Gestion de projets informatiques complexes

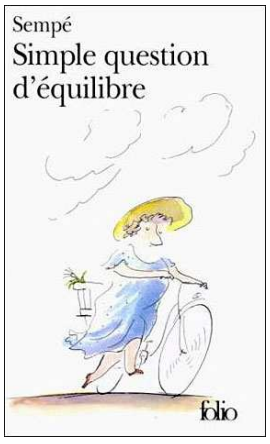
une brève histoire du management des systèmes complexes en environnement distribué, racontée aux petits enfants.

«Tout problème simple a une solution complexe…
… qui ne fonctionne pas».
O.Lockert

Antoine CELIER

Projets informatiques

POURQUOI UNE ARCHITECTURE LOGIQUE
Simple question d'équilibre **Référence : CARAPACE-157 Version 10/28/2018**



Comprendre d'où l'on vient pour savoir où l'on va

L'industrie du Logiciel
- La Crise du Logiciel

La réponse : le Génie Logiciel
- Les buts du Génie Logiciel
- Les principe du Génie Logiciel
- Méthodes de base du Génie Logiciel
- Les croyances erronées

Finalité du rôle de l'Architecte
- Finalité du Modèle de Composants
- Finalité du Modèle de Données
- Finalité du Modèle de Déploiement

La démarche

1 L'industrie du Logiciel

Données du Fonds monétaire international, 2017[3]

Rang	Pays ou territoire	PIB (en milliards de dollars)
1	États-Unis	19 390,60
-	Union européenne	17 308,86[1]
2	Chine	12 014,61
3	Japon	4 872,14
4	Allemagne	3 684,82
5	Royaume-Uni	2 624,53
6	Inde	2 611,01
7	France	2 583,56
8	Brésil	2 054,97
9	Italie	1 937,89
10	Canada	1 652,41

4 000 milliards de $

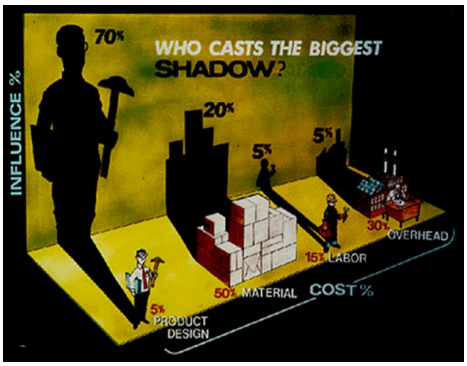
- **Chaque années, des centaines de milliards sont perdus dans les projets informatique**
  - 1/6 : succès
  - 1/2 : succès moyen (Qualité, Coût, Délai)
  - 1/3 : abandonné

1.1 La Crise du Logiciel

- **Dans les années 1970 et 1980 on a parlé de « Crise du Logiciel ».**
- **A l'époque, les symptôme de cette crise du logiciel sont les suivants**
  - Adéquation : les systèmes ne correspondent souvent pas aux besoins des utilisateurs
  - Fiabilité : Le logiciel tombe souvent en passe
  - Coût : les couts des logiciels sont rarement prévisibles et souvent perçus comme excessifs
  - Evolutivité : la maintenance du logiciel est souvent complexe, coûteuse et sujette aux erreurs
  - Ponctualité : Le logiciel est souvent en retard et livré avec des capacité inférieures à celles qui avaient été promis

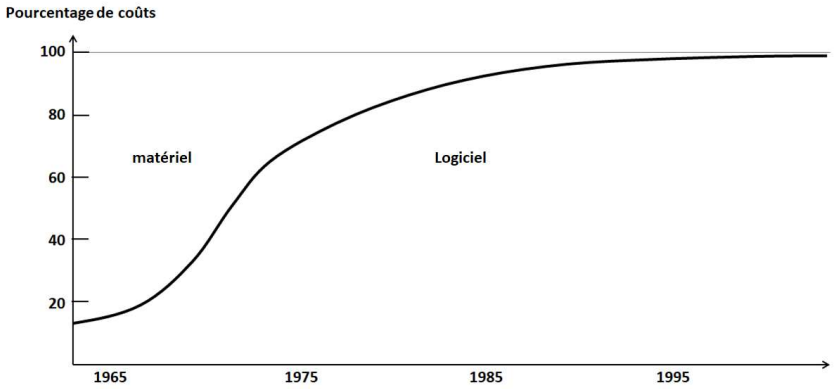
1.1 La Crise du Logiciel

- Transportabilité : le logiciel d'un système est rarement utilisé sur un autre système, même lorsque des fonctions similaires sont demandées
- Efficacité : les efforts de développement de logiciel n'utilisent pas de façon optimale les ressources disponibles (temps de calcul et espace mémoire)



