SOFTWARE ENGINEERING
(Week-12)

USAMA MUSHARAF

MS-CS (Software Engineering)

LECTURER (Department of Computer Science)

FAST-NUCES PESHAWAR

AGENDA OF WEEK # 12

- Architecture Evaluation
  - Qualitative Approach
  - Quantitative Approach

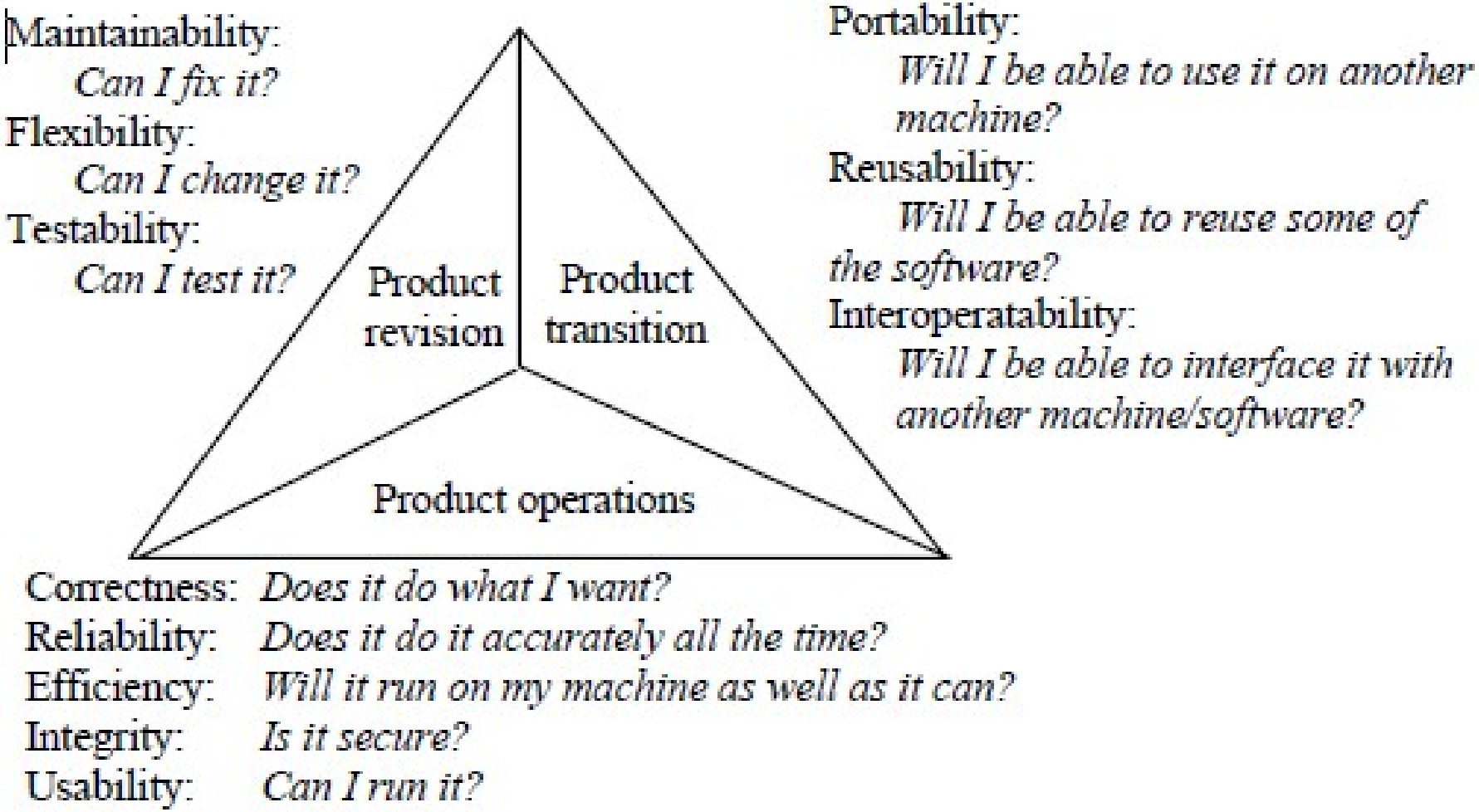


Figure 2.1 McCall's model of software quality

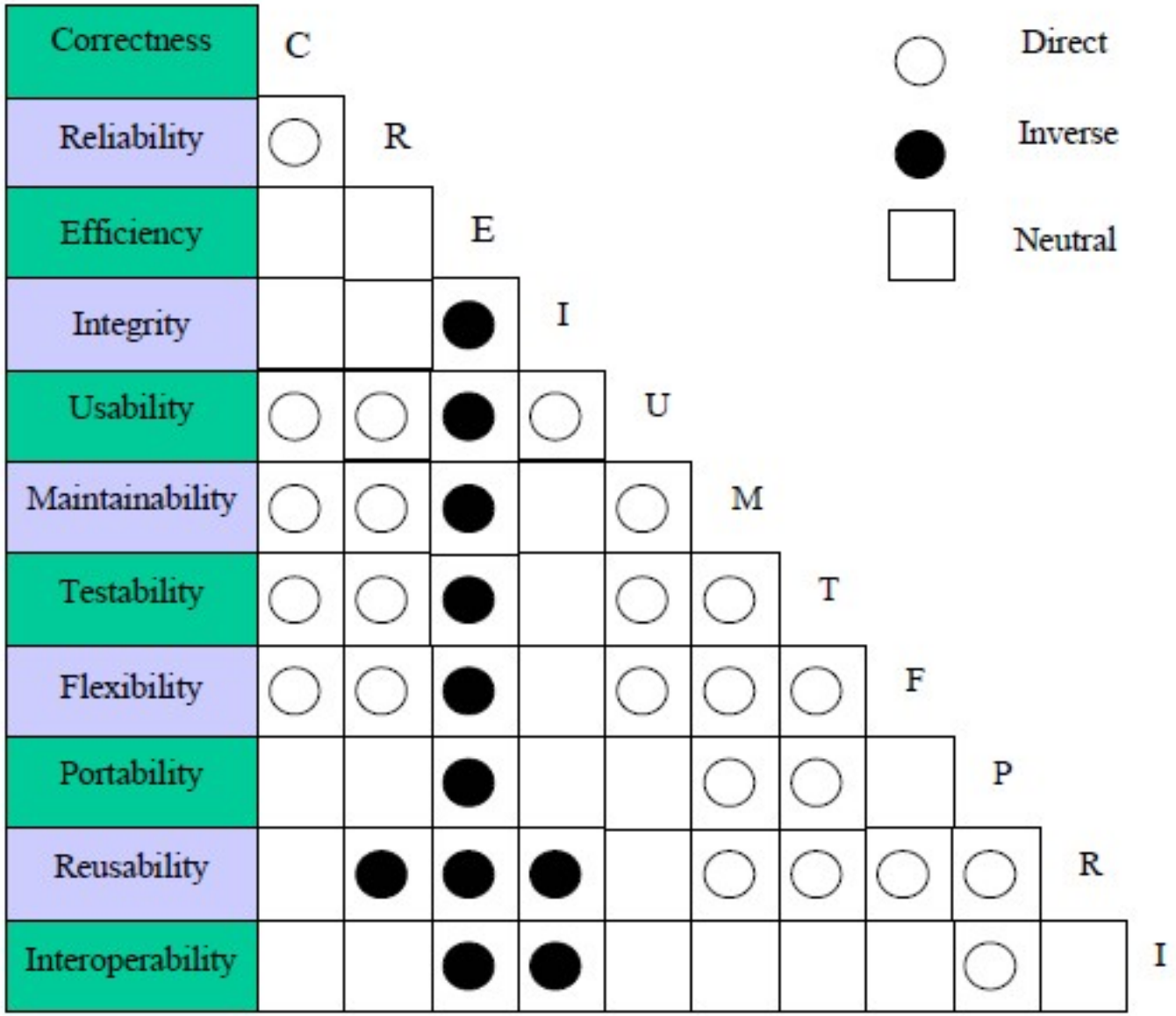


Figure 2.2 Perry's relational model of software quality

ARCHITECTURE EVALUATION

ARCHITECTURE EVALUATION

Software architecture evaluation in software systems is an important practice to develop quality software.

Evaluation is performed to analyze **software architecture** to reduce the possibility of risks and verify quality requirements, which are addressed during **software** design.

