Architecture and Design of Distributed Dependable Systems TI-ARDI

Validation and Documentation of Architectures

Version: 8.9.2013
Agenda

• Validation for fulfilling the requirements
  – ATAM (Architecture Tradeoff Analysis Method)
• Documentation of Architecture
Requirements for Distributed Systems

• Requirements are generally divided into
  – Functional requirements
  – Non-functional requirement

• Functional requirements are used as the main input for developing the domain model

• Both types should be taken into account for defining the system architecture
OO Analysis – OO Design

Requirement specification and Use Case Model

Actor 1

Use Case x

OO Analysis Model

OO Design Model

Solution 1.

Node

Task1

Task2

Solution 2.

Node1

Node2

Task1

Task2

Task3

Non funct. req.
Examples of Non-functional Requirements

- Distribution
- Performance
- Scalability
- Availability
- Reliability
- Portability
- Dynamic configuration
- Heterogeneity and Legacy systems
- Security
- + many more
Design Characteristics

• An analysis model can be implemented by a number of different design solutions

• It is a normal engineering discipline to suggest and investigate a number of different solutions and select the best compromise for the given situation

Remember also to document the investigated design alternatives!
System Architecture Model

• System & Software architects have to define **Architecture proposals** which fulfills all requirements
• Selecting the final architecture requires a validation technique as e.g. ATAM
• Architecture Tradeoff Analysis Method (**ATAM**)  
  – “The Architecture Tradeoff Analysis Method”, by R. Kazman et. al., Software Engineering Institute, Published in 1998 by IEEE.
Steps of the ATAM, an Iterative Model
ATAM Steps

• Step 1: Collect Scenarios
  – to elicit system usage scenarios from a representative group of stakeholders

• Step 2: Collect Requirements, constraints & environments

• Step 3: Describe Architectural Views
  – to generate candidate architectures for satisfying important quality attributes

• Step 4: Attribute-Specific Analyses

• Step 5: Identify Sensitivities (sensitivity points)

• Step 6: Identify Tradeoffs
Architecture Evaluation Process

Case Study: Architecture of a Furnace Server

Furnace RTS Server

- ADC
- Furnace Task 1
- Furnace Task 2
- ... (16 tasks)
- Comm

Connections:
- From ADC to some furnaces
- From Furnace Task 1 to Furnace Task 2
- From Furnace Task 2 to Furnace Task 16
- From Comm to clients
Architecture Proposal – Option 1.

Client-Server
Architecture Proposal – Option 2.

Client-Server-Server

Duplicated (backup) servers
Architecture Proposal – Option 3.

Client-Intelligent Cache (IC)-Server
Refined Architectural option

New proposal based on a new ATAM cycle concerning security analyses

E/D: Encryption/decryption
ATAM conclusions

• ATAM enables development of candidate architectures
• ATAM enables *rational choices* among competing architectures
• Results in an enhanced understanding of, and confidence in, a systems ability to meet its requirements
• We have a *documented rationale* for the *architectural choices made*
• ATAM helps in determining the architectural tradeoff points
Architecture and SysML/UML

Design

Architectural Design

Integration and Test

Testing

Unit Testing

Coding

Detailed Design

Mechanistic Design

Object Analysis

Systems Engineering

SysML

Analysis

Requirements Analysis

Iterative Prototypes

Party!

Translation
Architectural Views

• Different view models are described in literature:
  – Kruchten, 4+1 View Model
  – Nokia model, 3+1 View Model
  – B.P. Douglass, 5 View Model

• All authors agree on the main view concept
  – different views are needed
  – each view shows a specific characteristic of the system and has different target groups
  – but names and contents vary

• It is useful to supplement with a ”+1” view, describing scenarios or Use Cases

• There can be more views
  – e.g. RUP (Rational Unified Process) operates with both a Data view and a Security view as optional views
  – e.g. B.P. Douglass operates with a Safety & Reliability view, and a Distribution view
“4+1 View” Model for SW Architecture

Ref.: Philippe Kruchten, ”The 4+1 View of Architecture, ”IEEE Software, 12(6) Nov. 1995
Nokia’s ”3+1” View Model

- **Logical View**
  - system
  - product
  - applications

- **Run-Time View**
  - computers
  - processes
  - protocols
  - persistent objects

- **Development View**
  - components
  - interfaces
  - layers

- Scenarios
  - collaborations
  - design rationale

= Process + Deployment View

Ref: [Jaaksi99]
BPDs Five Views of Physical Architecture

Ref: [Douglass2003]
Comparison between View Models

Logical View
- Classes, Packages, Interfaces
- Use cases

Implementation View (development)
- Components, Layers

Process View
- Processes, Threads, Tasks

Deployment View (Physical)
- Nodes

Use Case View

Subsystem and Component View

Distribution View

Safety and Reliability View

Concurrency and Resource View
Summary on Architectural Views

• View models are very useful
• Can be used to drive development and evaluation of architecture
• 4+1 view model – the most known – but not a holy grail
• View models are used to describe and document the architecture