

# MIF01 – Génie Logiciel et Gestion de Projet

## Contexte, méthodes et processus

---

Aurélien Tabard  
Département Informatique  
Université Claude Bernard Lyon 1

Basé sur le cours de Yannick Prié

# Aspects pratiques

---

Page web de l'UE :

.: <http://tabard.fr/cours/2017/mif01>

Intervenants :

.: Aurélien TABARD – [aurelien.tabard@univ-lyon1.fr](mailto:aurelien.tabard@univ-lyon1.fr)

.: Emmanuel Coquery

.: S.D. Gouraud de Thalès

.: J.R. Meunier de Alteca

# Est ce que vous connaissez

---

Un langage de programmation ?

.: Java

.: C++

Un système de gestion de code ?

.: SVN

.: Git

Des outils de modélisation ?

.: UML

Des méthodes d'organisation ?

.: Développement agile

# Pourquoi le génie logiciel ?

---

Le développement est une activité (relativement) facile



<https://www.flickr.com/photos/jacobavanzato/16152519186>

# Pourquoi le génie logiciel ?

---

On peut créer rapidement un premier prototype



<https://www.flickr.com/photos/78044378@N00/364003706>

# Pourquoi le génie logiciel ?

---

Mais l'étendre devient rapidement compliqué...



<https://www.flickr.com/photos/10402746@N04/7165270428>

# Pourquoi le génie logiciel

---

L'informatique est partout

→ des risques énormes de catastrophes

Des milliards d'€/ \$ de perdu

Des personnes frustrées, blessés ou mortes...

# Therac-25

---

Machine de thérapie par rayons pour des malades du cancer

Au moins 6 morts d'overdose de radiations

→ blessures sérieuses et morts



<http://cr4.globalspec.com/blogentry/19025/Failure-of-the-Therac-25-Medical-Linear-Accelerator>

By CBSNEWS / AP / May 25, 2010, 7:08 PM

# Toyota "Unintended Acceleration" Has Killed 89



A 2005 Toyota Prius, which was in an accident, is seen at a police station in Harrison, New York, Wednesday, March 10, 2010. The driver of the Toyota Prius told police that the car accelerated on its own, then lurched down a driveway, across a road and into a stone wall. (AP Photo/Seth Wenig) / **AP PHOTO/SETH WENIG**

[Comments](#) / [f Share](#) / [Tweet](#) / [Stumble](#) / [@ Email](#)

Unintended acceleration in Toyota vehicles may have been involved in the deaths of 89 people over the past decade, upgrading the number of deaths possibly linked to the massive recalls, the government said Tuesday.

# Le crash du réseau d'AT&T

---

Crash du réseau d'AT&T aux États Unis en 1990  
11h de downtime, ~ 60 million de \$ de dommages

[www.att.com/gen/press-room?pid=6158&cat=46&u=484](http://www.att.com/gen/press-room?pid=6158&cat=46&u=484)



LIVE BBC NEWS CHANNEL



News services Your news when you want it



Last Updated: Thursday, 22 September 2005, 16:18 GMT 17:18 UK

E-mail this to a friend

Printable version

# Deadly plague hits Warcraft world

By Mark Ward  
Technology Correspondent, BBC News website

## A deadly virtual plague has broken out in the online game World of Warcraft.

Although limited to only a few of the game's servers the numbers of characters that have fallen victim is thought to be in the thousands.



Players get the chance to be heroes in World of Warcraft

Originally it was thought that the deadly digital disease was the result of a programming bug in a location only recently added to the Warcraft game.

However, it now appears that players kicked off the plague and then kept it spreading after the first outbreak.

Since its launch in November 2004, World of Warcraft (Wow) has become the most widely played massively multiplayer online (MMO) game in the world.

Its creator, Blizzard, claims that now more than four million people are regular players.

### Last rites

Wow is an online game that gives players the chance to adventure in the fantasy world of Azeroth that is populated by the usual mixture of humans, elves, orcs and other fantastic beasts.

### SEE ALSO:

- China imposes online gaming curbs  
25 Aug 05 | Technology
- 'My life as an online gamer'  
31 Aug 05 | Technology
- Buying success in online gaming  
05 Aug 05 | Click Online
- S Korean dies after games session  
10 Aug 05 | Technology
- Warcraft sets gaming standards  
14 Feb 05 | Technology
- Lethal guinea pig kills virtual people  
12 May 00 | Sci/Tech

### RELATED INTERNET LINKS:

- World of Warcraft
- Blizzard
- worldofwar.net

The BBC is not responsible for the content of external internet sites

### TOP TECHNOLOGY STORIES

- US lifts lid on WikiLeaks probe
- Bing gains market share in search
- 'Virtual human' makes Xbox debut
- News feeds

- News Front Page
- World
- UK
- England
- Northern Ireland
- Scotland
- Wales
- Business
- Politics
- Health
- Education
- Science & Environment
- Technology**
- Entertainment
- Also in the news
- 
- Video and Audio
- 
- Have Your Say
- Magazine
- In Pictures
- Country Profiles
- Special Reports
- RELATED BBC SITES
- SPORT
- WEATHER
- CBBC NEWSROUND
- ON THIS DAY
- EDITORS' BLOG

