# INF5153 – Génie logiciel : conception

# Conception architecturale

Jacques Berger

# Objectifs

Introduire un patron de conception architecturale

Discuter d'applications distribuées

# Prérequis

Aucun

# Architecture logicielle

Organisation des classes sous forme de packages, sous-systèmes et couches

Représentation UML : diagramme de composants; diagramme de packages

# Architecture de déploiement

Organisation des sous-systèmes sur les différentes machines et systèmes d'exploitation

Comment le système complet sera déployé

Représentation UML : diagramme de déploiement

# Couches

L'architecture logicielle la plus commune est l'architecture en couches

# Couches

Une couche est un ensemble de classes, de sous-systèmes et de packages cohésifs qui remplissent une responsabilité du système

Les couches offrent des fonctionnalités à forte granularité

# Couches

En théorie, la communication entre les couches devrait être du haut vers le bas

Les couches de plus haut niveau invoquent les services des couches de plus bas niveau

# Couches

Couches (découpage grossier)

Présentation
Logique applicative
Services techniques

# Couches

Présentation

L'interface utilisateur

# Couches

Logique applicative

Les objets du domaine; la logique d'affaires

# Couches

Services techniques

Services utilitaires; sous-systèmes et fonctionnalités génériques (réutilisables)

Ex. : accès à une base de données, logging

# Couches

Architecture en couches stricte : une couche ne peut invoquer que la couche inférieure

Architecture en couches lâche : une couche peut invoquer n'importe quelle couche inférieure

# Avantages

Les modifications sont localisées et demandent moins d'adaptations extérieures

Favorise la réutilisation de la logique applicative et des services de bas niveau

Facilite la séparation des tâches dans une équipe de développement

# Avantages

Diminue le couplage et les dépendances

Augmente la cohésion

Augmente la clarté de la conception

Encapsulation de la complexité

# Avantages

Favorise le remplacement de certaines couches

Des couches peuvent être distribuées

# Inconvénients

Une trop grande quantité de couches peut nuire à la performance

# Couches

Couches plus détaillées

Présentation
Application
Domaine
Services techniques
Fondation

# Couches

Présentation

Aussi nommée : Vue, IHM

Fenêtres; interface graphique
Interface web
Rapports
Affichage à la console

# Couches

Application

Aussi nommée : Contrôleur

Réception des demandes de la couche présentation
Sessions
Transitions entre les fenêtres et pages
Transformation des données de présentation

# Couches

Domaine

Aussi nommée : Logique applicative; Métier

Règles du domaine
Services du domaine

# Couches

Services techniques

Aussi nommée : Services techniques de haut niveau

Persistance
Sécurité

# Couches

Fondation

Aussi nommée : Services de base

Structures de données
Threads
Fonctions mathématiques
E/S; fichiers; BD et réseau

# Évolutif

Un modèle en couches peut être simple au début du projet et se raffiner au cours des itérations

Des couches peuvent s'ajouter

# Domaine

Modèle du domaine : modélisation des concepts

Couche du domaine : Classes qui implémentent le domaine

Il y a une forte relation entre le modèle du domaine et la couche du domaine

# Domaine

En conservant les termes utilisés dans le modèle du domaine pour les classes de la couche du domaine, on évite le décalage des représentations

# Séparation Modèle-Vue

Principes :

Les objets de l'interface utilisateur ne doivent pas être utilisés par les autres objets

La logique applicative ne doit pas être dans les objets de l'interface utilisateur

# Séparation Modèle-Vue

Le modèle correspond à la couche du domaine

La vue correspond à la couche de présentation

# Séparation Modèle-Vue

Modèles cohésifs centrés sur le domaine plutôt que l'interface utilisateur

Les vues évoluent plus rapidement que les modèles; favorise cette évolution

Connecter facilement une nouvelle interface utilisateur

# Séparation Modèle-Vue

Permet plusieurs vues sur un même objet du domaine

Portabilité du modèle

# Couplage

Les couches de haut niveau auront toutes des dépendances vers les couches Services techniques et Fondation

La couche Présentation n'appelle pas directement la couche Domaine, sauf s'il n'existe pas de couche Application

# Couche Application

Elle devient nécessaire si :

Plusieurs interfaces utilisateurs

Système distribué avec sessions

Mettre en place la Séparation Modèle-Vue

# Patron Couches

Le patron Couches possède une centaine de variantes

# Architecture à 3 niveaux

Architecture classique

3 niveaux :
Interface
Logique applicative
Stockage

# Architecture à 2 niveaux

2 niveaux :
Interface
Stockage

On code directement dans les vues

# Pipe & filter

On enchaîne les tâches les unes après les autres

Chaque tâche effectue des transformations sur les données ou des sélections pour réduire le jeu de données

# Pipe & filter

Utile lorsqu'on a plusieurs transformations à appliquer à un jeu de données

Robuste et facile à effectuer en parallèle

# Systèmes distribués

Un système est distribué si certains de ses composants communiquent au travers d'un réseau

La conception architecturale est très importante pour ces systèmes car une mauvaise architecture sera très dispendieuse à rectifier

# Client-serveur

L'architecture client-serveur est la plus commune

Un serveur offre un service

Plusieurs clients consomment ce service

# Client-serveur

La communication passe toujours par le serveur

Différents types de client-serveur :

- Ad hoc
- RPC
- SOAP
- REST

# Client-serveur

Très simple à imaginer

Facile à mettre en oeuvre

# P2P

Peer-to-peer

Les pairs se découvrent sur le réseau (souvent grâce à un serveur)

Ensuite, la communication se fait de pair à pair

L'un des pairs devient le serveur, et l'autre le client

# P2P

Robustesse du réseau beaucoup plus forte

# Microservices

Les services sont décomposés en applications de petite taille

Les applications communiquent entre elles (souvent par un API REST)

# Microservices

Permet une excellente mise à l'échelle

Couplage très faible qui facilite le déploiement des applications

# Plus loin...

UML2 et les design patterns, 3ème édition
Craig Larman
Pearson, 2005
Chapitres 12 et 28