C9 - Architecture logicielle avec UML

Objectifs

- Découvrir les principales méthodes et langages d'architecture logicielle
- Découvrir les architectures de systèmes à base de composants
- Comprendre comment créer une architecture logicielle efficace
- Apprendre à appliquer les patterns d'architecture logicielle
- Apprendre à décrire des architectures logicielles à l'aide de perspectives et de vues
- Découvrir comment évaluer les définitions d'architecture.
- Découvrir les principaux processus d'architecture

Environnement du cours

- Un PC pour deux stagiaires

Prérequis

- Bonne connaissance du langage UML

Audience visée

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

Plan du cours

Premier jour

Introduction

- Définitions de l'architecture
  - Architecture système
  - Architecture métier
  - Architecture logicielle
  - Architecture technique
  - Architecture de ligne de produits
  - Architecture d'entreprise
- Qu'est-ce que l'architecture logicielle
  - Quel problème vise-t-elle
  - Ce qui n'est pas de l'architecture logicielle
- Pourquoi avons-nous besoin d'architecture logicielle
- La norme ANSI/IEEE-1471-2000
  - Architecture et descriptions architecturales
  - Le cadre conceptuel de l'IEEE 1471
  - Vues et points de vue
- Éléments et principes d'architecture
  - Modules
  - Composants et connecteurs
  - Parties prenantes et préoccupations
  - Styles d'architecture
- Langages de description d'architecture

- Vision architecturale du développement logiciel
  - Architectures système et logicielle
  - Architecture logicielle et cycles de développement
  - Architecture logicielle et conception

L'architecture logicielle

- Définitions
- L'architecture n'est pas la conception
- Composants et relations
  - Interfaces
  - Modèles d'architecture
- Le processus de base de conception d'architecture
  - Les grandes étapes
  - Les préoccupations clés
  - Ce qu'il faut faire et ne pas faire
- Les décisions d'architecture
  - Portée
  - Impact
- La qualité d'une architecture
  - Bonnes et mauvaises architectures
  - Avoir raison
  - Réussir
- Faire vivre les architectures
  - L'attitude de la direction
  - L'attitude des développeurs

UML et les descriptions d'architecture

- Définir les exigences fonctionnelles
- Définir les exigences non fonctionnelles
- Identifier les composants
- Modéliser le comportement du système
- Créer et documenter les interfaces
- Allouer les composants
- Valider les descriptions d'architecture

Deuxième jour

Les structures architecturales

- Structure à base de modules
  - Décomposition
  - Utilisation
  - En couches
  - Objet
- Structure composants et connecteurs
  - Processus communicants
  - Concurrence
  - Données partagées
  - Client-serveur
- Structure d'allocation
  - Déploiement
  - Implémentation
  - Répartition du travail

Les vues architecturales

- Les vues d'architecture
  - Vues et parties prenantes
  - Points de vue
- Le modèle 4+1 vues de Kruchten
  - Vue logique
  - Vue processus
  - Vue développement
  - Vue physique
  - Vue cas d'utilisation
- Le modèle 4 vues de Siemens (S4V)
  - Vues conceptuelle, module et code
  - Vue d'exécution et architecture matérielle
- Le modèle 3 vues du Software Engineering Institure (SEI)
  - Module
  - Composants et connecteurs
  - Allocation
- La justification de conception et la vue décision
  - La nécessité de capturer la justification de conception
  - La vue décision et les autres modèles de vues

Troisième jour

Styles et patterns architecturaux

- Patterns, modèles de référence et architectures de référence
- Patterns de base
  - Event-driven
  - Pipes and filters
  - Architecture en couches
  - Architecture trois tiers (MVC)
  - Client-serveur
  - Pair à pair
  - Share-nothing
  - Plugins
- Architecture orientée objet
  - Classes et relations
  - Composants et paquetages
  - Interfaces et dépendances

Les architectures de systèmes répartis

- Les contraintes et compromis du réparti
  - Client-serveur
  - Absence d'état
  - Cacheabilité spécifiée
  - Interface et opérations uniformes
  - Système en couches
  - Ajout dynamique de code
- Les architectures orientées services
  - Composants et services
  - Domaines et cycles de vie
  - Programmation par contrat
  - Annuaires et courtiers
  - Couplage lâche ou tardif

    - L'exemple OSGi/Java
- Architecture orientée ressources : ReST (Representational Resource Transfer)
    - Que sont les ressources
    - Noms de ressources
    - La notion de représentation de ressource
    - Contraintes d'interface
    - État de l'application

Quatrième jour

Les processus d'architecture

- Le Rational Unified Process (RUP)
    - Artefacts, rôles, workflows et résultats
    - Fondamentaux et bonnes pratiques
    - Les quatre phases
    - Pourquoi RUP n'a-t-il pas eu tant de succès ?
- L'Eclipse Process Framework et OpenUP
    - OpenUP comme variante du Unified Process
    - OpenUP comme processus agile
    - EPF Composer comme outil de support d'OpenUP
- Le Two Tracks Unified Process (2TUP)
    - Le cycle en Y
    - La branche technique
    - La branche fonctionnelle
    - La branche de réalisation
- Le Visual Architecting Process (VAP)
    - Le processus technique
    - Le processus organisationnel
    - Les principes directeurs
- Diriger, suivre ou s'effacer ?
    - Leadership et management
    - Diriger implique de suivre
    - Pourquoi il faut savoir s'effacer