# Déploiement de healthcare.gov

---

Lancement de la plateforme d'assurance santé aux États Unis

Retards et crash à répétition :

∴ cout politique et personnes non assurées



Mikey Dickerson Velocity NY 2014 Keynote: "One Year After healthcare.gov..."



dotScale 2014 - Robert Kennedy - Life in the Trenches of healthcare.gov

# Échec des méthodes agiles à la Macif

---

Mauvais application de méthodes de développement :

*“Alors qu’elle était plaignante, la Macif vient d’être condamnée à payer à un éditeur de logiciel 1.45 millions d’euros” (et 4 ans de développement).*

<https://www.linkedin.com/pulse/saffranchir-du-cycle-en-v-agile-canada-dry-ou-comment-maxime-blanc>

# Objectifs du cours

---

*Comprendre les enjeux du Génie Logiciel  
et de la gestion de projet*

- .: Méthodes agiles
- .: Maîtrise d'outils de conception / spécification
- .: Patrons de conception
- .: Tests
- .: Éthique

# Considéré comme acquis

---

## 1. Les bases de la programmation orientée objet

[http://tabard.fr/courses/2015/mif17/2015/MIF17\\_Rappel\\_objet.pdf](http://tabard.fr/courses/2015/mif17/2015/MIF17_Rappel_objet.pdf)

## 2. La modélisation UML de base :

.: Diagrammes statiques (classes, objet, package, etc.)

<http://tabard.fr/courses/2015/mif17/2015/UML-Statique.pdf>

.: Diagrammes dynamiques (séquences, états-transitions, etc.)

<http://tabard.fr/courses/2015/mif17/2015/UML-Dynamique.pdf>

# Plan des cours

---

.: Généralités sur les méthodes	aujourd'hui
.: Outil de gestion de code	demain
.: Cas d'utilisation	jeudi
.: Méthodes agiles	vendredi
.: Patrons de conception (Design patterns)	lundi
.: Tests	mercredi
.: Éthique et informatique	jeudi
.: Gestion de projet	vendredi

# Répartition

---

.: 8 x 2h Cours

.: 2 x 2h TD :

.: Cas d'utilisation

.: Méthode SCRUM

.: 5 x 2h TP :

.: mise en route,

.: outils de build et de gestion de projet

.: tests

.: design patterns / refactoring (2x)

# Plan de ce cours

---

## **Avant-propos**

Génie logiciel

Méthodes

Activités

Outils

Documentation de projet

# La chose à retenir

---

1. Les systèmes informatique sont complexes et demandent des méthodes de conception pour être menés à bien.
2. Ces méthodes doivent combiner rigueur et adaptation à l'inconnu et au changement.

# La modélisation (i.e. UML) dans tout ça

---

Savoir modéliser n'est pas suffisant pour bien concevoir

Il faut en plus

- .: maîtriser des techniques de conception et de programmation
- .: coder de manière accessible et réutilisable sur le long terme

Méthodes de conception

- .: propositions de cheminements à suivre pour concevoir
- .: pas de méthode ultime non plus
  - .: certains bon principes se retrouvent cependant partout

# Ce qu'il faut aimer pour arriver à concevoir

---

- .: Être à l'écoute du monde extérieur
- .: Dialoguer et communiquer avec les gens qui utiliseront le système
- .: Observer et expérimenter :  
une conception n'est jamais bonne du premier coup
- .: Travailler sans filet :  
en général, il y a très peu de recettes toutes faites
- .: Abstraire
- .: Travailler à plusieurs : un projet n'est jamais réalisé tout seul
- .: Aller au résultat :  
le client doit être satisfait, il y a des enjeux financiers

# Avertissement

---

.: Beaucoup de concepts dans ce cours

.: proviennent du domaine du développement logiciel

.: ancien (plusieurs décennies) -> plus récent

.: Tout l'enjeu est de comprendre

.: ce qu'ils décrivent / signifient

.: comment ils s'articulent

.: Méthodes

.: construire petit à petit une compréhension globale

.: lire et relire

.: chercher de l'information par soi même

.: poser des questions

.: pratiquer



# A propos

---

- .: Interaction Humain Machine (IHM)
- .: UX Design
- .: Visualisation de données

<http://tabard.fr>





## Interaction multi-dispositifs



## Visualisation de l'activité dans les lieux publics

## Faciliter les déplacements à Paris



# Plan

---

Avant-propos

**Génie logiciel**

Méthodes

Activités

Outils

Documentation de projet

# Génie logiciel

---

## Définition

.: ensemble de méthodes, techniques et outils pour la production et la maintenance de composants logiciels de qualité

## Pourquoi ?

.: logiciels de plus en plus gros, technologies en évolution, architectures multiples

## Principes

.: rigueur et formalisation, séparation des problèmes, modularité, abstraction, prévision du changement, généricité, incréments

# Qualité du logiciel (1)

---

## Facteur externes (usages)

### .: Correction (validité)

- .: aptitude à répondre aux besoins et à remplir les fonctions définies dans le cahier des charges

### .: Utilisabilité

- .: facilité d'apprentissage et d'utilisation, de préparation des données, de correction des erreurs d'utilisation, d'interprétation des retours

### .: Robustesse (fiabilité)

- .: aptitude à fonctionner dans des conditions non prévues au cahier des charges, éventuellement anormales

# Qualité du logiciel (2)

---

## Facteurs externes (environnement)

### .: Compatibilité

.: facilité avec laquelle un logiciel peut être combiné avec d'autres

### .: Extensibilité

.: facilité avec laquelle de nouvelles fonctionnalités peuvent être ajoutées

### .: Efficacité

.: utilisation optimale des ressources matérielles (processeur, mémoires, réseau, ...)

