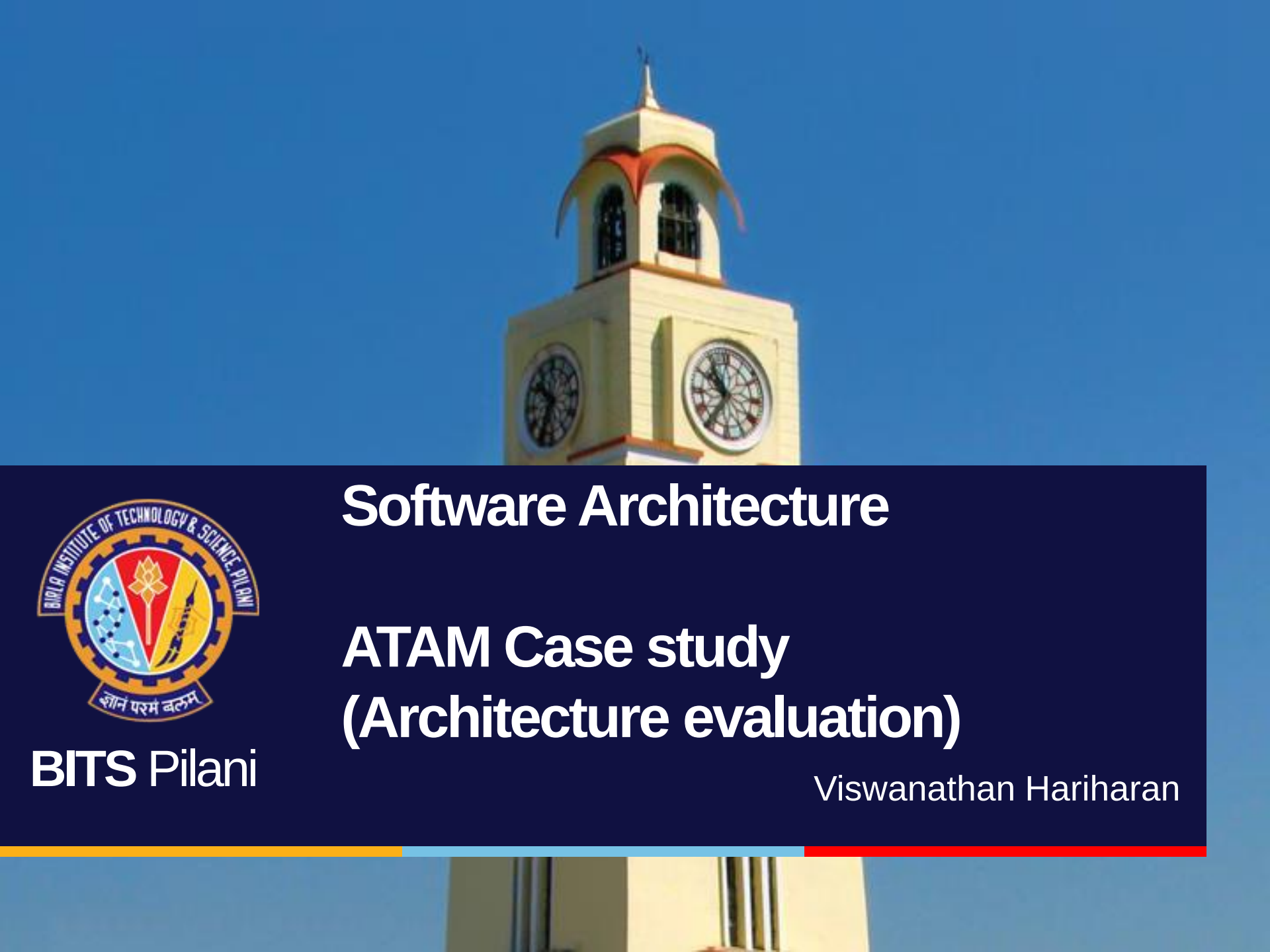


Software Architecture

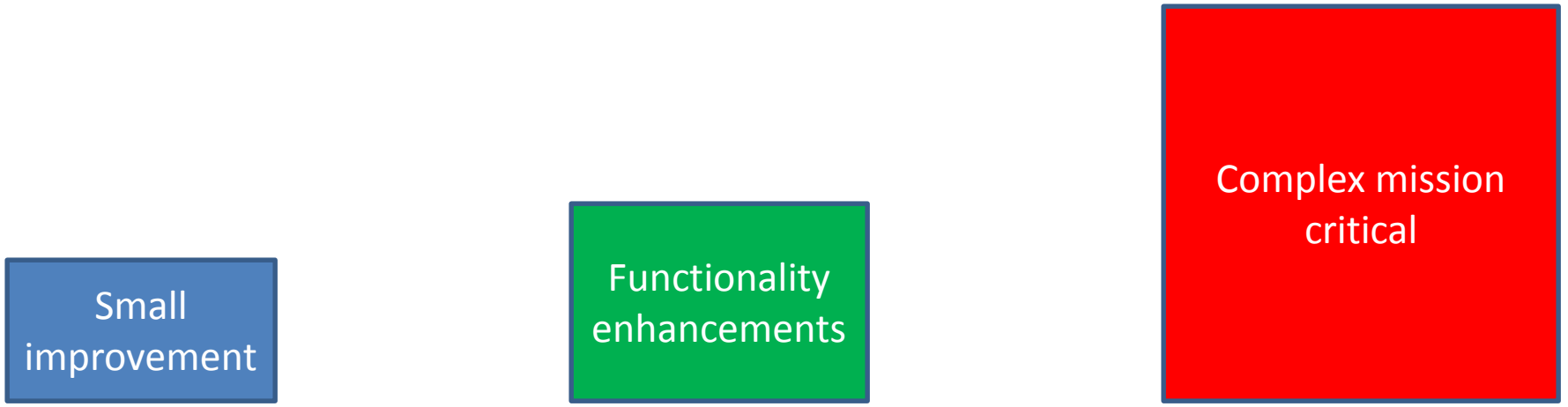
ATAM Case study (Architecture evaluation)

