cohesion
  - The scenarios are of different classes and the module can be subdivided
    - Architecture may not be represented at the correct level of detail
  - The scenarios are different but the module cannot be divided
    - Potential problem in the architecture from improper separation of concerns
- Interaction related to metrics include structural complexity, coupling and cohesion