Software Architectures and Quality

Architecture Evaluation and Transformation

Javier González Huerta

javier.gonzalez.huerta@bth.se

BLEKINGE TEKNISKA HÖGSKOLA · BTH ·

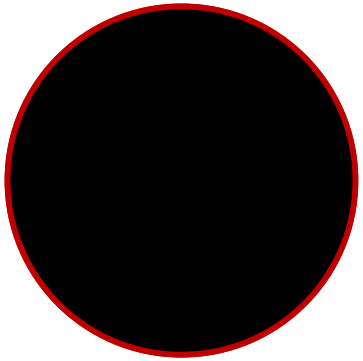
in real life

Objectives

- Understand the different reasons and different times for architectural evaluations
- Discuss some of the most widely-used architectural evaluation methods
- Discuss the problems around architecture evaluations
- Introduce the concept of architectural transformations and its usage together with the architecture evaluations

Reflection

- **Discuss during 2 minutes in groups of 2-3 persons about:**
- **What are the main purposes of performing architecture evaluation?**
- **What aspects are going to be considered during an architecture evaluation?**



Purpose of **[Early]** Architecture Evaluations

- Aiming to answer [some of] the following questions:
  - Are the quality attribute requirements met?
  - Do the stakeholders share a common understanding on the system?
  - Are all the quality requirements accounted for?
  - Does the architecture have any evident weak spot?
  - Can the system (or the architecture) be improved?
  - Does the team have all the needed resources?
  - Is the system / architecture feasible?
  - Should we continue with the project?

Purpose of [Late] Architecture Evaluations

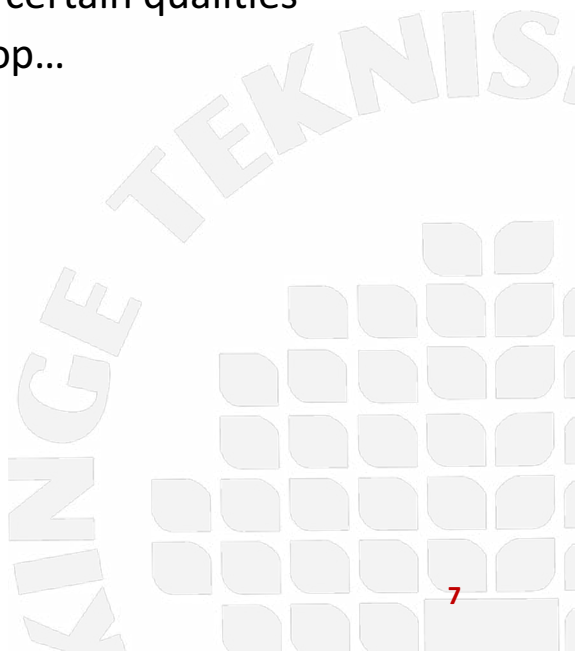
- More hard metrics
- Analyzing the architecture of the existing software
- Focusing on what needs to be improved for next releases
- Analysis of what needs to be improved
  - Looking for refactorings to improve certain weak points

Early Architecture Evaluation Methods

- Experiences and Logical Reasoning
  - Unstructured revisions
- Scenario based
  - Software Architecture Analysis Method (SAAM)
  - Architecture Trade of Analysis Method (ATAM)
  - Global Analysis (as a whole)
  - QASAR (BTH-way)

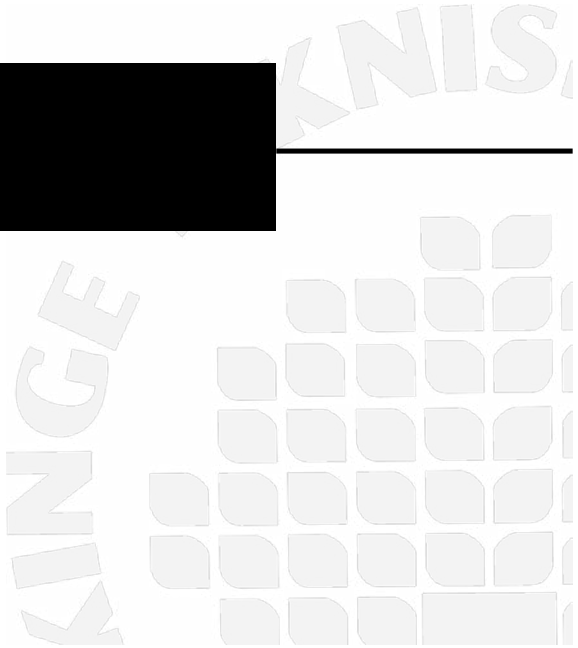
Early Architecture Evaluation Methods

- Simulation-based analysis
  - Build parts of the architecture
  - Build models of the architecture
  - ADL-based assessment
    - We can use some of the analyses offered by the ADLs to analyze certain qualities
    - Examples: AADL (Latency, Dependability), Meta-H, Rapide, Aescop…
- Mathematical models
  - Domain specific
  - Example: ABAS
- Software Inspections