### .: Intégrité (sécurité)

.: aptitude d'un logiciel à se protéger contre des accès non autorisés.

# Qualité du logiciel (3)

---

## Facteurs internes (concepteur)

### .: Ré-utilisabilité

.: aptitude d'un logiciel à être réutilisé, en tout ou en partie, pour d'autres applications

### .: Vérifiabilité

.: aptitude d'un logiciel à être testé (optimisation de la préparation et de la vérification des jeux d'essai)

### .: Portabilité

.: aptitude d'un logiciel à être transféré dans des environnements logiciels et matériels différents

### .: Lisibilité

### .: Modularité

# Projet en général (1/2)

---

## Définition

- ∴ ensemble d'actions à entreprendre afin de répondre à un besoin défini (avec une qualité suffisante), dans un délai fixé, mobilisant des ressources humaines et matérielles, possédant un coût.

## Maître d'ouvrage (MOA)

- ∴ personne physique ou morale propriétaire de l'ouvrage. Il détermine les objectifs, le budget et les délais de réalisation.

## Maître d'oeuvre (MOE)

- ∴ personne physique ou morale qui reçoit mission du maître d'ouvrage pour assurer la conception et la réalisation de l'ouvrage.

# Projet en général (2/2)

---

## Conduite de projet

- .: organisation méthodologique mise en oeuvre pour faire en sorte que l'ouvrage réalisé par le maître d'oeuvre réponde aux attentes du maître d'ouvrage dans les contraintes de délai, coût et qualité.

## Direction de projet

- .: gestion des hommes : organisation, communication, animation
- .: gestion technique : objectifs, méthode, qualité
- .: gestion de moyens : planification, contrôle, coûts, délais
- .: prise de décision : analyse, reporting, synthèse

# Projet logiciel

---

## Problème

.: comment piloter un projet de développement logiciel ?

-> Définir et utiliser des méthodes

.: spécifiant des processus de développement

.: organisant les activités du projet

.: définissant les artefacts du projet

.: se basant sur des modèles

# Cycle de vie d'un logiciel

---

## Cycle de vie d'un logiciel

- ∴ débute avec la spécification et s'achève sur les phases d'exploitation et de maintenance

## Modèles de cycle de vie

- ∴ organiser les différentes phases du cycle de vie pour l'obtention d'un logiciel fiable, adaptable et efficace
- ∴ guider le développeur dans ses activités techniques
- ∴ fournir des moyens pour gérer le développement et la maintenance
  - ∴ ressources, délais, avancement, etc.

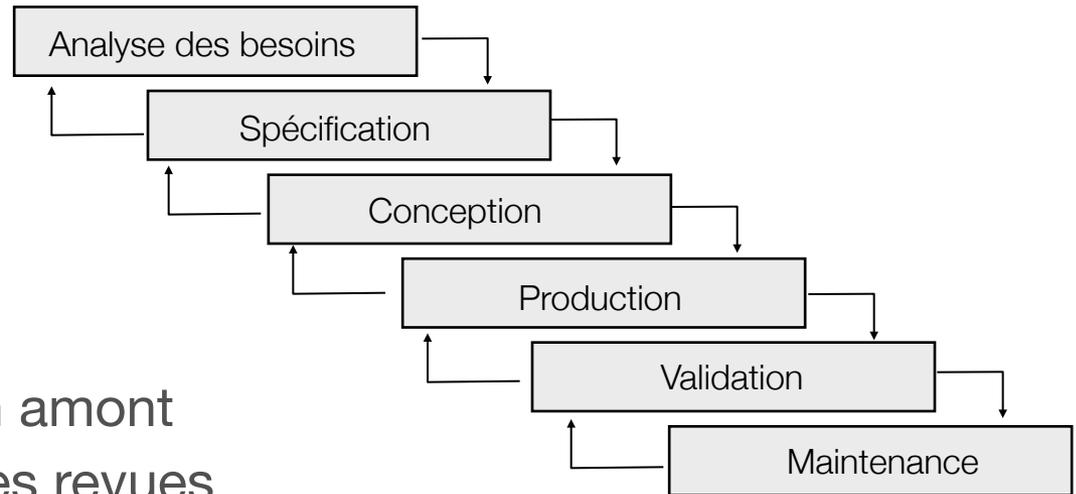
## Deux types principaux de modèles

- ∴ Modèle linéaires
  - ∴ en cascade et variantes
- ∴ Modèles non linéaires
  - ∴ en spirale, incrémentaux, itératifs



