Cours 1

Université Paris 13- PRES Sorbonne Paris cité

Modélisation objet avec UML

(Unified Modeling Language)

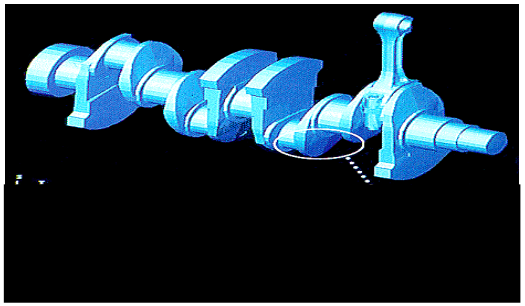
A.Osmani

Constat

- Les entreprises sont dépendantes de leur information
- Le volume des informations augmente plus rapidement que les capacités à les traiter
- Le client cherche des solutions qui répondent à ses besoins, adaptées à l'évolution rapide de l'environnement et adaptées aux contraintes d'exploitation.
- Les coûts de la maintenance sont élevés.
- Complexité croissante des systèmes informatiques

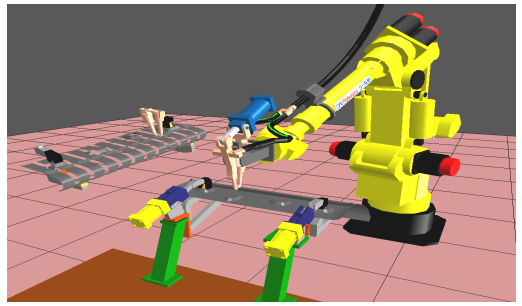
D'ou l'intérêt grandissant des méthodes d'analyse, de modélisation et de conception.