Architecture Evaluation and Transformation

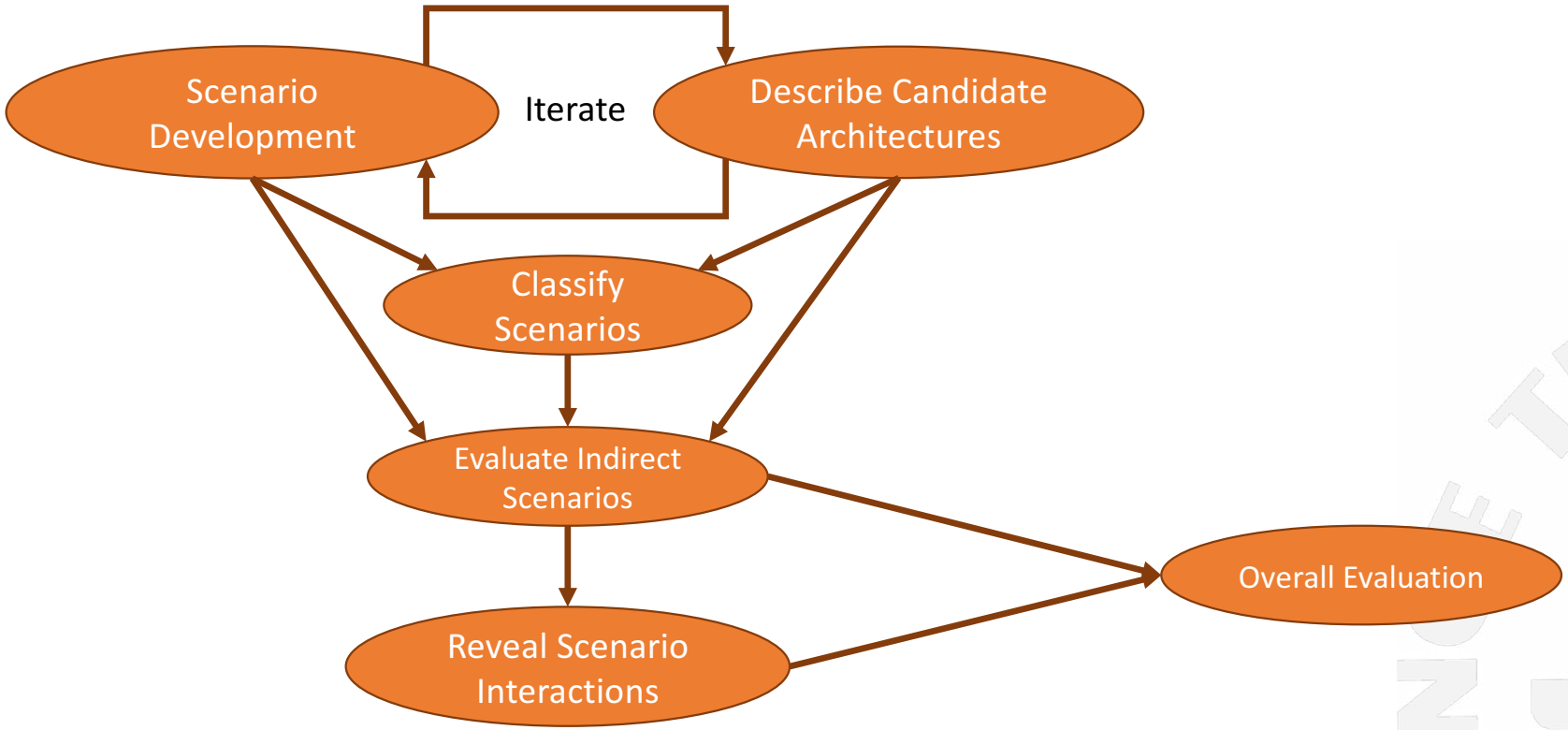
Evaluation Methods: SAAM



SAAM: Software Architecture Analysis Method I

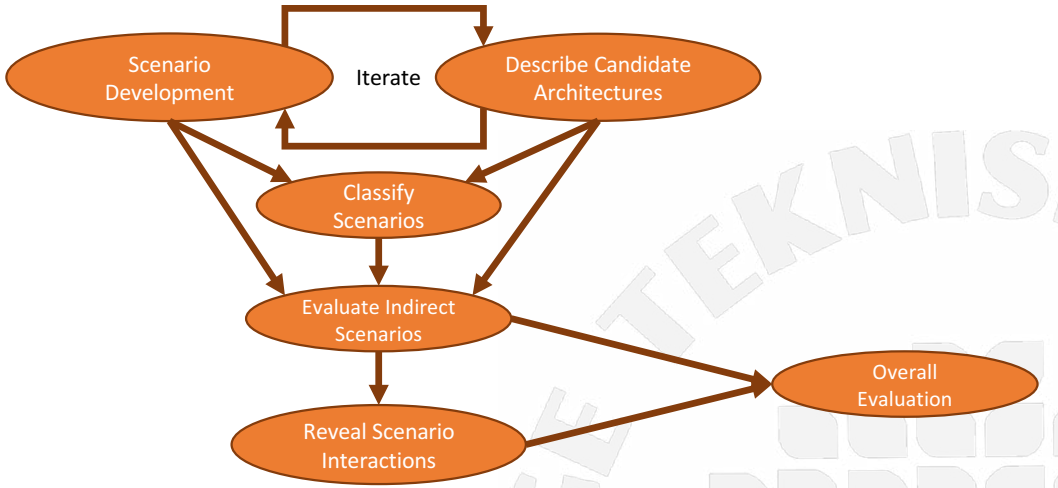
1. Develop Scenarios
2. Describe Candidate Architecture
3. Classify Scenarios
   - Directly supported vs Indirectly supported
4. Perform Scenario Evaluations for Indirect Scenarios
   - Identify required changes to support the scenario (Architectural Transformations)
   - Assess the cost of the change
5. Reveal Scenario Interactions
   - Identify scenarios requiring changes in the same components -> components overload?
6. Overall Evaluation
   - Prioritize Scenarios

SAAM: Software Architecture Analysis Method II



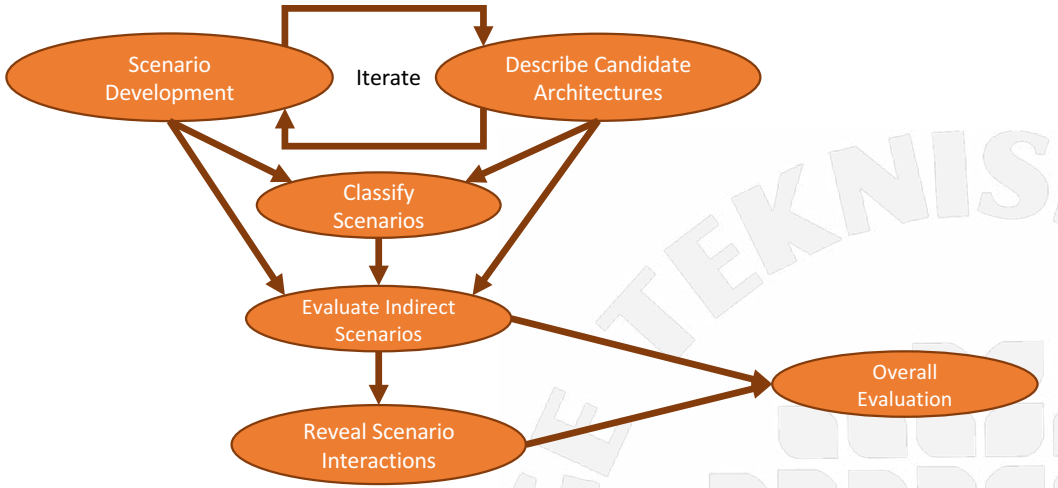
SAAM: Software Architecture Analysis Method III

- **Classify Scenarios:**
  - The architecture supports the scenario without changes -> Direct Scenarios
  - We should add new components, new interactions or both -> indirect scenarios
- **Evaluate Indirect Scenarios:**
  For each indirect scenario, list the changes to the architecture that are necessary



SAAM: Software Architecture Analysis Method IV

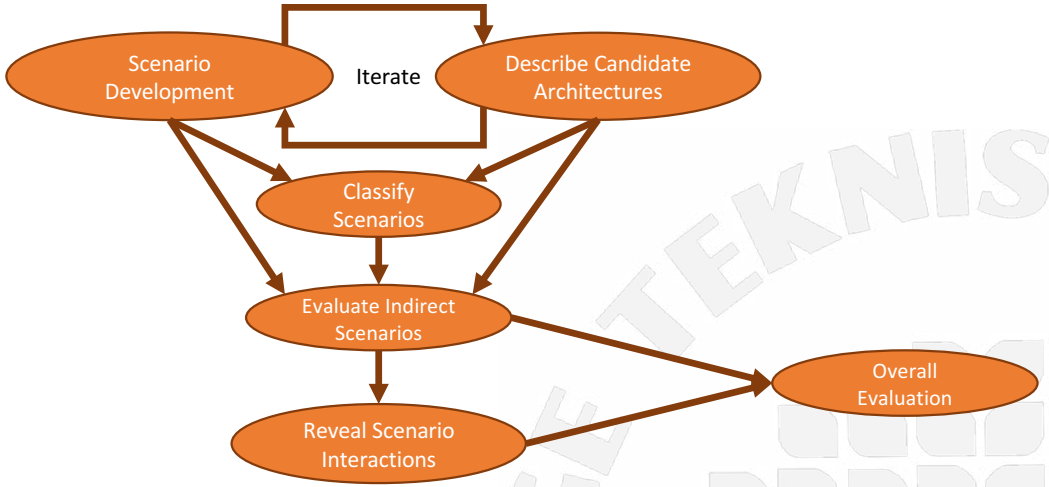
- **Interactions:** When two or more indirect task scenarios require changes to a single component of a system, they are said to interact in that component



SAAM: Software Architecture Analysis Method V

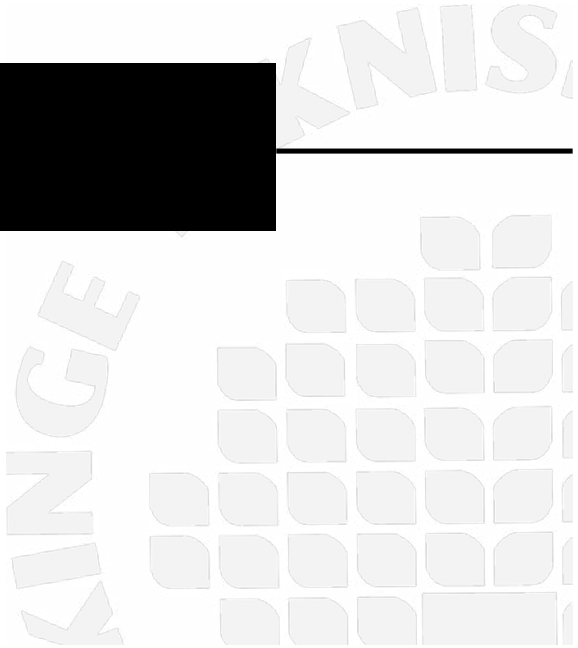
- **Overall Evaluation:**
  - Assign weight to each scenario in terms of relative importance
  - Use that weighting to determine an overall ranking.
  - This resolves the situation 1srt architecture scores well on ½ scenarios, and the 2nd architecture scores better on the other ½.

- Given this set of mini-metrics, SAAM can be used (and in fact was developed with the intent to) compare competing architectures on a per-scenario basis.



Architecture Evaluation and Transformation

Evaluation Methods: ATAM



Architecture Trade-Off Analysis Method (ATAM)

- Scenario based evaluation method **based on SAAM**
- The purpose of ATAM is to assess the **consequences** of **architectural design decisions** in the light of **quality attributes**.
  - ATAM helps in foreseeing how an **attribute** of **interest** can be **affected** by an **architectural design decision**.
  - The Quality Attributes of interest are clarified by **analyzing** the stakeholder's **scenarios** in terms of stimuli and responses.
  - Once the scenarios had been defined, the next step is to define which ***architectural approaches***[1] can **affect** to those **quality attributes**.

[1] In ATAM, the term ***architectural approaches*** are used to refer both to architectural styles or patterns

Architecture Trade-Off Analysis Method (ATAM)

- Participants:
  - **Evaluation Team:** Team Leader, Evaluation Leader, Scenario Scribe, Proceedings Scribe, Timekeeper, Process Observer, Process Enforcer, Questioner
  - **Project Decision Makers:** Project manager, Customer?, Architect, Commissioner of Evaluation
  - **Architecture Stakeholders:** Developers, Testers, Integrators, Maintainers, Performance Engineers, Users, Builders of Related Systems, etc.