ARCHITECTURE EVALUATION

The goal of *software architecture evaluation* is to provide effective means to determine quality attributes characteristics, identify potential risks in architecture design, and sometimes to understand the trade-off in architecture design.

ARCHITECTURE EVALUATION APPROACHES

Several methods and techniques have been proposed to **evaluate software architecture** based on quality attributes.

Architecture evaluation can be broadly divided into two approaches.

Qualitative Approach

Quantitative Approach

1. QUALITATIVE APPROACH

QUALITATIVE APPROACH

Qualitative approaches are basically scenario-based architecture evaluation methods.

1. ATAM, Architecture Trade-off Analysis Method
2. SAAM, Software Architecture Analysis Method
3. CBAM, Cost Benefit Analysis Method
4. ALMA, Architecture Level Modifiability Analysis
5. FAAM, Family – Architecture Analysis Method

ARCHITECTURE TRADE OFF ANALYSIS METHOD

ATAM

Architecture Trade-off Analysis Method (ATAM):

ATAM is a scenario-based architecture method for assessing quality attributes such as: modifiability, portability, maintainability, etc.

ATAM analyses how well software architecture satisfies particular quality goals.

PREREQUISITES AND INPUTS FOR ATAM

The evaluators must understand the system architecture, recognize the architectural parameters, define their implications with respect to the system quality attributes, and compare these implications against the requirements.

Problem areas were so called “sensitivity points”, “tradeoff points” and risks. These must be carefully identified.

PREREQUISITES AND INPUTS FOR ATAM

A sensitivity point is a collection of components in the architecture that are critical for achievement of a particular quality attribute.

A trade-off point is a sensitivity point that is critical for the achievement of multiple quality attributes.

ATAM is a context-based evaluation method in which quality attributes of the system must be understood. This can be achieved employing descriptive scenarios for evaluating the quality attributes.

ATAM PHASES

ATAM is divided into four phases:

1. *Presentation Phase*
2. *Investigation and Analysis Phase*
3. *Testing Phase*
4. *Reporting Phase*

ATAM STEPS

1. ***Presenting Phase:***
   1. Present the ATAM
   2. Present business drivers
   3. Present architecture
2. ***Investigation and Analysis Phase:***
   4. Identify architectural approaches
   5. Generate quality attribute utility tree
   6. Analyze architectural approaches
3. ***Testing Phase:***
   7. Brainstorm and prioritize scenarios
   8. Reanalyze architectural approaches
4. ***Reporting Phase:***
   9. Present results

STEP 1: PRESENT THE ATAM

ATAM is explained to those involved in evaluation for the purpose of understanding.

Using a standard presentation, the leader describes the ATAM steps in brief and the outputs of the evaluation.

STEP 2: PRESENT BUSINESS DRIVERS

Appropriate system representative(s) present an overview of the system, its requirements, business goals, and context, and the architectural quality attribute drivers.

Everyone involved in the evaluation needs to understand the context for the system and the primary business drivers motivating its development.

The presentation should describe the following:

- The system's most important functions.
- The business goals and context as they relate to the project.
- The architectural drivers (that is, the architecturally significant requirements).

STEP 3: PRESENT THE ARCHITECTURE

In this step the system or software architect (or another lead technical person) presents the architecture.

The architect have to describe the following technical aspects.

- Architectural approaches (or patterns, or tactics,)
- Current state of the art technologies used in architecture.
- Architecture Views
- Quality Attributes

STEP 4: IDENTIFY ARCHITECTURAL APPROACHES

Step 4 of the ATAM captures the list of architectural approaches gleaned from the architecture presentation, as well as from the evaluation team's pre-exercise review of the architecture documentation.