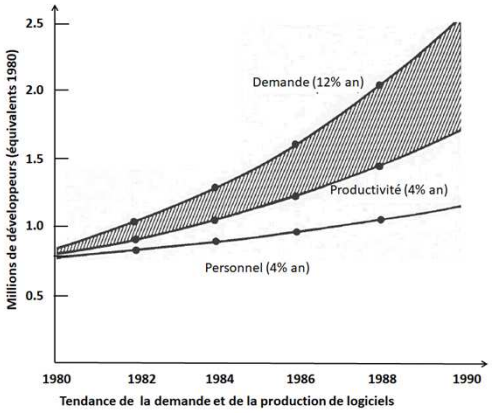
1.1 La Crise du Logiciel

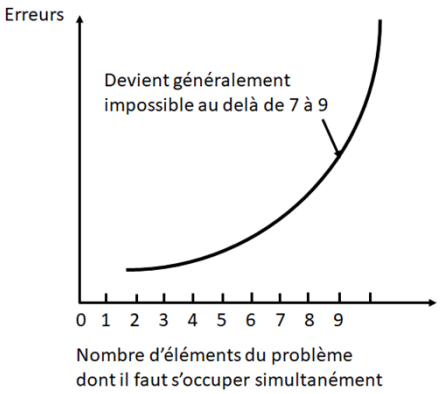
- **Il y a plusieurs causes à cette crise**
  - Un explosion du coûts des logiciels



1.1 La Crise du Logiciel

- Un tension sur le marché des développeurs
- La difficulté à gérer la complexité





2 La réponse : le Génie Logiciel

- **La réponse à la crise du logiciel va être la formalisation d'une méthode de Gestion de Projet : le cycle en V**
- **Mais surtout la formalisation de bonnes pratiques de développement : le Génie Logiciel**
  - En partant d'un constat : les spécifications évoluent généralement durant le développement du système
  - Et par la suite, les systèmes logiciels ne meurent pas, ils ne font que se modifier
  - Les principes du Génie Logiciel doivent donc transcender les effets du changement.

2.1 Les buts du Génie Logiciel

- **Evolutivité**
  - Pour cause d'évolution des spécifications (durant la conception initiale ou dans le cadre des versions successives)
  - Pour raison de maintenance
- **Efficacité**
  - Utiliser de façon optimale les ressources disponibles
  - Les ressources en temps
  - Les ressources en espace (d'adressage)
  - Les ressources périphériques
- **Fiabilité**
  - Eviter les défauts de conception et de développement
  - Récupérer les pannes et les problèmes de performance

2.1 Les buts du Génie Logiciel

- **Intelligibilité**
  - Si nous devons appliquer notre savoir à un problème difficile
  - Alors il faut une solution ayant une architecture efficace et perceptible.

2.2 Les principes du Génie Logiciel

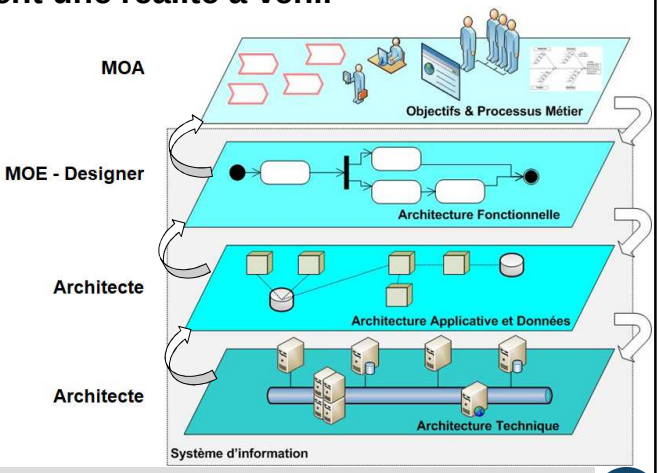
- **Abstraction**
  - Pour les données
  - Pour les traitements
- **Dissimulation d'information**
  - Masquer la façon dont la donnée ou le traitement sont implémentés
- **Modularité**
  - Module fonctionnel (orienté procédure)
  - Modules déclaratif (orienté objet)
  - Une structuration à objectif
  - Du Quoi (module de haut niveau)
  - Au Comment (module de bas niveau)

