Software architectures and Framework evaluation

Rafael. A. Calvo

Edited by Ian Chunfeng Liu

Elec5619 Object Oriented Application Frameworks

THE UNIVERSITY OF SYDNEY

Agenda

- Software Architecture: Definitions, Importance
- Architecture Tradeoff Analysis Method (ATAM)
- Case study: Learning Management Systems

Software Architecture

“the fundamental organization of a system, embodied in

its components,

their relationships to each other

and the environment, and the principles governing its design and evolution”. (IEEE 1471)

Software Architecture

- Architecture is important
- Architecture can be prescribed
- Architecture is central for communicating
- Architecture is expensive to change (analyze early!)
- Architecture affects the entire project
- Requirements can be understood early

System Quality Attribute

- Performance
- Availability
- Usability
- Security

End User’s view

- Maintainability
- Portability
- Reusability
- Testability

Developer’s view

- Time to market
- ROI
- TCO
- Legacy systems
- Internal expertise

Business view

Architecture Tradeoff Analysis Method (ATAM)

- ATAM was first described in the classic book ‘Software Architecture in Practise’ by Len Bass [1].
- ATAM is a thorough and comprehensive way to evaluate a software architecture, and it is a **risk-mitigation** process.
- The Goal: to reveal how an architecture satisfies **quality goals** defined by stakeholders, and **where it stands with the common trade-offs**.
- Participants:
  - The Evaluation team (Group 1)
  - Project Managers (Group 2)
  - Architecture stakeholders. (Group 3)

Expected Outcomes

- A concise presentation of the architecture
- Articulation of the business goals.
- Quality requirements in terms of a collection of scenarios.
- Mapping of architectural decisions to quality requirements.
- A set of identified sensitivity and tradeoff points.
- A set of risks and non-risks.

Requirement 1 - Scenario

A university wants to procure a new Learning Management System. After a call for Expression of Interests (EOI) 3 are preselected:

- Sakai
- Moodle
- Blackboard

Phases

An evaluation following ATAM should follow these phases:

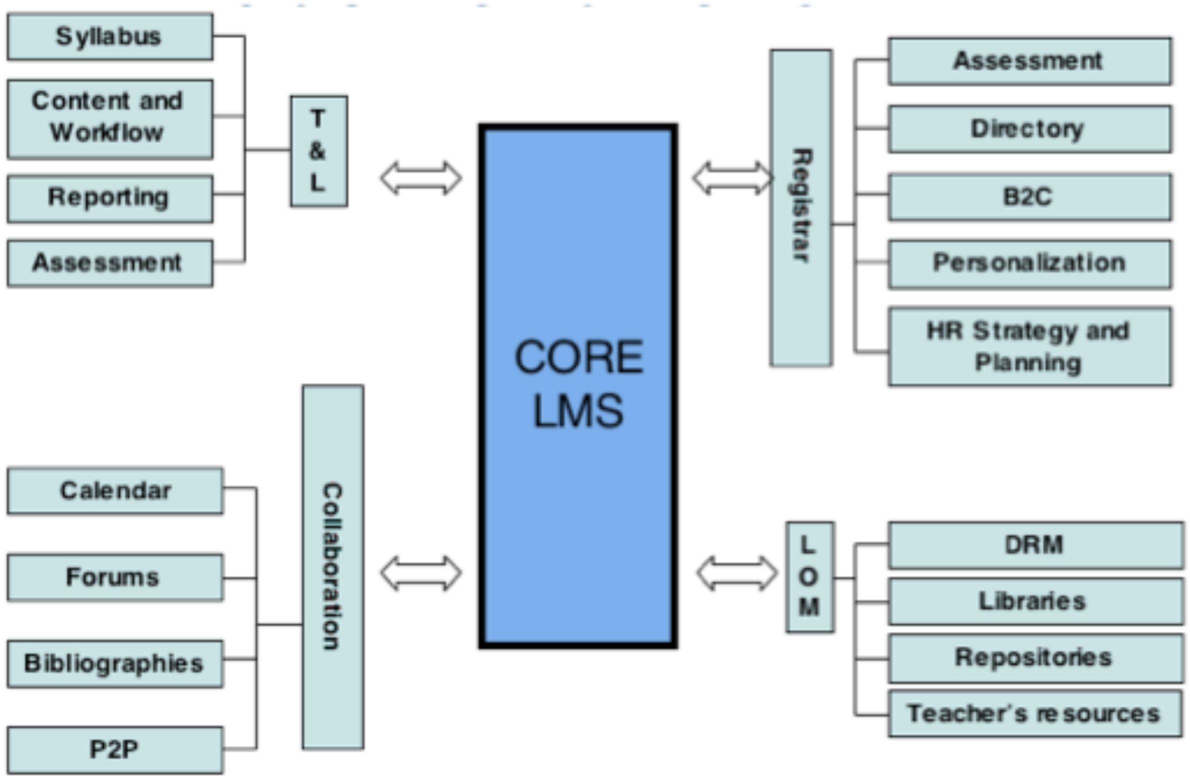
1. Present ATAM
2. Present Business Drivers
3. Present Architecture
4. Identify Architectural Approaches
5. Generate Quality Attribute Utility Tree
6. Analyze architectural approaches
7. Brainstorm and prioritize scenarios
8. Analyze architectural approaches
9. Present results