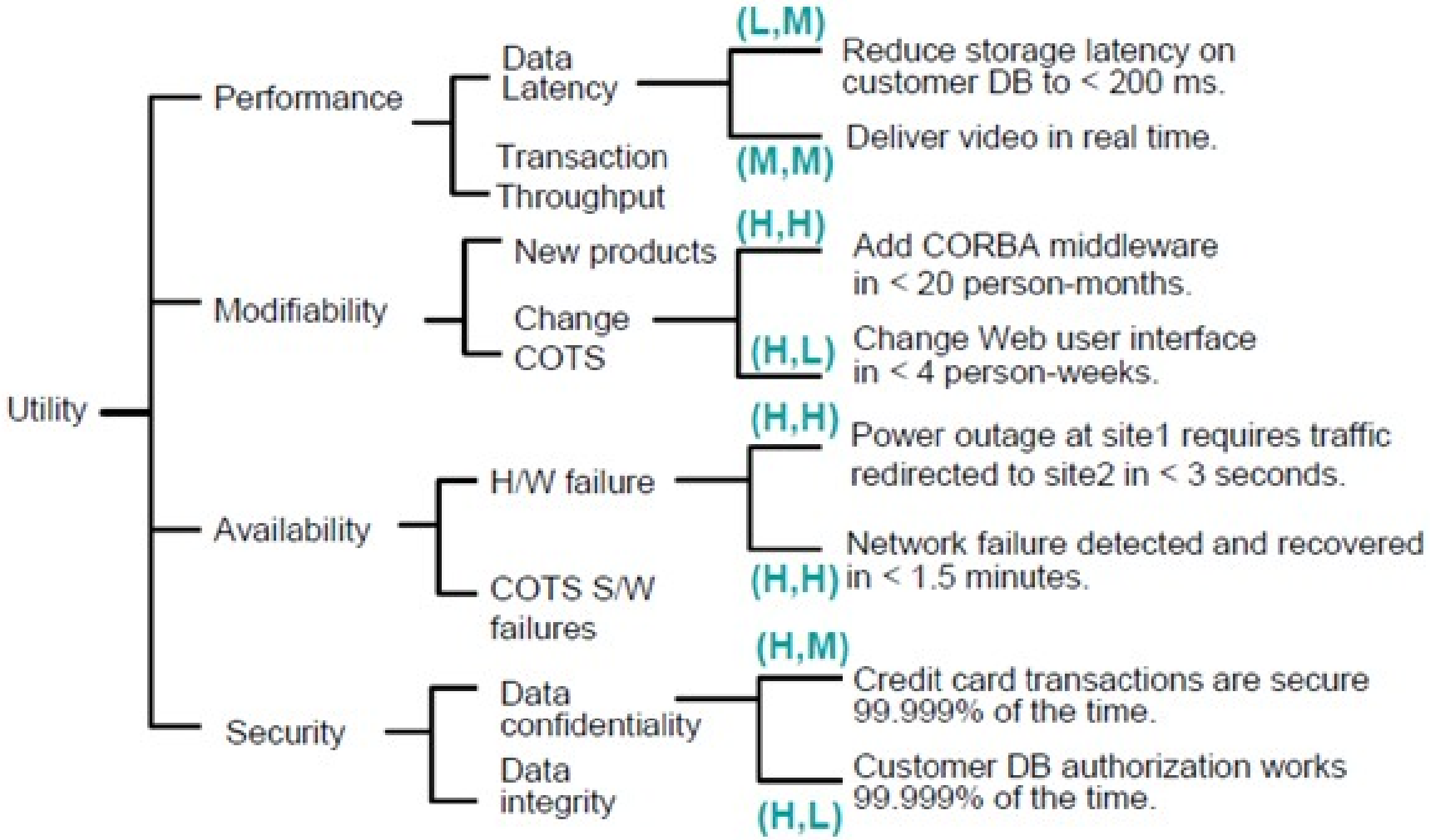
The ATAM focuses on analyzing an architecture by understanding its architectural approaches, especially patterns.

The system or software architect presents general architectural approaches to achieving specific qualities.

STEP 5: GENERATE UTILITY TREE

Step 5 of the ATAM produces a quality attribute utility tree. That tree provides a vehicle for translating the quality attribute goals articulated in the business drivers presentation to “testable” quality attribute scenarios.

UTILITY TREE EXAMPLE



USING SCENARIOS

- Scenarios are a technique developed at the SEI to tease out issues concerning an architecture through manual evaluation and testing.

- Scenarios are related to architectural concerns such as quality attributes, and they aim to highlight the consequences of the architectural decisions that are encapsulated in the design.

USING SCENARIOS

The ATAM work describes scenarios and their generation in great detail.

- Scenarios involve defining some kind of stimulus that will have an impact on the architecture.
- The scenario then involves working out how the architecture responds to this stimulus.
- If the response is desirable, then a scenario is deemed to be satisfied by the architecture. If the response is undesirable, or hard to quantify, then a flaw or at least an area of risk in the architecture may have been uncovered.

