Software Architecture Evaluation

Daniel Soares and Ricardo Vilela

SSC5944-1 – Software Architecture, 1st Semester 2016

Agenda

- Goals
- Why, When and Who
- Expected Results
- Quality Attributes
- Architecture Evaluation Benefits
- Architecture Tradeoff Analysis Method – ATAM
- Decision-Centric Architecture Review - DCAR

Goals

- Is the software architecture suitable to system for which it was designed?
  - Will the software achieve the quality requirements?
  - Can the software be developed with the available resources?

- To evaluate architectural decisions with respect to the impact on the quality requirements.

Why Evaluate an Architecture?

- The earlier you find a problem, the better.
- Architecture evaluation is a cheap way to avoid disaster.
- However:
  - Implementation might diverge from the architecture.
  - Architecture can not determine all system's qualities.

When Can an Architecture be Evaluated?

- Classical application:
  - After the architectures is specified
  - Before the implementation starts

Early	Late
• Evaluation does not need to wait the architecture be fully specified • Iterative evaluation of architecture decisions done and pending • Cost	• The architecture is specified and the implementation completed • Understanding of legacy systems and checking if they can meet quality requirements.

Who's Involved?

Stakeholders

- Member of the development team: coders, integrators, testers, maintainers, etc
- Project decision makers: Architect, designers of components, customers, and project's manager

Evaluation teams

- They will conduct the evaluation and perform the analysis
- Not recommended to drawn evaluation team from the project staff

What result does it produce?

- Report of the answers for this question:
  - Is the software architecture suitable to system for which it was designed?
- Quality Requirements (prioritized)
  - Can it be captured from requirement document?
    - Complete and updated?
    - Express the requirements for right system?
  - The completeness and reliability of the evaluation depend on the completeness and reliability of the architectural description
- It does not tell you yes/not, good/bad, 6/10.
- It tells you where are the risks

"If you don't know where you are going
any road can take you there."

Why Are Quality Attributes Too Vague for Analysis?

Basis for architectural evaluation , but by themselves is not sufficient to judge an architecture.

- Often, requirements statements are like the following:

The system shall be robust.	The system shall be highly modifiable.	The system shall exhibit acceptable performance.

The point is that quality attributes are not absolute quantities, they exist in the context of specific goals.

Why Are Quality Attributes Too Vague for Analysis?

- In particular:
    - A system is modifiable (or not) with respect to a specific kind of change.
    - A system is secure (or not) with respect to a specific kind of threat.
- In a perfect world, the quality requirements would be completely and unambiguously specified in a requirements document.

Most of us do not live in such a world.

Why Are Quality Attributes Too Vague for Analysis?

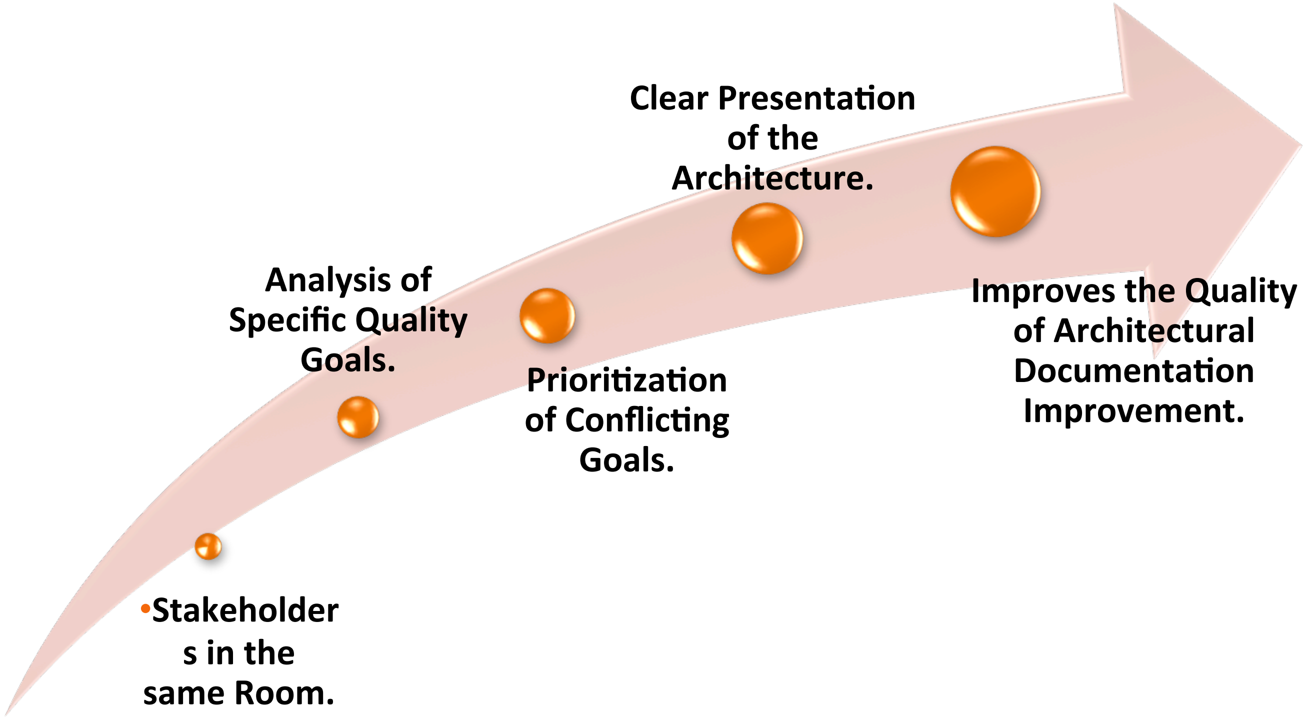
- An architecture evaluation elicits the specific quality goals against the architecture will be judged.
  - If possible, Wonderful!
  - Otherwise, we ask the stakeholders to help. (By Scenarios)
- A scenario is a short statement describing an interaction of one of the stakeholders with the system.
  - Each scenario, then, is associated with a particular stakeholder.
  - Furthermore, each scenario also addresses a particular quality, but in specific terms.