Phase 1 – Present ATAM

- Evaluation leader (Group 1)
- Describe ATAM (steps and outputs) to participants (slides above)

Phase 2 – Business drivers

- Project Manager (Group 2)
- Main business driver: Improve learning experience of students
- Many others linked to functionalities and quality attributes



Phase 3: Present architecture

- Architect (Group 3)
- Presents an overview of the architecture
  - Technical constrains such as OS, hardware or middle-ware prescribed for use
  - Other systems with which the system must interact
  - Architectural approaches (or patterns) used to meet the requirements

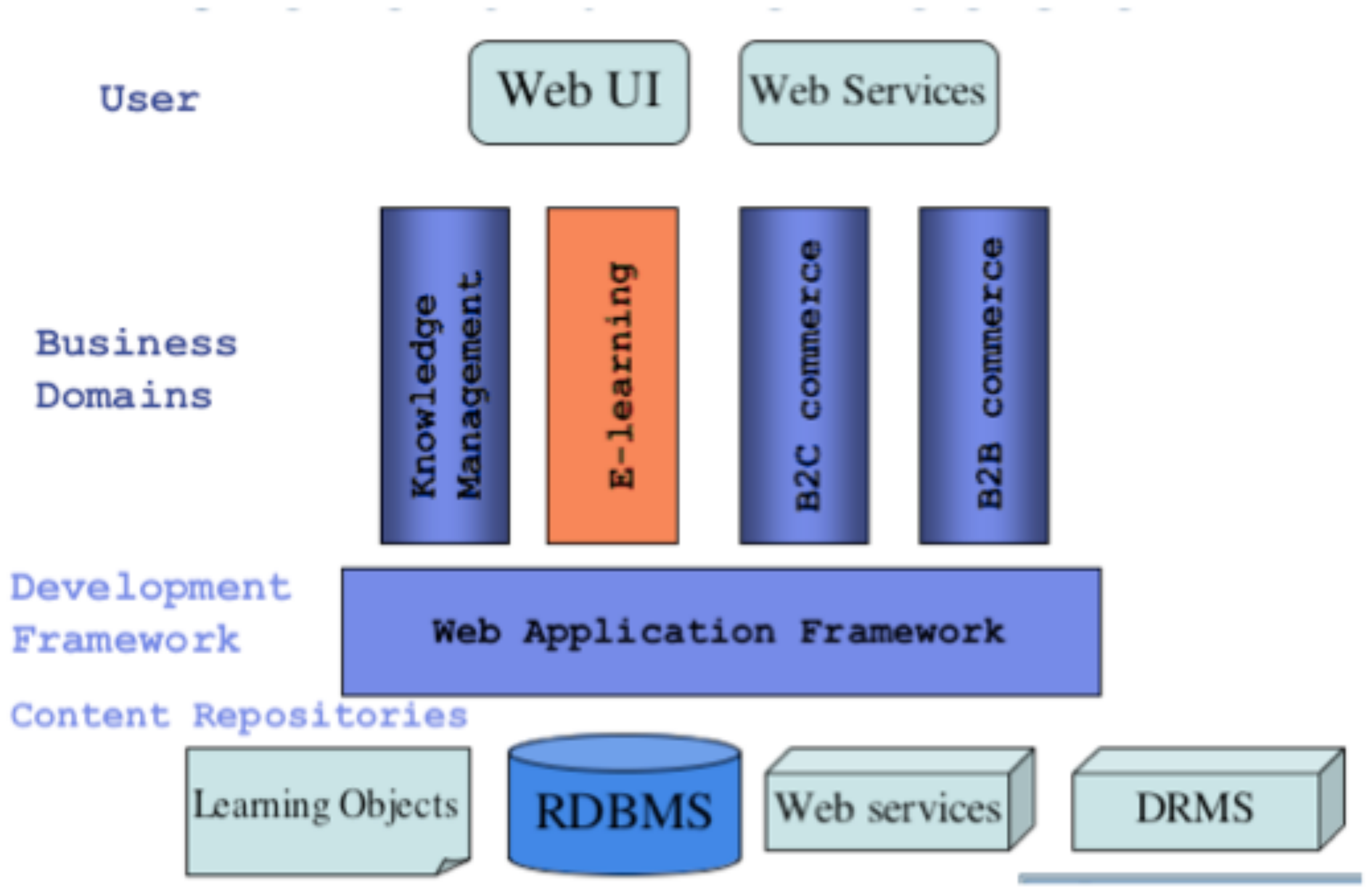
- The Evaluation team begins probing for and capturing risks.