Viswanathan Hariharan

BITS Pilani

Introduction

Software projects come in different colours and shapes



Improve response time

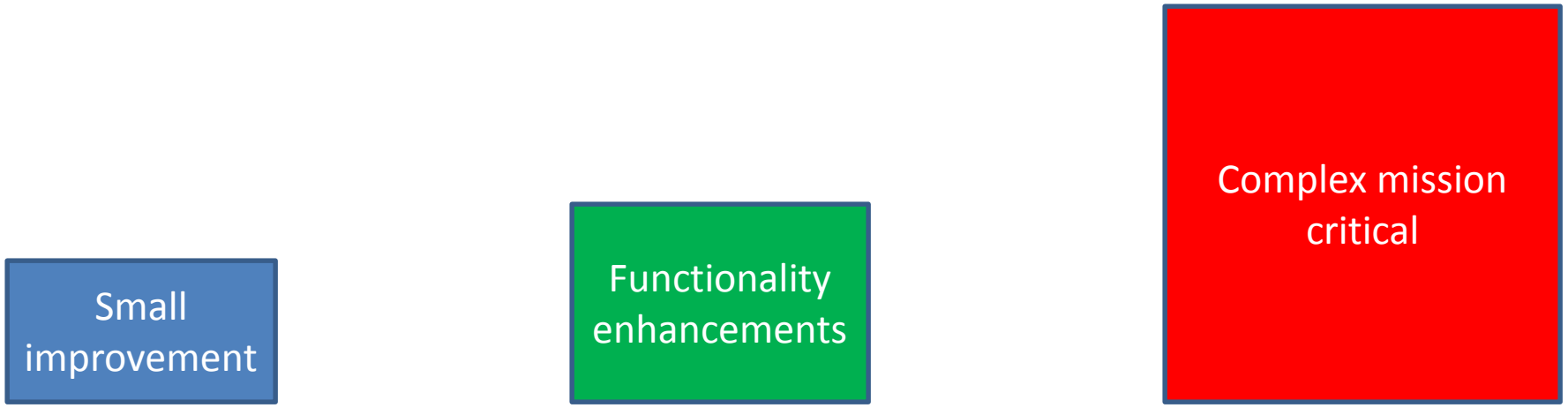
Add Loyalty module

Build a satellite system

Introduction

innovate achieve lead

Review techniques differ



Today…

ATAM

Architecture Trade-off Analysis Method

Method + Case study

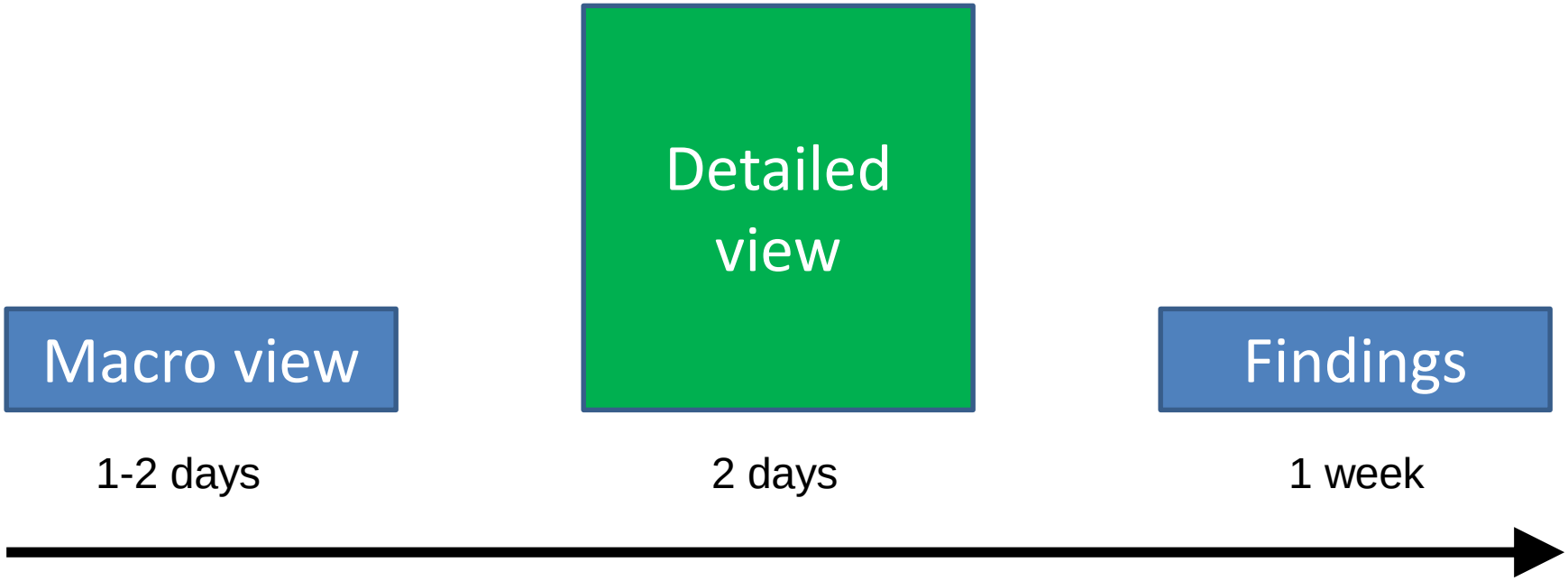
Method

Reviewers: External, from different organization

Unbiased opinion

Independent perspective

Process steps…



Process steps…

Macro view

- Goals & Business functions
- Architectural approach
- Utility tree

Detailed view

- Detailed scenarios
- Prioritization
- Tactics used

Findings

- Risks
- Sensitivity
- Trade-offs

Process steps

Macro view | Detailed view | Findings

innovate achieve lead

Judiciary system

- Goal: Improve efficiency of court operations
- Functions: Filing case, Proceedings, Judgement
- Quality attributes: Security of information, Usability

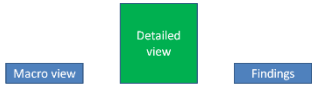
Macro view

- Goals & Business functions
- Architectural approach
- Utility tree

Utility tree

Quality attribute	Attrubute refinement	Scenario	Business value	Architecture impact
Security	Integrity	The proceedings should not be lost or tampered with by unauthorized people, including those working in the court (1)	High	High
Usability	Workflow	Once the staff enter the proceedings, it should come to the concerned judge for review, changes and approval (3)	High	High