Architecture Trade-Off Analysis Method (ATAM)

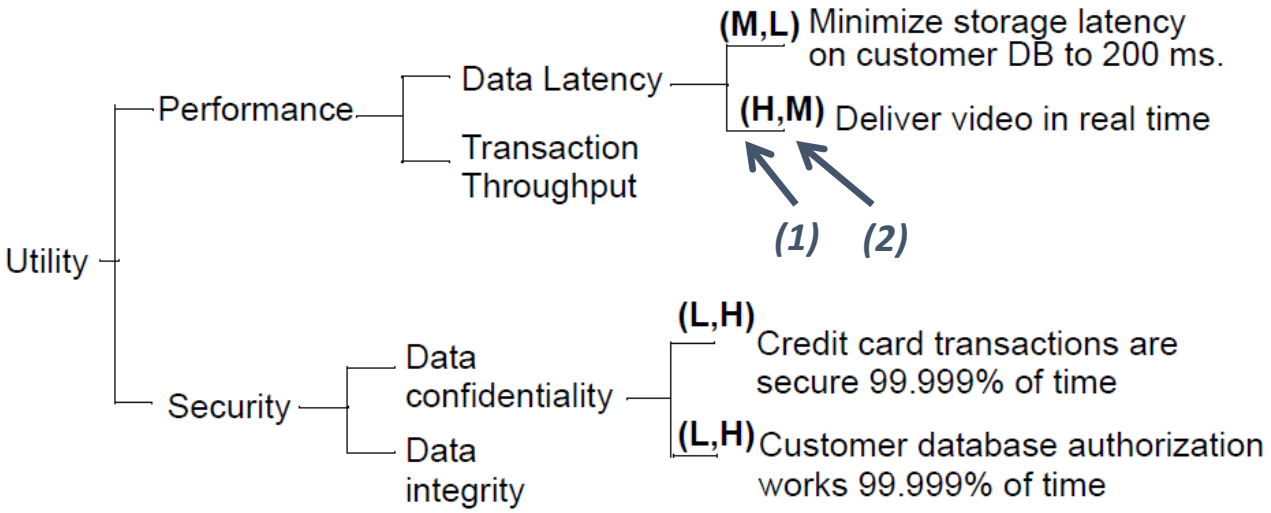
- The outputs of ATAM are:
  - A concise **presentation of the architecture**
  - A ***Utility Tree***: a top-down mechanism for directly and efficiently translating the business drivers of a system into concrete quality attribute scenarios.
  - A set of ***architectural approaches*** identified and or applied: Sometimes we can identify architectural approaches that cannot be applied on our architecture
  - A ***set of scenarios*** identified and the subset that had been effectively mapped in the architecture.
  - A ***set of questions about the quality attributes*** in the architecture and the ***answers to these questions***.
  - The mapping between ***architectural decisions*** to ***quality requirements.***
  - The ***risks identified***: risks that the architecture is able to mitigate and the risks that threaten i) the system and ii) the business goals.

ATAM Scenarios

- In ATAM both the scenarios that had been identified and those that had been mapped into the architecture are documented in terms of ***stimuli and response***.
  - In some cases, we can identify scenarios that describe requirements that cannot be effectively mapped into the architecture.
  - We document both the ones that had been identified and those that had been mapped into the architecture.
- ***Stimuli*** are the events that cause the architecture to respond or change.
- To analyze an architecture for adherence to quality requirements, those requirements need to ***be expressed in terms that are concrete and measurable or observable***.

Utility Tree

- The utility tree contains attribute goals concrete enough for prioritization.
- Those scenarios will make these architectural response goals even more specific. The prioritization can be done by using ***Low, Medium or High*** literals associated to each leaf node.
- This prioritization is done using two dimensions:
  - ***(1) The importance of each node in the success of the system***
  - ***(2) The degree of perceived risk associated to the achievement of that goal.***



ATAM Phases and Steps I

- ***Phase 1: Presentation***
  1. **Present ATAM:** The method is described to the evaluators and stakeholders.
  2. **Present business drivers:** business goals which motivate the development effort and hence that will be the primary architectural drivers are presented.
  3. **Present the architecture:** describe the proposed architecture, focusing on how it addresses the business drivers.
- ***Phase 2: Investigation and Analysis***
  4. **Identify architectural approaches:** Architectural approaches supporting the system and business goals are identified by the architect, but not analyzed
  5. **Generate quality attribute utility tree:** Quality factors that comprise system "utility" (performance, availability, security, etc.) are elicited, specified down to scenarios level, annotated with stimuli and responses, and prioritized.
  6. **Analyze architectural approaches:** The stakeholders and the architect analyze how the architectural approaches affect to the quality factors identified in Step 5.

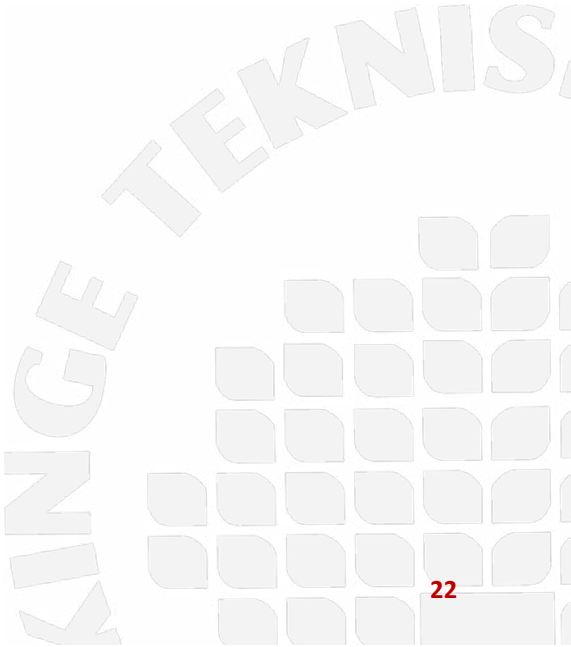
ATAM Phases and Steps II

- ***Phase 3: Testing***
  7. **Brainstorm and prioritize scenarios:** If required, more scenarios are generated and prioritized together with the scenarios on the utility tree.
  8. **Analyze architectural approaches:** This step reiterates step 6, but here the highly ranked scenarios from Step 7 are considered and we should decide which architectural patterns are applied and how the architecture will be modified.
- ***Phase 4: Reporting***
  9. **Present results:** Based on the information collected (patterns, scenarios, attribute-specific questions, the utility tree, risks, sensitivity points, tradeoffs) the team presents the findings to the stakeholders and writes a report detailing this information.

ATAM in a Mission Control System

Phase 1:

1. **ATAM Presentation**
2. **Business Goals Presentation:** In our example we are going to deal with a **mission control system**, and the business goals, are among others, the high **reliability** and **performance**.



ATAM in a Mission Control System

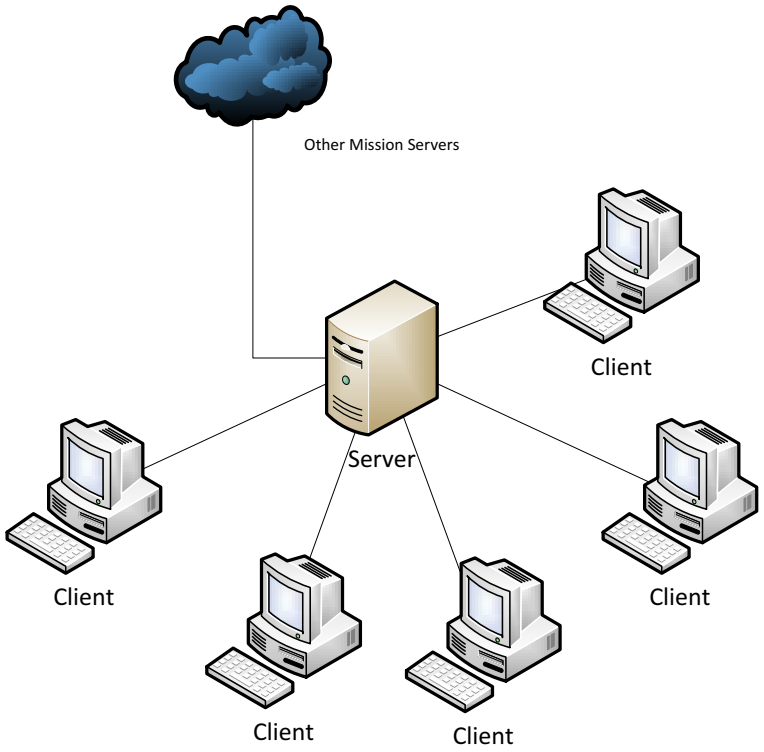
Phase 1:

3. Architecture Presentation:

- The system of our example is the ***Battlefield Control System***. This system is used to control the **movements** of **troops** in the battlefield. There are **two types of nodes** in its architecture:
  - **Server Node (commander):** supports those taking decisions. It handles a database with the soldiers and battlefield status and transmits orders to client nodes. Is capable of communicating with other Server nodes to send and receive orders
  - **Client Nodes (Soldiers):** make requests to the server and update the server's database
- The internode communication between the clients and the server is only though encrypted messages sent via a shared radio communication channel
  - 9600bps limited bandwidth
  - Only one node can be broadcasting at any moment
  - The size database is usually around 55Kb