Exemple de situation industrielle complexe



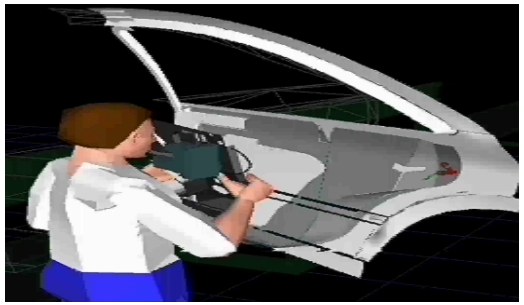
CAO

Réalité virtuelle

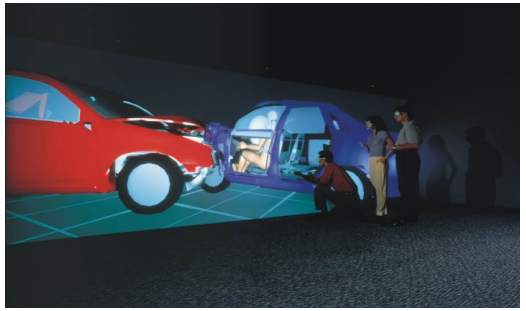


FAO

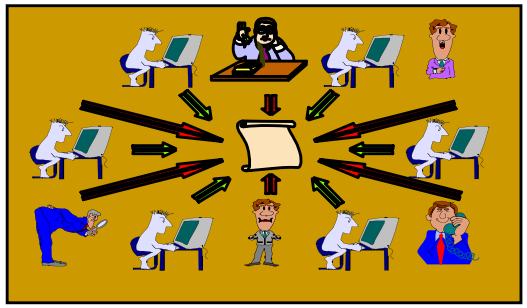
Maquette numérique



Usine numérique

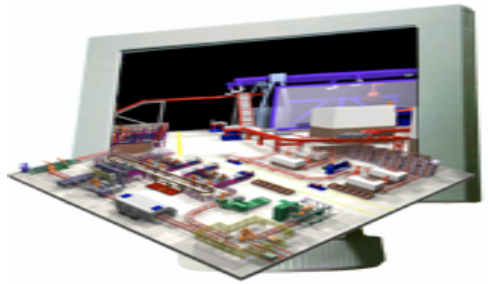


Calculs scientifiques



Gestion des connaissances

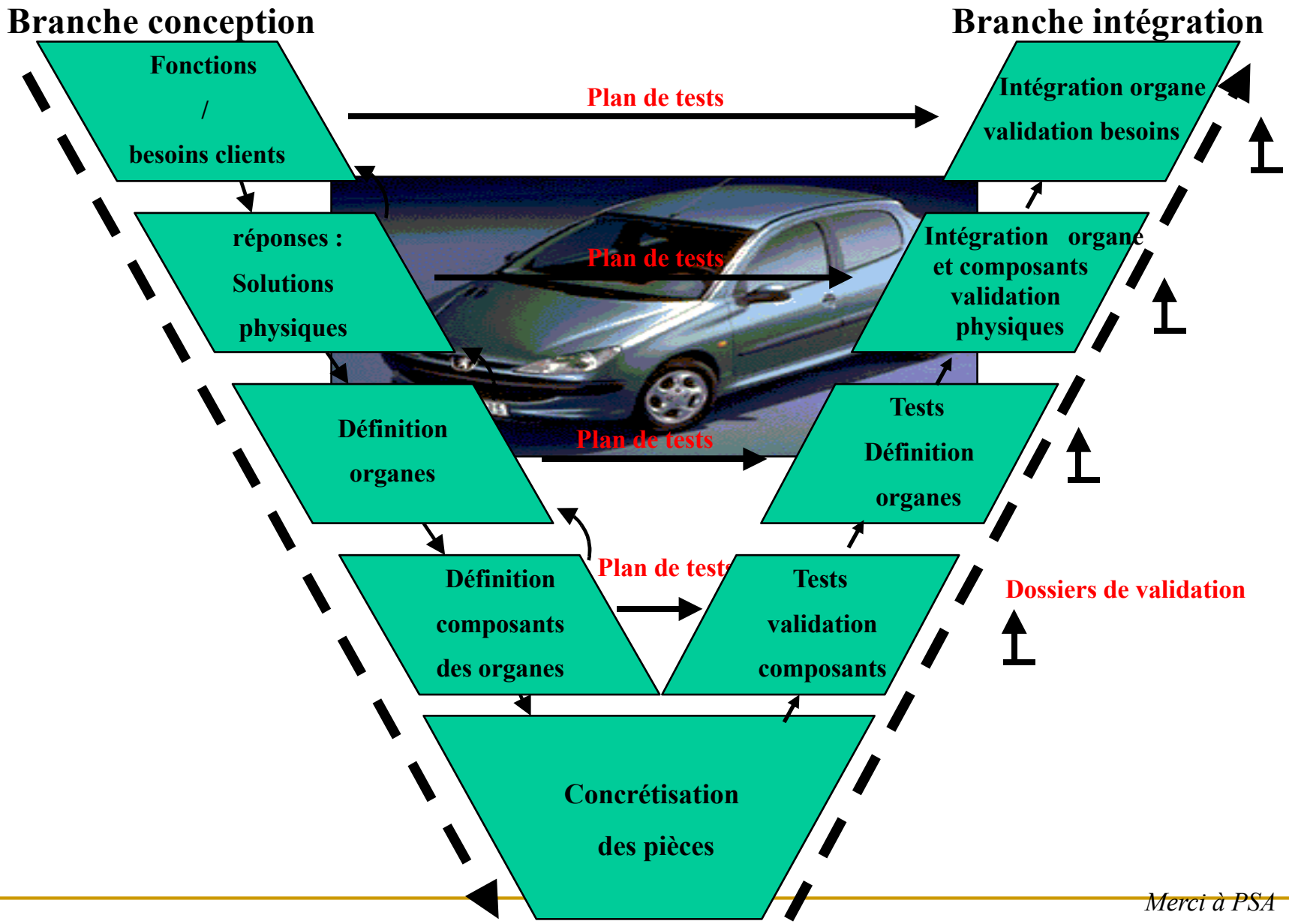
Gestion des données techniques



Architecture syst. Info.

Merci à PSA

Exemple de cycle de développement de produits industriels



Merci à PSA

Objectifs du cours

Ce cours a pour objectifs de vous faire :

- Comprendre les enjeux du Génie Logiciel ainsi que l'importance des activités de spécification et de conception
- Comprendre l'intérêt de la modélisation
- Capturer les besoins des utilisateurs
- Connaître la notation UML
- Analyser ces besoins en UML
- Spécifier et concevoir des logiciels avec UML en vue d'une programmation en Java

Déroulement du cours (5 séances)

- Introduction à la modélisation et au génie logiciel
  Modélisation UML et cas d'utilisation
- Diagramme des classes et diagramme d'objets
- Diagrammes d'interaction : séquences et collaboration
- Diagramme des états/transitions de diagramme d'activités
- Diagramme de deploiment et diagramme de composants
  Eléments du langage OCL (Objects Constraints language)
  Modélisation UML / programmation Java
  Introduction aux design patterns

Génie logiciel

Définition Ensemble de moyens techniques (méthodes, modèles, outils) utilisés pour spécifier, concevoir, réaliser (programmer), installer et maintenir des applications informatiques sûres, efficaces, conviviales, évolutives et économiques.

Cycle de vie du logiciel :

A. L'étude préalable décrit les objectifs initiaux
   - *pourquoi développer le logiciel ?*

B. Le développement qui définit et construit le logiciel
   - *que doit faire le logiciel et comment le réaliser ?*

C. L'exploitation qui assure les tâches de fonctionnement et de maintenance
   - *comment exploiter et faire évoluer le logiciel ?*

Génie logiciel

Au cours de ces 3 phases seront menées (souvent simultanément) des activités :

- d'*ingénierie du logiciel* qui définissent, réalisent ou mettent à jour les programmes formant le logiciel,
- de *production de la documentation* qui élaborent les documents techniques internes et les manuels d'installation et d'utilisation,
- de *gestion de la qualité* qui visent à garantir la simplicité d'emploi, les performances, les possibilités d'évolution, la portabilité, la lisibilité et la conformité du logiciel vis à vis des objectifs initiaux,
- de *gestion de la sécurité* qui contrôlent les éventuelles erreurs du logiciel et vérifient si ce dernier résiste aux données et manipulations incorrectes,
- de *gestion de projet* qui comprennent la gestion des coûts et l'organisation du travail (planification, contrôle et évaluation des tâches et des ressources humaines et matérielles).

Processus de Développement

Processus de développement = ensemble d'étapes pour obtenir un logiciel ou faire évoluer un logiciel existant.

Objectifs : obtenir

- des logiciels de qualité répondant aux attentes des utilisateurs
- maîtriser les coûts et le temps

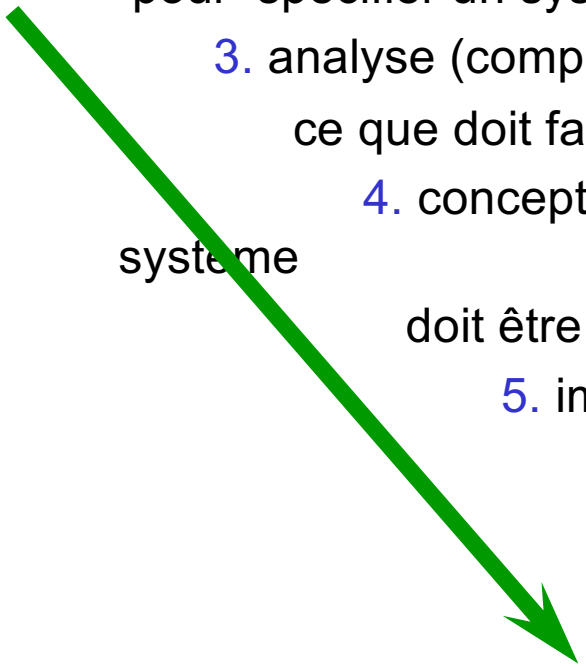
Le processus de développement se décompose selon deux axes :

- développement technique : objectif qualité de la production
- gestion du développement : mesure et prévision des coûts et des délais

Cycle de vie d'un logiciel

les 3 étapes de création d'un logiciel décrites précédemment assurent :

