Evaluation of Software Architecture

Truong Ho-Quang
truongh@chalmers.se

Schedule

Week		Date	Time	Lecture	Note
36	L1	Wed, 2 Sept	13:15 – 15:00	Introduction & Organization	Truong Ho
37	L2	Wed, 9 Sept	13:15 – 15:00	Architecting Process & Views	Truong Ho
37	S1	Thu, 10 Sept	10:15 – 12:00	<< Supervision/Assignment>>	TAs
38	L3	Wed, 16 Sept	13:15 - 15:00	Requirements & Quality Attributes	Sam Jobara
38	S2	Thu, 17 Sept	10:15 – 12:00	<< Supervision/Assignment>>	TAs
38	L4	Fri, 18 Sept	13:15 – 15:00	Architectural Tactics & Roles and Responsibilities	Truong Ho
39	S3	Wed, 23 Sept	13:15 – 15:00	<< Supervision/Assignment>>	TAs
39	L5	Thu, 24 Sept	10:15 – 12:00	Functional Decomposition & Architectural Styles P1	Truong Ho
39	L6	Fri, 25 Sept	13:15 – 15:00	Architectural Styles P2	Truong Ho
40	S4	Wed, 30 Sept	13:15 – 15:00	<< Supervision/Assignment>>	TAs
40	L7	Thu, 1 Oct	10:15 – 12:00	Architectural Styles P3	Sam Jobara
40	L8	Fri, 2 Oct	13:00 – 15:00	Guest Lecture: Scaling DevOps – GitHub's Journey from 500+ to 1500+ People	Johannes
41	S5	Wed, 7 Oct	13:15 – 15:00	<< Supervision/Ass	
41	L9	Thu, 8 Oct	10:15 – 12:00	Current Industrial SW Architecture Issues Architectures of Blockchain with Case Stu	ara
42	L10	Wed, 14 Oct	13:15 – 15:00	Design Principles	o
42	S6	Thu, 15 Oct	10:15 – 12:00	<< Supervision/Assig	
42	L11	Fri, 16 Oct	13:15 – 15:00	Guest Lecture: Architecture changes at Volvo Truck's Application System (TAS)	Anders Magnusson
43	L12	Wed, 21 Oct	13:15 – 15:00	Clarification: Deployment Diagram, Solution Ass.1	Truong Ho
43	L13	Thu, 22 Oct	10:15 – 12:00	Architecture Evaluation	Truong Ho
43		Fri, 23 Oct	13:00 – 15:00	To be determined (exam practice?)	Teachers
44	Exam	30 Oct	8:30 – 12:30		

We are HERE!

Canvas page of the final exam is published!

- Check the link
  https://chalmers.instructure.com/courses/13072
- If you cannot access, contact student office (student_office.cse@chalmers.se) for help!
- If you have special needs/requests during the exam, come talk to me!

Outline of Topics for Today's Lecture

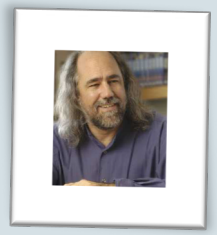
- Clarification: Forces and Drivers
- Evaluation of Software Architecture
  - **What** is architecture evaluation!
  - Evaluation **approaches**!
  - **Benefits** and **limits** of architecture evaluation!
  - **ATAM** as evaluation method!
    - Architecture Tradeoff Analysis Method
  - **Example** Evaluation

FORCES & ARCHITECTURAL DRIVERS

Forces that affect the Design

"In physics, a force is any influence that causes an object to undergo a certain change, either concerning its movement, direction, or geometrical construction.“
(wikipedia, Force)

The "software forces" image of below is from Grady Booch's Models09 keynote, The Other Side of Model Driven Development (2009):



Example of forces

- Business **constrains**
  - Time/Schedule
  - Budget
  - Team composition
  - Software licensing restrictions or requirements
- Technical **constrains**
  - Programming language
  - Operating system or platforms supported
  - Use of a specific library or framework

Tip: *"Seperate the constraints you are given from the constraints you give yourself"*

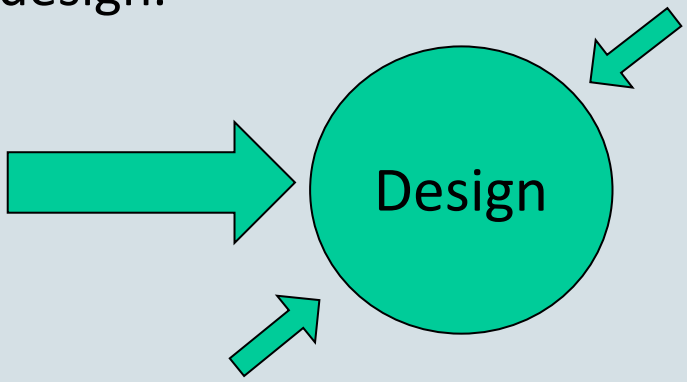
Michael Keeling in this blog post

Architectural Drivers

- **Architectural drivers** are the design **forces** that will influence the **early design decisions** the architects make

- Architectural drivers are not all of the requirements for a system, but they are those requirements that are **most influential** to the architectures design.

- The ’art’ of the architect is to identify which forces have the strongest effect on the architecture-design.



Forces vs Drivers

- There is no clear separation between forces and drivers
- Identification of architectural drivers is very contextual. This often bases on:
  - Architect's experience
    - Pitfalls: Noone knows everything!
  - A thorough architectural reviews/evaluations
    - Business value, architectural impacts

What to keep in mind?

- **Always** mind **what** you are/will be architecting!
  - Input/output
  - What constrains are relevant?

Enterprise architecture

System architecture

Subsystem

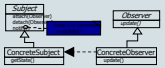
Application architecture

Application

Macro-architecture

Frameworks

Micro-architecture



Design patterns

Is ‘cost’ an architectural driver?

- This is an ‘ultimate’ driver to any aspects of software development projects
- ‘Cost’ affects
  - Functionalities (quality & quantity)
  - Quality of the system
  - Technical choices
- What happens when considering ‘cost’ in any design decision?
  - As an architect, you cannot decide everything!

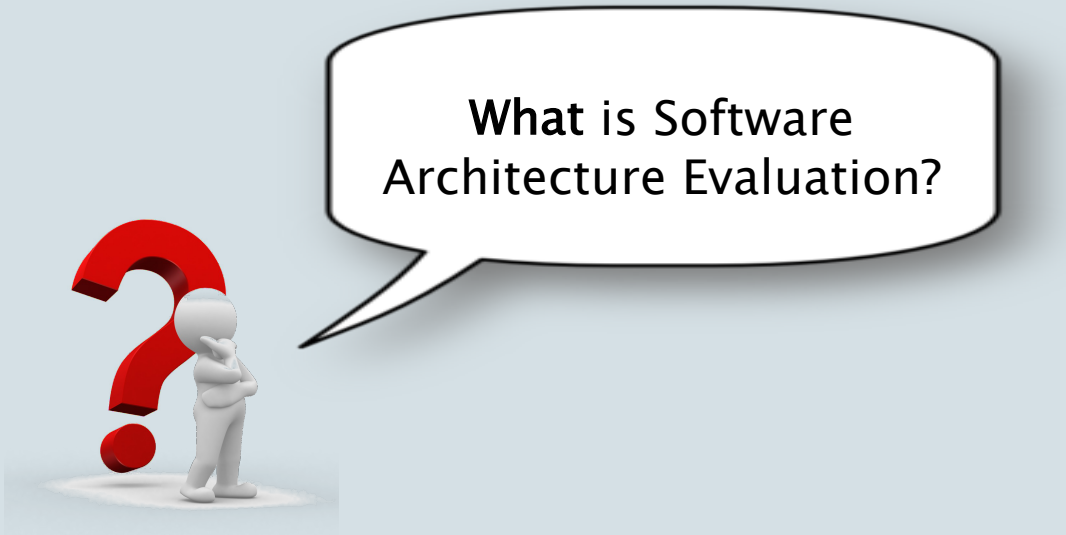
- My advice:
  - Cost should be treated in project management level.
  - Ask “stakeholders” to break down the cost-constrains to concrete functional and non-functional constrains (as input for requirements).

Is 'Quality' an arch. driver?

- YES, but 'Quality' itself is too generic!
  - 'Quality' cannot be measured!
- Quality is often viewed through specific set of quality attributes
- It's important to point out what aspect of quality the software/system should fullfil:
  - Performance
  - Availability
  - Maintainability
  - …

Is ‘Functionality’ an arch. driver?

- YES, but it is an ‘ultimate’ driver, too.
- Functionalities affects the design in many ways
  - Functional sub-system/components
  - Domain-specific logics
  - Interaction between these components
  - …
- Many tools are being used to address the functional aspect of sw system
  - Funtional decomposition
  - Functional testing
  - …


What is Software Architecture Evaluation?

What is Architecture Evaluation?

Architecture Evaluation is the **process** of determining **how well** the current design or a portion of it **satisfies** the **requirements** derived during analysis.

- Key **questions**:
  - How can you be sure whether the architecture chosen for your software is a **right one**?
  - How can you be sure that it **won't lead to calamity** but instead will pave the way through a **smooth development** and **successful product**?


What to evaluate in Software Architecture?

What to Evaluate?

"fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution."

ISO/IEC/IEEE 42010

"The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them.."

Len Bass

What to Evaluate?

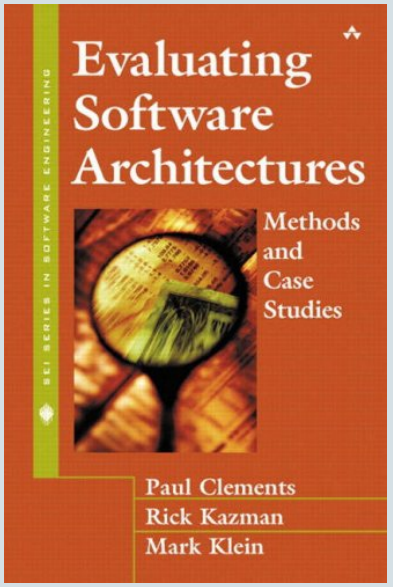
If architectural decisions determine a system's **quality attributes**, then it is possible to evaluate **architectural** respect to their **impact** on those attributes**decisions** with.

Architects pay more attention to **qualities** that arise from architecture choices.

Architectures allow or preclude nearly all of the system's **quality attributes**.

What to Evaluate?