Modifiability	Criteria specification	The cases should be intelligently scheduled considering the age of the case, criticality, etc. (4)	Medium	Medium
Usability	Status notification	The system should send notification to the concerned lawyer and parties when hearings are scheduled (5)	High	High
Performance	Response time	Proceedings will consist of text, photos, images & videos (b)	Medium	Medium
Usability	Understanding user model	Information can be categorized as evidence, arguments, facts, etc. System should aid in the entry of such information and make the data entry efficient (2)	High	Low
Usability	Intuitiveness	Registering a case should be very easy (2)	High	Medium
Usability	Status notification	I should be notified about the hearing date (5)	High	High
Usability	Understanding user model	I should be able to upload documents (2)	High	Low
Usability	Status notification	I should be notified when the proceedings are posted and I should be able to view them in chronological order (2)	High	High
Usability	Intuitiveness	I should be able to file affidavits and petitions online (2)	High	Low

Process steps

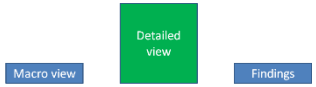


Detailed view

- Detailed scenarios
- Prioritization
- Tactics used

Voting

Process steps



- Risks: Ex. Certain data access services are not secure enough. Hackers can get access to private data (such as date of birth) using these services.
- Sensitivity: Ex. An eComm system's interface to telecom gateway is sensitive to changes in gateway interface
- Trade-offs: Ex. Multiple levels of security (user pwd, txn pwd, OTP) may impact usability.