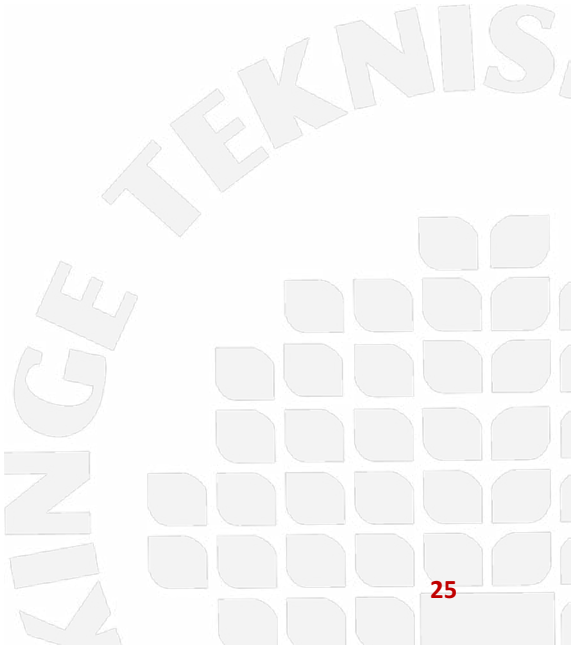
3. Architecture Presentation

Original Architecture to be evaluated:



Architectural Limitations

- This system architecture has a reliability problem, the system depends totally on the server
- If the server fails the system becomes totally useless
- We should consider which architectural transformations can be applied to this architecture to solve this problem



Step 4: Architectural approaches identification: Backup Server

Problem addressed:

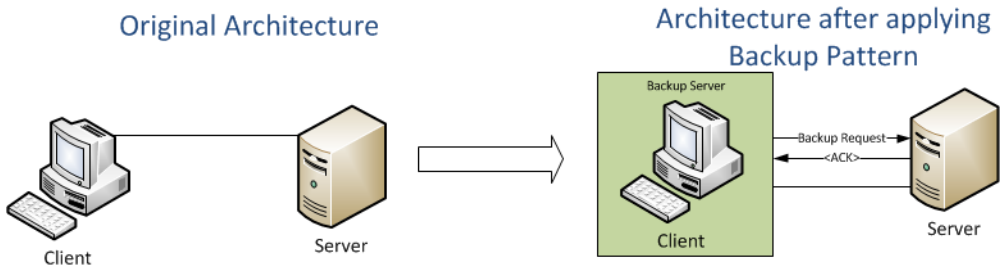
- In some systems a failure on the server means that the system goes down and the service is interrupted.

Structure:

- A client node can promote to backup server by sending a request and by receiving the corresponding ACK from the server.
- The backup server mirrors the server database each 10 minutes and monitors the communications between the server and the client nodes (stores every message sent). If the server fails automatically promotes to server.

Consequences:

- Improves the availability of the system
- Increases the communications through the channel, since the backup server has to receive a copy of the server's database



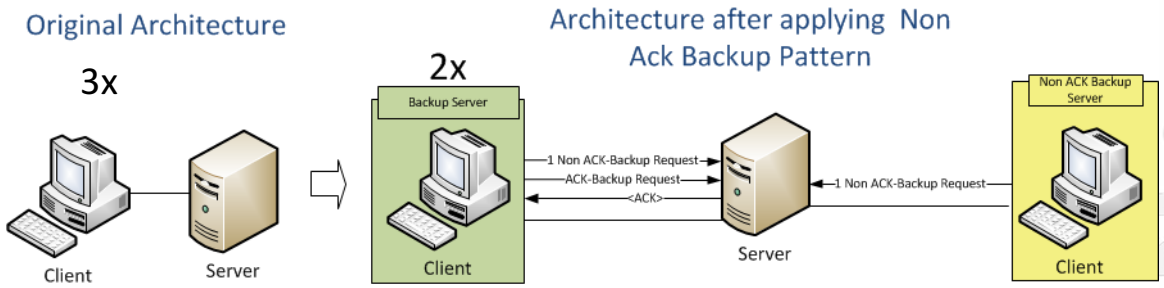
Step 4: Architectural approaches identification: Non ACK Backup (1/2)

Problem:

- In some systems a failure on the server means that the system goes down and the service is interrupted.

Structure:

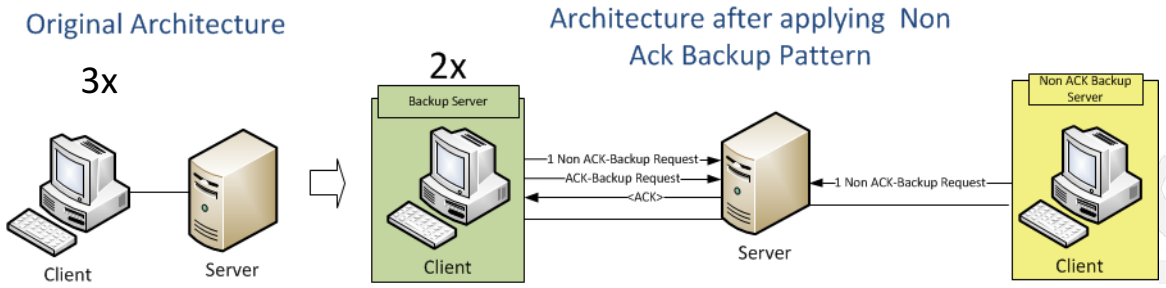
- A client node can promote to Non-ACK backup server by sending a request to the server.
- A Non-ACK Backup Server node can promote to backup server by sending a request and by receiving the corresponding ACK from the server.
- The Non-ACK Backup Server only monitors the communications, storing a copy of the messages sent through the communication channel. If a Non-ACK Backup Server needs to promote directly to Server, it will ask the other clients to resend information about their state.
- The backup server mirrors the server database each 10 minutes and monitors the communications between the server and the client nodes (stores every message sent). If the server fails automatically promotes to server.



Step 4: Architectural approaches identification: Non ACK Backup (2/2)

Consequences:

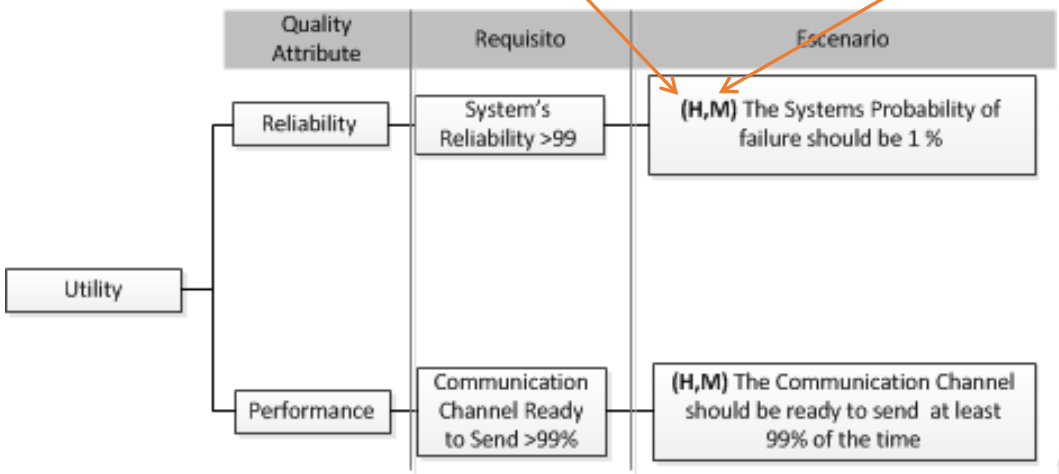
- Improves the system's availability. The system survives to an eventual failure both on the server and on the backup servers before a backup server has promoted to server.
- Maintain two backup servers mirroring the server's database increases the network load, since the two backup servers have to receive a copy of the server's database (in our case 55Kb)
- If there is a simultaneous failure on the server and on each single backup server require the Non-ACK backup to ask the other client nodes to resend the state, increasing the recovery time.