What Are the Outputs of an Architecture Evaluation?

- The ATAM method produces the followings outputs:

- Prioritized Statement of Quality Attribute Requirements.
- Mapping of Approaches to Quality Attributes.
- Risks and Nonrisks.
- Catalog of Architectural Approaches Used
- Sensitivity Points and Tradeoff Points

What Are the Benefits and Costs of performing an Architecture Evaluation?


•Stakeholders in the same Room.
Analysis of Specific Quality Goals.
Prioritization of Conflicting Goals.
Clear Presentation of the Architecture.
Improves the Quality of Architectural Documentation Improvement.

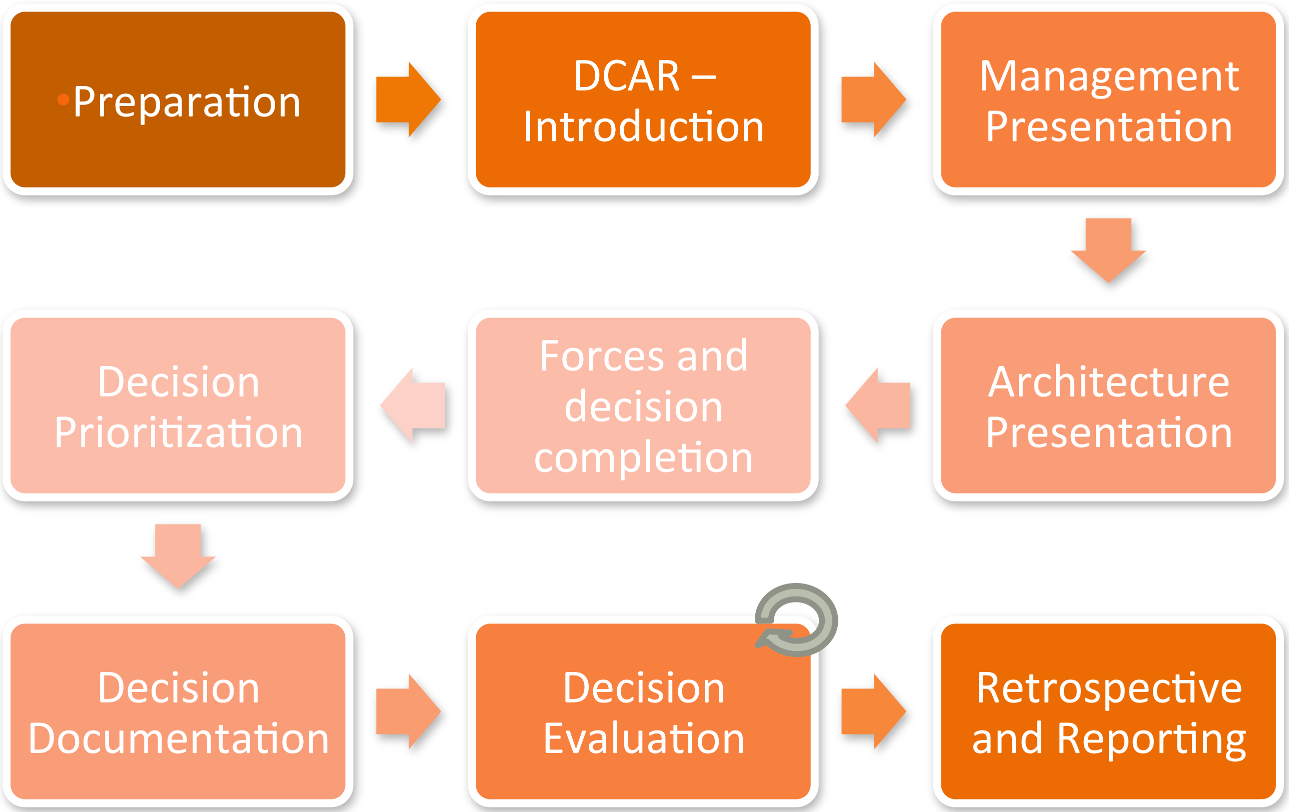
Architecture Tradeoff Analysis Method - ATAM