# Modèle en cascade

---



Années 70

Linéaire, flot descendant

Retour limité à une phase en amont

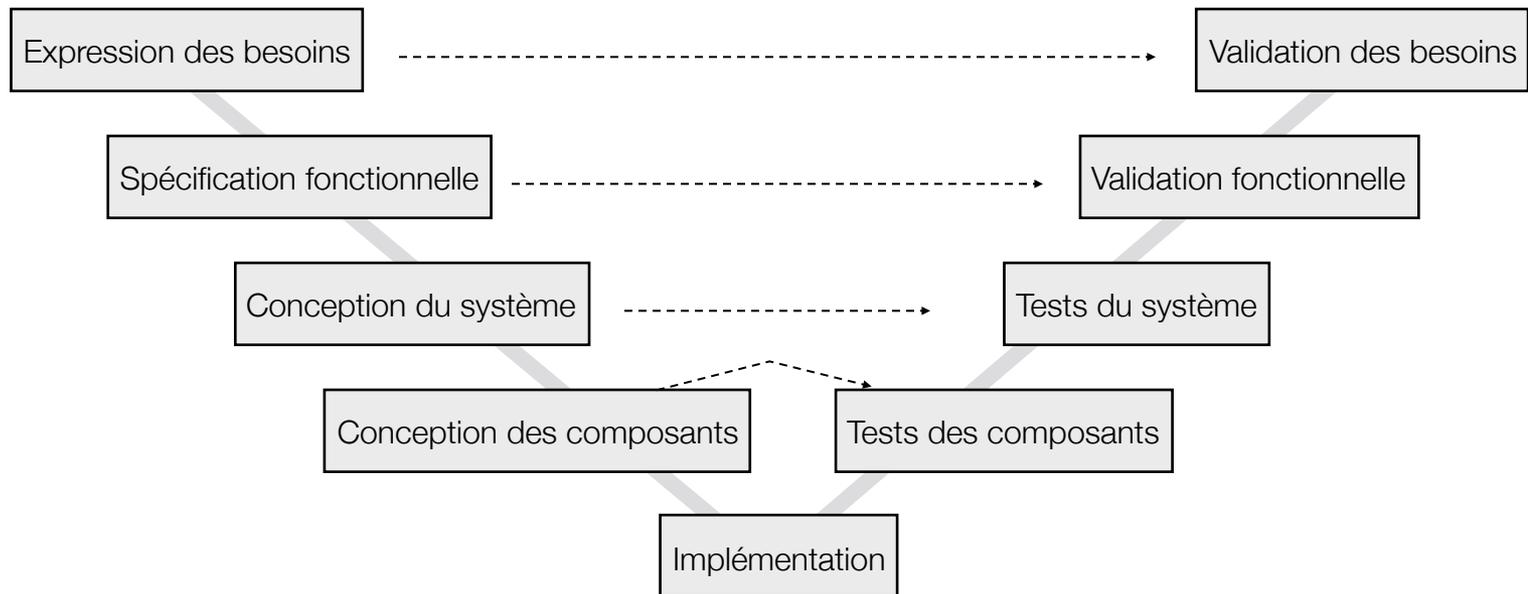
Validation des phases par des revues

Échecs majeurs sur de gros systèmes

- .: délais longs pour voir quelque chose qui tourne
- .: test de l'application globale uniquement à la fin
- .: difficulté de définir tous les besoins au début du projet

Bien adapté lorsque les besoins sont clairement identifiés et stables

# Modèle en V



Variante du modèle en cascade

Tests bien structurés

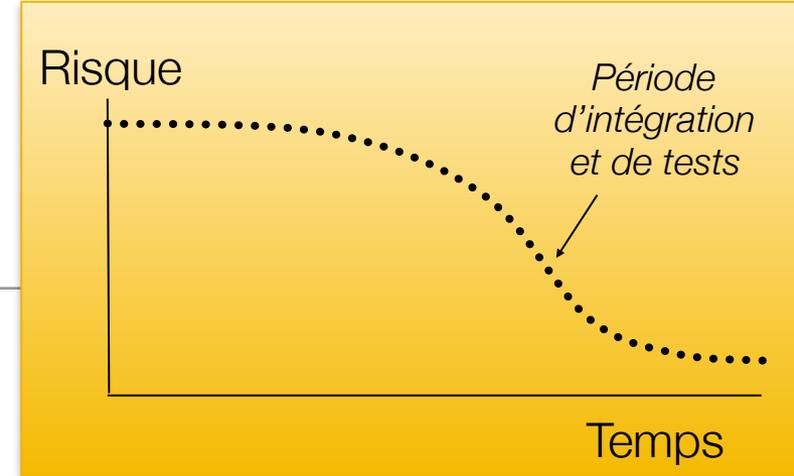
Hiérarchisation du système (composants)

Validation finale trop tardive (très coûteuse s'il y a des erreurs)

Variante : W (validation d'un maquette avant conception)