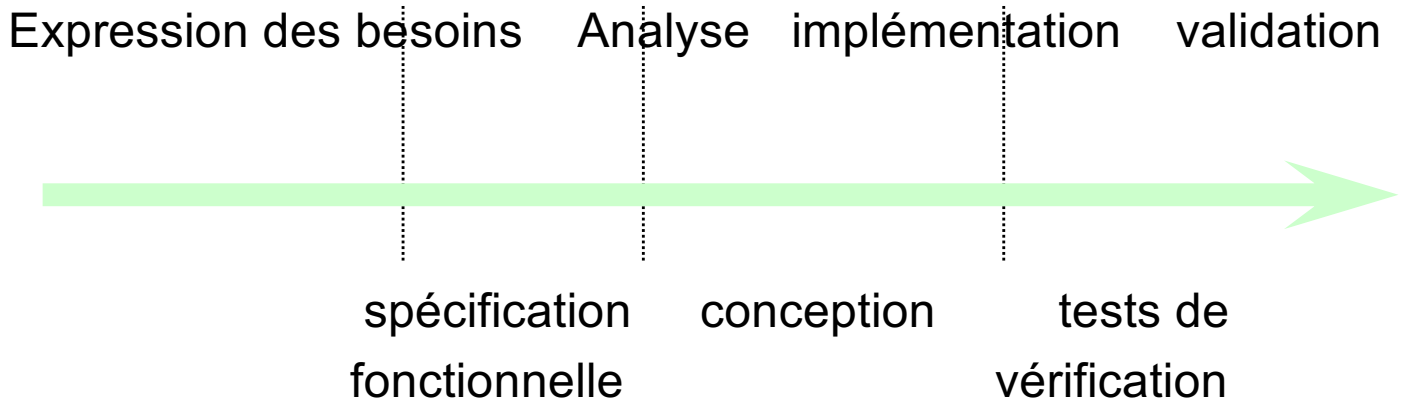
1. Expressions des besoins
2. spécifications (ce que c'est? les uses cases sont une approche pour spécifier un système)
3. analyse (comprendre le métier, comment l'élaborer ce que doit faire un système)
4. conception (s'accorder sur la manière dont le système doit être construit et non plus sur ce qu'il doit faire)
5. implémentation
6. tests et vérification : s'assurer de la qualité technologique d'un produit
7. validation
8. maintenance et évolution



Les cycles de vie d'un logiciel

Le cycle de vie d'un logiciel a pour but d'expliquer le séquencement des différentes étapes, ainsi que la manière dont elles coopèrent.

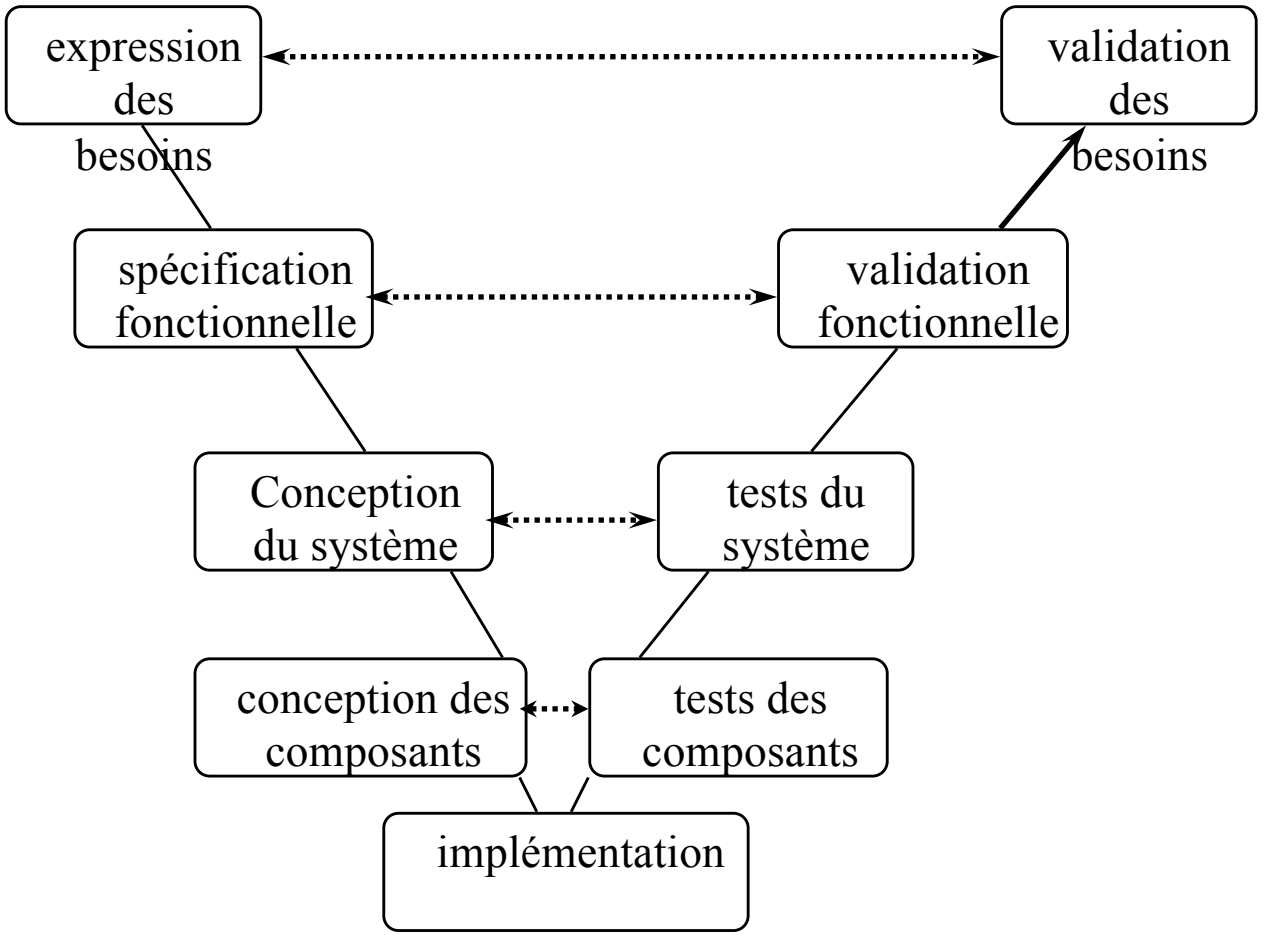
La plupart des applications traditionnelles non objets utilisent deux types de modèles : le modèle linéaire et le modèle en V.