Phase 3: Present architecture - Blackboard

- Proprietary
- Used BEA weblogic (now part of Oracle)
- Uses Apache Tomcat
- “Not much documentation available”
- “Stealth techniques indicate about 450 tables and complex proprietary DB model.”
- Some scalability problems:

Note: This was valid as of Oct 2010

Phase 3: Present architecture - Blackboard



Phase 3: Present architecture - Sakai

Some of Sakai highlight characteristics.

1. Based on existing systems from Indiana, Michigan and Stanford Universities. Released in 4Q 2003.
2. About 100 universities using it. Many part of The Sakai Partners Program
3. Core packages internationalized. Translated to over 5-6 (?) languages
4. Standards compliant
5. Uses all the new Java frameworks: Spring, Hibernate, JSF

For more information see: http://www.sakaiproject.org

Sakai Architecture (1)

› **Client**

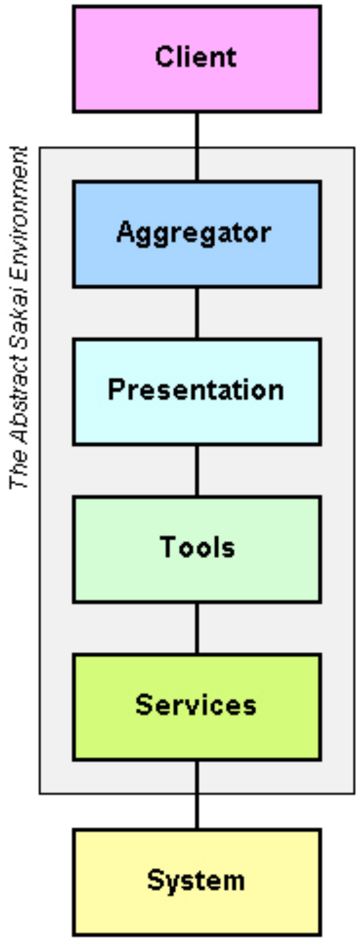
output aggregated and presented to the client using a markup language, such as HTML. Specialized clients may communicate directly with Sakai services provided they are enabled for such transactions (content authoring, for example).

› **Aggregator**

Allocates and manages screen real estate and certain user interface transactions

› **Presentation**

combines data from a Sakai tool and a user interface description to create a mark up fragment which is aggregated before delivery to the user. The user interface description is ideally contained in a resource external to the software and makes use of standard user interface elements designed to deliver a consistent Sakai user experience.