# Les problèmes du cycle en cascade

---



- .: Risques élevés et non contrôlés
  - .: identification tardive des problèmes
  - .: preuve tardive de bon fonctionnement
- .: Grand laps de temps entre début de fabrication et sortie du produit
- .: Décisions stratégiques prise au moment où le système est le moins bien connu
- .: Non-prise en compte de l'évolution des besoins pendant le cycle

# Échecs des cycles en cascades

---

- .: 25 % des exigences d'un projet type sont modifiées (35-50 % pour les gros projet) (Larman 2005)
- .: 45% de fonctionnalités spécifiées ne sont jamais utilisées (Larman 2005, citant une étude 2002 sur des milliers de projets)
- .: le développement d'un nouveau produit informatique n'est pas une activité prévisible ou de production de masse
- .: la stabilité des spécifications est une illusion
- .: Distinction entre activités trop stricte
- .: modèle théoriquement parfait, mais inadapté aux humains

# Anti-cascade

---

Nécessité de reconnaître que le changement est une constante (normale) des projets logiciels

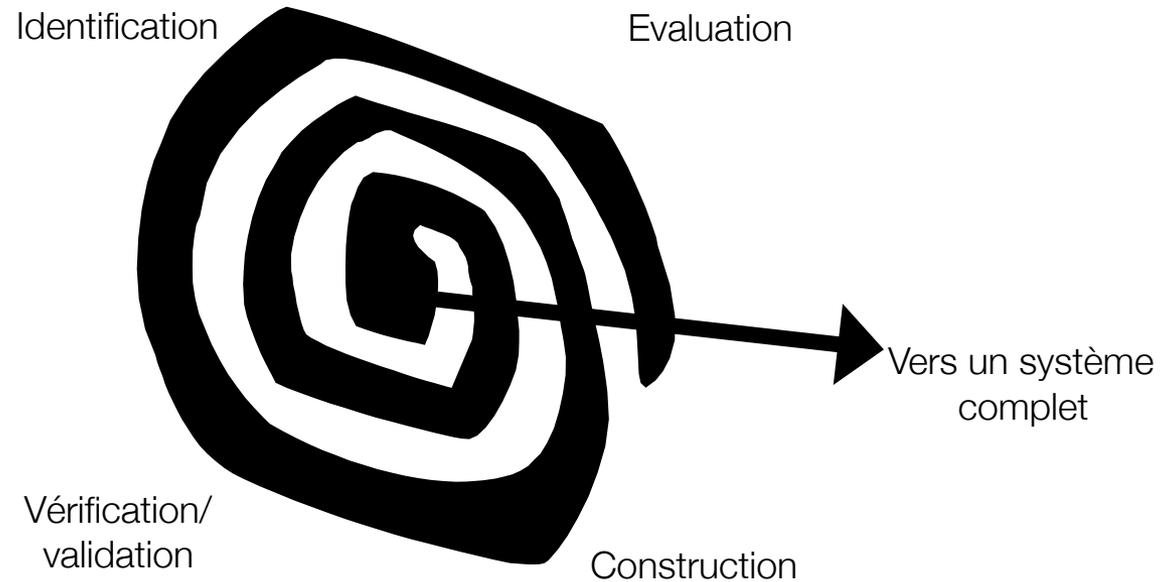
- .: feedback et adaptation : décision tout au long du processus
- .: convergence vers un système satisfaisant

## Idées

- .: construction du système par incréments
- .: gestion des risques
- .: passage d'une culture produit à une culture projet
- .: souplesse de la démarche

# Modèle en spirale

---



Incréments successifs → itérations

Approche souvent à base de prototypes

Nécessite de bien spécifier les incréments

Figement progressif de l'application

Gestion de projet pas évidente

Les méthodes  
objet en dérivent

# Plan de ce cours

---

Avant-propos

Génie logiciel

**Méthodes**

Activités

Outils

Documentation de projet

# Qu'est ce qu'une méthode ?

---

## Définitions

- .: guide plus ou moins formalisé
- .: démarche reproductible permettant d'obtenir des solutions fiables à un problème donné

## Capitalise

- .: l'expérience de projets antérieurs
- .: les règles dans le domaine du problème

## Une méthode définit

- .: des concepts de modélisation (obtenir des modèles à partir d'éléments de modélisation, sous un angle particulier, représenter les modèles de façon graphique)
- .: une chronologie des activités (construction de modèles)
- .: un ensemble de règles et de conseils pour tous les participants

## Description d'une méthode

- .: des gens, des activités, des résultats

# Méthodes en génie logiciel

---

## Grandes classes de méthodes (Bézivin)

- .: méthodes pour l'organisation stratégique
- .: méthodes de développement
- .: méthodes de conduite de projet
- .: méthodes d'assurance et de contrôle qualité

## Méthodes de développement pour

- .: construire des systèmes opérationnels
- .: organiser le travail dans le projet
- .: gérer le cycle de vie complet
- .: gérer les coûts
- .: gérer les risques
- .: obtenir de manière répétitive des produits de qualité constante

# Évolution des méthodes

---

4 vagues :

- .: Méthodes de modélisation par les fonctions
- .: Méthodes de modélisation par les données
- .: Méthodes de modélisation orientée-objet
- .: Méthodes agiles