Les cycles de vie d'un logiciel

Le modèle en V est l'un des modèles les plus utilisé actuellement

Expression des besoins Analyse implémentation validation

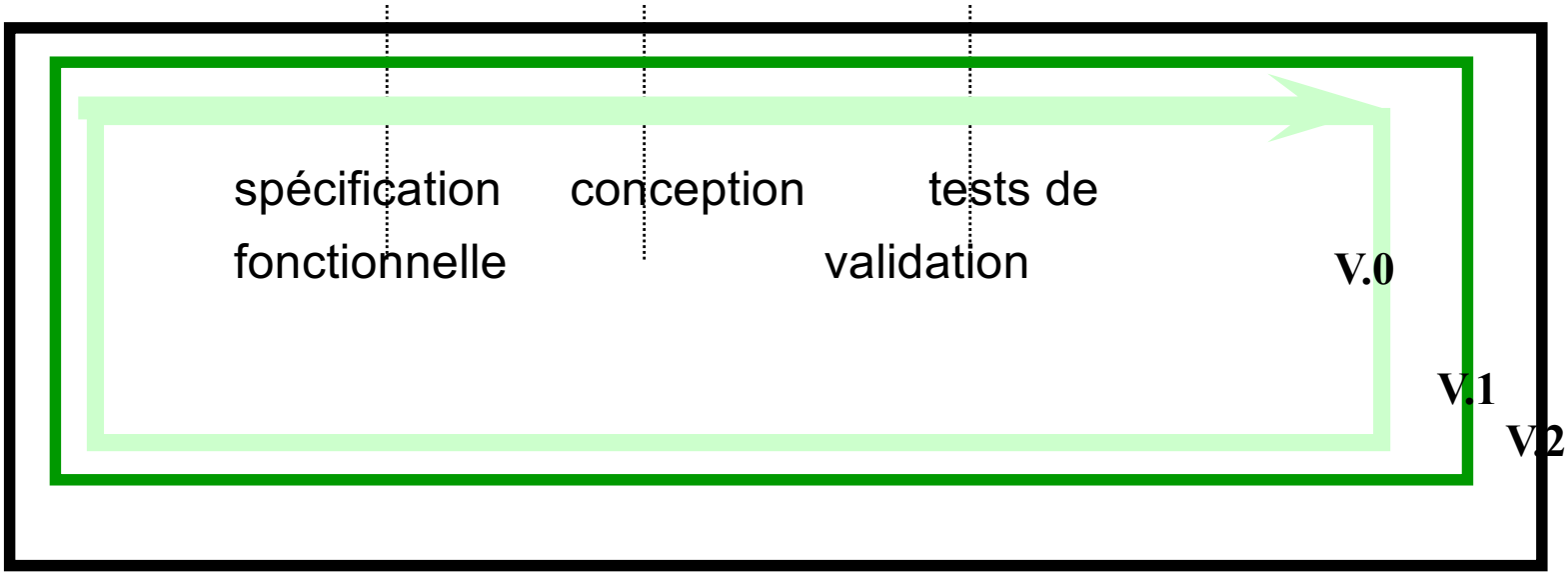


Les cycles de vie d'un logiciel

le cycle de vie d'un projet à objets possède trois caractéristiques fondamentales

- la traçabilité : correspondance entre les éléments définis dans deux phases successives
- le caractère itératif : si validité non satisfaisante alors répéter toutes les phases
- le caractère incrémental: réalisation d'une série de prototypes