SCENARIO TYPES

Scenarios are used to exercise the architecture against current and future situations:

Use case scenarios reflect the normal state or operation of the system.

Growth scenarios are anticipated changes to the system (e.g., double the message traffic, change message format.

Exploratory scenarios are extreme changes to the system. These changes are not necessarily anticipated or even desirable situations (e.g., message traffic grows 100 times, replace the operating system).

EXAMPLE - SCENARIOS

Scenarios should be as specific as possible:

Stimulus . ***Environment*** . **Response**.

- Use case scenario

  System keeps doors locked for protection. **When a crash occurs**, **system unlocks doors**.

-

- Growth scenario

  Customer B needs a function, ***which was developed for customer A.*** **Reuse it within 1 week**.

- Exploratory scenario

  Reuse a 10-year-old function ***in the new software generation*** **within 1 month**.

STIMULUS ENVIRONMENT RESPONSE

Example Use Case Scenario:

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

Example Growth Scenario:

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

Example Exploratory Scenario:

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

UTILITY TREE EXAMPLE

Quality Attribute	Stimulus	Response
Modifiability	The *Customer System* packaged application is updated to an Oracle database.	The *Validate* component must be rewritten to interface to the Oracle system.
Availability	The email server fails.	Messages build up in the *OrderQ* until the email server restarts. Messages are then sent by the *SendEmail* component to remove the backlog. Order processing is not affected.
Reliability	The *Customer* or *Order* systems are unavailable.	If either fails, order processing halts and alerts are sent to system administrators so that the problem can be fixed.

STEP 6: ANALYZE ARCHITECTURAL APPROACHES

The evaluators and the architects map the utility tree scenarios to the architecture to see how it responds to each important scenario.

The analysis is not meant to be comprehensive. The key is to elicit sufficient architectural information to establish a link between the architectural decisions made and the quality attribute requirements that need to be satisfied.

STEP 7: BRAINSTORM AND PRIORITIZE SCENARIOS

The stakeholders brainstorm scenarios that are operationally meaningful with respect to the stakeholders' individual roles.

A user will probably come up with a scenario that expresses useful functionality or ease of operation.

A quality assurance person will propose a scenario about testing the system or being able to replicate the state of the system leading up to a fault.

STEP 7: BRAINSTORM AND PRIORITIZE SCENARIOS

The purpose of scenario brainstorming is to take the pulse of the larger stakeholder community: to understand what system success means for them.

Once the scenarios have been collected, they are prioritized by voting.

The list of prioritized scenarios is compared with those from the utility tree exercise.

- If they agree, it indicates good alignment between what the architect had in mind and what the stakeholders actually wanted.
- If additional driving scenarios are discovered—and they usually are—this may itself be a risk, if the discrepancy is large. This would indicate that there was some disagreement in the system's important goals between the stakeholders and the architect.

STEP 8: RE-ANALYZE ARCHITECTURAL APPROACHES

In this step the evaluation team performs the same activities as in step 6, using the highest-ranked, newly generated scenarios.

The evaluation team guides the architect in the process of carrying out the highest ranked new scenarios.

The architect explains how relevant architectural decisions contribute to realizing each one.

STEP 9: PRESENT RESULTS

The collected information from the evaluation is summarized and presented to stakeholders.

The following outputs are presented:

- The architectural approaches documented.
- The set of scenarios and their prioritization from the brainstorming.
- The utility tree.
- The risks discovered.
- The non-risks documented.
- The sensitivity points and tradeoff points found.

ARCHITECTURE TRADEOFF ANALYSIS METHOD

- The ATAM is a method that helps stakeholders ask the right questions to discover potentially problematic architectural decisions
- Discovered risks can then be made the focus of mitigation activities: e.g. further design, further analysis, prototyping.
- The purpose of the ATAM is NOT to provide precise analyses . . . the purpose IS to discover risks created by architectural decisions.
- We want to find *trends*: correlation between architectural decisions and predictions of system properties.

2. QUANTITATIVE APPROACH

WHY DO WE MEASURE?

To indicate the quality of the product

To assess the productivity of the people who produce the product

To assess the benefits derived from new software engineering methods and tools

DEFINITIONS

Measure - quantitative indication of the extent, amount, dimension, capacity, or size of some attribute of a product or process.