# 1ère génération : Modélisation par les fonctions

---

Décomposition d'un problème en sous-problèmes

Analyse fonctionnelle hiérarchique : fonctions et sous-fonctions

- .: avec fonctions entrées, sorties, contrôles (proche du fonctionnement de la machine)
- .: les fonctions contrôlent la structure : si la fonction bouge, tout bouge
- .: données non centralisées

Méthodes de programmation structurée

- .: IDEF0 puis SADT

Points faibles

- .: focus sur fonctions en oubliant les données, règles de décomposition non explicitées, réutilisation hasardeuse

# 2ème génération : Modélisation par les données

---

Approches dites « systémiques »

SI = structure + comportement

Modélisation des données et des traitements

- .: privilégie les flots de données et les relations entre structures de données (apparition des SGBD)
- .: traitements = transformations de données dans un flux (notion de processus)

Exemple : MERISE

- .: plusieurs niveaux d'abstraction
- .: plusieurs modèles

Points forts

- .: cohérence des données, niveaux d'abstraction bien définis.

Points faibles

- .: manque de cohérence entre données et traitements, faiblesse de la modélisation de traitement (mélange de contraintes et de contrôles), cycles de développement trop figés (cascade)

# génération actuelle : Modélisation orientée-objet

---

Mutation due au changement de la nature des logiciels

.: gestion > bureautique, télécommunications

Approche « systémique » avec grande cohérence données/traitements

Systeme

.: ensemble d'objets qui collaborent

.: considérés de façon statique (ce que le système est : données) et dynamique (ce que le système fait : fonctions)

.: évolution fonctionnelle possible sans remise en cause de la structure statique du logiciel

Démarche

.: passer du monde des objets (du discours) à celui de l'application en complétant des modèles (pas de transfert d'un modèle à l'autre)

.: à la fois ascendante et descendante, récursive, encapsulation

.: abstraction forte

.: orientée vers la réutilisation : notion de composants, modularité, extensibilité, adaptabilité (objets du monde), souples

# génération émergente : Méthodes agiles

---

## Méthodes adaptatives (vs. prédictives)

- .: itérations courtes
- .: lien fort avec le client
- .: fixer les délais et les coûts, mais pas la portée

## Insistance sur les hommes

- .: les programmeurs sont des spécialistes, et pas des unités interchangeables
- .: attention à la communication humaine
- .: équipes auto-organisées

## Processus auto-adaptatif

- .: révision du processus à chaque itération

# Plan de ce cours

---

Avant-propos

Génie logiciel

Méthodes

**Activités**

Outils

Documentation de projet

# Développement logiciel et activités

---

Cinq grandes activités qui ont émergé de la pratique et des projets

- .: spécification des besoins
- .: analyse
- .: conception
- .: implémentation
- .: tests

# Spécification des besoins

---

Fondamentale mais difficile

Règle d'or

∴ les informaticiens sont au service du client, et pas l'inverse

Deux types d'exigences

∴ Exigences fonctionnelles

∴ à quoi sert le système

∴ ce qu'il doit faire

∴ Exigences non fonctionnelles

∴ performance, sûreté, portabilité, etc.

∴ critères souvent mesurable

Notion de conception participative

# Besoins : modèle FURPS

---

## Fonctionnalités

.: fonctions, capacité et sécurité

## Utilisabilité

.: facteurs humains, aide et documentation

## Fiabilité (Reliability)

.: fréquence des pannes, possibilité de récupération et prévisibilité

## Performance

.: temps de réponse, débit, exactitude, disponibilité et utilisation des ressources

## Possibilité de prise en charge (Supportability)

.: adaptabilité, facilité de maintenance, internationalisation et configurabilité

# Besoins : modèle FURPS+

---

## FURPS mais aussi :

- .: implémentation : limitation des ressources, langages et outils, matériel, etc.
- .: interface : contraintes d'interfaçage avec des systèmes externes
- .: exploitation : gestion du système dans l'environnement de production
- .: conditionnement
- .: aspects juridiques : attribution de licences, etc.

# Activités : analyse / conception

---

Un seule chose est sûre :

.: l'analyse vient avant la conception

## Analyse

.: plus liée à l'investigation du domaine, à la compréhension du problème et des besoins, au quoi

.: recherche du bon système

## Conception

.: plus liée à l'implémentation, à la mise en place de solutions, au comment

.: construction du système

## Frontière floue entre les deux activités

.: certains auteurs ne les différencient pas

.: et doutent qu'il soit possible de distinguer

.: d'autres placent des limites

.: ex. : analyse hors technologie / conception orientée langage spécifique

# Activités : implémentation / tests

---

## Implémentation

- .: dans un ou plusieurs langage(s)
- .: activité la plus coûteuse

## Tests

- .: tests unitaires
  - .: classe, composant
- .: test du système intégré
- .: non régression
  - .: ce qui était valide à un moment doit le rester
  - .: impossible à réaliser sans outils

# Plan de ce cours

---

Avant-propos

Génie logiciel

Méthodes

Activités

**Outils**

Documentation de projet

# Outils et processus

---

Une méthode spécifique

- .: des activités
- .: des artefacts à réaliser

Il est souvent vital de disposer d'outil(s) soutenant le processus en

- .: pilotant / permettant les activités
- .: gérant les artefacts du projet

Les outils peuvent être plus ou moins

- .: intégrés à la méthode
- .: inter-opérables
- .: achetés / fabriqués / transformés...