Expression des besoins Analyse implémentation validation



Les cycles de vie d'un logiciel

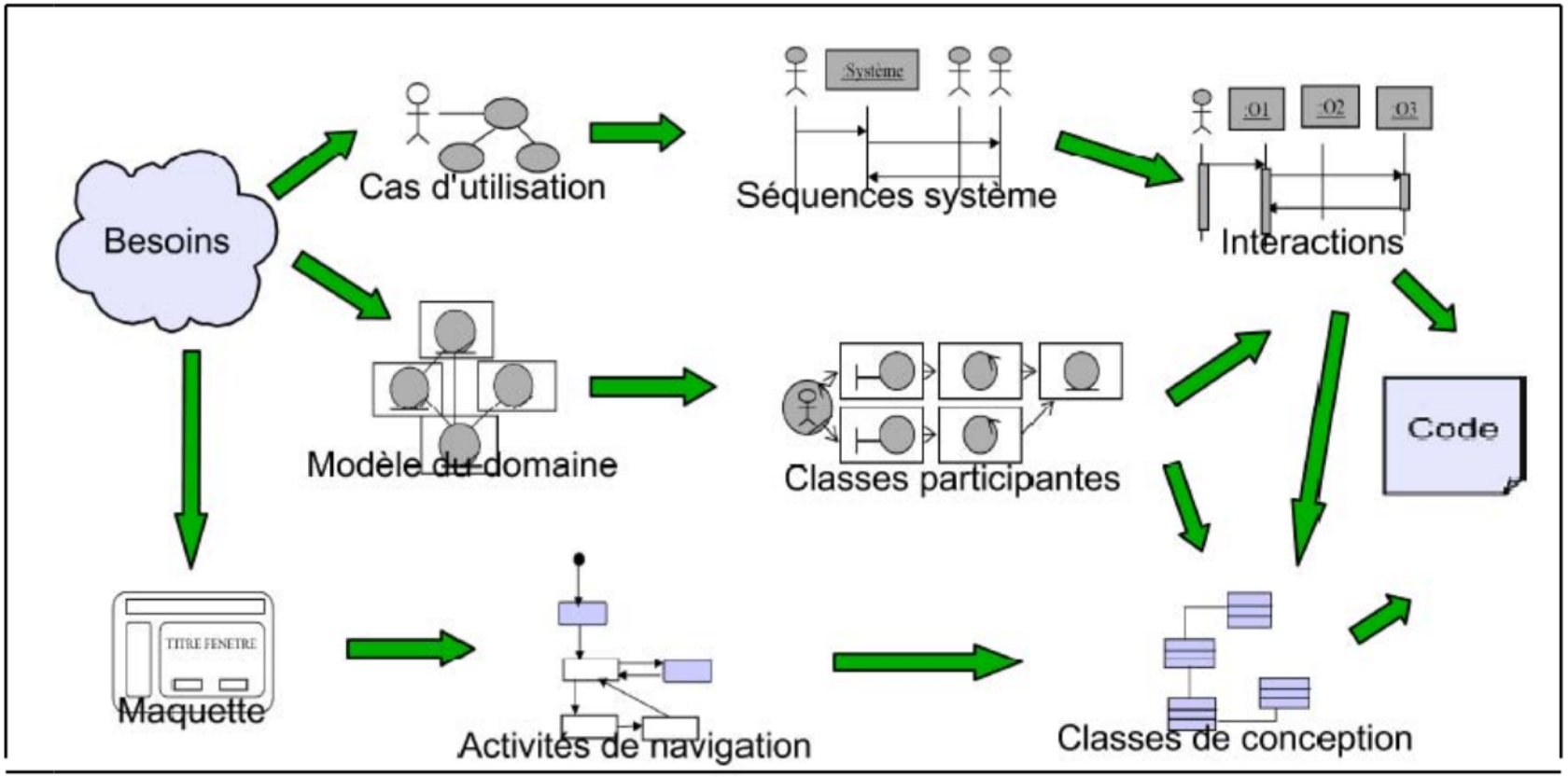


Figure 3-1 : La démarche Agile

La création du modèle

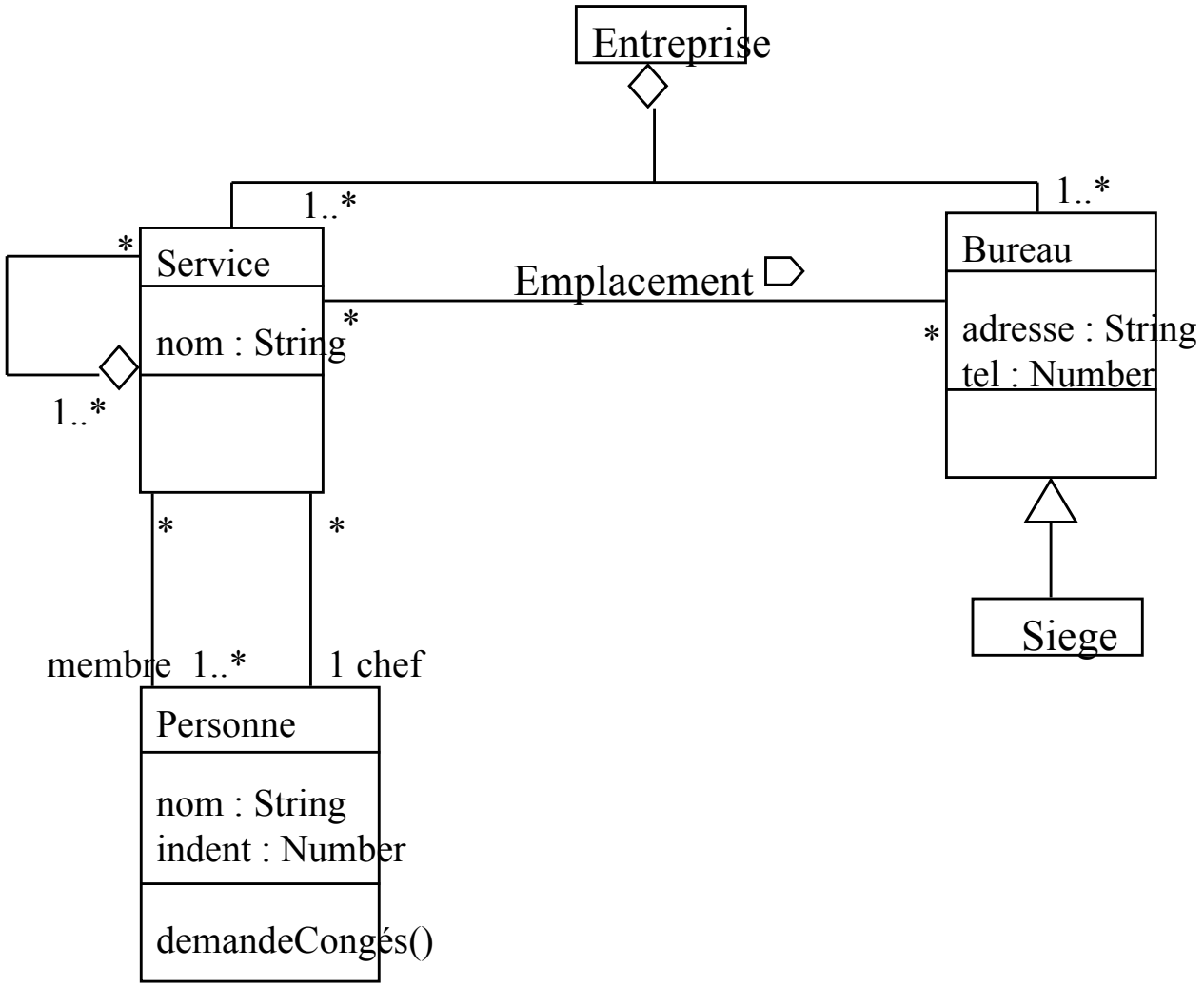
Les phases d'analyse et de conception donnent naissance à un ensemble de modèles du système.

Ceux-ci seront implémentés dans un langage de programmation.

Nous allons voir

- Un exemple de modèle
- Quelques définitions
- Quels sont les objectifs de la modélisation ?
- Quelles sont les propriétés d'un modèle ?
- La modélisation dans le domaine du logiciel
- Quelques langages de modélisation

Exemple de modèle (UML)



Exemple de modèle (UML)