Step 5: Utility Tree Generation

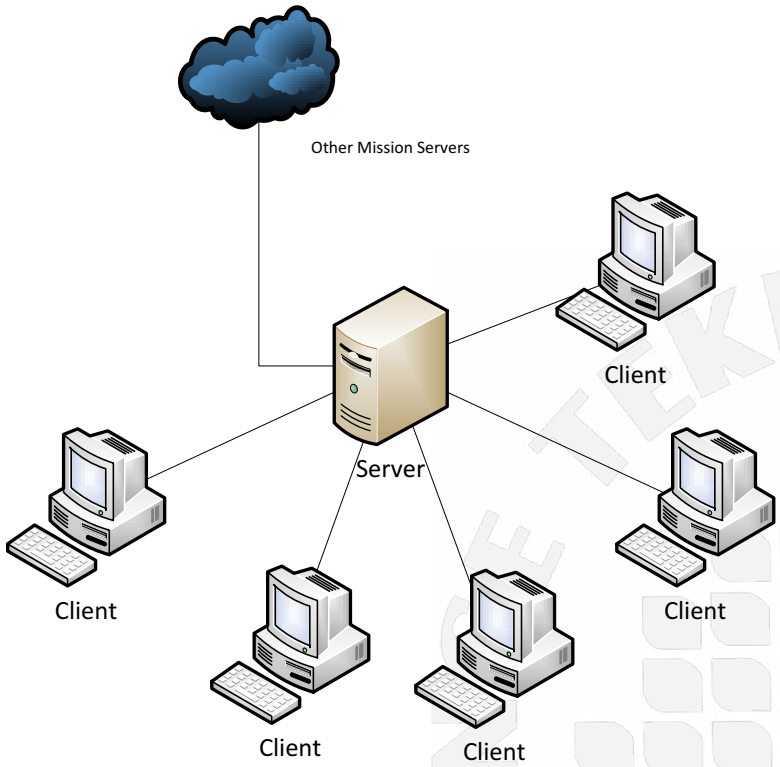
How Important it is (**High/Medium/Low**)

How difficult to achieve it is (**High/Medium/Low**)



Initial assessment

- **Reliability Analysis:**
  - Probability of failure of a node (and of the server) is 5%
    - Probability of failure of the system ≃ P(Server) = 5%
- **Performance Analysis:**
  - The communication channel is only used for transition between server and clients and then it is ready to send a 100% of the time.



Architecture Evaluation

Reliability: Probability of Failure

Value Obtained | 5%

Performance: % of Time Ready To Send

Value Obtained | 100%

Step 6. Architectural Approaches Analysis

Architectural Approaches Analysis

- How will the approaches identified in Step 4 improve the quality factors presented in the utility tree?
  - The application of the Backup Pattern will increase the reliability of the system, but on the contrary, it makes use of the communication channel to mirror the server's database.
  - The application of the Non-ACK Backup will improve much more the reliability, but on the other hand the communication channel will probably be busy too much time.

Step 7 and 8: Scenario Prioritization and Approach Analysis

Phase 3:

7. Scenario Prioritization

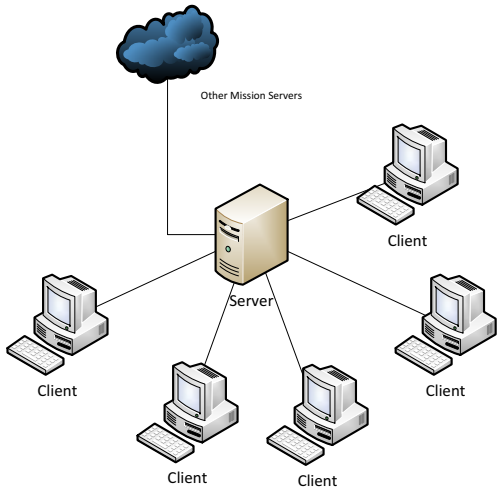
- In our case since we did not aggregate new scenarios we only have to revise the priorities on the two existing scenarios.

8. Architectural Approaches Analysis

- Considering the architecture, the architectural approaches and the high priority scenarios we should decide which approach is more appropriated, and detect the changes on the architecture.

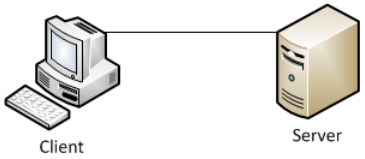
Step 8: Approach Analysis II

Original Architecture

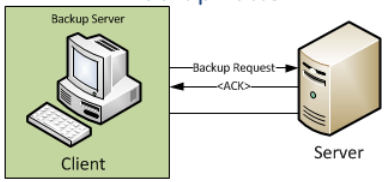


Option 1

Original Architecture

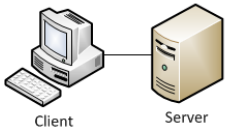


Architecture after applying Backup Pattern

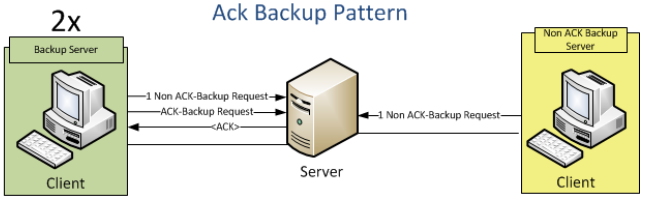


Option 2

Original Architecture



Architecture after applying Non Ack Backup Pattern



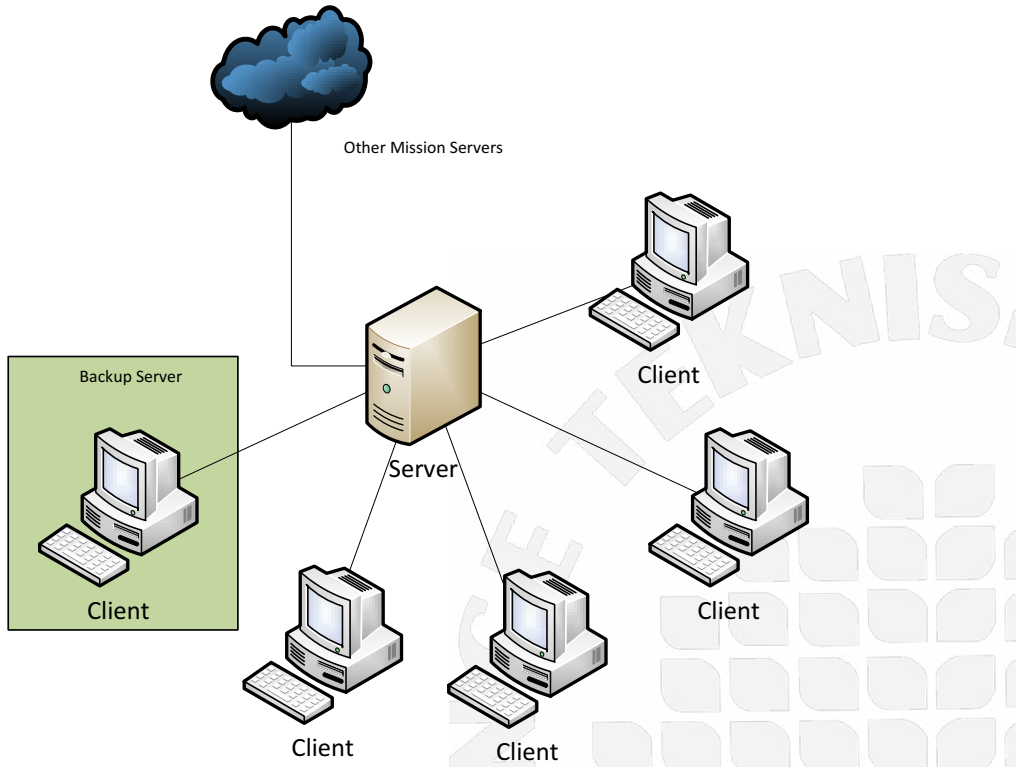
Scenarios

(H,M) The Systems Probability of failure should be 1 %

(H,M) The Communication Channel should be ready to send at least 99% of the time

Step 8: Approach Analysis III: Backup Server Alternative

- Analyze how the different approach give support to the quality attribute scenarios



Step 8: Approach Analysis III: Backup Server Alternative

Reliability Analysis:

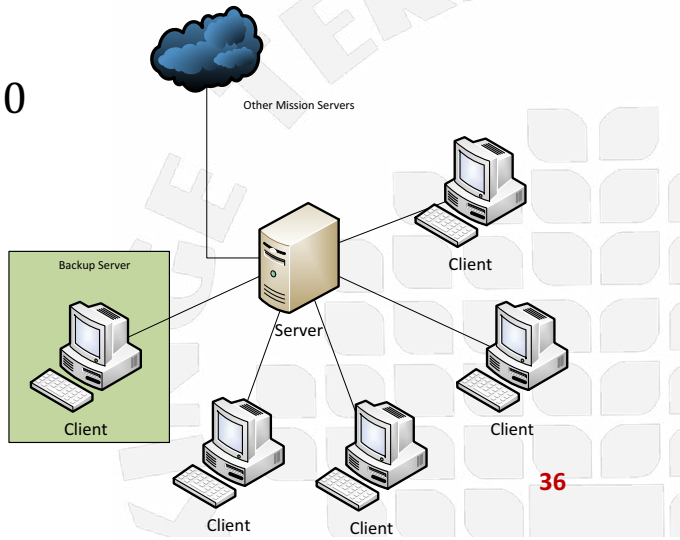
- The probability of failure of the system is now the cumulative probability of a simultaneous failure both on the server and on the backup server. P(System) = P(Server) * P(Backup Server)

Performance Analysis:

- We should analyze the percentage of the time in which the communication channel is used with mirroring purposes.
  - The backup server mirrors once each 10 minutes

$$Mirroring\ Ussage = \frac{{}^{55Kb}/_{9.600Kbps}}{10 * 60} * 100$$

$\%\ Ready\ To\ Send$ = 100- $Mirroring\ Ussage$



Step 8: Approach Analysis III: Backup Server Alternative

- **Reliability: Probability of Failure**

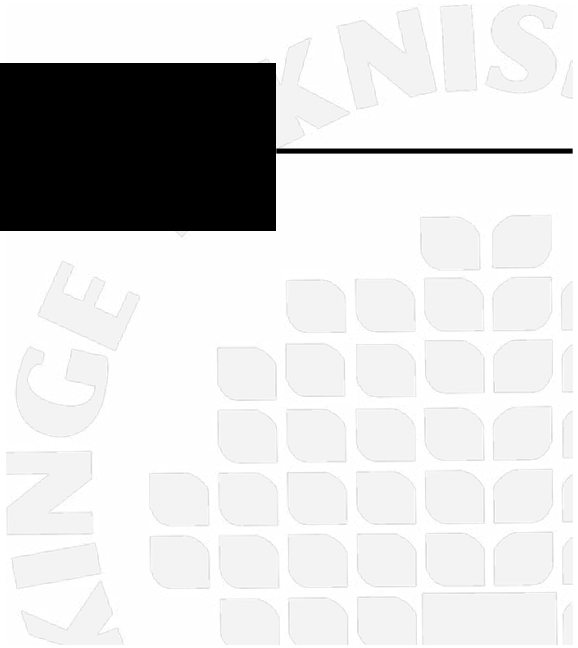
  Value Obtained: 0,25%

- **Performance: % of Time Ready To Send**

  Value Obtained: 99%

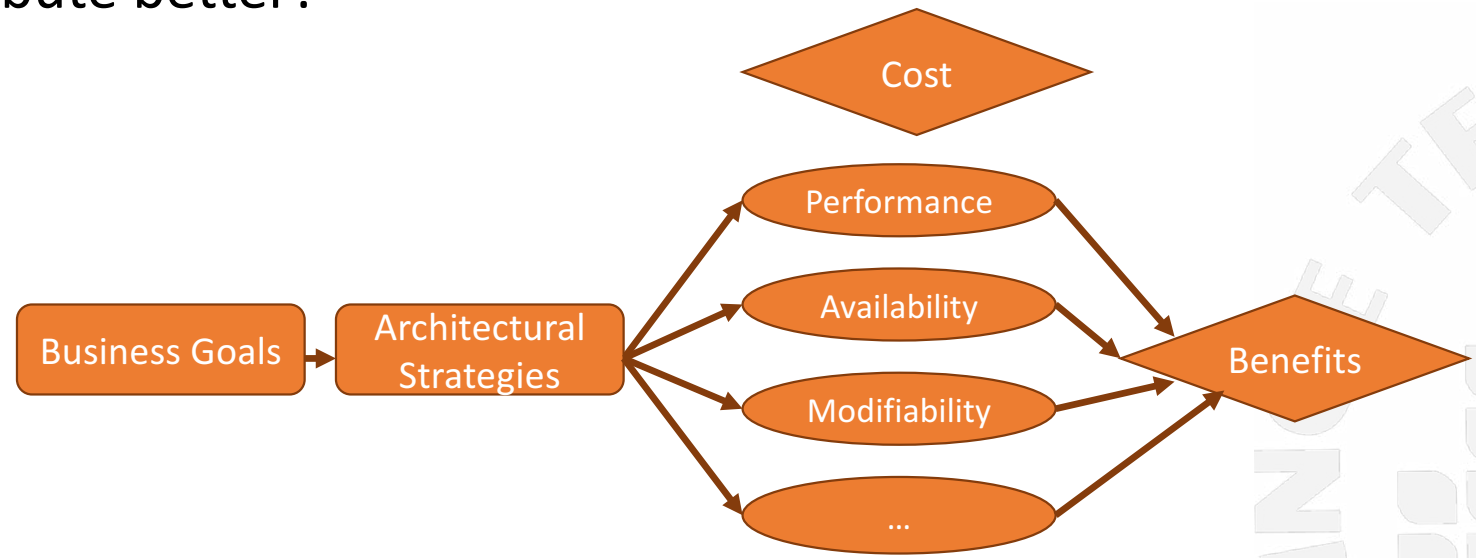
Architecture Evaluation and Transformation

Evaluation Methods: CBAM



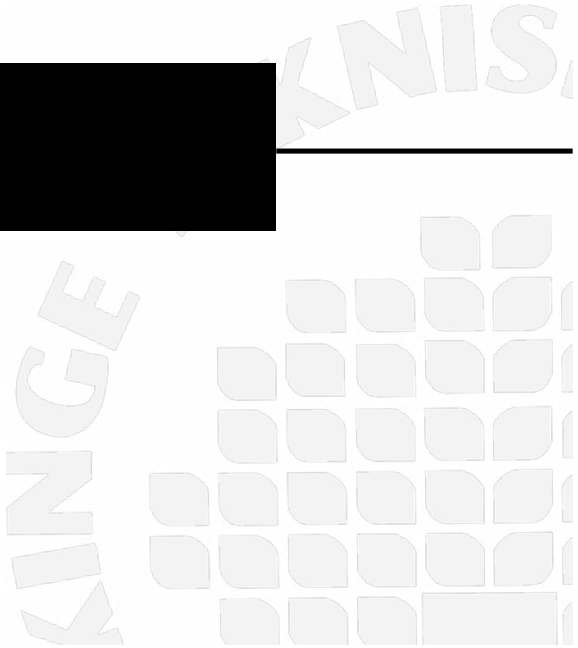
CBAM: Cost Benefit Analysis Method

- Starts where ATAM ends
- Analyzes what is the $Utility = \frac{Cost_i}{Benefit_i}$ of supporting a quality attribute better?

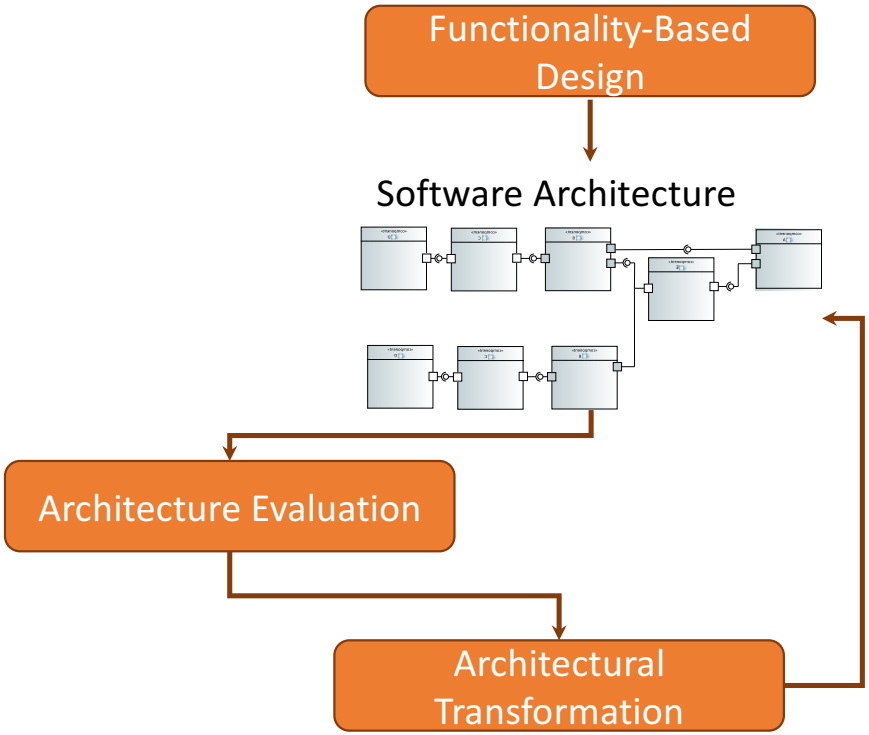


Architecture Evaluation and Transformation

Evaluation Methods: QASAR



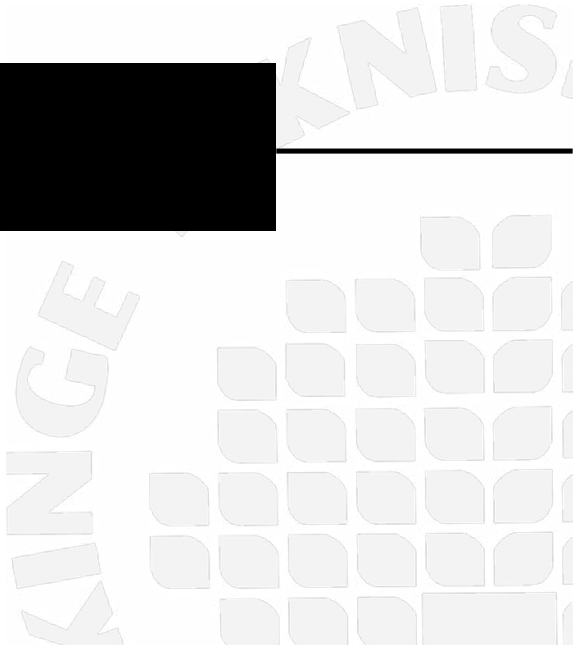
QASAR



J. Bosch, *Design and Use of Software Architecture - Adopting and evolving a product-line approach*. Pearson Education, 2000.