2.2 Les principes du Génie Logiciel

- **Localisation**
  - Un module doit être à faible couplage et à forte cohésion
- **Uniformité**
  - Une notation cohérente
- **Intégralité**
  - Tous les éléments importants sont présents
- **Testabilité**
  - La décomposition du système en modules rend ceux-ci aisément testables

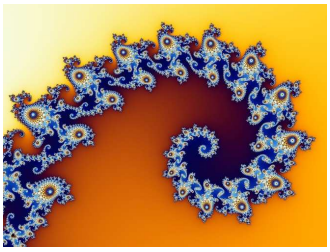
2.3 Méthodes de base du génie Logiciel

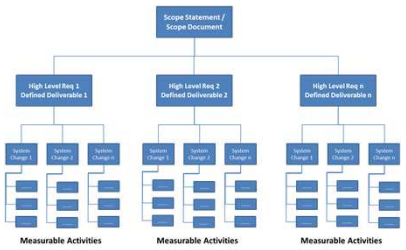
- **Définition du projet : action spécifique, nouvelle, de durée limitée, qui structure méthodiquement et progressivement une réalité à venir**



2.3 Méthodes de base du génie Logiciel

- **Approche fractale**
- **On décompose**
  - Le système en sous-systèmes
  - Les sous systèmes en composants
  - Les composants en modules
  - Les modules en classes…
- **On s'arrête quand**
  - L'atome a une finalité claire
  - L'atome a une frontière
  - L'atome a un responsable
  - L'atome a un budget
  - L'atome a un planning



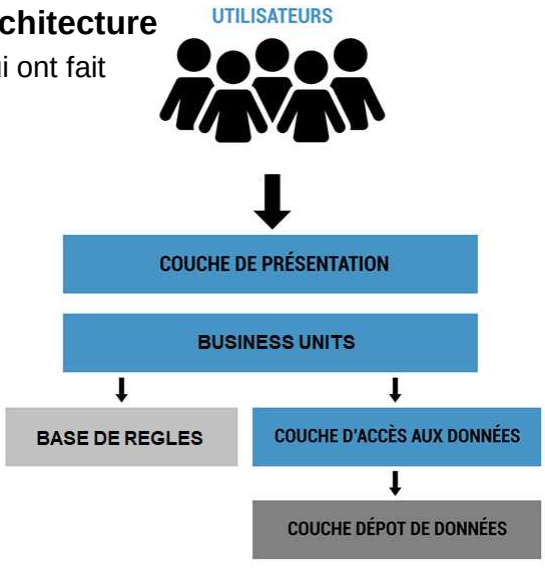


2.3 Méthodes de base du génie Logiciel

- **Réutilisation**
  - Pourquoi faire aujourd'hui ce qu'un autre a fait hier
  - Pourquoi forcer l'autre à faire demain ce que vous faites aujourd'hui
- **La réutilisation est générale**
  - Design pattern
  - Classes
  - Bases de données
  - Composants externes

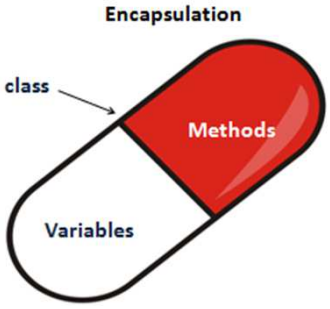
2.3 Méthodes de base du génie Logiciel

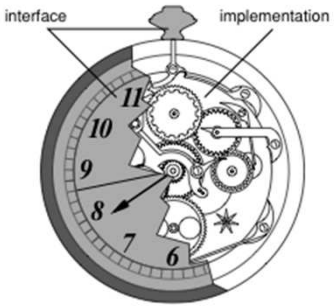
- **1ère réutilisation : l'architecture**
  - Utiliser des patterns qui ont fait leurs preuves



2.3 Méthodes de base du génie Logiciel

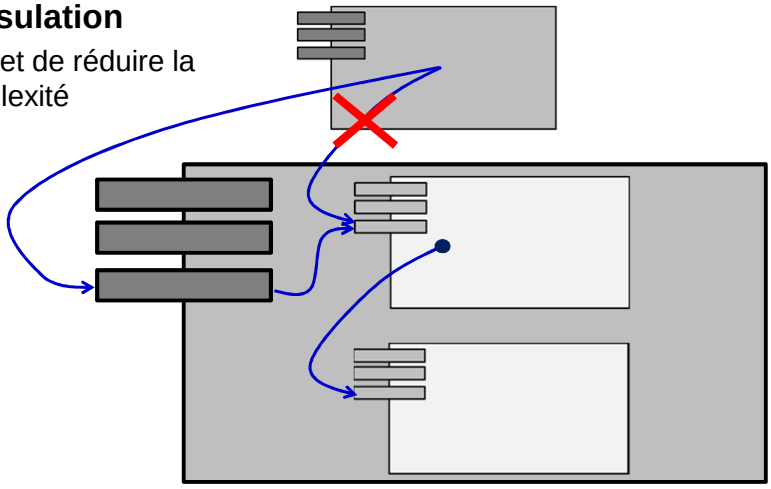
- **Encapsulation**
  - Des données
  - Des traitements