Steps		1st Meeting	2nd Meeting
Preparation	1. Present the ATAM	Evaluation Team and Project Decision Makers	Evaluation Team; Project Decision Makers and All Stakeholders
	2. Present Business Drivers		
	3. Present Architecture		
Investigation and Analysis	4. Identify architectural decisions		
	5. Generate quality attribute utility tree		
	6. Analyze architectural decisions		
Testing	7.Brainstorm and prioritize scenarios		
	8. Analyze architectural decisions		
Reporting	9. Present results		

Decision-Centric Architecture Review - DCAR

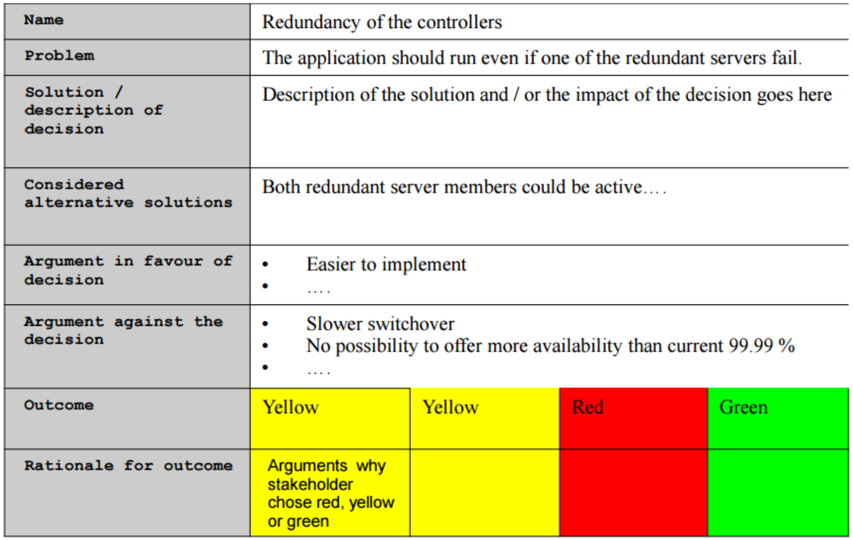
- Goal:
  - It determines the soundness of architectural decisions
- Decision-based Architecture Evaluation
  - Software architecture is the composition of set of architectural design decisions
  - Architecting is making decisions
- Benefits
  - Lightweight (it takes 4 hours + lunch)
  - Incremental
  - Keep the benefits of ATAM (Communication, documentation)
- Downsides
  - Concept of decisions is not still used in industry
  - It does not take future aspect into account
  - It requires expertise from the evaluators

Decision-Centric Architecture Review - DCAR



ATAM X DCAR

Análise da Decisão Arquitetural - 3				
Nº Cenário: R3	Cenário: A arquitetura deve permitir a troca do script de integração com pouco esforço			
Atributo(s): Modificabilidade				
Ambiente: Rotina de manutenção				
Estímulo: Um stakeholder substitui/altera o script de integração modificando o arquivo correspondente, sem alterar o formato de suas entradas e saídas.				
Resposta: A arquitetura deve permitir a troca do script de integração com pouco esforço.				
Decisões Arquiteturais	**Sensibilidade**	**Tradeoff**	**Risco**	**Não risco**
D3 - Separação do script de integração dos demais serviços	S3.1	T3.1, T3.2		N3.1
Sensibilidade: S3.1: A separação do script permite a substituição do mesmo sem a alteração do serviço. **Tradeoff:** T3.2: Interoperabilidade: Uma vez que somente é aceito scripts escritos na linguagem R T3.1: Performance: Um algorítimo de integração interno provavelmente seria mais eficiente **Não Risco:** N3.1: Boa decisão uma vez que se mostrou a melhor alternativa para se alcançar o requisito em questão. Porem, um grande esforço deverá ser alocado quando necessária a alteração dos padrões de entrada/saída de um novo script. **Sugestão:** Encapsular mais o R e usar os padrões da OGC.				



Name	Redundancy of the controllers			
Problem	The application should run even if one of the redundant servers fail.			
Solution / description of decision	Description of the solution and / or the impact of the decision goes here			
Considered alternative solutions	Both redundant server members could be active….			
Argument in favour of decision	• Easier to implement • ….			
Argument against the decision	• Slower switchover • No possibility to offer more availability than current 99.99 % • ….			
Outcome	Yellow	Yellow	Red	Green
Rationale for outcome	Arguments why stakeholder chose red, yellow or green			

OBRIGADO!

Daniel Soares and Ricardo Vilela