C9 - Software Architecture with UML

Objectives

- Discover the main software architecture methods and languages
- Discover component based system architectures
- Understand how to create an effective software architecture
- Learn how to apply software architecture patterns
- Learn to describe software architectures using perspectives and views
- Discover how to evaluate architecture definitions.
- Discover leading architectural processes

Course environment

- One PC for two trainees

Exercise : Labs will be conducted under Enterprise Architect Business and Software Engineering Edition

Pre-requisites

- Good knowledge of the UML language

Environnement du cours

- Cours théorique
  - Support de cours imprimé et au format PDF (en anglais).
  - Le formateur répond aux questions des stagiaires en direct pendant la formation et fournit une assistance technique et pédagogique.
- Au début de chaque demi-journée une période est réservée à une interaction avec les stagiaires pour s'assurer que le cours répond à leurs attentes et l'adapter si nécessaire

Audience visée

- Tout ingénieur ou technicien en systèmes embarqués possédant les prérequis ci-dessus.

Plan du cours

First Day

Introduction

- Definitions of Architecture
  - System architecture
  - Business architecture
  - Software architecture
  - Technical architecture
  - Product line architecture
  - Enterprise architecture
- What is Software Architecture
  - What problem does it target
  - What is not Software Architecture
- Why do we need Software Architecture

- The ANSI/IEEE-1471-2000 standard
  - Architecture and Architectural descriptions
  - The IEEE 1471 Conceptual Framework
  - Views and Viewpoints
- Architecture elements and principles
  - Modules
  - Components and connectors
  - Stakeholders and concerns
  - Architecture styles
- Architecture Design Languages
- Architectural view of software development
  - System and Software Architectures
  - Software Architecture and the development cycles
  - Software Architecture and design

Software Architecture

- Definitions
- Architecture is not Design
- Components and relationships
  - Interfaces
  - Architecture models
- The basic architecture design process
  - The main steps
  - Key concerns
  - What to do and not to do
- Architectural decisions
  - Scope
  - Impact
- Architecture quality
  - Good and bad architectures
  - Being right
  - Being successful
- Making architectures work
  - The management attitude
  - The developers attitude

UML and Architecture Descriptions

- Define functional requirements
- Define non-functional requirements
- Identify components
- Model system behavior
- Create and document interfaces
- Allocate components
- Validate Architecture descriptions

Second Day

Architectural structures

- Module-based structure
  - Decomposition
  - Uses
  - Layered
  - Object
- Component and connector structure

    - Communication processes
    - Concurrency
    - Shared data
    - Client-server
- Allocation structure
    - Deployment
    - Implementation
    - Work assignment

Architectural views

- Architecture views
    - Views and stakeholders
    - Viewpoints
- Kruchten s 4+1 view model
    - Logical view
    - Process view
    - Development view
    - Physical view
    - Use cases view
- Siemens 4 view model (S4V)
    - Conceptual, Module and Code views
    - Execution view and Hardware architecture
- Software Engineering Institure (SEI) 3 view model
    - Module
    - Components and Connectors
    - Allocation
- Design rationale and the Decision view
    - The need for capturing the design rationale
    - The decision view and other view models

Third Day

Architectural styles and patterns

- Patterns, Reference models and Reference Architectures
- Basic patterns
    - Event-driven
    - Pipes and filters
    - Layered architecture
    - Three-tier architecture (MVC)
    - Client-server
    - Peer to Peer
    - Share-nothing
    - Plugins
- Object oriented architecture
    - Classes and relations
    - Components and packages
    - Interfaces and dependences

Distributed systems architectures

- The distributed constraints and tradeoffs
    - Client-server
    - Statelessness
    - Specified cacheability
    - Uniform interface and operations

    - Layered System
    - Dynamic code add-in
- Service oriented Architectures
    - Components versus Services
    - Domains and Lifecycles
    - Programming by contract
    - Registrars and brokers
    - Loose or late coupling
    - The OSGi/Java example
- Resource Oriented Architecture: ReST (Representational Resource Transfer)
    - What are resources
    - Resource names
    - The notion of resource representation
    - Interface constraints
    - Application state

Fourth Day

Architectural processes

- The Rational Unified Process (RUP)
    - Artefact, Roles, Workflows and Outcomes
    - Fundamentals and Best practices
    - The four phases
    - Why RUP was not so popular?
- The Eclipse Process Framework and OpenUP
    - OpenUP as an Unified Process variant
    - OpenUP as an agile process
    - EPF Composer as an OpenUP support tool
- The Two Tracks Unified Process (2TUP)
    - The Y-shaped cycle
    - The Technical track
    - The Functional track
    - The Build track
- The Visual Architecting Process (VAP)
    - The technical process
    - The organisational process
    - Guiding principles
- Leading, following or getting out of the way?
    - Leadership vs management
    - Leading implies following
    - Why getting out of the way is needed