2.3 Méthodes de base du génie Logiciel

- **Encapsulation**
  - Permet de réduire la complexité

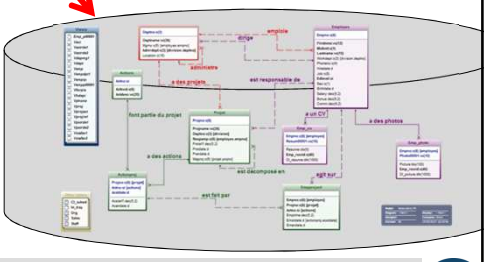


2.3 Méthodes de base du génie Logiciel

- **L'encapsulation permet le développement en parallèle**
  - En particulier  dans du Développement Agile

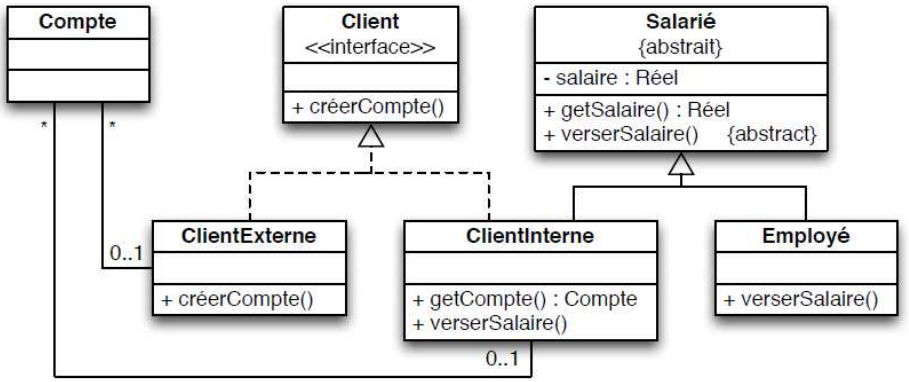
1ère implémentation : un fichier XML

2nde implémentation : une Base de Données Relationnelle



2.3 Méthodes de base du génie Logiciel

- **L'encapsulation permet l'émergence des classes abstraites, de l'héritage**

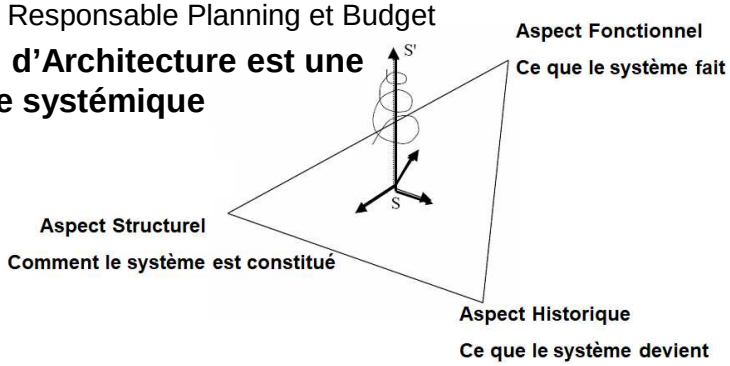


2.4 Les croyances erronées

- **Un logiciel peut être construit uniquement par assemblage de fonctions et de standards**
  - Les standard et fonctions existantes sur le marché sont une aide utile, mais ne sont pas suffisamment complets et adaptables pour permettre la construction complète du logiciel.
- **Ajouter des personnes à une équipe d'ingénieurs permet de rattraper le retard**
  - C'est le contraire: les personnes qui arrivent doivent être formées et informées sur le logiciel en cours de construction par les autres ingénieurs, ce qui entraîne des retards supplémentaires.
- **Le travail est terminé quand le logiciel est livré**
  - L'expérience montre que la majeure partie du travail commence après la livraison du logiciel au client. (phase de maintenance)

3 Finalité du Rôle de l'Architecte

- **L'Architecte accumule tous les emmerdes**
  - Ceux du MOE : donner du sens à chaque delivery
  - Ceux du Designer : implémenter les fonctionnalités
  - Ceux du Responsable Qualité : performance, fiabilité, convivialité, installabilité…
  - Ceux du Responsable Planning et Budget
- **Le travail d'Architecture est une démarche systémique**