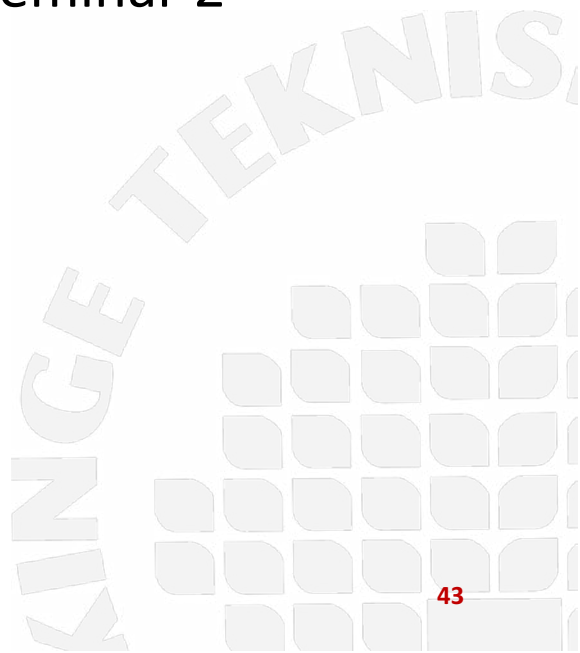
Architecture Evaluation and Transformation

Evaluation Methods: BTH 4-hours



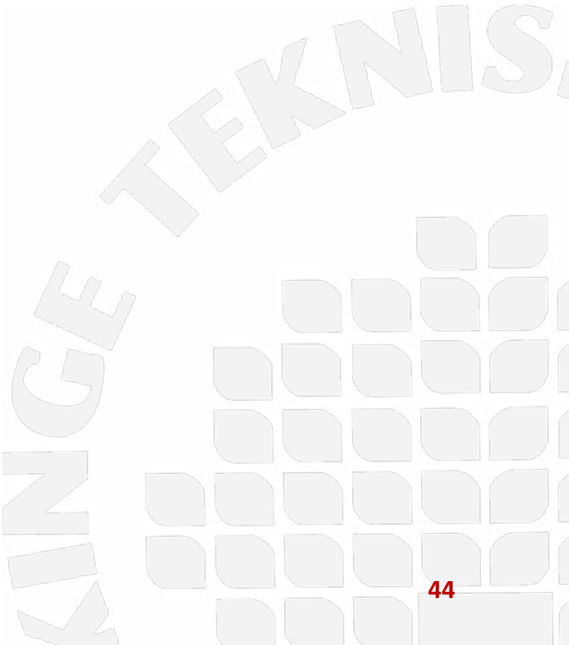
BTH 4-hour Architecture Evaluation

- Shorter variant of SAAM/ATAM
- Used primarily for student projects
- Applied also among industrial partners
- Will be applied in the peer-reviewed evaluation on Seminar 2



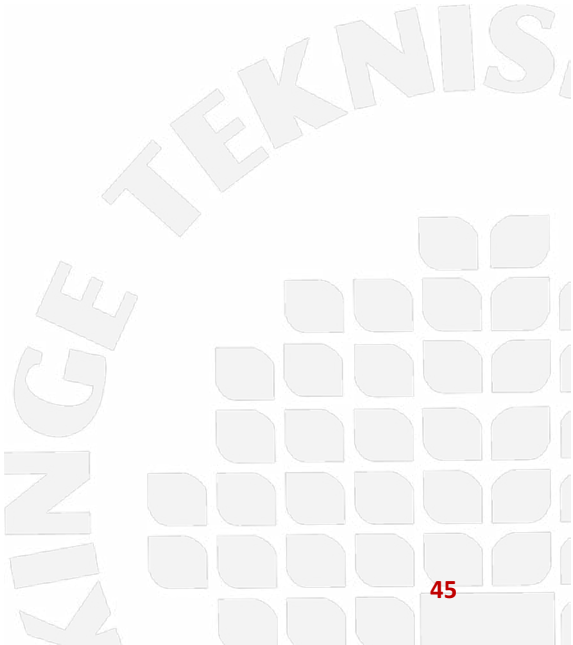
BTH 4-hour Architecture Evaluation

Step	Name	Time
1	Introduction	10min
2	Identify Quality Requirements	20min
3	Elicit Scenarios	30min
4	Architecture presentation	2x15min
5	Break	20min
6	Scenario and Architecture Analysis	2x40min
7	Analyze Approaches	2x20min
8	Conclusion	15min



Evaluation Experiences

- Size of Development Team
- Clear Objective
- Present Recipients of Evaluation Document
- Moderate Pursuit of Issues
- Use Extreme Scenarios
- The Impact of the Project Manager
- Summarize Often



Size of the evaluation team

- 3-4 persons in the Evaluation Team
  - Fewer is harder
  - More may intimidate the team being evaluated
- Task division:
  - One person documents
  - The others take turn in thinking / pursuing issues

Clear Evaluation Objective: Present Recipients of Evaluation

- Open target = no end criterion
- Need clear objective to decide on most appropriate method and most appropriate participants
- Avoid guessing about the objectives
  - For example, *"You are here to fail us and stop the project!"*
- Make sure there are clear benefits for the project

Moderate Pursuit of Issues

- Conflict: You are there to find flaws, but if you do not know when to "let go" the project will become defensive and uncooperative.
- Knowing when to back down is not only a technical skill.
- Difficult to identify and investigate ripple effects.
  - **Solution:** Leave warnings in the evaluation documentation.

Use extreme scenarios

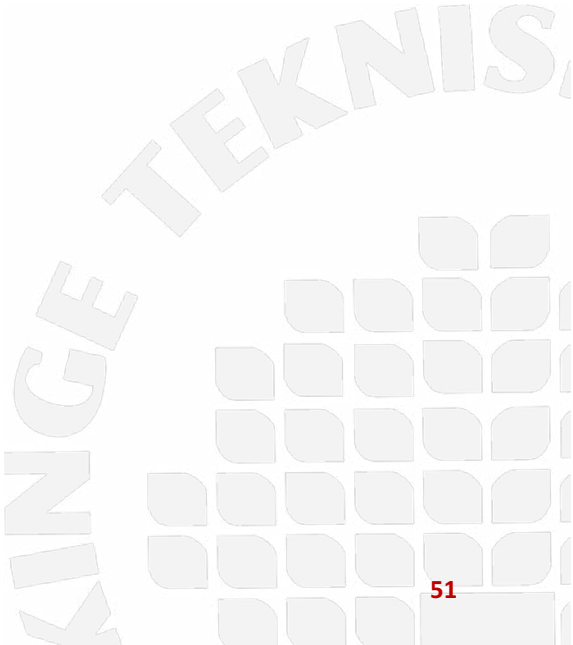
- Use Reduction ad absurdum:
  - The absurd may jolt the project into defining limits
- Investigate a normal scenario and several extremes, where the boundaries of the requirements and the system are tested
- Be open to pursue promising paths
  - For example, with even more extreme scenarios even if it is out of the scope of what you planned

The Impact of the Project Manager in Evaluations

- It is *absolutely vital* that the project manager understands the benefits of the evaluation.
- The project manager is the lest technical of the project members (not always)
  - Perceives pressure from mid-level management
- Group issues: Do the other project members dare speak up against their project manager?

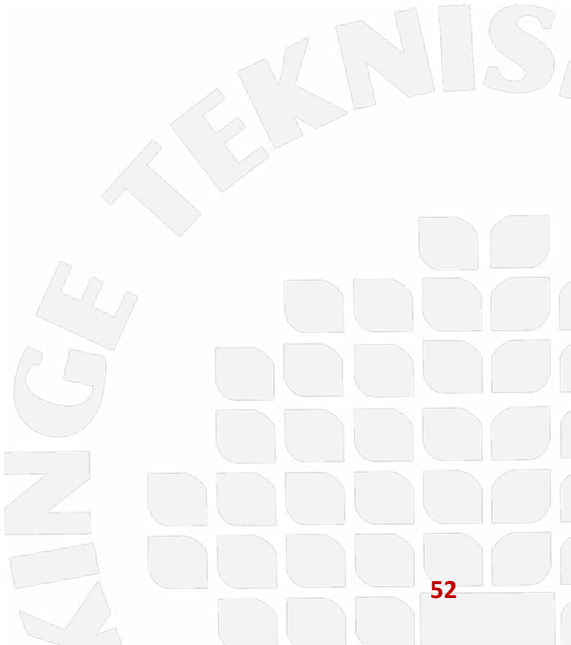
Summarize often

- Keep the evaluation and the project “on track”
- Make sure that found issues are clearly presented and understood.