"… the evaluator is able to conclude that **a quality goal is sensitive** to certain properties of the architecture. A goal of any architecture evaluation is to make this reasoning **explicit** and to **record** it for posterity." *

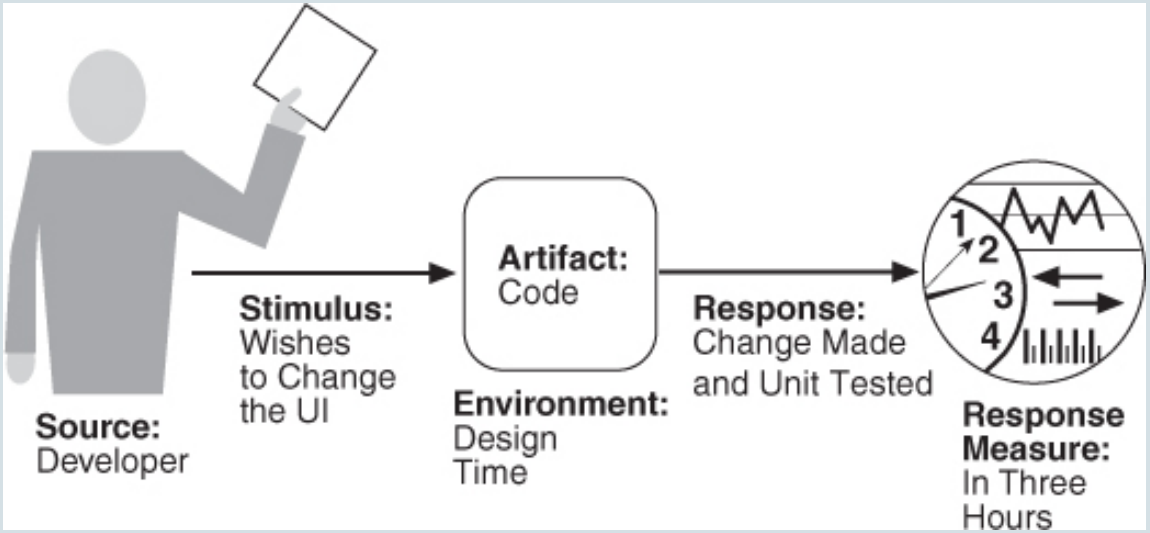


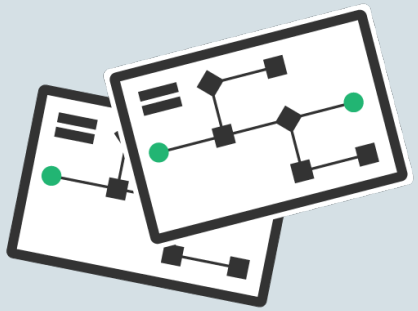
* Clements et al.


How to evaluate
Software Architecture?

Evaluating Quality Attributes

- Quality attributes can be evaluated through:
  - **Scenario-based evaluation**: for example *change scenarios* for assessing maintainability





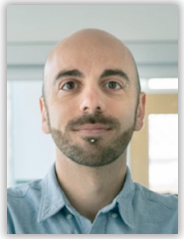
ICT is unsustainable

Total number of views:

3,362,297,996

Total energy per view:

0.2 kWh



Slide by
Ivano Malavolta

Total energy consumed: ~672 GWh in less than 7 years

x27,000 for 7 years!

https://www.youtube.com/watch?v=9bZkp7q19f0
http://shrinkthatfootprint.com/average-household-electricity-consumption

Where does this energy go?

Battery charge efficiency: 90%

CPU: 500 - 2,000 mW

GSM: 800 mW

Display: 400 mW

GPS: 176 mW

Gyroscope: 130 mW

Microphone: 101 mW

Bluetooth: 100 mW

Accelerometer: 21 mW

Evaluating Quality Attributes

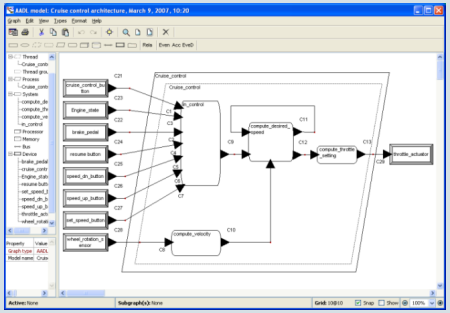
- Quality attributes can be evaluated through:
  - **Simulation**: for example *Prototyping* is a form of simulation where a part of the architecture is implemented and executed in the actual system context

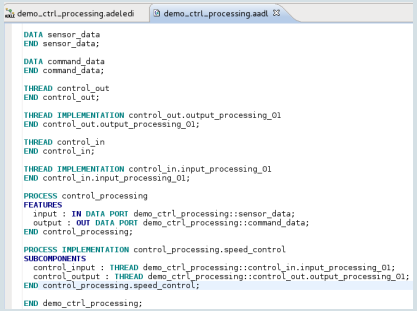
E.g. Usability

Armour Driving Simulator (ADS)

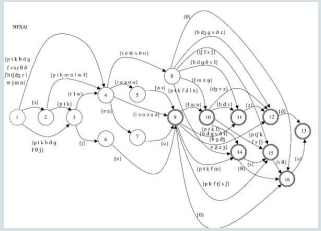
Evaluating Quality Attributes

- Quality attributes can be evaluated through:
  - **Mathematical modeling**: for example, checking for *potential deadlocks*





Architecture Description Languages



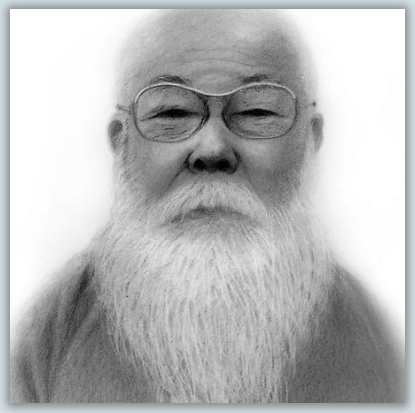
Performance
e.g. Queueing Networks

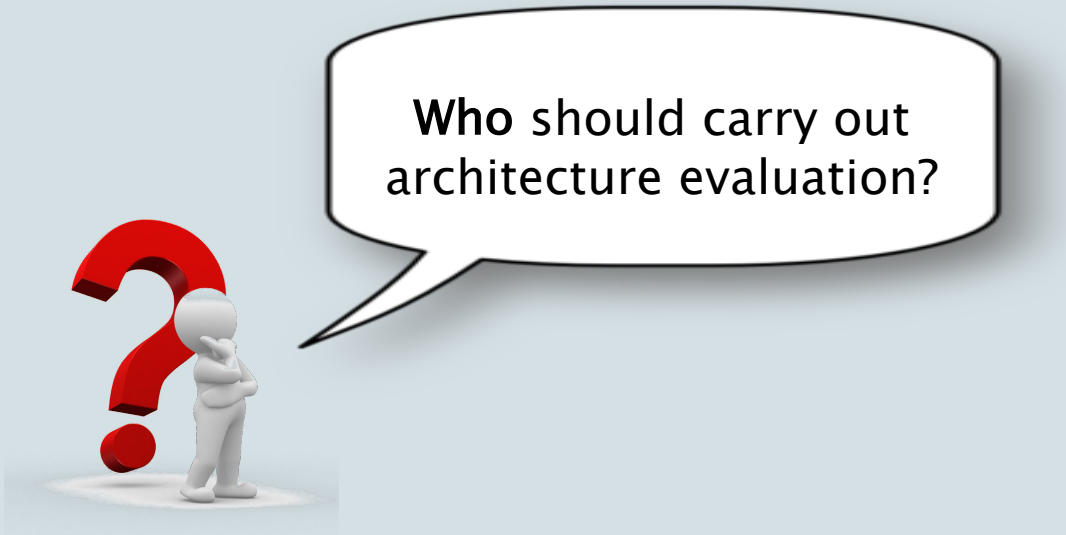
Safety
e.g. Fault-Tree Analysis

Evaluating Quality Attributes

- Quality attributes can be evaluated through:
  - **Experience-based assessment**:

    this is based on *subjective factors* like intuition, experience and expertise of software engineers




Who should carry out architecture evaluation?

Who!

- Evaluation by the **designer**
  - Every time a key design decision or a design milestone is completed.

- **Advantages**:
  - Familiarity with the system
  - Minimal overhead
- **Limitations**:
  - Personal bias
  - Dominant architect perspective

Who!

- **Peer** review
  - Peer = experienced colleague on the project, but not the architect
  - At any point of the design process where a candidate architecture exists.

- **Advantages**:
  - Familiarity with the system
  - Multiple perspectives
- **Limitations**:
  - Organization bias
  - Limited availability

Who!

- Analysis by **outsiders**
  - Architecture-specialists and experts.

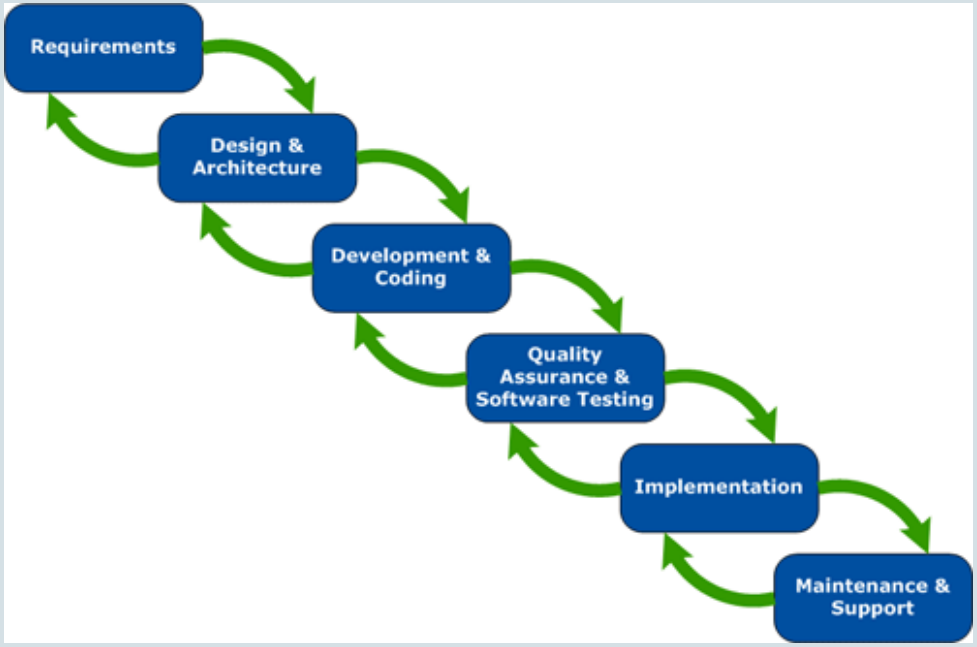
- **Advantages**:
  - Minimal bias
  - Expert recommendations

- **Limitations**:
  - Start-up time / getting up to speed
  - High expenses
  - Confidentiality issues


When to carry out software architecture evaluation?

When?

- **Early**: Examine those architectural decisions already made and choose among architectural options that are pending.