Sakai Architecture (2)

› **Tools**

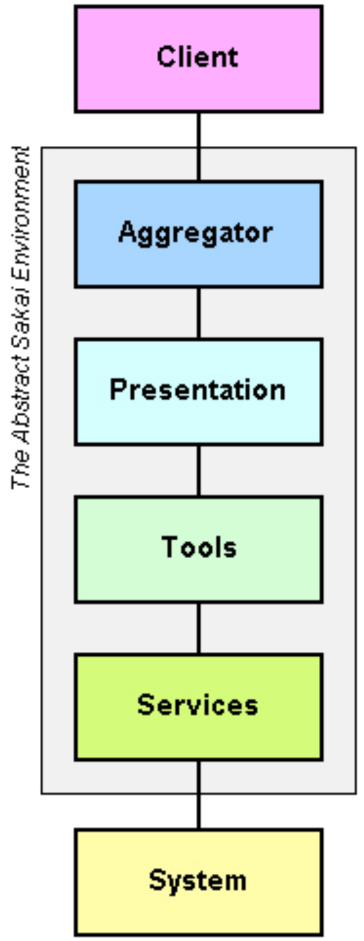
An application which marries presentation logic with application logic contained in services. Tools provide code to respond to user interface requests and events, which may (or not) modify data managed by services. The tool may draw on services to provide data to the presentation layer.

› **Services**

A collection of classes which manage data via a defined set of behaviors. This data may or may not be persistent across user sessions. Where possible data should be modeled and represented using accepted industry standards. Behavior is represented using a published Application Programming Interface (API). Services may call other services, creating dependencies on them. Services are intended to be modular, reusable and portable across Sakai environments, and potentially to non-Sakai environments also.

› **System**

› The system is the server environment which the Sakai environment resides, plus any remote capabilities available to it. This environment may include web servers,database servers, operating systems, file and resource repositories, enterprise and back office systems, etc.



Phase 3: Present architecture - Moodle

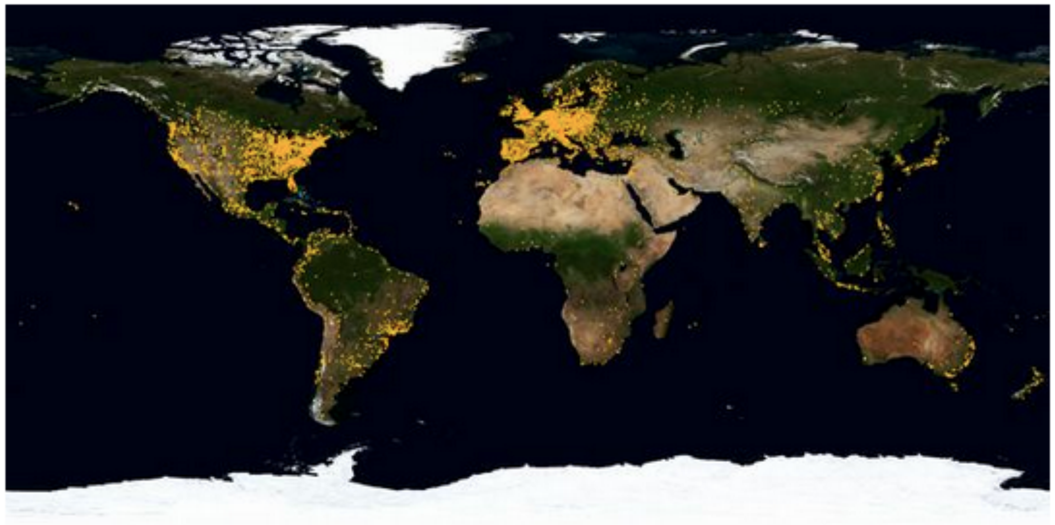
Some of Moodle highlight characteristics.

1. Created and maintained by Australian researcher Martin Dougiamas. Version 1.0 released in 2002.
2. Hundred of universities use it
3. Translated to over 10 languages.
4. Many Standards supported
5. Strong user community
6. Apache/PHP and MySQL (other DB supported by ODBC)