# Des outils pour gérer un projet (1)

---

Outils de planification

Outils de gestion des versions

Outils de gestion de documentation

Outils de maquettage

Outils de gestion des tests

# Des outils pour gérer un projet (2)

---

Outils de modélisation

Ateliers de développement logiciel

Outils de vérification

Outils de communication

...

# Plan de ce cours

---

Avant-propos

Génie logiciel

Méthodes

Activités

Outils

**Documentation de projet**

.: Spécification fonctionnelles

.: Document d'Architecture logicielle

# Cahier des charges

---

## Spécification d'un système à réaliser

- .: Fonctionnalités, description du contexte, contraintes de délais, de prix, préférences...

## Document très important

- .: Permet de se mettre d'accord en interne sur le système à construire
- .: Permet de lancer un appel d'offre
- .: Est une partie du contrat entre le demandeur et le prestataire
- .: Sert de base et de guide au chef de projet
- .: ...

# Cahier des charges fonctionnel

---

Préciser les orientations et le champ du domaine étudié,  
Analyser l'existant au niveau organisation, documents utilisés, traitements effectués, données manipulées,  
Proposer des solutions d'organisation, fonctionnelles et techniques répondant aux exigences et besoins exprimés,

# Cahier des charges fonctionnel

---

Obtenir une description globale du système (organisationnelle, fonctionnelle, technique, contraintes majeures de sécurité, de performance, interfaces avec d'autres systèmes...),

Vérifier la faisabilité organisationnelle et technique,

Aboutir à un choix argumenté d'une solution type de développement.

# Spécifications fonctionnelles

---

## Besoins fonctionnels métier

- .: Liste de fonctions que le système devra remplir
  - .: Scénarios
  - .: Cas d'utilisation
  - .: ...
- 
- .: Ne pas parler de réalisation technique

# Spécifications techniques

---

## Comment réaliser les choses

- .: Liste de fonctions à coder
- .: Architecture
- .: ...

## Attention

- .: le mélange technique / métier est facile à faire !
- .: Un exemple
  - .: <http://www.pcinpact.com/actu/news/63190-hadopi-specification-fonctionnelles.htm>

# Architecture ?

---

## Difficile à définir

- ∴ Ex. bâtiment : plombier, électricien, peintre, ne voient pas la même chose.

## Définitions

- ∴ Software architecture is not only concerned with structure and behavior, but also with usage, functionality, performance, resilience, reuse, comprehensibility, economic and technological constraints and tradeoffs, and esthetics. (RUP, 98)
- ∴ A Technical Architecture is the minimal set of rules governing the arrangement, interaction, and interdependance of the parts or elements that together may be used to form an information system. (U.S. Army 1996)