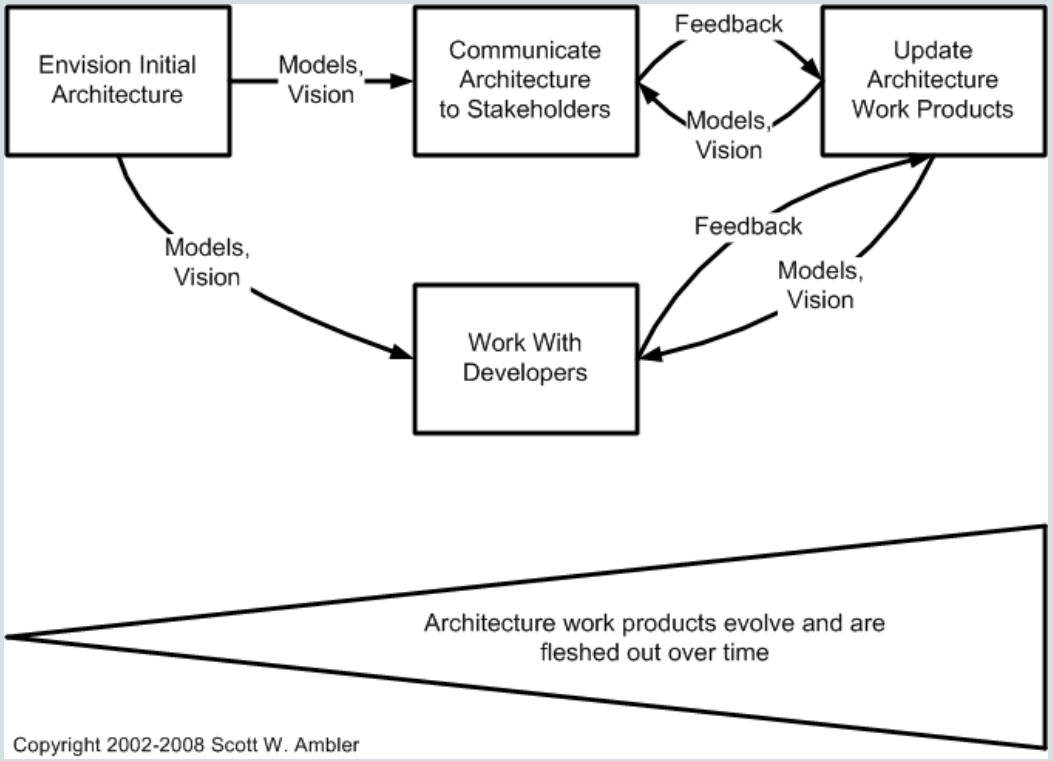
When?

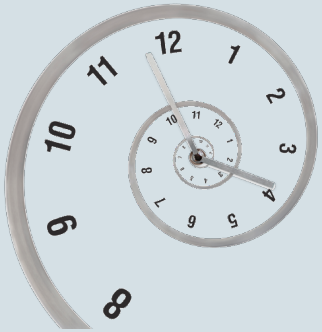
- **Late**: The implementation is complete (e.g. using a legacy system).

Open Source Software

When?

- **Continuous**: Evaluation at each development iteration.



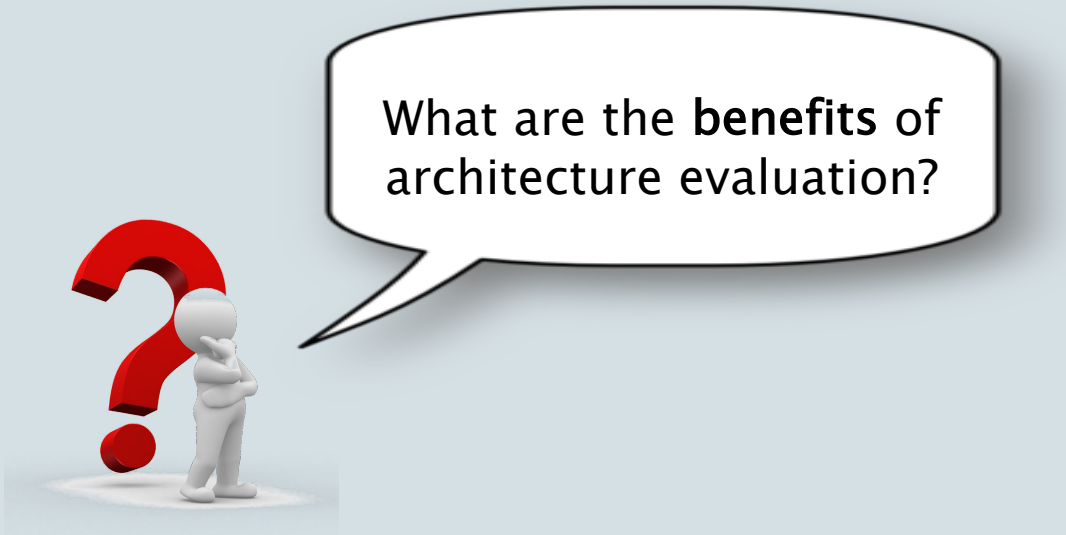


When commissioning/buying a system

buyer

Which of the offered systems fits best in my system ?

sellers


What are the **benefits** of architecture evaluation?

Results of Software Evaluation

- Is this **architecture suitable** for the system for which it was designed?

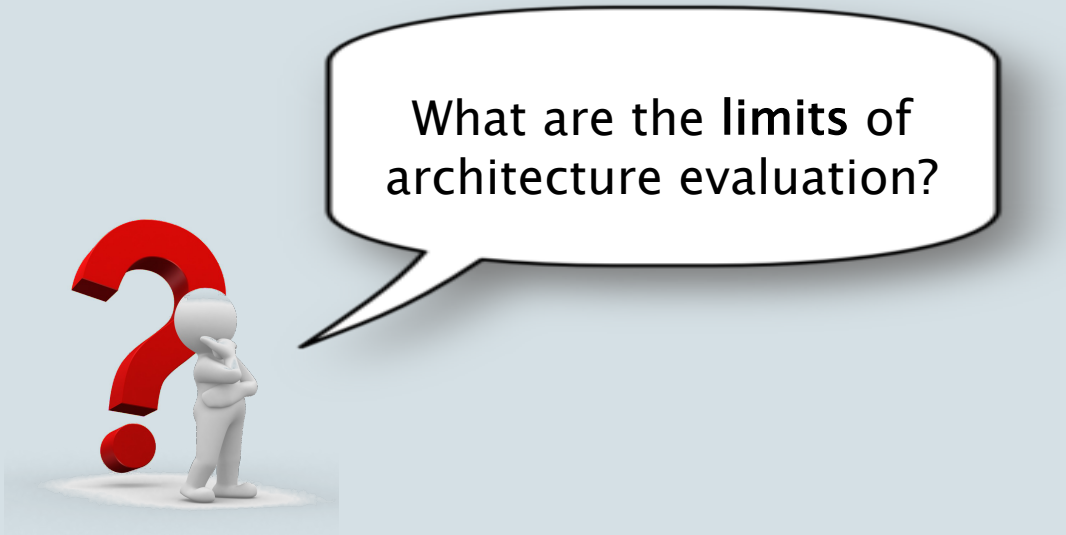
- **Which among several** competing architectures is the most suitable one for the system at hand?
  - System will meet its **quality goals**
  - System will provide the required **behavioural function**
  - System will be developed according to its **design constraints**
  - System can be built using the **resources** at hand

An architecture evaluation doesn't tell you
"yes" or "no," "good" or "bad," or "6.75 out of 10."
*It tells you **where you are at risk.***

* Clements et al.

Benefits of Architecture Evaluation

- Puts stakeholders in the same room
- Forces an articulation of specific quality goals
- Results in the prioritization of conflicting goals
- Forces a clear explication of the architecture
- Improves the quality of architectural documentation
- Uncovers opportunities for cross-project reuse
- Results in improved architecture practices


What are the **limits** of architecture evaluation?

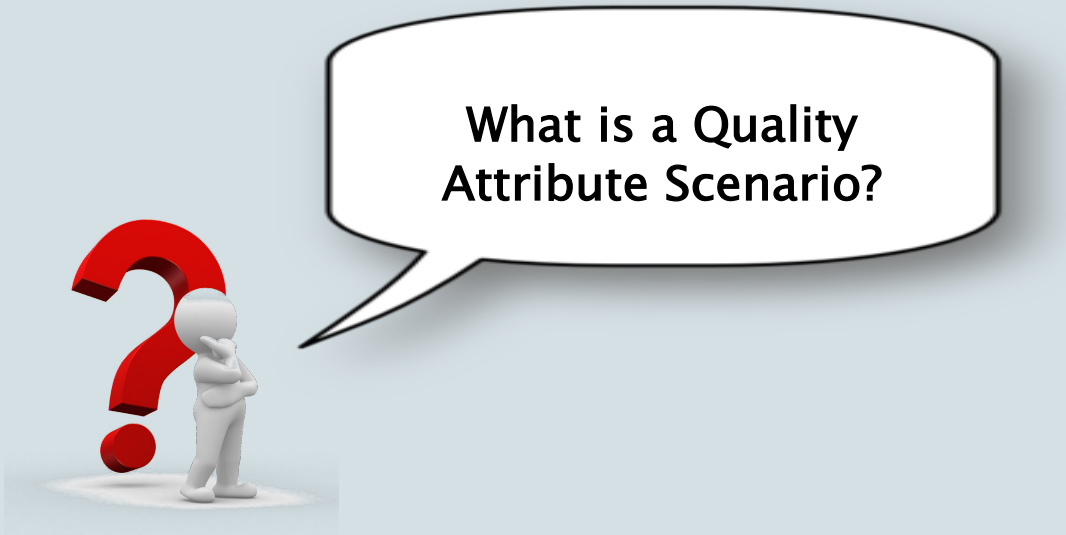
Evaluation Challenges

- What **artefacts** are available?
- What **resources** are available?
- Who **sees** the results?
- Who **performs** the evaluation?
- Which **stakeholders** will participate?
- What are the **business goals**?
- What **tools** are available?


What is ATAM?

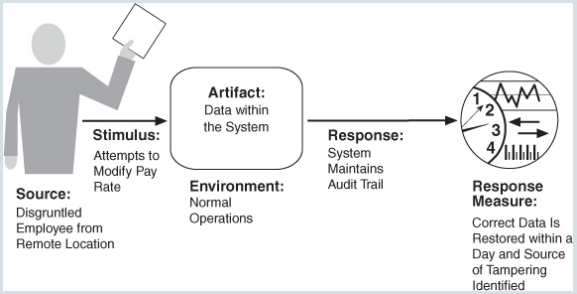
Architecture Tradeoff Analysis Method – ATAM

- ATAM: Architecture Tradeoff Analysis Method
  - A **scenario-based** architecture method for assessing quality attributes such as: modifiability, availability, and security.
- Evaluators need **not be familiar** with the architecture or its business goals
- System need **not** yet be **constructed**
- A large number of **stakeholders** are involved

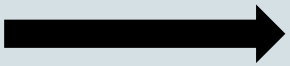

What is a Quality Attribute Scenario?

ATAM: Quality Attribute Scenario

- A Quality Attribute Scenario is a quality attribute specific requirement.
  - Source of stimulus (e.g., human, computer system, etc.)
  - Stimulus – a condition that needs to be considered
  - Environment – what are the conditions when the stimulus occurs?
  - Artifact – what elements of the system are stimulated.
  - Response – the activity undertaken after arrival of the stimulus.
  - Response measure – when the response occurs it should be measurable so that the requirement can be tested.