For more information see: http://www.moodle.org

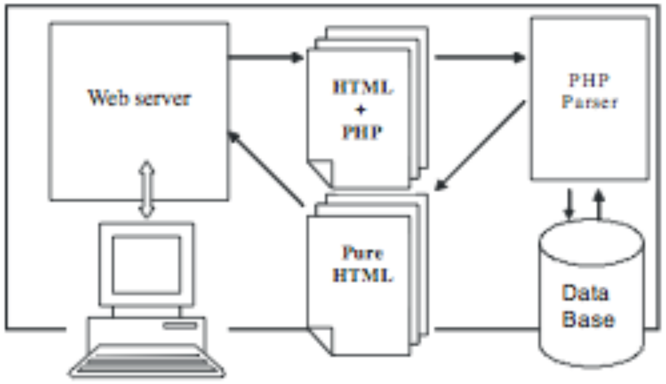
Registered moodle sites

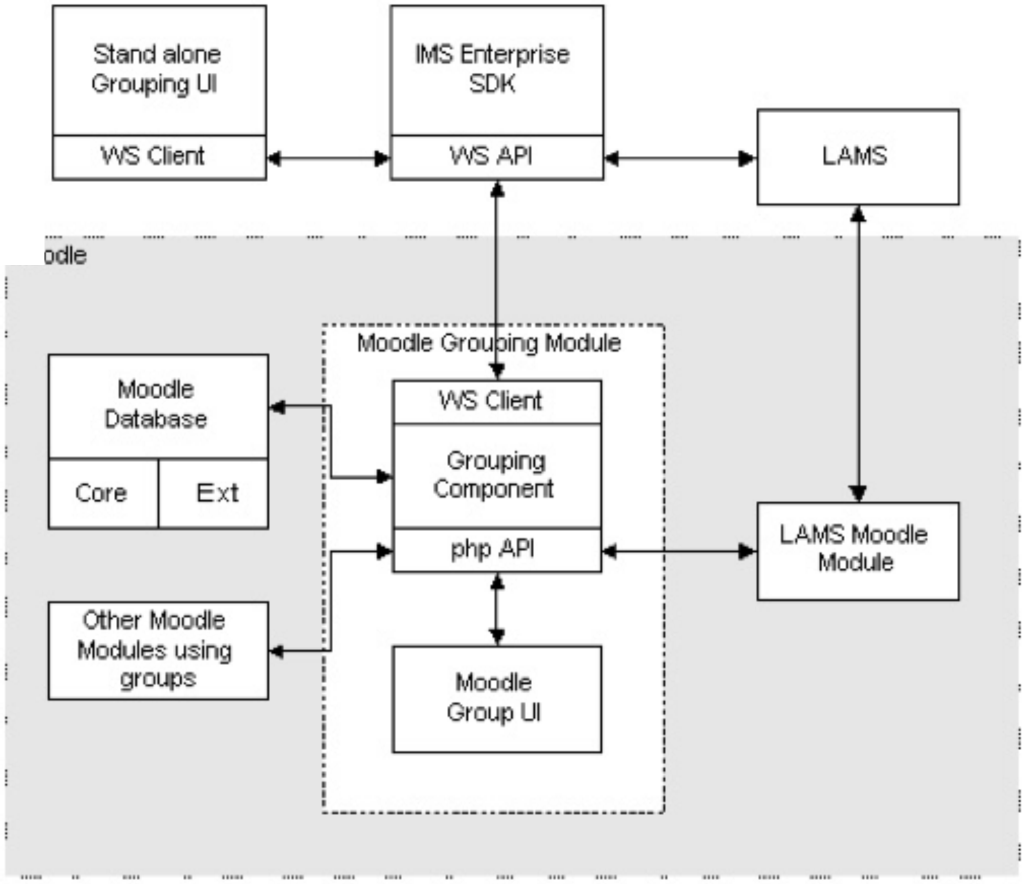
Some of the growing community of Moodle users are listed below.
To add or update your site, just use the "Registration" button on your Moodle admin page.
(Note: we check these sites regularly and remove unreachable or invalid sites)



There are 68653 currently active sites that have registered from 221 countries.
15426 of these have requested privacy and are not shown in the lists below.

Moodle architecture





Quantitative Evaluation of Frameworks for Web Applications Thirumalai Selvi, et.al

Phase 4: Identify Architectural Approaches

- **Evaluation team (Group 1)**
- **Start to identify places in the architecture that are key for realizing quality attribute goals.**
- **Identify any predominant architectural approaches – for example:**
  - Client-server
  - 3-tier
  - Proxy
  - Publish-subscribe
  - Redundant hardware