Findings

- Risks
- Sensitivity
- Trade-offs

Case study: CAAS

Common Avionics Architecture System

Rockwell Collins

Building trust every day

Case study: CAAS
Common Avionics Architecture System

https://youtu.be/da9MHLeTwvY

Case study: CAAS
Common Avionics Architecture System

CAAS Cockpit integrates multiple sub-systems - communications, navigation, weapons, mission sensor

Ref:
https://www.rockwellcollins.com/Products_and_Services/Defense/Avionics/Integrated_Cockpit_Solutions/Common_Avionics_Architecture_System.aspx

Rockwell Collins avionics management system caters to different types of helicopters





Background

Two different proprietary avionics systems were in use

This resulted in greater effort to enhance and maintain

Case study: CAAS
Common Avionics Architecture System

innovate achieve lead

Goal

Create a scalable system that meets the needs of multiple helicopter cockpits to address modernization issues

1. Easier to maintain
2. Allow third-party upgrades
3. Provide a common ‘look and feel’

Case study: CAAS
Common Avionics Architecture System

innovate achieve lead

Approach

Use a single, open, common avionics architecture system for all platforms to reduce the cost of ownership

CAAS: Quality attributes

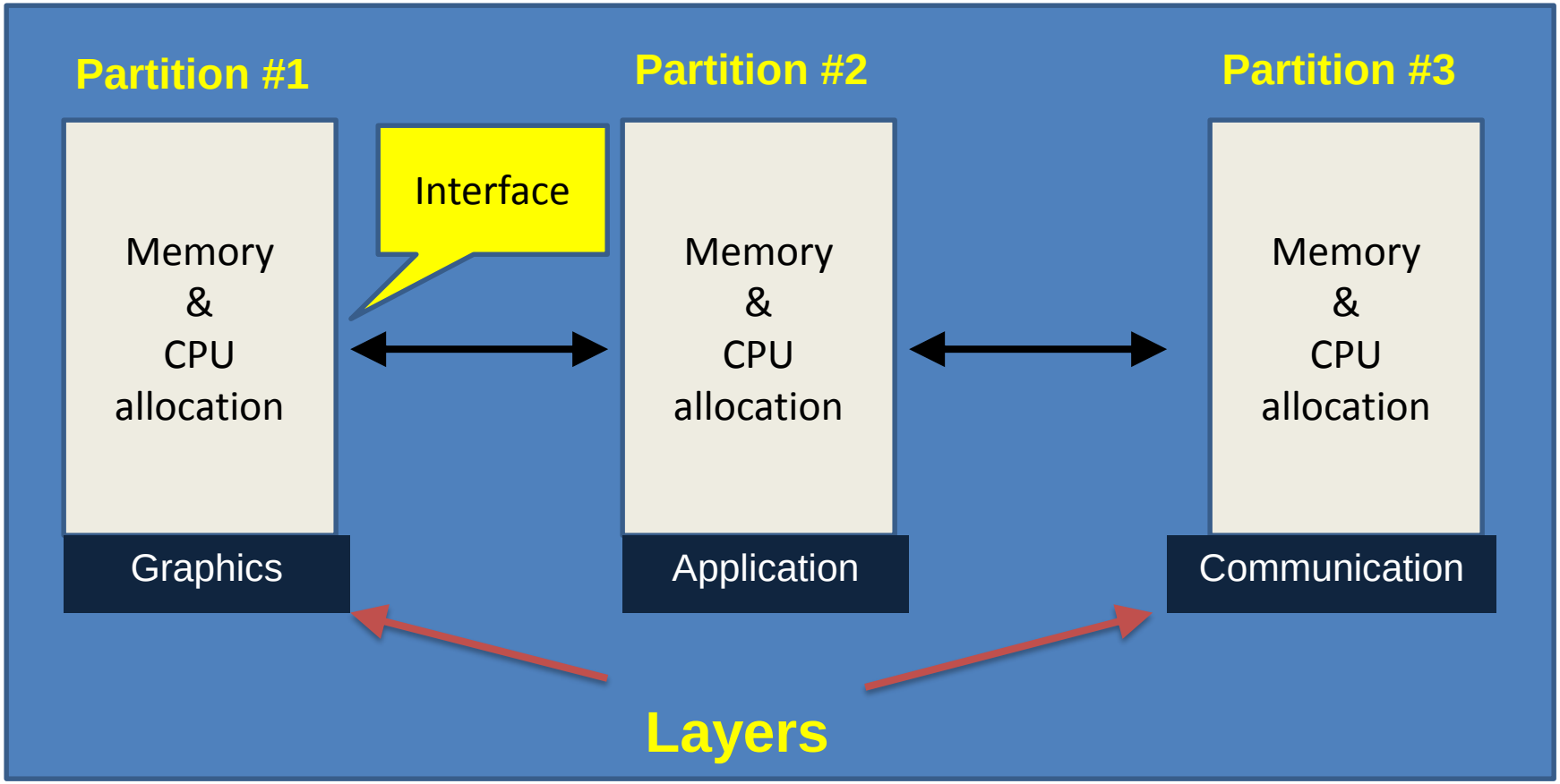
Availability

Performance

Maintainability

CAAS: Architectural approach

Sample structure within a system