Example Quality Scenario for Security



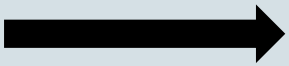
Artifact

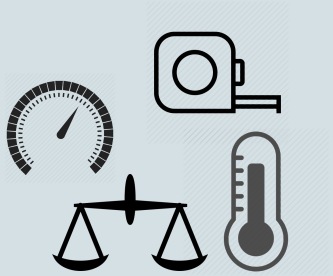
System services

Data within the system

Component/resource of the system

Data produced/consumed by the system

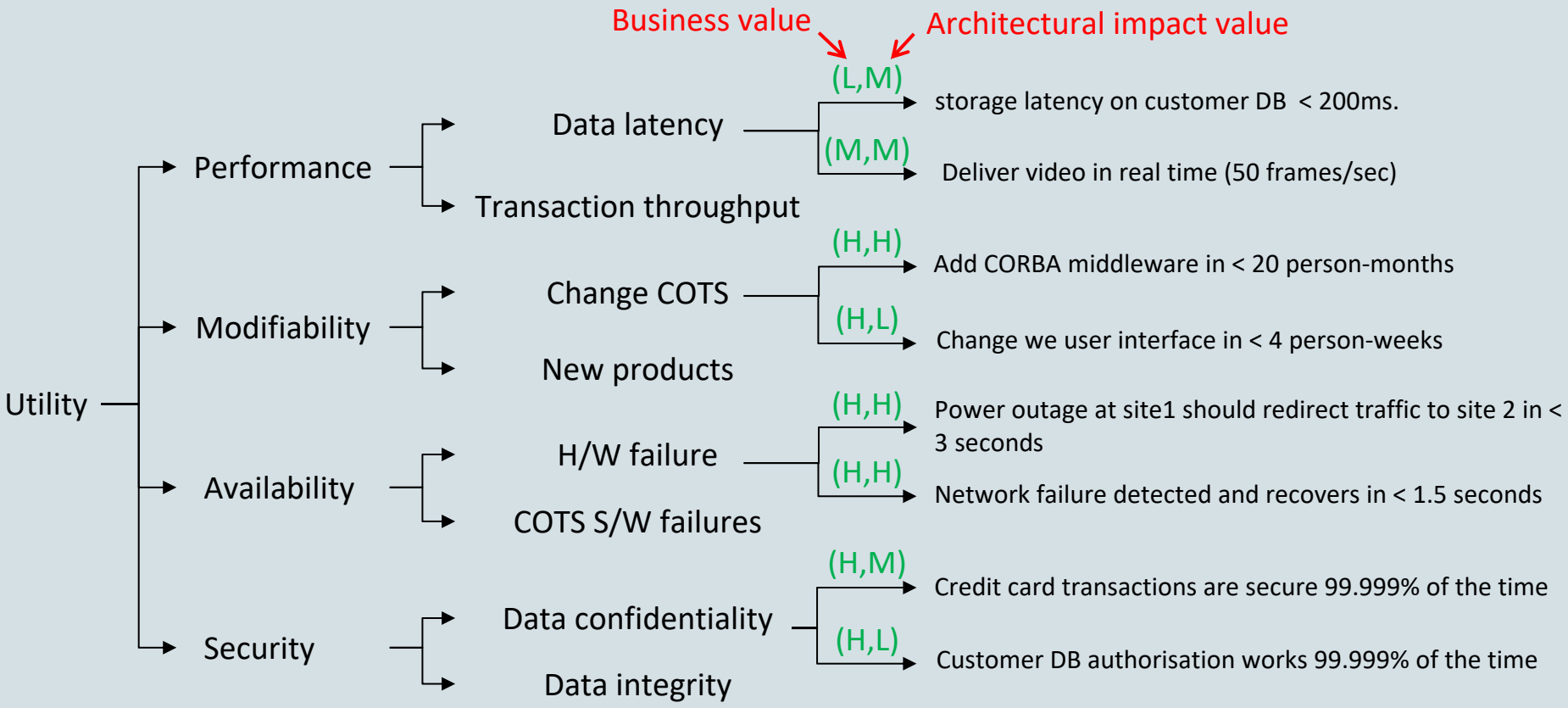




Source	Stimulus	Environment	Response	Measure
Identified user	Attempt to display data	Normal mode	Lock Computer	Latency
Unknown user	Attempt to modify data	Overload mode	Maintain Audit trail	Deadline
Hacker from outside the organisation	Attempt to delete data	Reduced capacity mode		Throughput
Hacker from inside the organisation	Access system services	Emergency mode		Jitter
	Change system's behaviour	Peak mode	Should be SMART!	Miss rate
	Reduce availability			Data loss


But how to elicit and identify Quality Attribute Scenarios?

Utility Tree

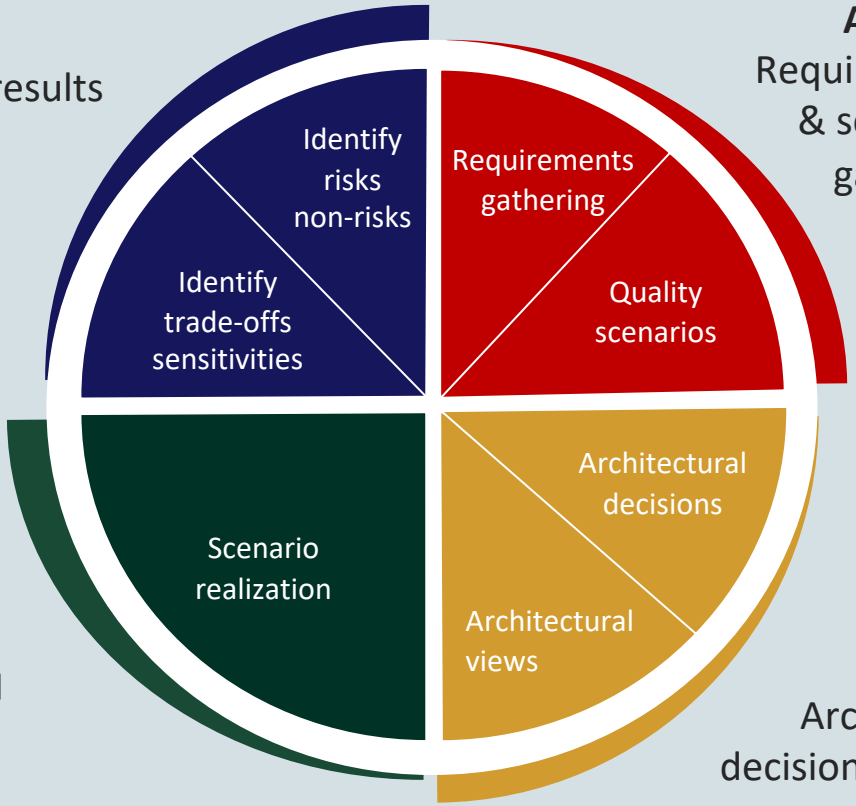



What are the activities involved in ATAM?

ATAM Activities

Activity IV
Evaluation results

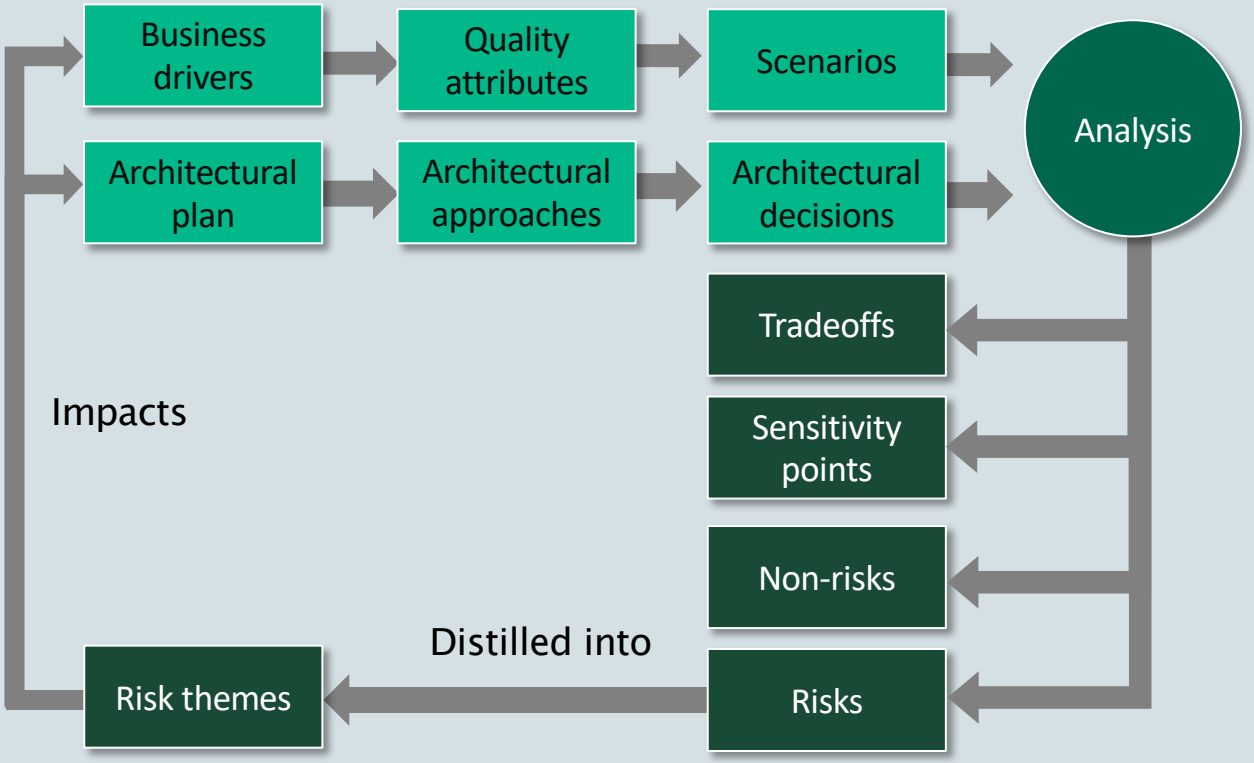
Activity I
Requirements & scenarios gathering



Activity III
Analysis

Activity II
Architectural decisions & views

ATAM Conceptual Flow



ATAM Output

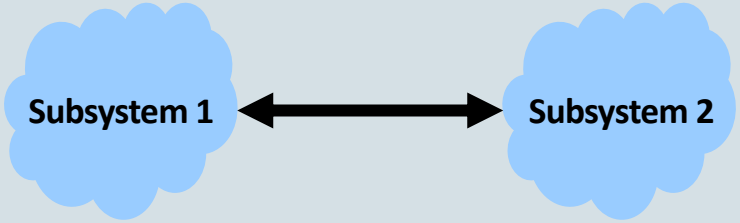
Item	Description
Sensitivity point	A property of one or more components (and/or component relationships) that is critical for achieving a particular quality attribute response
Tradeoff point	An architectural decision that affects more than one quality attribute (possibly in opposite ways)
Risk	Architectural decision that may lead to undesirable consequences
Non risk	Architectural decision that is deemed safe
Risk theme	A general concern of a group of interrelated risks in a design, assigned its own risk value

Sensitivity Point

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

A system requires

- high performance



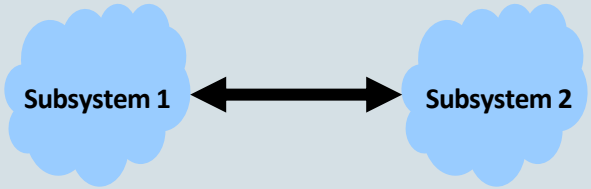
Suppose throughput depends on one channel

increase channel speed → increase performance

Trade-off point

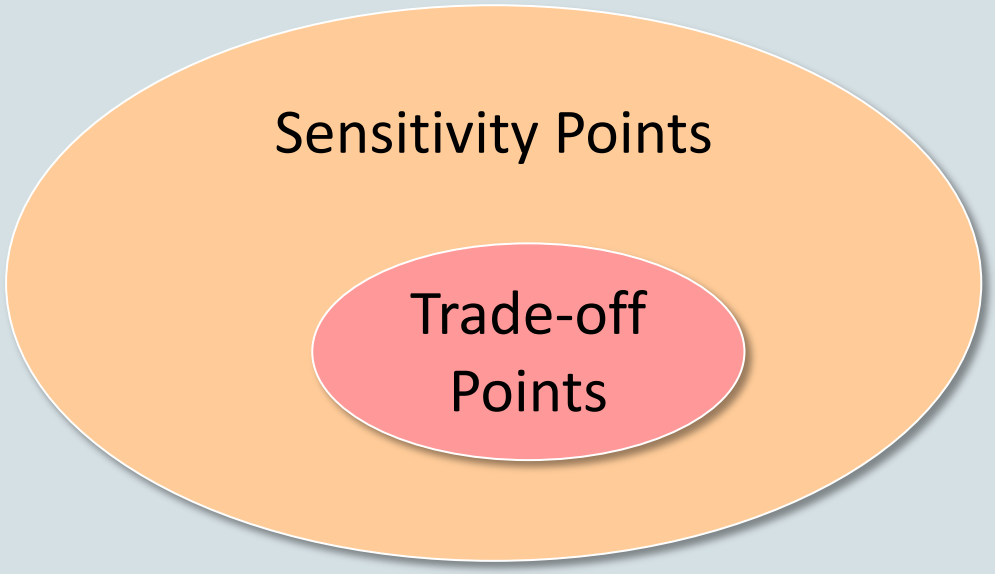
A **trade-off point** is a parameter of the architecture that affects multiple quality attributes in opposite directions.