Phase 5: Generate Utility Tree

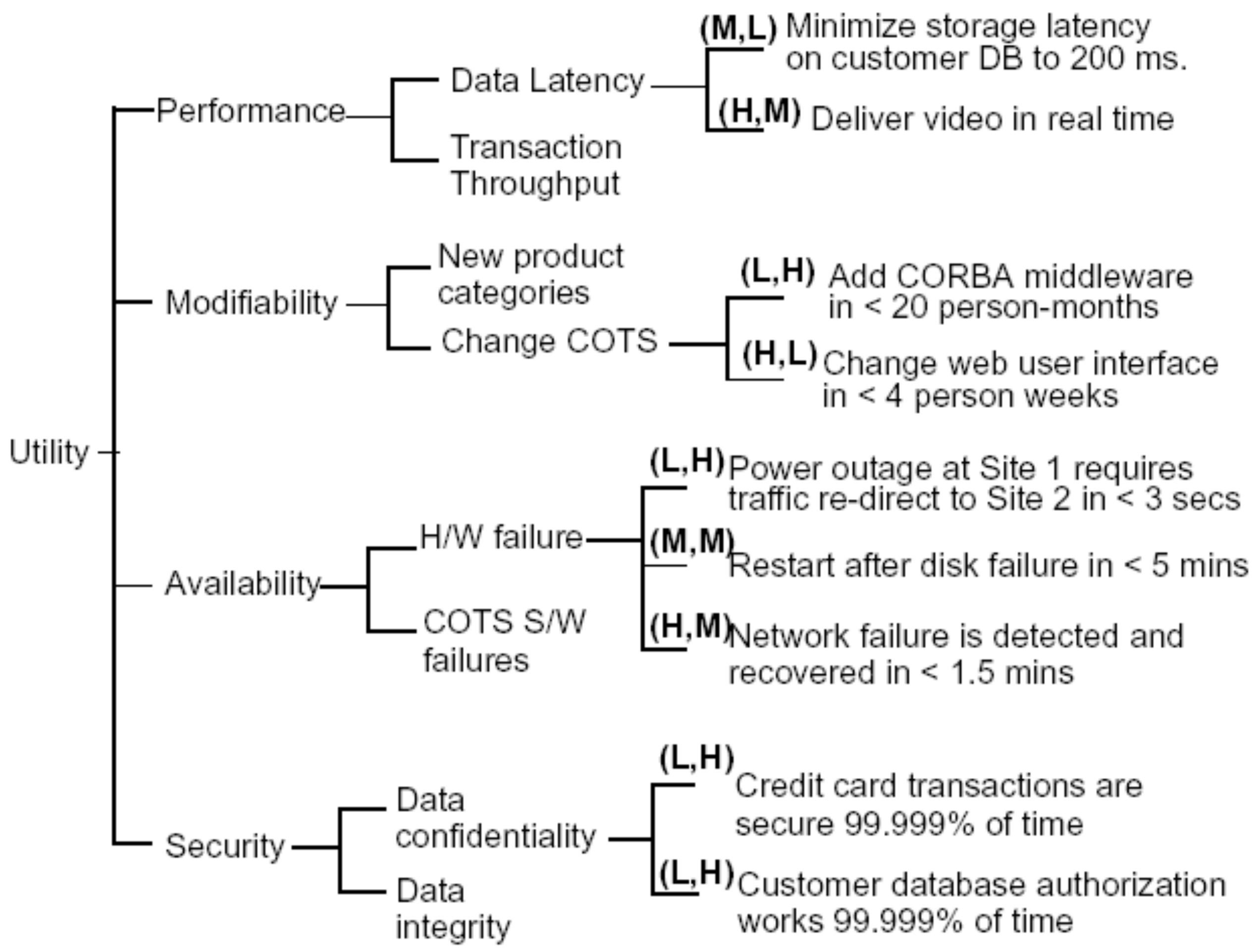
- **Evaluation team (group 1) & project manager (group 2)**
- **Identify, prioritize, and refine the most important quality attribute goals by building a *utility tree*.**

  - A utility tree is a top-down vehicle for characterizing the “driving” attribute-specific requirements

  - Select the most important quality goals to be the high-level nodes (typically performance, modifiability, security, and availability)

  - Scenarios are the leaves of the utility tree
    - Output: a characterization and a prioritization of specific quality attribute requirements.

Phase 5: Utility Tree /cont.