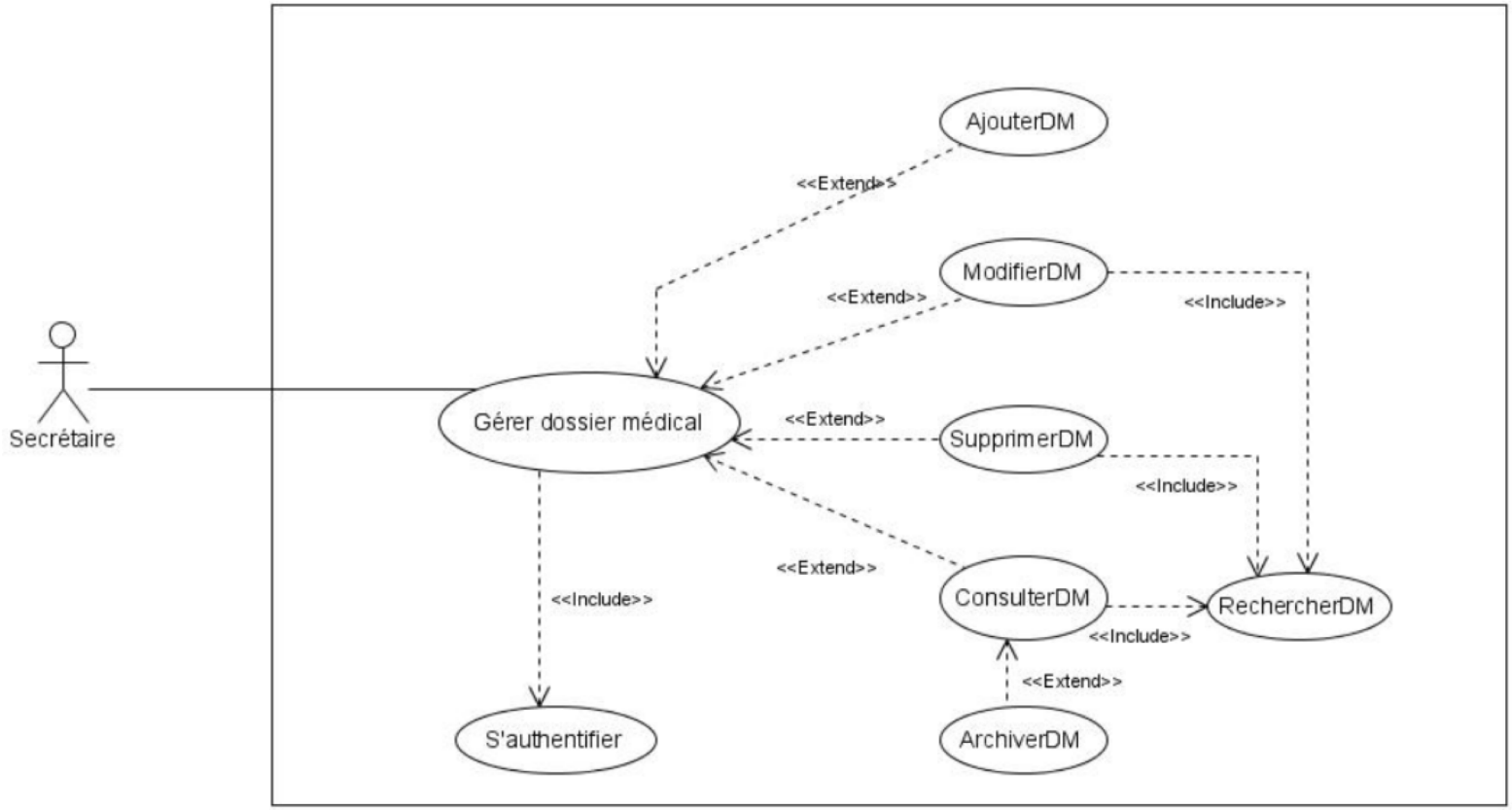


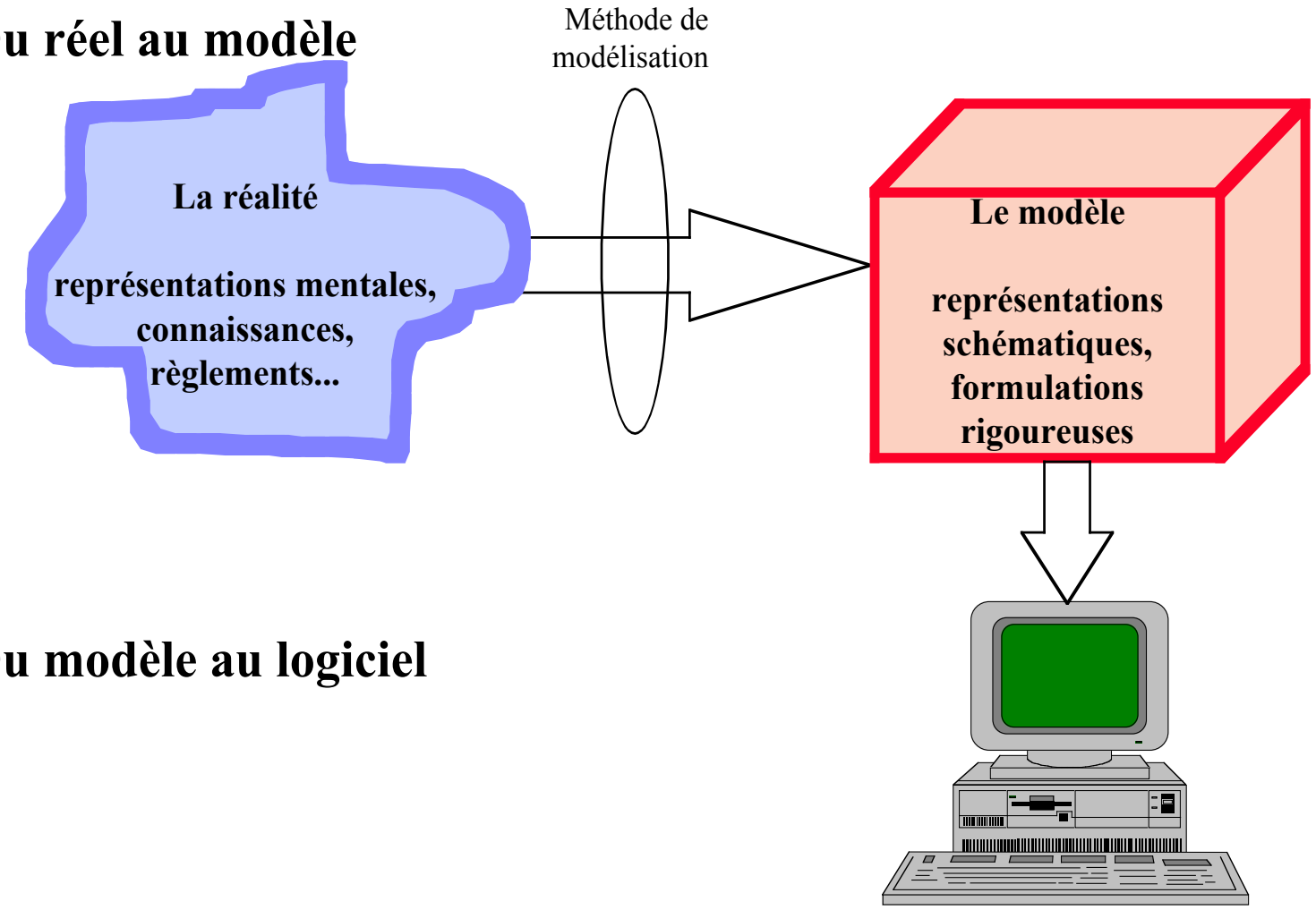
FIGURE 1.5 – Diagramme de cas d'utilisation "Gérer dossier médical"