Measurement - the act of determining a measure

Metric - a quantitative measure of the degree to which a system, component, or process possesses a given attribute (IEEE)

WHAT CAN BE MEASURED?

Direct measures

Lines of codes (LOC), speed, cost, memory size, errors, ...

Indirect measures

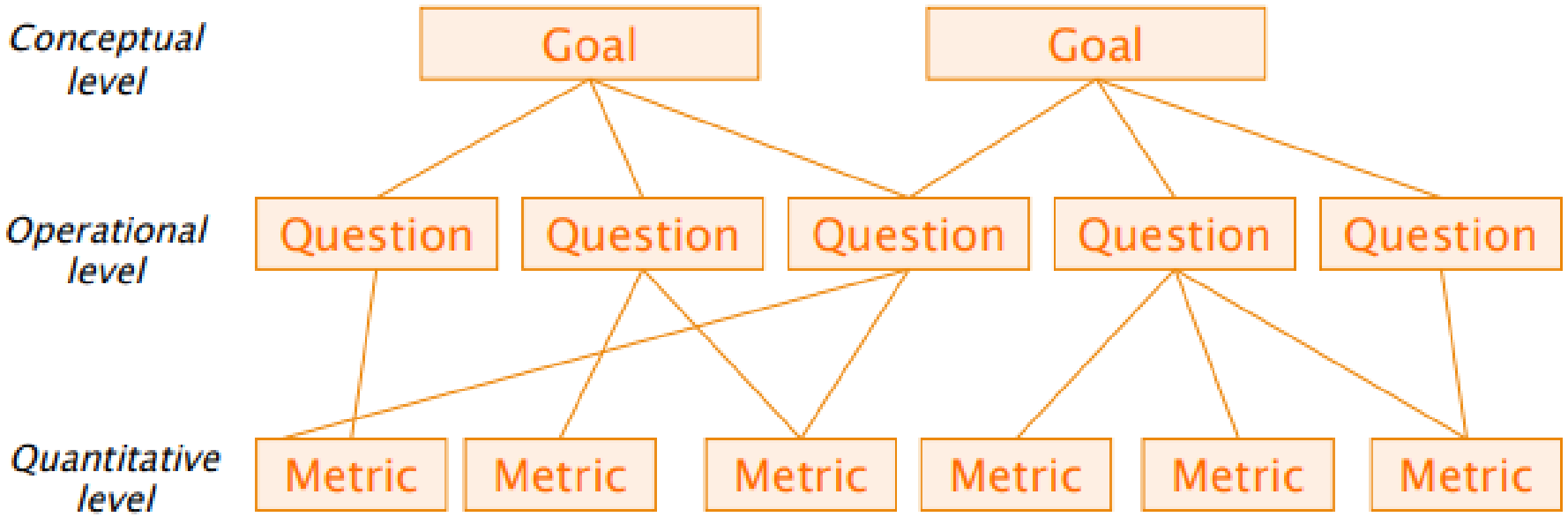
Quality, functionality, complexity, reliability, maintainability, etc ...

Example:

length of a pipe is a direct measure.

the quality of the pipes can only be measured indirectly by finding the ratio of the accepted pipes to the rejected.

GOAL QUESTION METRIC (GQM)



GOAL DEFINITION

analyze the use of stereotyped UML diagrams, with the purpose of evaluating their usefulness in Web application comprehension for different categories of users. The quality focus is to ensure high comprehensibility, while the perspective is both of Researchers, evaluating how effective are the stereotyped diagrams during maintenance for different categories of users, and of Project managers, evaluating the possibility of adopting the Web modeling technique WAE in her organization, depending on the skills of the involved developers. The context of the experiment consists of two Web applications (objects) and four groups of subjects: research associates, students from an undergraduate course, and students from two graduate courses.

GOAL DEFINITION

Analyze	Objects(s) of study stereotyped UML diagrams
for the purpose of	Purpose evaluating their usefulness
with respect to their	Quality focus comprehension
from the point of view of	Perspective researcher and project manager
in the context of	Context research associates, undergraduate students graduate students

QUESTIONS

Software architects often ask the following questions before deployment to understand the performance interrelationships in their design:

Q1: How does the architecture affect response times and throughputs for the expected workload?