Utility
Performance
Data Latency
Transaction Throughput
(M,L) Minimize storage latency on customer DB to 200 ms.
(H,M) Deliver video in real time
Modifiability
New product categories
Change COTS
(L,H) Add CORBA middleware in < 20 person-months
(H,L) Change web user interface in < 4 person weeks
Availability
H/W failure
COTS S/W failures
(L,H) Power outage at Site 1 requires traffic re-direct to Site 2 in < 3 secs
(M,M) Restart after disk failure in < 5 mins
(H,M) Network failure is detected and recovered in < 1.5 mins
Security
Data confidentiality
Data integrity
(L,H) Credit card transactions are secure 99.999% of time
(L,H) Customer database authorization works 99.999% of time

Phase 5 – Scenario examples

- **Use case scenario**
  - Remote user requests a database report via the Web during peak period and receives it within 5 seconds.
- **Growth scenario**
  - *Add a new data server to reduce latency in scenario 1 to 2.5 seconds within 1 person-week.*
- **Exploratory scenario**
  - Half of the servers go down during normal operation without affecting overall system availability.
- **Scenarios should be as specific as possible.**

Phase 6: Analyze Architectural Approaches

- **Evaluation Team (group 1)**
- **Probes architectural approaches from the point of view of specific quality attributes to identify risks.**
  - Identify the approaches that pertain to the highest priority quality attribute requirements
  - Generate quality-attribute specific questions for highest priority quality attribute requirement
  - Ask quality-attribute specific questions
  - Identify and record risks and non-risks, sensitivity points and tradeoffs

Phase 6: Analysis /cont.

- **Quality attribute questions probe styles to elicit architectural decisions which bear on quality attribute requirements.**

Examples

- **Performance**
  - How are priorities assigned to processes?
  - What are the message arrival rates?
  - What are transaction processing times?

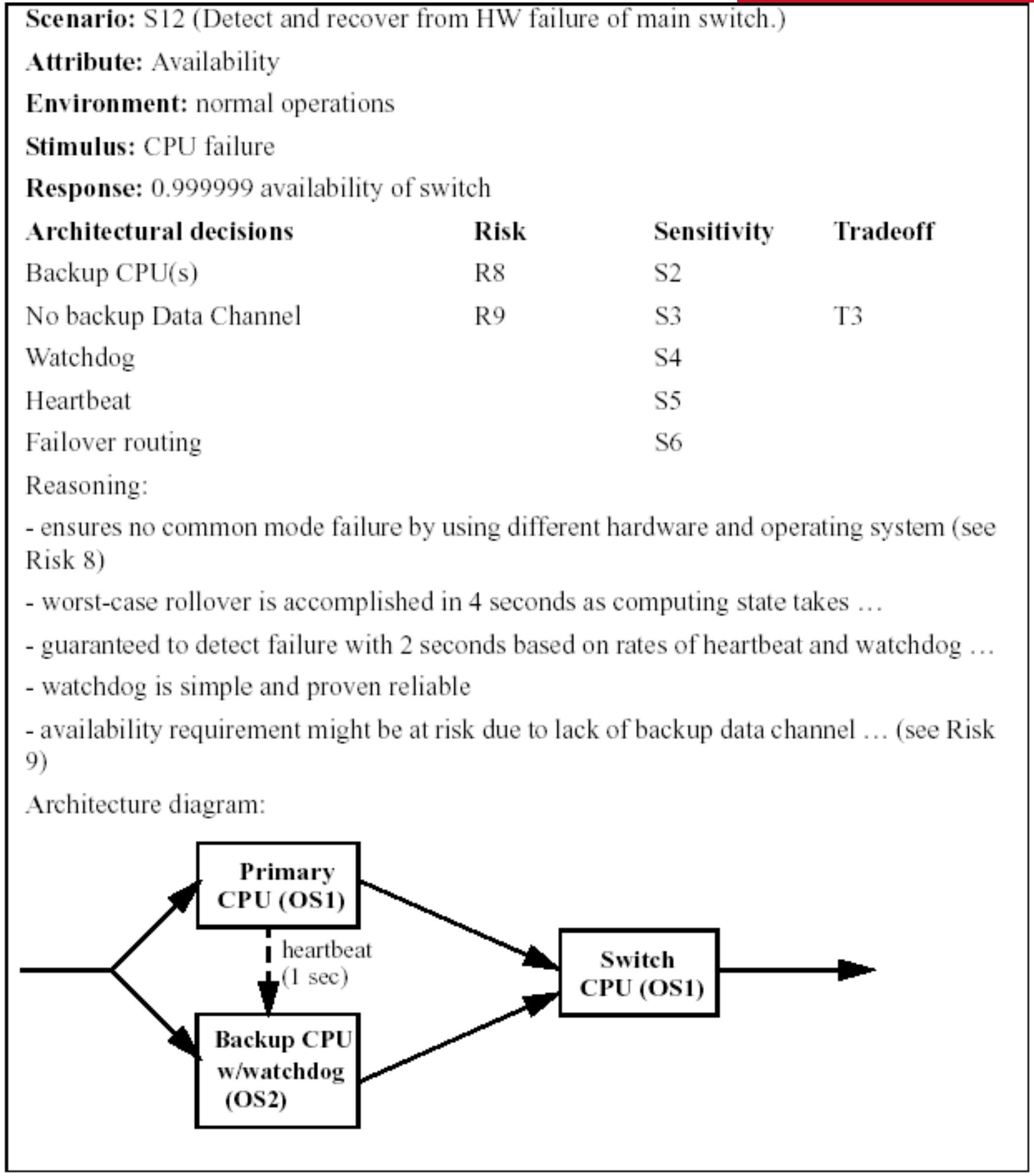
- **Modifiability**
  - Are there any places where layers/facades are circumvented ?
  - What components rely on detailed knowledge of message formats?
  - What components are connected asynchronously?