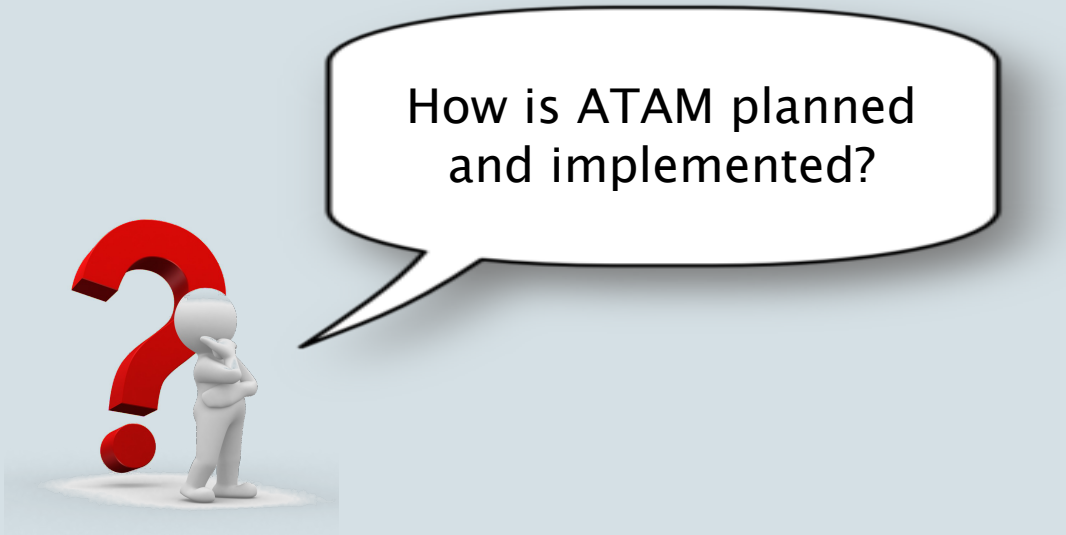
A system requires
- high performance,
- high reliability
- high security



increase channel speed → increase performance ↑ & decrease reliability ↓

increase encryption → increase security ↑ & decrease performance ↓


Sensitivity Points
Trade-off Points


How is ATAM planned and implemented?

ATAM Phases

Evaluation client

Evaluation team

Project decision makers

Stakeholders

ATAM Phases

Evaluation client

Evaluation team

Project decision makers

Stakeholders

Phase 0
Partnership and preparation

Proceeds informally as required, perhaps over a few weeks

ATAM Phases

Evaluation client

Evaluation team

Project decision makers

Stakeholders

Phase 0
Partnership and preparation

Proceeds informally as required, perhaps over a few weeks

Phase 1
Initial evaluation

1-2 days followed by hiatus of 1-3 weeks

1. Present ATAM
2. Present business drivers
3. Present the architecture
4. Identify architectural approaches
5. Generate quality attribute utility tree
6. Analyse architectural approaches

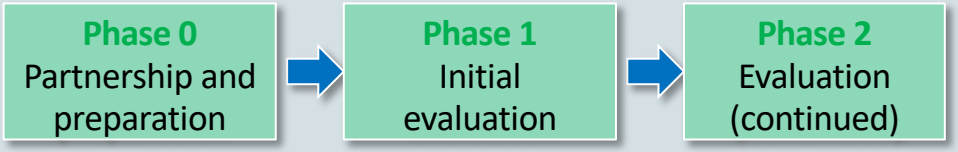
ATAM Phases

Evaluation client

Evaluation team

Project decision makers

Stakeholders



Proceeds informally as required, perhaps over a few weeks

1-2 days followed by hiatus of 1-3 weeks

2 days

1. Brainstorm and prioritize scenarios
2. Analyse architectural approaches
3. Present results: provide all documentation to the stakeholders

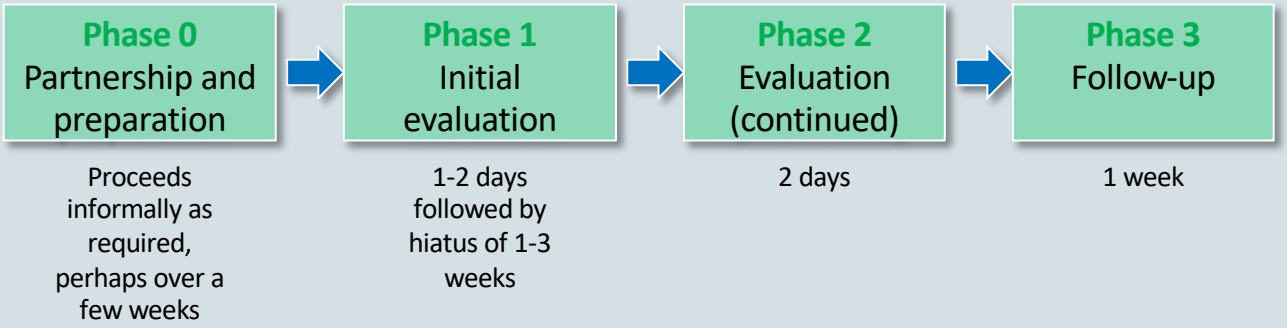
ATAM Phases

Evaluation client

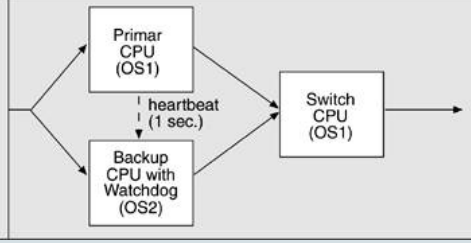
Evaluation team

Project decision makers

Stakeholders



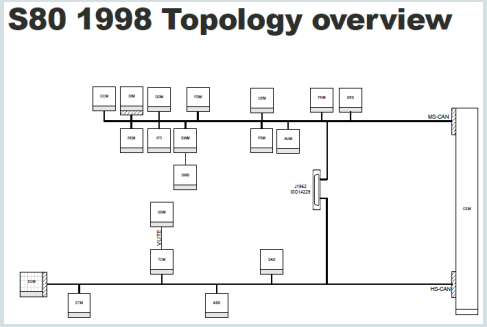
ATAM Example Analysis

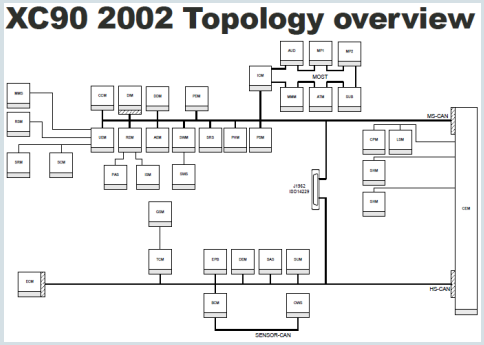
Scenario #: A12	Scenario: Detect and recover from HW failure of main switch.			
Attribute(s)	Availability			
Environment	Normal operations			
Stimulus	One of the CPUs fails			
Response	0.999999 availability of switch			
Architectural decisions	Sensitivity	Tradeoff	Risk	Nonrisk
Backup CPU(s)	S2		R8	
No backup data channel	S3	T3	R9	
Watchdog	S4			N12
Heartbeat	S5			N13
Failover routing	S6			N14
Reasoning	Ensures no common mode failure by using different hardware and operating system (see Risk 8) Worst-case rollover is accomplished in 4 seconds as computing state takes that long at worst Guaranteed to detect failure within 2 seconds based on rates of heartbeat and watchdog Watchdog is simple and has proved reliable Availability requirement might be at risk due to lack of backup data channel ... (see Risk 9)			
Architecture diagram				