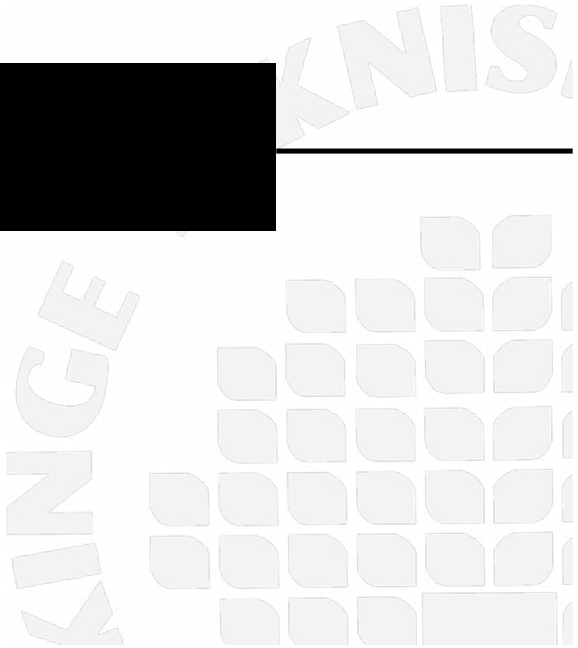
Evaluation Experiences

- Size of Development Team
- Clear Objective
- Present Recipients of Evaluation Document
- Moderate Pursuit of Issues
- Use Extreme Scenarios
- The Impact of the Project Manager
- Summarize Often
- Guidance – not Lecturing
- Avoid Grading Tension
- Make the Architecture Matter
- Encourage Peer Evaluation



Architecture Evaluation and Transformation

Architectural Transformations



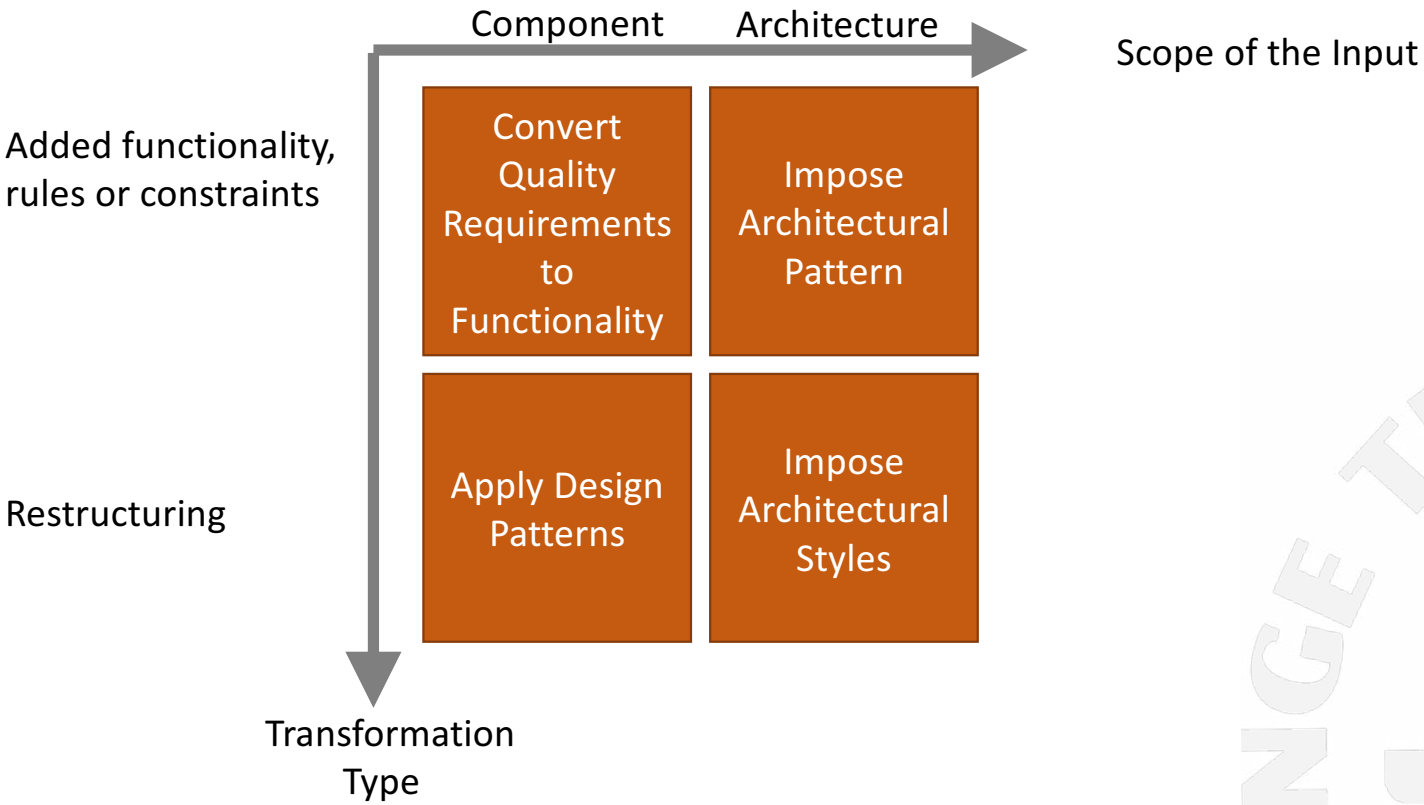
Architectural Transformations

- If during the evaluation we identify weak points in the architecture, we change it (to better “armor” the architecture)
  - Provided that the ROI of making the change makes it worthwhile
  - And provided is not easier to renegotiate the requirements
- Bosch labels this modifications as *architectural transformations*
- Modify the architecture keeping the domain functionality invariant
  - Only impacting the quality attributes of the system
- A given transformation might improve some quality attributes but might deteriorate some others

Architectural Transformations: Steps

- Identify quality requirements that are not yet fulfilled
- Identify at what components or locations in the architecture is the quality attribute inhibited
- Select transformations that will solve the identified problems in the architecture
- Transform the architecture accordingly

Architectural Transformations



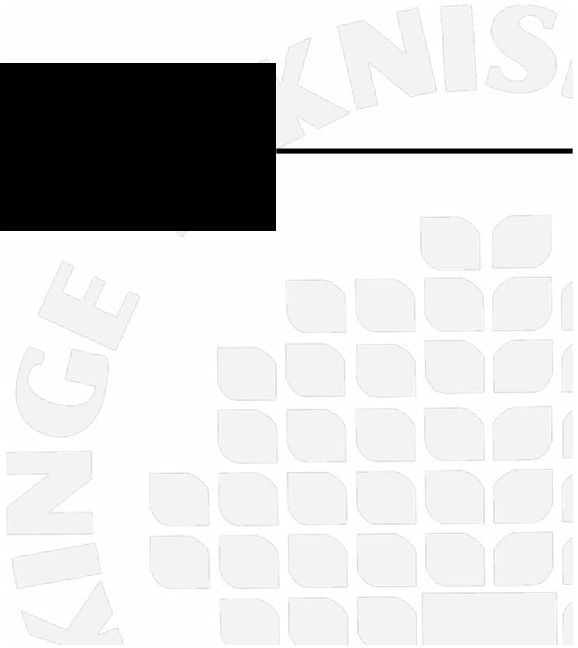
J. Bosch, *Design and use of software architectures: adopting and evolving a product-line approach*. ACM Press/Addison-Wesley Publishing Co., 2000.

Types of Architectural Transformations

- From bigger scope to smaller
  - **Impose architectural style**
    - Reorganize the whole architecture (of the whole system or of a given subsystem)
    - **Examples:** Pipes and filters, layers, blackboard, model view controller
  - **Impose architectural pattern**
    - Maintain the components and their functionality but introduced some new behavior
    - **Examples:** Introduce concurrency, introduce redundancy, persistency, distributed communication mechanisms
  - **Apply design patterns**
    - Reorganize a single component, or a small set of components according to the principles of a design pattern.
    - **Examples:** Facade, Observer, Abstract Factory
  - **Convert Quality Requirements into functionality**
    - Add functionality not from the application domain to solve some quality issues
    - **Examples:** add undo/history functionality, self monitoring, exception handling

Architecture Evaluation and Transformation

Summary



Summary

- We have different times and different reasons to carry out architecture evaluation
- The purpose and time restricts the method to be used
- Some of the most well-known, commonly-used methods are SAAM and ATAM for scenario-based evaluations
- BTH has developed some contributions to the field, with the QASAR method and the BTH 4-evaluation method
- There are other metric-based ways of evaluating architectures, including some model-based evaluations (such as the ones supported by AADL or the QuaDAI method)

Software Architectures and Quality

Architecture Evaluation and Transformation

Javier González Huerta

javier.gonzalez.huerta@bth.se

BLEKINGE TEKNISKA HÖGSKOLA · BTH ·

in real life