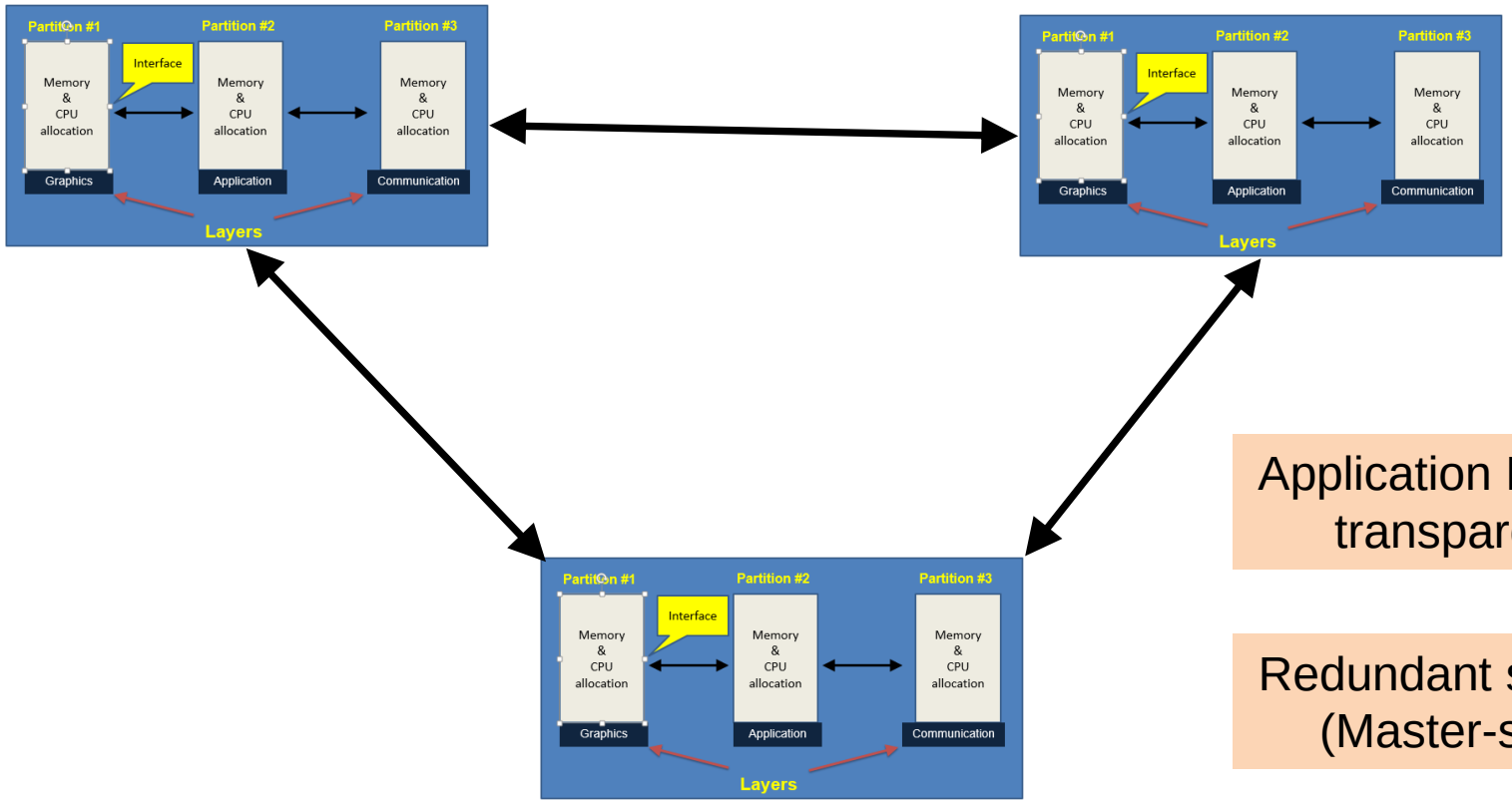


POSIX based system

CAAS: Architectural approach

innovate achieve lead

Distributed system



Application Location transparency

Redundant software (Master-slave)

CAAS Utility tree

innovate achieve lead

Table 1: Utility Tree for the Availability Quality Attribute

Phase 1: Quality Attribute Utility Tree	
Quality Attribute	availability
Attribute Concerns	The OFP doesn't crash.
Scenarios	1. Invalid data is entered by the pilot, and the system does not crash.
	2. Invalid data comes from an actor on any bus, and the system does not crash.
	3. When a 1.9-second power interruption occurs, the system will execute a warm boot and be fully operational in 2 seconds.
Attribute Concerns	graceful degradation in the presence of failures
Scenarios	1. A loss of Doppler occurs, the pilot is notified, and the Doppler timer begins a countdown (for multi-mode radar [MMR] validity).
	2. A partition fails, the rest of the processor continues working, and the system continues to function.
Attribute Concerns	no degradation in the presence of failures for which there are redundant components/paths
Scenarios	1. The data concentrator suffers battle damage, and all flight-critical information is still available.
	2. The mission processor in the outboard MFD fails, and that display and the rest of the system continue to operate normally.