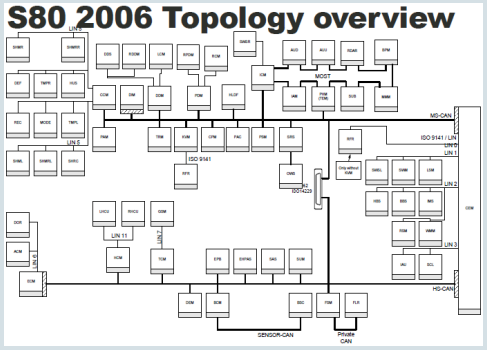
Example

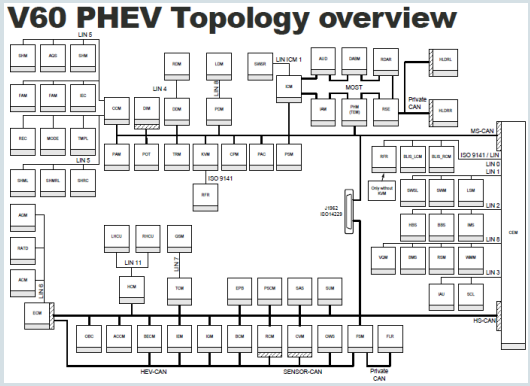
Automotive Software Architecture

Increasing amount of software in systems

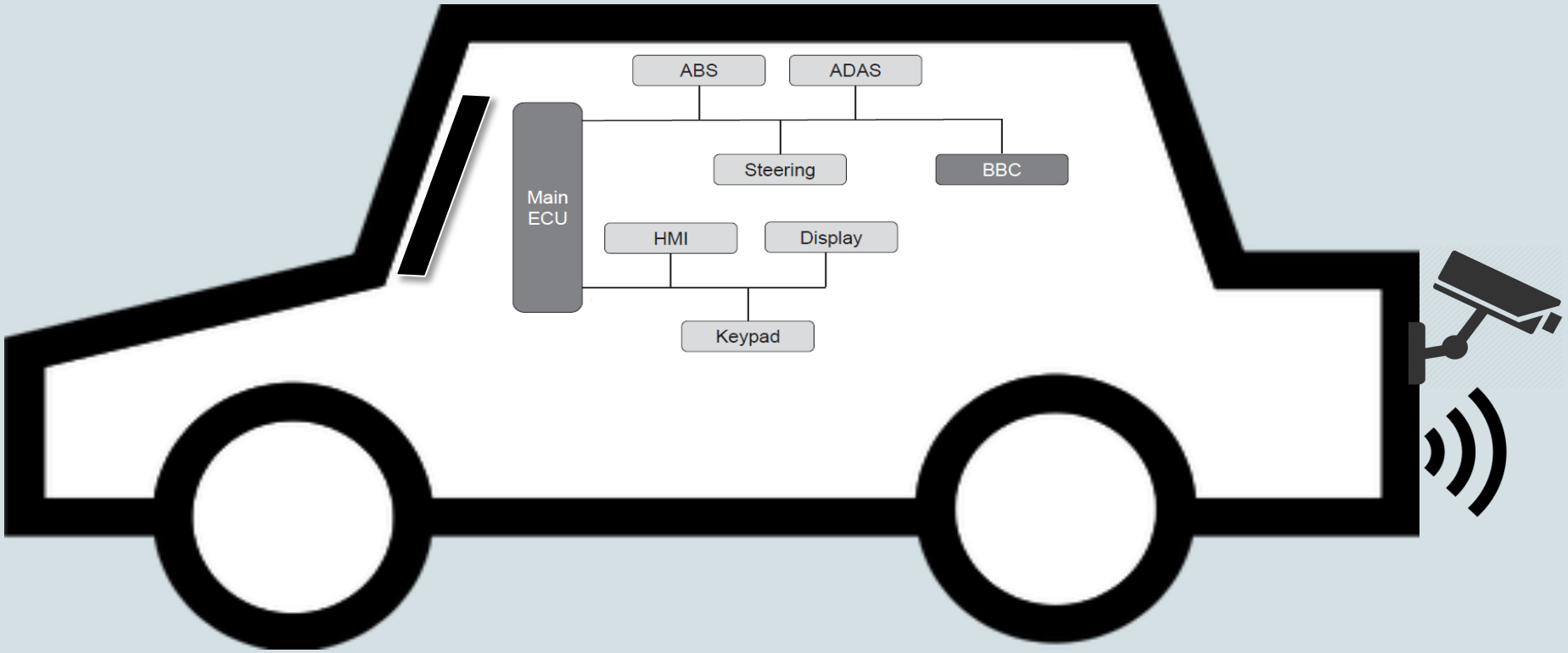




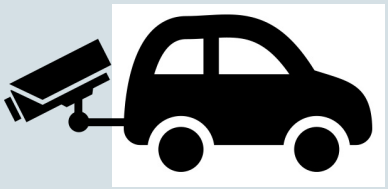


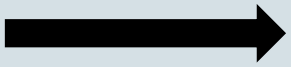


Example Quality Scenario for Safety

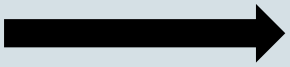


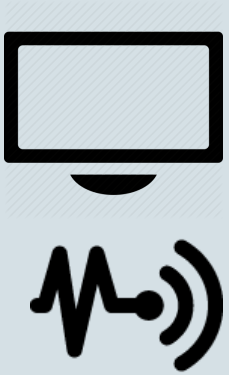
Example Quality Scenario for Safety





Artifact
Main ECU, BBC ECU, Flexray bus





Source
Rear-camera

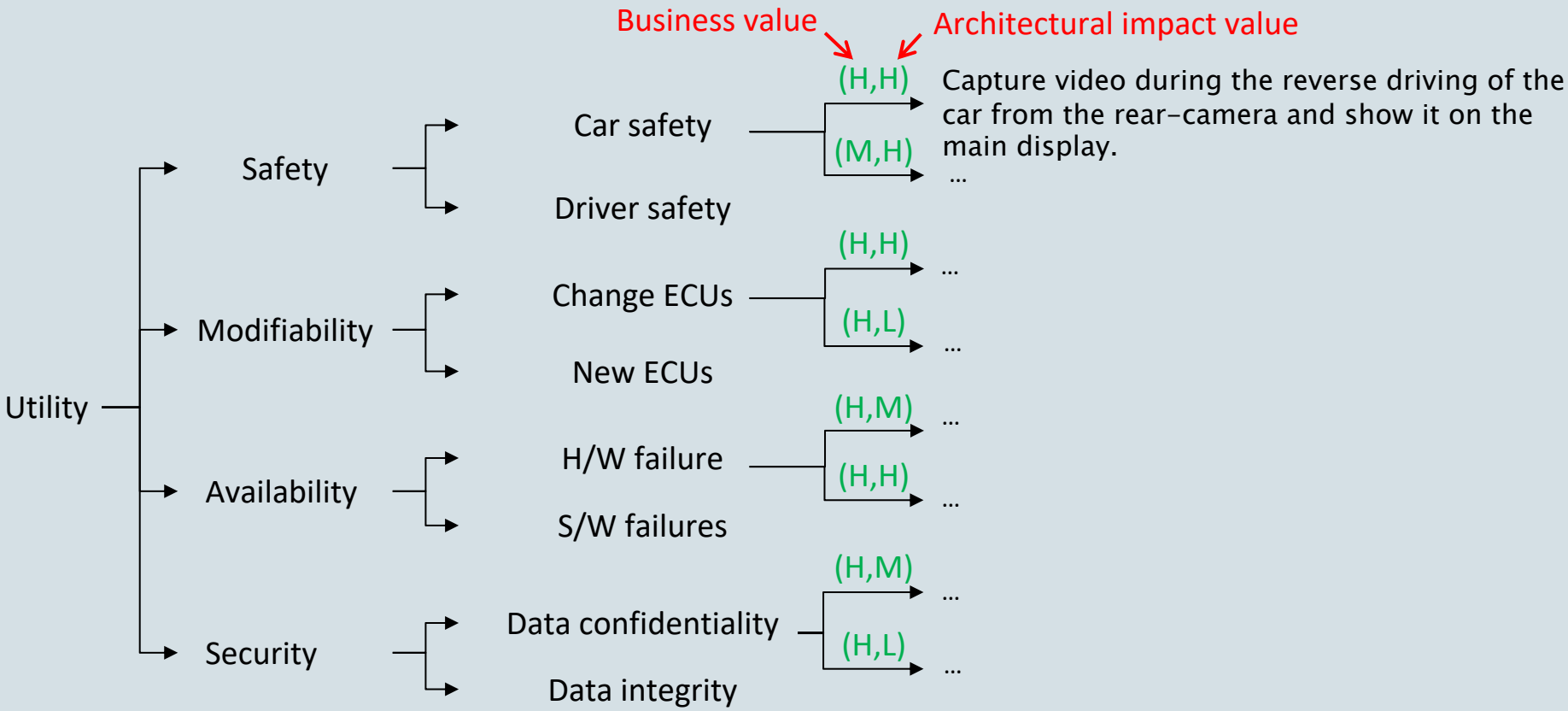
Stimulus
Camera feed

Environment
Car in reverse driving

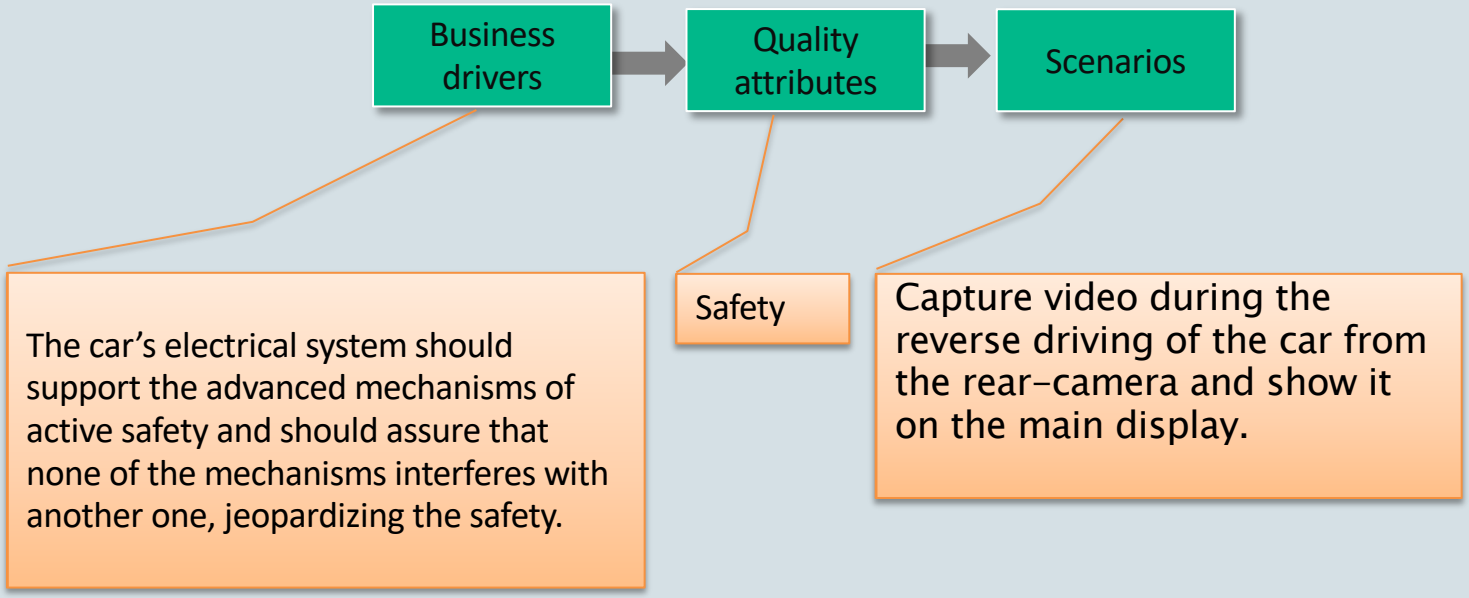
Response
Process video data and show it on the display

Measure
Video displayed in real-time and no loss of safety signals from parking sensors

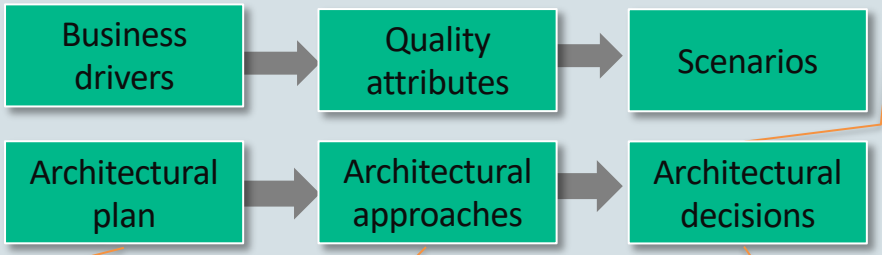
Utility Tree



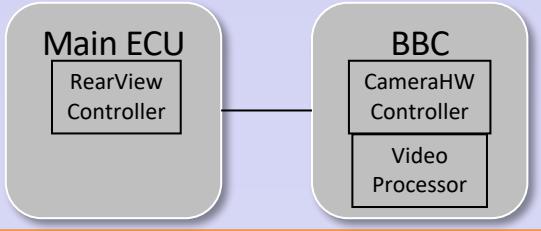
ATAM Conceptual Flow

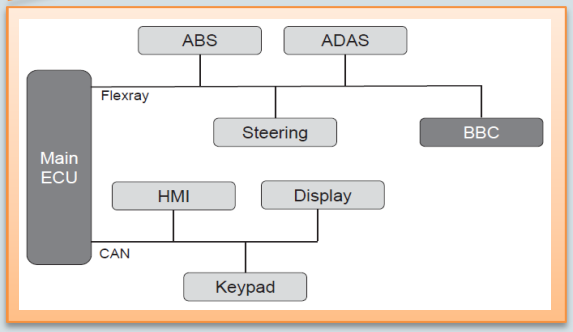


ATAM Conceptual Flow



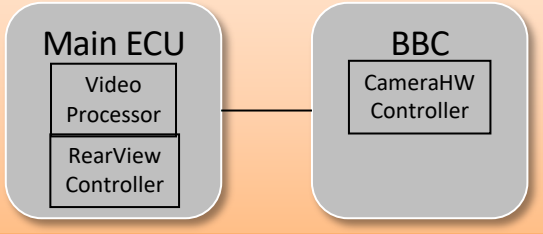
Placing the processing of the video feed on BBC