Le modèle

- Un modèle est une simplification de la réalité.
  - C'est une abstraction souvent sémantiquement complète d'un système.
  - Un modèle représente un point de vue à un certain niveau de détail
- un modèle peut aussi modéliser l'environnement du système (quels sont les utilisateurs et quelles sont les interfaces fournies par le système)
- Le modèle
  - Nous aide à visualiser le système comme nous voulons le voir (la lune est belle pour certains, elle fait un centième de la terre pour d'autres)
  - Nous permet de spécifier la structure ou le comportement du système
  - Nous donne les composants nous permettant de construire le système
  - Nous guide dans le processus de prise de décisions.

Qu'est-ce qu'un modèle ?

- **Du réel au modèle**



- **Du modèle au logiciel**

But de la modélisation ?

- Une raison fondamentale pour laquelle on modélise est de permettre de mieux comprendre le système que nous voulons développer.
- modéliser un système avant sa réalisation. Ceci permet de
  - comprendre le fonctionnement du système
  - de maîtriser sa complexité
  - d'assurer sa cohérence
  - pouvoir communiquer au sein de l'équipe de réalisation
- Dans le domaine de l'ingénierie du logiciel, le modèle
  - permet de bien répartir les taches et d'automatiser certaines d'entres elles
  - est un facteur de réduction de coût de de délais
  - d'assurer un bon niveau de qualité et une maintenance efficace
- Le modèle n'est rarement un produit fini, c'est un outil de travail. Il permet
  - d'organiser les tâches de réalisation
  - de contrôler l'avancement de la réalisation
  - De servir de support de communication entre l'analyste, le client et les utilisateurs

Propriétés des modèles

- Le choix du modèle a une influence capitale sur le type de problème à résoudre et sur les solutions à obtenir. Il faut donc bien choisir le modèle
- Chaque modèle doit être décrit à différents niveaux d'abstraction
- Les meilleurs modèles doivent avoir un comportement proche de la réalité
- Un seul modèle est souvent insuffisant. Les systèmes non-triviaux sont mieux modéliser par un ensemble de modèles indépendants.