Phase 6: Analysis /cont.

› Scenarios

Vs.

Architecture



Scenario: S12 (Detect and recover from HW failure of main switch.)

Attribute: Availability

Environment: normal operations

Stimulus: CPU failure

Response: 0.999999 availability of switch

Architectural decisions	**Risk**	**Sensitivity**	**Tradeoff**
Backup CPU(s)	R8	S2	
No backup Data Channel	R9	S3	T3
Watchdog		S4	
Heartbeat		S5	
Failover routing		S6	

Reasoning:

- ensures no common mode failure by using different hardware and operating system (see Risk 8)

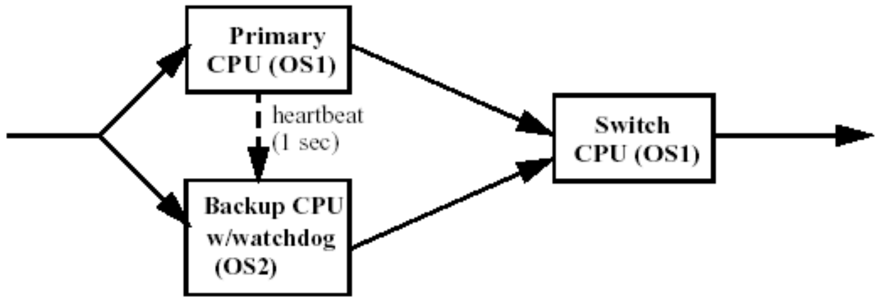
- worst-case rollover is accomplished in 4 seconds as computing state takes …

- guaranteed to detect failure with 2 seconds based on rates of heartbeat and watchdog …

- watchdog is simple and proven reliable

- availability requirement might be at risk due to lack of backup data channel … (see Risk 9)

Architecture diagram:



Phase 6: Sensitivity & Tradeoffs

More Examples

- **Sensitivity points** – A property of a component that is affects the success of system.
  - "The number of simultaneous database clients will affect the number of transaction a database can process per second. This assignment is a sensitivity point for the performance"

- **Tradeoff** - A property that affects more than one attribute or sensitivity point.

Keeping the backup database affects performance also so it's a trade-off between reliability and performance.

Phase 6: Risks and Non-Risks

- **Risk**
  - The decision to keep backup is a risk if the performance cost is excessive
- **Non Risk**
  - The decision to keep backup is a non-risk if the performance cost is not excessive

Phase 7: Brainstorm & Prioritize Scenarios

› **Stakeholders generate scenarios using a facilitated brainstorming process.**

- Scenarios at the leaves of the utility tree serve as examples to facilitate the step.

- The new scenarios are added to the utility tree

Phase 8: Analyze Architectural Approaches

› Identify the architectural approaches impacted by the scenarios generated in the previous step.

› This step continues the analysis started in step 6 using the new scenarios.

› Continue identifying risks and non-risks.

› Continue annotating architectural information.

Phase 9: Present ATAM results

- Architectural approaches
- Utility tree
- Scenarios
- Risks and “non-risks”
- Sensitivity points and tradeoffs

www.rgoarchitects.com/Files/ATAM.ppt

Questions?