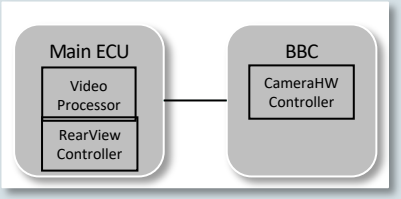


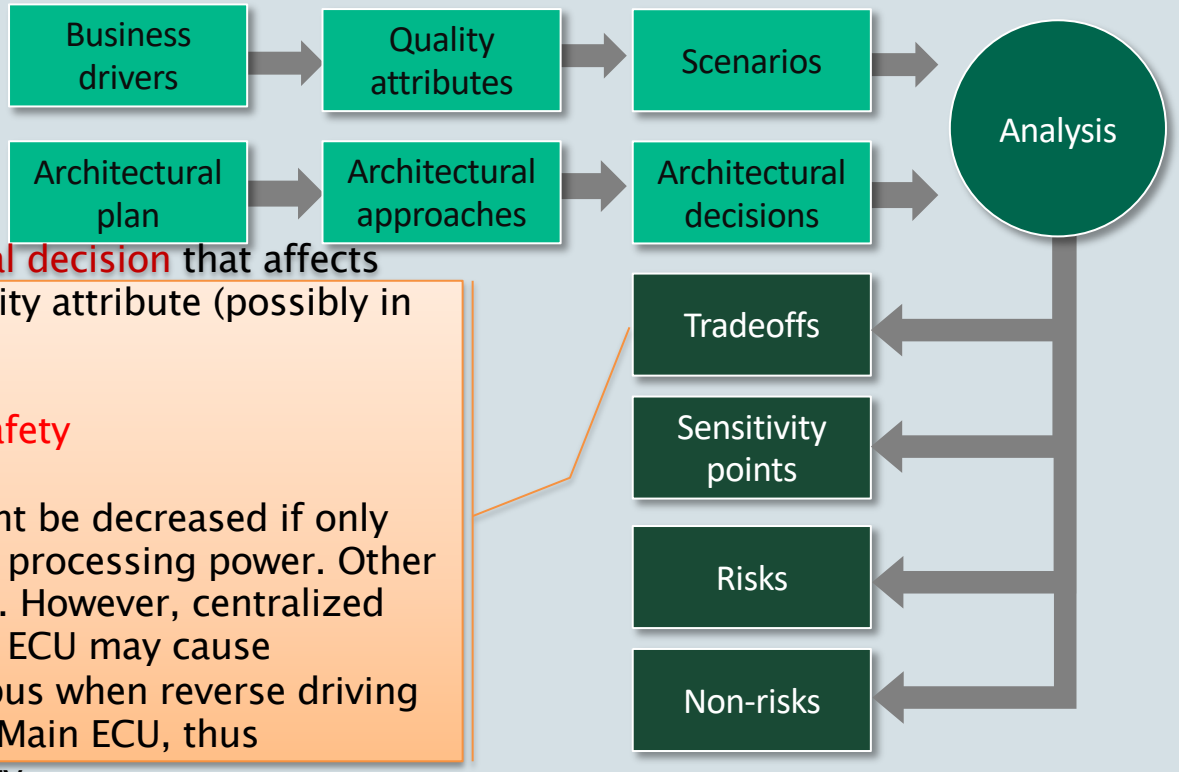
Multi-tiered deployment of processing components over multiple ECUs.

Placing the processing of the video feed on the Main ECU



ATAM Conceptual Flow



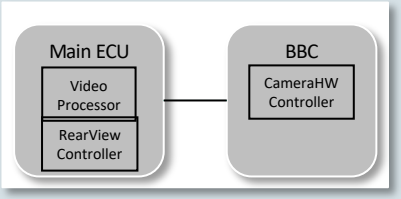


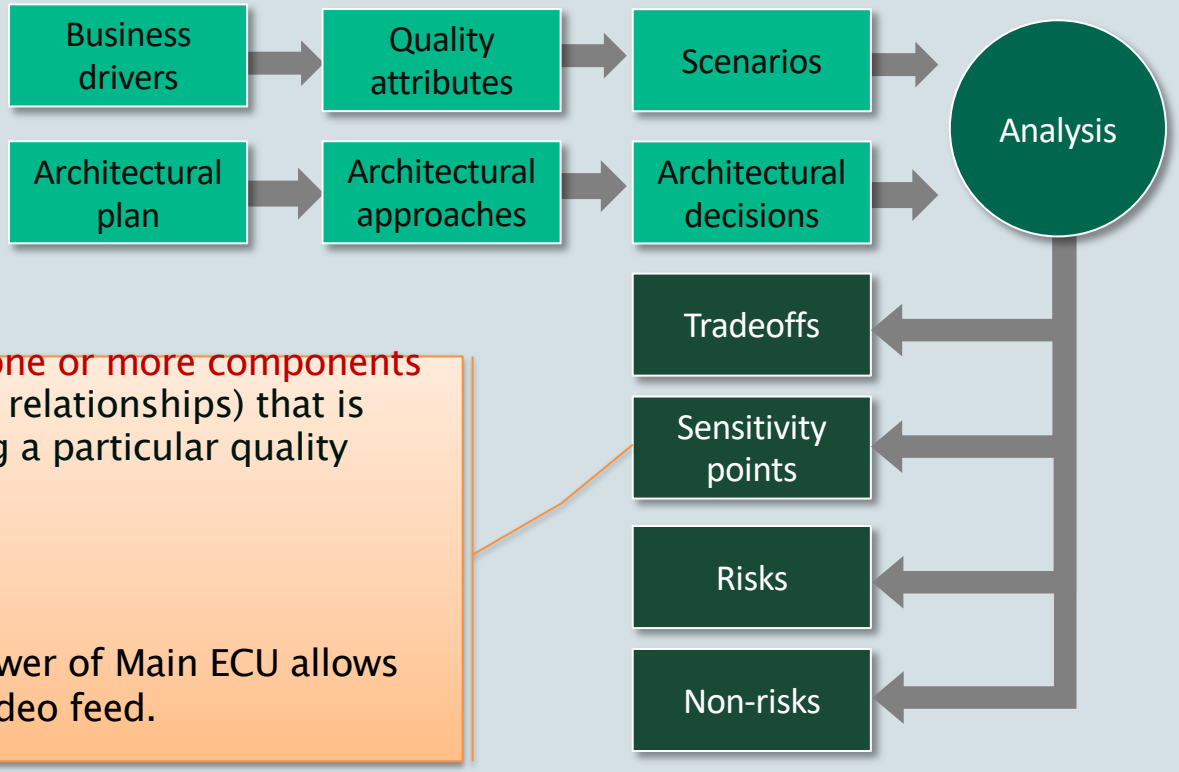
Def: An architectural decision that affects more than one quality attribute (possibly in opposite ways).

Example: Cost vs Safety

Cost of the car might be decreased if only Main ECU is of high processing power. Other ECUs don't need be. However, centralized processing on Main ECU may cause congestion on the bus when reverse driving and overloading of Main ECU, thus compromising safety.

ATAM Conceptual Flow



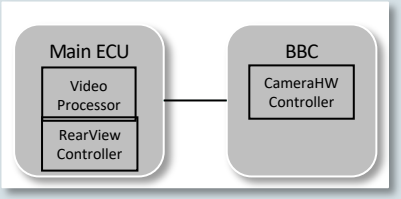


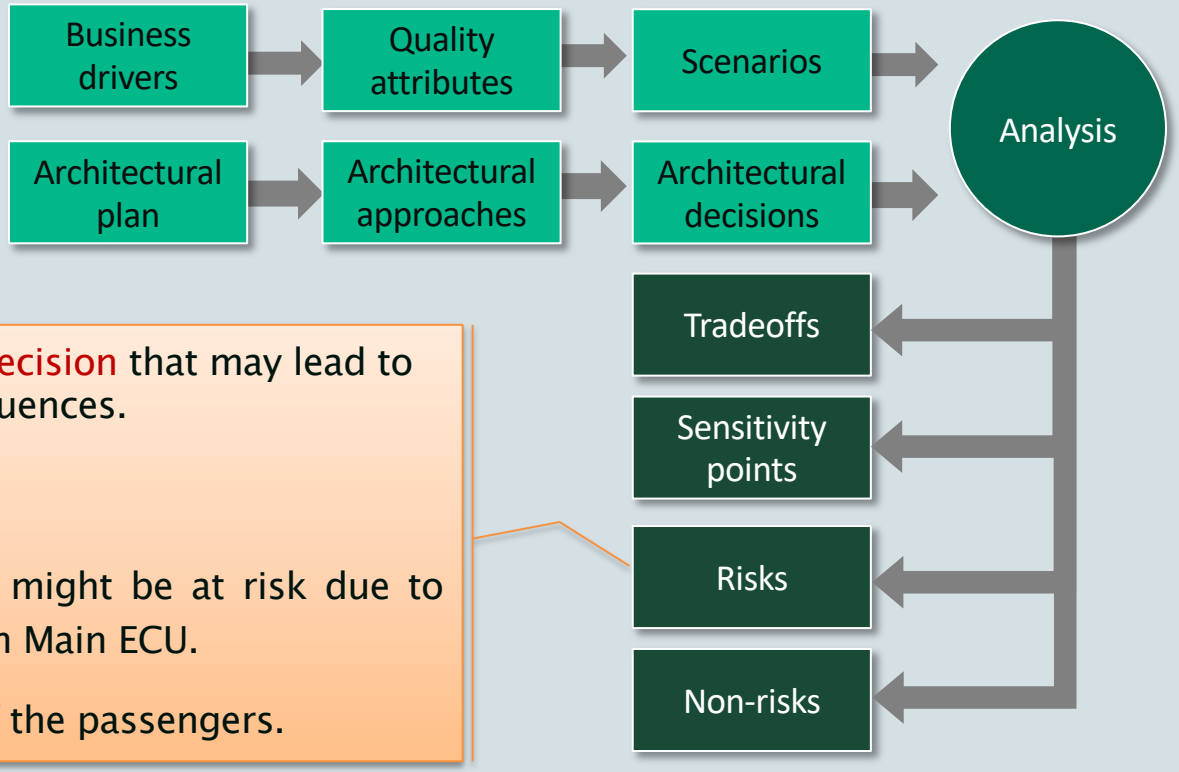
Def: A property of one or more components (and/or component relationships) that is critical for achieving a particular quality attribute response.

Example:

High processing power of Main ECU allows for processing of video feed.

ATAM Conceptual Flow





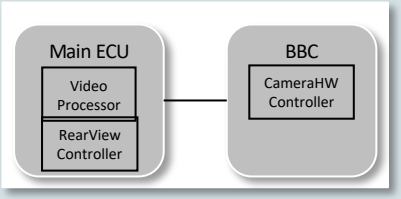
Def: Architectural decision that may lead to undesirable consequences.