Q2: How does the implementation of specific components influence performance?

Q3: How does the allocation of components to resources influence performance?

Q4: How does the system perform if its workload increases unexpectedly?

ARCHITECTURAL DESIGN METRICS

SIZE METRICS

1. No of Components (NC)

- No of components (NC) metric is used to count the number of components in a system. The NC metric indicates the system size that is used in estimating the complexity of the system.

$$NC(SyS) = \sum_{C \in SyS}^{n} 1$$

SIZE METRICS

2. No of Operations(NO)

- NO is a count of all the operations of a component to indicate the complexity of a component.

$$NO(C) = \sum_{O \in C}^{n} 1$$

- To calculate the overall NO of all components in a system.

$$NO(SyS) = \sum_{O \in SyS}^{n} 1$$

COUPLING METRICS

1. Direct Coupling

- The direct coupling can be defined as the direct interactions between consumers and providers. Direct coupling can be calculated by counting all the direct consumer calls for a specific provider (component) as shown below.

$$DC\ (p) = C\ (p)$$

2. Indirect Coupling (IC)

- The indirect coupling was calculated by counting the direct consumers calls for specific providers and then assume consumers as providers and calculate their coupling.

$$IC(p) = DC\ (p) + \sum_{c(p) \epsilon P} \mathrm{IC(c(p))}$$

COUPLING METRICS

3. Total Coupling

- The total coupling can be calculated by the addition of direct and indirect coupling, as shown in the equation below.

$$TCoup = DC(p) + IC(p)$$

4. Coupling Factor (CoupF)

- The quantitative result of total coupling could not be interpreted directly, because this number may give a good indication if the system is big or worse if it is small. This means that the size of the system is needed to understand the value of this metric. If we suppose f = NC (SyS) + NO (SyS) then:

$$CopF(SyS) = \frac{TCoup(SyS)}{f2 - f}$$

COHESION METRICS

1. Cohesion Metric

This metric was used to measure the degree of cohesion for specific component(C). Cohesion is a relationship between the operations of a component.

CM (C) = counts all the consumers that consume the functionalities

The consumer here refers to the operations of a service. The remaining operations are the value of this metric.

COHESION METRICS

2. Cohesion Factor

- Just like the coupling factor, the result given by the cohesion metric could not be interpreted directly, rather it depends on the system size. If the system is big, the result of this metric may provide a good indication, else the indication would be bad. Let f = NC(SyS) + NO(SyS) and i = NC(c) + NO(c) then

$$CohF(C) = \frac{CM(C) * (i^2 - i)}{f2 - f}$$

$$CohF(SyS) = \sum_{C \in SyS} \frac{CM(C) * (i^2 - i)}{f2 - f}$$

COMPLEXITY METRICS

1. Total Complexity Metric for a Component

- This metric calculates the complexity of component(c) from coupling and cohesion metrics.

$$TCM(C) = \frac{TCoup(C) + NC(C) + NO(C)}{CM(C)}$$

2. Complexity Factor (ComF)

- This metric calculates the complexity factor by using coupling and cohesion factors.

$$ComF(C) = \frac{CopF(C)}{CohF(C)}$$

COMPLEXITY METRICS

3. Total Complexity Metric for a System

- This metric calculates the complexity of the entire system by using the previous two metrics.

$$TCM(SyS) = \sum_{s \epsilon SOS} TCM(C) * ComF(C)$$

REUSABILITY METRICS

1. Reusability Metric

- Reusability can be calculated in terms of coupling, by counting the entire direct and indirect consumer calls for a specific provider.

$$Res = TCoup = DC(p) + IC(p)$$

2. Reusability Factor:

- The reusability factor can be calculated by comparing reusability values (in terms of coupling) with the system size by measuring the cohesion of operations.

$$ResF = \frac{CM(SyS)}{TCoup(SyS)}$$

MAINTAINABILITY METRICS

In the literature, the most common characteristics related to maintainability are coupling, cohesion and complexity.

Cohesion

[Cohesion] positively impacts (+) → [maintainability]

Coupling

[Coupling] negatively impacts (-) → [maintainability]

No of complex component (NCC)

[NCC] negatively impacts (-) → [maintainability]

Complexity

[Complexity] negatively impacts (-) → [maintainability]

HAVE A GOOD DAY!