CAAS: Scenario generation & prioritization

innovate achieve lead

Table 2: Brainstormed Scenarios from Step 7

Phase 2: Brainstormed Scenarios		
Scenario Number	**Scenario Text**	**Number of Votes**
2	Changes to the CAAS are reflected in the simulation and training system concurrently with the airframe changes, without coding it twice (simulation and training stakeholder).	5
3	No single point of failure in the system will affect the system's safety or performance (system architect stakeholder).	10
5	Multiple versions of the system must be fielded at the same time. Those versions should be distinguishable and should not have a negative impact on the rest of the system (system implementer stakeholder)	1
9	75% of the CAAS is built from reused components increasing new business opportunities (from Phase 1, program manager stakeholder).	9
13	Given maximum "knob twiddling" to the level that the system's performance is degraded, the system can prioritize its flight-critical functions, so they are NOT degraded (safety stakeholder).	6
15	Given the need for a second ARC231, the radio can be incorporated into the existing system by reusing existing software at minimal or no cost (requirements stakeholder).	2
20	An application doesn't crash, but starts producing bad data. The system can detect the errant data and when applications crash (reliability stakeholder).	3

CAAS: Sample observations from analysis

Risk	There are no built-in hooks to connect to simulators. So the software can not drive both the simulators and the actual helicopters
Sensitivity	Isolating operating system dependencies will enhance portability.
Trade-off	Letting pilots set parameters such as turbine gas temperature limits, increases flexibility but decreases safety

CAAS: Sample observations from analysis

innovate achieve lead

Themes

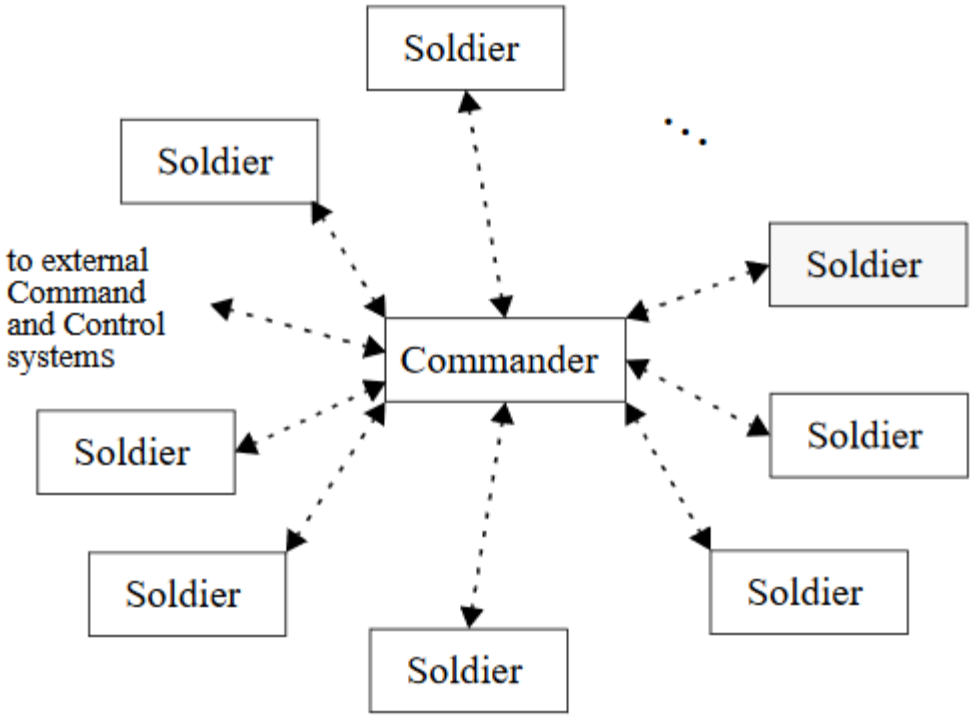
Several scenarios dealt with performance. However performance requirements were not spelled out clearly

Additional risks

External evaluation can also reveal additional risks not previously imagined

Case study: Battlefield Control system

This system is used by army battalions to control the movement and operations of troops in real time in the battle-field.