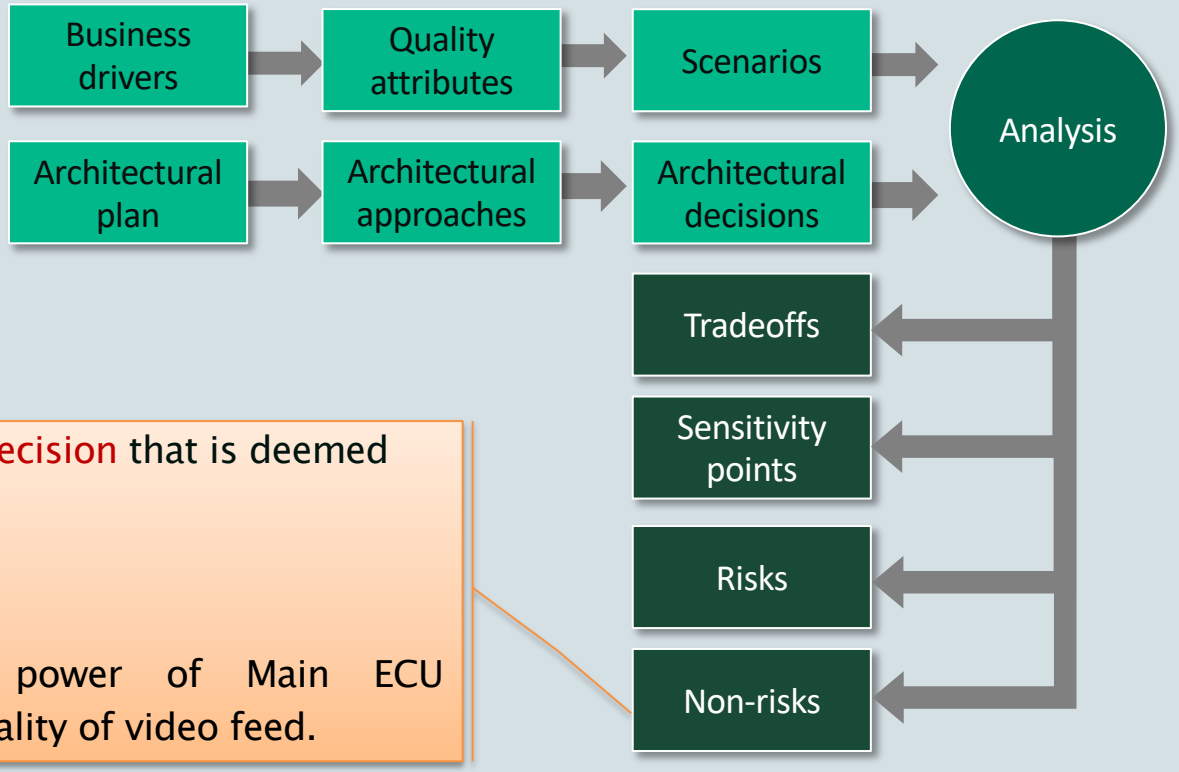
Example:

Safety requirement might be at risk due to heavy processing on Main ECU.

→ Impact: health of the passengers.

ATAM Conceptual Flow



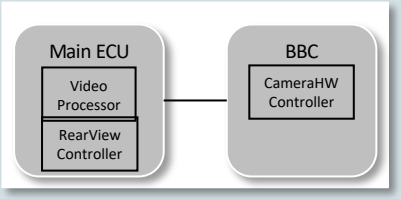


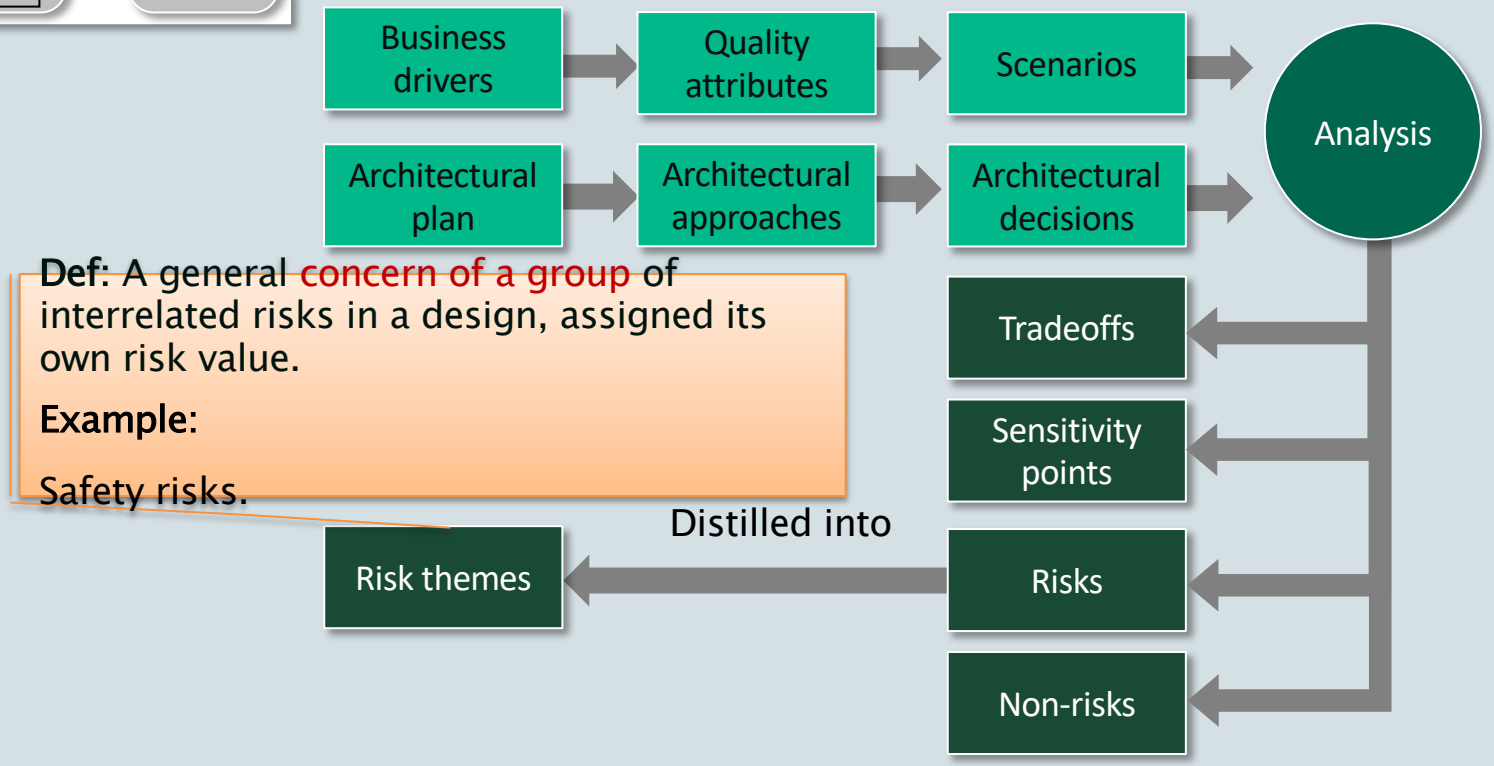
Def: Architectural decision that is deemed safe.

Example:

High processing power of Main ECU guarantees high quality of video feed.

ATAM Conceptual Flow



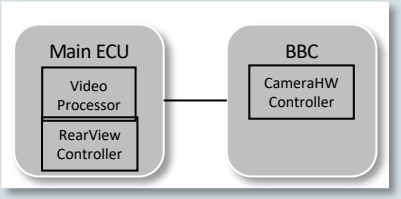


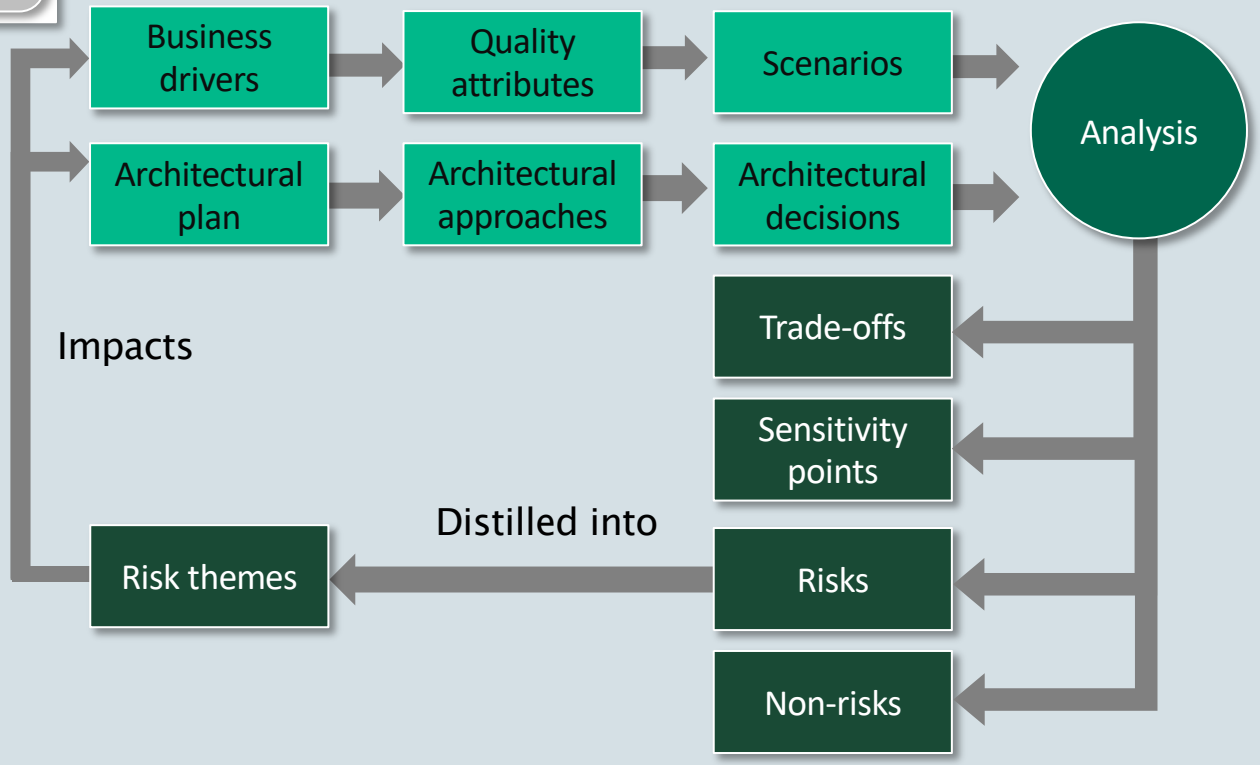
Def: A general concern of a group of interrelated risks in a design, assigned its own risk value.

Example:

Safety risks.

ATAM Conceptual Flow





Summary

- We have learned:
  - **What** is software architecture evaluation!
  - **How to plan** software architecture assessment!
  - What are the **results** and **benefits** of architecture evaluation
  - **ATAM** – Architecture Tradeoff Analysis Method

Summary

- ATAM:
  - is a **scenario-based** scenario-based architecture evaluation method that focuses on a system's **quality goals**
  - is a **qualitative** evaluation approach
  - is **not** an evaluation of requirements
  - is **not** a code evaluation
  - does **not** include actual system testing
  - is works with possible **areas of risks**