3 Finalité du Rôle de l'Architecte

- **Dans le cours Architecture Logique que je vous ai fait il y a à peu près un mois, j'avais mis en exergue les 3 dimensions fondamentales à prendre en compte dans le travail de l'Architecte**
  - Le Modèle de Composants
  - Le Modèle de Données
  - Le Modèle de Déploiement (ou Architecture Physique)
- **Bref des contraintes de Build, des contraintes de Run**

3.1 Finalité du Modèle de Composants

- **La décomposition du système en composants**
  - Interface
  - Implémentation

  va permettre
  - La distribution du travail, l'adéquation aux savoir faire
  - Le travail en parallèle
  - La réduction du niveau de complexité
  - La fiabilité
  - La testabilité
  - La réutilisation
  - Contrôle de cohérence
  - Evolutivité
  - Transportabilité et efficacité

3.2 Finalité du Modèle de Données

- **Ce qui reste à la fin de la session, ce sont les données**
- **La base de données**
  - Recherche avancée
  - Sécurité – Contrôle de Process (exemple : non éditabilité d'une donnée publiée)
  - Collaboration
  - Réplication
  - Analyse
    - Big Data
    - I.A.

3.3 Finalité du Modèle de Déploiement

- **Le dimensionnement**
  - Des machines
  - Des espaces de stockage
  - Des bandes passantes

  va permettre de tenir les objectifs non fonctionnels
  - Performance
  - Sécurité
  - Haute Disponibilité…
- **Les architecture Clients / Serveurs vont permettre**
  - La mise à niveau facile des clients
  - L'adaptation du comportement du produit à la configuration du client

3 Finalité du Rôle de l'Architecte

- **A l'évidence**
  - Le Modèle de Composants
  - Le Modèle de Données
  - Le Modèle de Déploiement (ou Architecture Physique)

  sont liés ensemble.

- **Néanmoins, pour des raisons de simplicité pédagogique, je traiterai les 3 points successivement**

- **Dans ce cours, notre démarche est une démarche de Gestion de Projet**


Un projet c’est rigueur, bon sens,


Un projet c’est rigueur, bon sens et communication.

3 Finalité du Rôle de l'Architecte

- **La Rigueur, c'est que nous tentons de couvrir tous les aspects de la démarche systémique**
- **Le bon sens, c'est le fait que nous cherchons des solutions simples en posant correctement les questions**
- **La communication, c'est le fait que le résultat doit être facile à comprendre pour :**
  - Le développeur qui sera en charge de spécifier puis de développer les composants
  - Le Responsable de Base de Données, qui devra implémenter les structures en Base
  - L'Administrateur du Site Client, qui devra déployer et administrer la solution

4 La Démarche

- **Comment allons-nous procéder aujourd'hui**
  - Une heure sur le Modèle de Composants
    - Exercices
    - Présentation des patterns les plus classiques
  - Une heure sur le Modèle de Données
    - Exercices et présentation des patterns les plus classiques
  - Une heure où nous définissons l'exercice du cours : un ERP pour une entreprise qui fabrique des voitures et préparons le terrain
    - Architecture fonctionnelles
    - Contraintes non fonctionnelles
- **On se retrouve 15 jours plus tard pour :**
  - Analyser vos propositions
  - Identifier les points à améliorer.

4 La Démarche

- **Quelques remarques**
  - Pour ne pas complexifier la compréhension du modèle, je vous prendrai souvent des exemple du DMP
  - Je laisse volontairement de côté les technologies utilisées pour chaque composant
    - C'est votre espace de liberté
    - Certaines vous sont imposées : JEE, Android
    - Parler technologie conduit à penser implémentation sans avoir achevé la démarche de conception
- **Vous avez ensuite 15 jours pour construire en équipe la solution au problème**
  - Un Modèle de Composants
  - Un Modèle de Données
  - Un Modèle de Déploiement (ou Architecture Physique)

4 La Démarche

- **Ensuite**
  - Deux heures sur le Modèle de Déploiement
    - Je vous pose des exercices
    - Je vous présente les patterns les plus classiques
  - Deux heures où on parle de votre architecture d'ERP
  - Deux heures de conclusion et de questions/réponses
- **Le travail se fait par l'ensemble du groupe**
  - Un projet se gagne en équipe
- **3 règles**
  - Faire simple
  - Faire simple
  - Faire simple

4 La Démarche