# Architecture

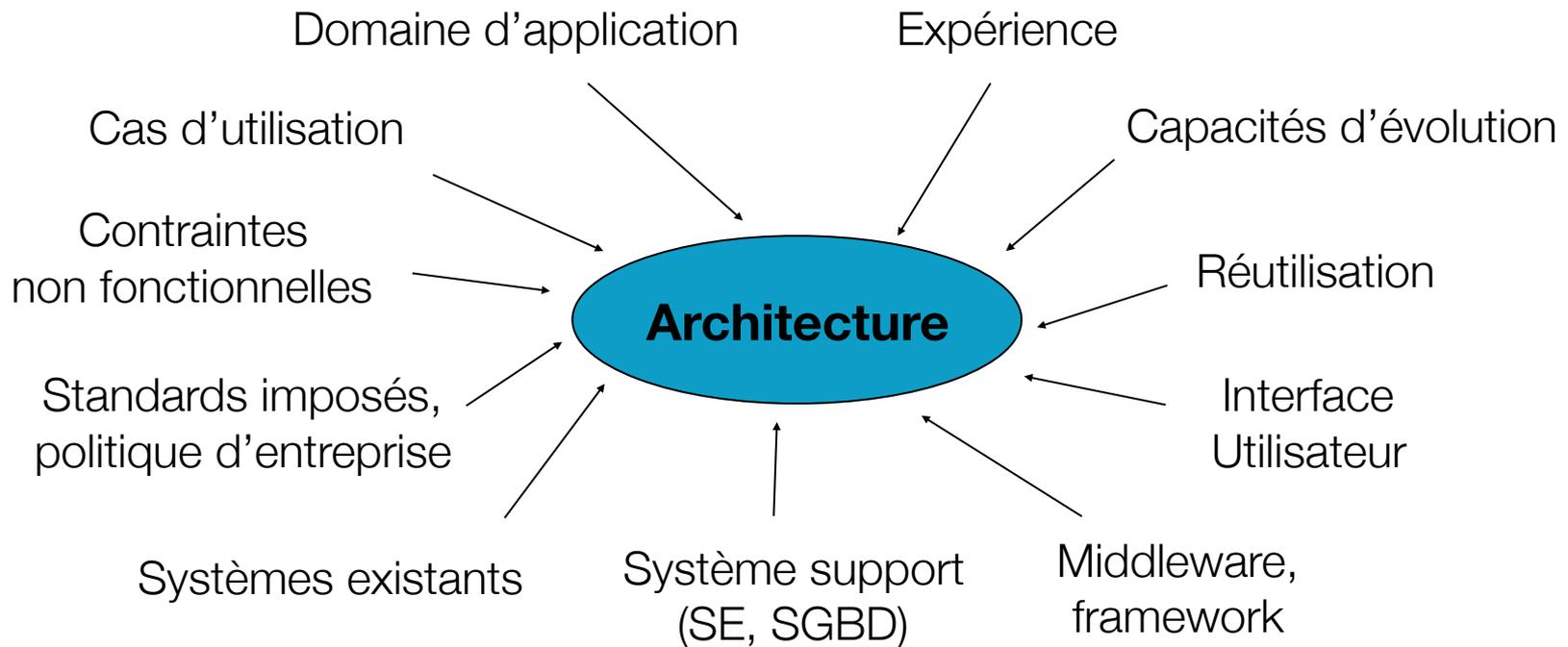
---

## Définition pour ce cours

- .: art d'assembler des composants en respectant des contraintes, ensemble des décisions significatives sur
  - .: l'organisation du système
  - .: les éléments qui structurent le système
  - .: la composition des sous-systèmes en systèmes
  - .: le style architectural guidant l'organisation (couches...)
- .: ensemble des éléments de modélisation les plus signifiants qui constituent les fondations du système à développer



# Facteurs influençant l'architecture



## Points à considérer

Performances, qualité, testabilité, convivialité, sûreté, disponibilité, extensibilité, exactitude, tolérance aux changements, robustesse, facilité de maintenance, fiabilité, portabilité, risque minimum, rentabilité économique...

# Axes pour considérer l'architecture

---

## Architecture logicielle (ou architecture logique) :

- .: organisation à grande échelle des classes logicielles en packages, sous-systèmes et couches
  - .: architectures client/serveurs en niveaux (tiers)
  - .: architectures en couches
  - .: architecture à base de composants

## Architecture de déploiement

- .: décision de déploiement des différents éléments
- .: déploiement des fonctions sur les postes de travail des utilisateurs (entreprise : central/départemental/local)

## Existence de patterns architecturaux

# Description de l'architecture (1)

---

L'architecture doit être une vision partagée

- .: sur un système très complexe
- .: pour guider le développement
- .: tout en restant compréhensible

Il faut donc mettre en place une description (ou documentation) explicite de l'architecture

- .: qui servira de référence jusqu'à la fin du cycle (et après)
- .: qui doit rester aussi stable que l'architecture de référence



# Description de l'architecture (2)

---

## Comment décrire l'architecture ?

- .: décrire les facteurs qui influencent l'architecture
  - .: facteurs architecturaux
- .: décrire les choix qui ont été faits
  - .: mémos techniques
- .: décrire l'architecture
  - .: document d'architecture du logiciel

# Description de l'architecture

## Mémos techniques

---



Les choix architecturaux doivent prendre en compte les facteurs architecturaux

Il est important de décrire les solutions choisies et leur motivation

- ∴ assurer la traçabilité des décisions architecturales

  - ∴ raisons des choix

  - ∴ alternatives étudiées, etc.

- ∴ Les « mémos techniques » décrivent les choix architecturaux

  - ∴ texte + diagrammes

Mémo technique :

Nom du problème étudié

Résumé de la solution

Facteurs architecturaux

Solution

Motivation

Problèmes non résolus

Autres solutions envisagées



# Description de l'architecture

---

## Vue architecturale

- .: vue de l'architecture du système depuis un point de vue particulier
  - .: texte + diagrammes
- .: se concentre sur les informations essentielles et documente la motivation
  - .: « ce que vous diriez en 1 minute dans un ascenseur à un collègue »
- .: description a posteriori

# DAL : Document d'Architecture du Logiciel

---

## Récapitulatif des décisions architecturales

- .: Résumé rapide de l'architecture

  - .: 1 diagramme + texte

- .: Facteurs architecturaux

  - .: quels sont les points qui ont eu une influence importante sur les choix d'architecture ?

- .: Ensemble de vues architecturales

  - .: description du matériel et du logiciel utilisés

- .: Mémos techniques

  - .: quelles sont les décisions qui ont été prises, pourquoi, etc.

# DAL :

## Contenu possible

---

- .: Context
- .: Functional View
- .: Process View
- .: Non-functional View
- .: Constraints
- .: Principles
- .: Logical View
- .: Interface View
- .: Design View
- .: Infrastructure View
- .: Deployment View
- .: Operational View
- .: Security View
- .: Data View
- .: Technology Selection
- .: Architecture Justification

# On a vu :

---

Avant-propos

Génie logiciel

Méthodes

Activités

Outils

Documentation de projet