[redacted]

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

- Theoretical course
  - PDF course material (in English) supplemented by a printed version.
  - The trainer answers trainees' questions during the training and provide technical and pedagogical assistance.
- At the start of each session the trainer will interact with the trainees to ensure the course fits their expectations and correct if needed

Pre-requisites

- Good knowledge of the UML language

Target Audience

- Any embedded systems engineer or technician with the above prerequisites.

Evaluation modalities

- The prerequisites indicated above are assessed before the training by the technical supervision of the traineein his company, or by the trainee himself in the exceptional case of an individual trainee.
- Trainee progress is assessed by quizzes offered at the end of various sections to verify that the trainees have assimilated the points presented
- At the end of the training, each trainee receives a certificate attesting that they have successfully completed the course.
  - In the event of a problem, discovered during the course, due to a lack of prerequisites by the trainee a different or additional training is offered to them, generally to reinforce their prerequisites,in agreement with their company manager if applicable.

Plan

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

Renseignements pratiques

Inquiry : 4 days