New risk discovered during evaluation…

The pattern of communication between Control and backup is distinct from communication with other nodes. They exchange far more data than other nodes

Risk

Enemy may detect this pattern and attack the Control node and Backup node

Conclusion

- In large and complex mission critical systems, external reviews add a lot of value
- Such reviews brings together all stakeholders
- Apart from risk identification, the exercise generates very useful artifacts about the system such as Utility tree, Scenarios, Architecture diagrams, etc.

Appendix

Reference:

Rockwell Collins case study:

https://resources.sei.cmu.edu/asset_files/TechnicalNote/2003_004_001_14150.pdf

Examples of self evaluation checklists

Availability:

- Do we have a mechanism to detect failure?
- Do we have a mechanism to switch to a backup component?
- Do we have a mechanism to inform the client about the failure?
- Do we have a mechanism to save state periodically?

Performance:

- What is the mechanism to add & remove more resources dynamically?
- If there is a common resource that is needed by multiple clients, what is the mechanism to reduce the bottleneck?

Example of Self evaluation models

How to evaluate the availability of a system that has 2 servers – one primary & one hot standby?

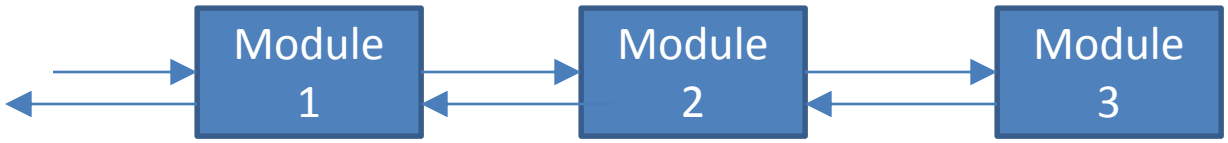
- If probability of server failure is 1% (1/100 = 0.01), what is the probability of 2 servers failing at the same time?
- Compare the availability of the system with one server and 2 servers

- Given that a server has failed, what is the probability that the second server also fails
- $P * P = P^2 = 0.01 * 0.01 = 0.0001$
- Availability = 1 - Probability of failure = 1 - .0001 = .9999 = 99.99% (Availability with one server = 1 – 0.01 = .99 = 99%)

Example of Self evaluation models

How to calculate the latency (performance)?

- Let us say, in order to satisfy a client request, the request needs to pass via 3 modules one after another.
- The latency of 1st module is 0.1 milli sec, latency of 2nd module is 0.2 milli second, latency of 3rd module is 0.3 milli second
- What is the latency experienced by the client?
- Sum of Latency of each module = 0.1 + 0.2 + 0.3 = 0.6 milli second



1 sec 2 sec 3 sec Total 6 seconds

Architecture Trade-off Analysis Method (ATAM)

- ATAM is a method used to evaluate architecture of large systems
- It assumes that reviewers are not familiar with the business goals and the architecture of the system
- It is suitable for many domains such as
  - Finance
  - Defence
  - Automotive
  - Etc.

ATAM

Participants

- Evaluation team
- Project decision makers – Business stakeholder, Project manager
- Arch stakeholders – Users, developers, testers, maintenance staff
- Scenario scribe – Writes down scenarios discussed in the workshop
- Proceedings scribe – Captures the entire proceedings including goals, architecture approach, evaluation observations

ATAM

Output of ATAM

- Concise presentation of architecture
- Business goals
- Prioritized quality attribute scenarios
- Set of Risks and Non-risks
- Set of risk themes
- Mapping of architecture decisions to quality requirements (scenarios)
- Sensitivity & Trade-off points

ATAM

innovate achieve lead

Phases

Phase	Activity	Participants	Duration
Phase 0	Partnership & Preparation	Eval team + Proj decision makers	Few weeks
Phase 1	Evaluation	Eval team + Proj decision makers + Architect	1-2 days
Phase 2	Evaluation	Eval team + Proj decision makers + Architect + Stakeholders (view & view points)	2 days
Phase 3	Follow up (Prepare report)	Eval team	1 week

ATAM - Steps

Phase 1

1. Present ATAM – Evaluation leader
2. Present business drivers – Proj decision maker (Bus goals, major functions)
3. Present architecture – Lead architect
4. Identify architectural approaches – Evaluation team
5. Generate utility tree – Eval team + Project decision makers
6. Analyse architectural approaches (sufficiency of architecture, risks, sensitivity & trade-off)

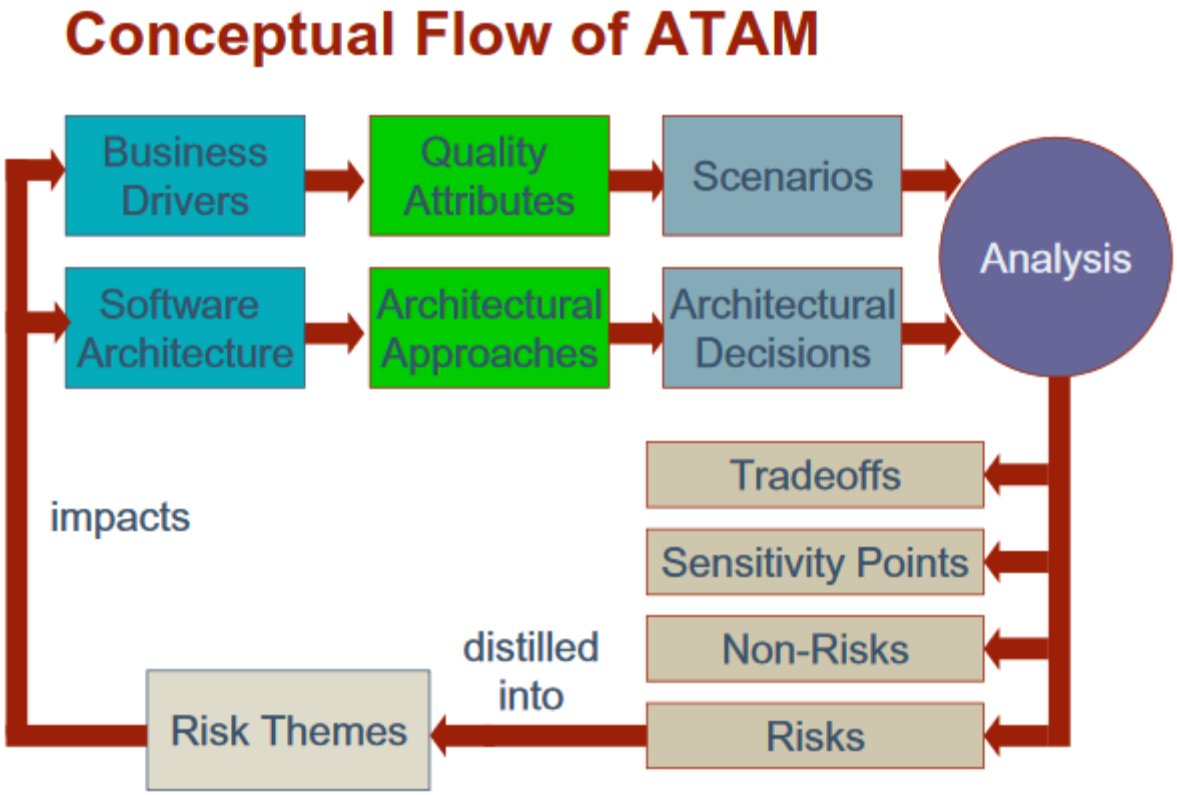
Phase 2

7. Brainstorm & prioritize business scenarios - Eval team + Project decision makers + Stakeholders
8. Analyze architectural approaches

Phase 3

9. Present results

Conceptual flow of ATAM



Ref: http://www.sei.cmu.edu/library/assets/best_practices.pdf

ATAM - Results

Results consist of

- Arch approaches (ex. Layering, distributed processing)
- Prioritized scenarios
- Risks, Non risks (Risks are arch decisions that may lead to undesirable consequences)
- Sensitivity points & Trade-offs (arch decisions that have a marked effect on one or more Quality attributes)
- Risk themes (Systemic weaknesses in architecture)

Case study: CAAS
Common Avionics Architecture System

innovate achieve lead

- Overall, the goal of the CAAS is to create a scalable system that meets the needs of multiple helicopter cockpits to address modernization issues.

- Its approach is to use a single, open, common avionics architecture system for all platforms to reduce the cost of ownership.

- This approach is based on Rockwell Collins' Cockpit Management System (CMS) in its Flight 2 family of avionics systems, augmented with IAS 2 functionality.

CAAS: Arch approaches

1. Partitioning: Partitioning of memory and utilization of CPU time (availability, safety, modifiability, testability, maintainability)
2. Encapsulation: used to isolate partitions. Between partitions, applications can share only their state via the network. The remote service interface (RSI) and remote service provider (RSP) are examples of encapsulation that isolate the network implementation details. (modifiability, availability)
3. Interface strategy: Accessing components only via their interfaces is strictly followed. Besides controlling interactions and eliminating the back-door exploitation of changeable implementation details, this strategy reduces the number of inputs and outputs per partition. (modifiability, maintainability)
4. Layers: used to partition and isolate high-level graphics services (portability, modifiability)
5. Distributed processing: Predominantly, a client-server approach is used to decouple “parts” of the system. Also, the Broadcast feature is used to broadcast information periodically. (maintainability, modifiability)