Exemple de modèles

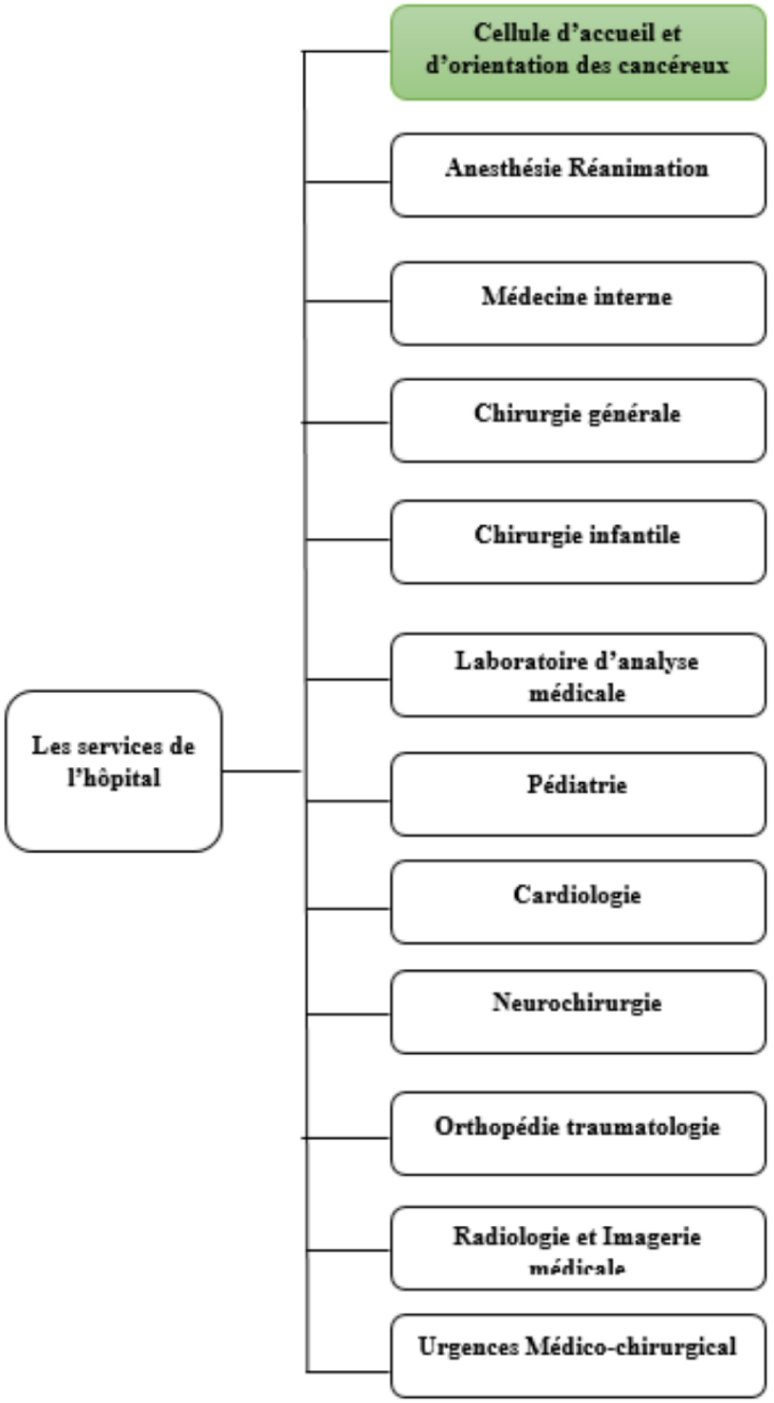


FIGURE 1.2 – Organigramme de l'hôpital

Modélisation Dans le domaine du logiciel

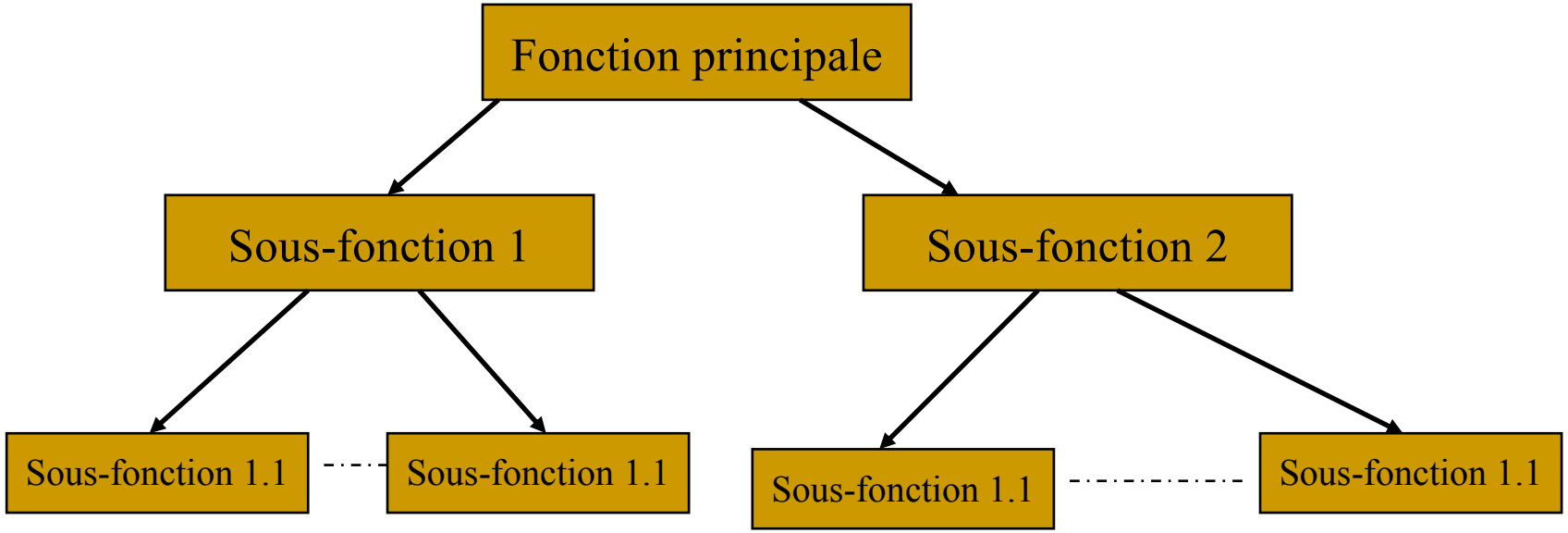
Dans le domaine du logiciel il existe deux principales approches de modélisation :

- Approche fonctionnelle. Approche traditionnelle utilisant des procédures et des fonctions. Les grand programmes sont ainsi décomposés. Cette approche est très dépendante du matériel avec toutes les conséquences que ceci implique

- Approche orientée objets. L'élément principal de tout logiciel est l'objet ou la classe.
  - L'objet est une instance et la classe. La classe est une description d'un ensemble d'objets ayant les mêmes propriétés.
  - Chaque objet possède une identité, un état, des données et un comportement.

Modélisation par décomposition fonctionnelle

La fonction donne la forme du système



Modélisation fonctionnelle (approche descendante)

Décomposer la fonction globale jusqu'à obtenir des fonctions simples
À appréhender et donc à programmer.

Remarque : ici un module sera composé des fonctions d'une même fonctionnalité. Exemple : module d'impression constitué des fonctions d'impression des acteurs, des salles de cinéma, des producteurs, etc.

Avantages

- Organisée, réfléchie, logique
- Ordonnée, réduit la complexité

Problèmes

- comment assurer l'évolution du logiciel
- Comment réutiliser les parties déjà développées
- Comment structurer les données

Modélisation orientée objets

La conception objet est la méthode qui conduit à des architectures logicielles fondées sur les objets que tout système manipule, plutôt que sur la fonction qu'il est censé réaliser.

On part des objets du domaine (briques de base) et on remonte vers le système global

Remarque : ici un module est constitué d'un objet de de toutes les fonctions associées, il est dit classe. Exemple : le classe acteur est constituée des attributs liés à l'acteur (nom, age, taille, etc.) et de toutes les fonctions associées (modifierAdresse(), ajouterRoleDansFilm(), etc.)

Avantages :

- Simple : peu de concepts de base
- Raisonnement par abstraction (objets du domaine)

Important :

Ne pas réduire l'approche objet à une décomposition objet. L'approche objet n'est pas seulement descendante

Modélisation Objets

- L'approche objet est moins intuitive que l'approche fonctionnelle
- Rien dans les concepts de base objets ne dicte comment modéliser la structure objet d'un sytème de manière pertinente
- L'application des concepts objets nécessite une grande rigueur pour éviter les ambiguïtés , les incompréhensions, etc.
- La pensée objet doit être dissociée de la programmation objet. Car le langage objet ne présente qu'une manière particulière d'implémenter certains concepts et ne valident en rien la conception.
- Connaître des langages comme Java ou C++ n'est pas une fin en soi, il faut savoir s'en servir en utilisant un moyen de modélisation, d'analyse et de conception adéquat.
- Sans les langages de haut niveau, il est difficile de comprarer plusieurs solutions de découpe objet.

Pour une utilisation efficace de l'approche objet, il faut donc

1. Un langage permettant de représenter les concepts abstraits, de limiter les ambiguités, de faciliter l'analyse
2. Une démarche d'analyse et de concetion objet

   ne pas effectuer une analyse fonctionnelle et se contenter d'une implémentation objet

   définir les différents modèles (ou vues) permettant de décrire tous les aspects du système

Langages de modélisation objet

Il est nécessaire de disposer d'un langage qui définit :

- la sémantique des concepts
- une notation pour la représentation de concepts
- des règles d'utilisation des concepts et de construction

L'industrie du logiciel dispose de nombreux langages de modélisation

- adaptés aux systèmes procéduraux (MERISE)
- adaptés aux systèmes temps réel (ROOM,SADT)
- adaptés aux systèmes à objets(OMT,Booch, UML)

Pour modéliser un système complexe, il est nécessaire

- de disposer d'outils spécialisés pour la construction de modèles
- de disposer de la description de processus de réalisation

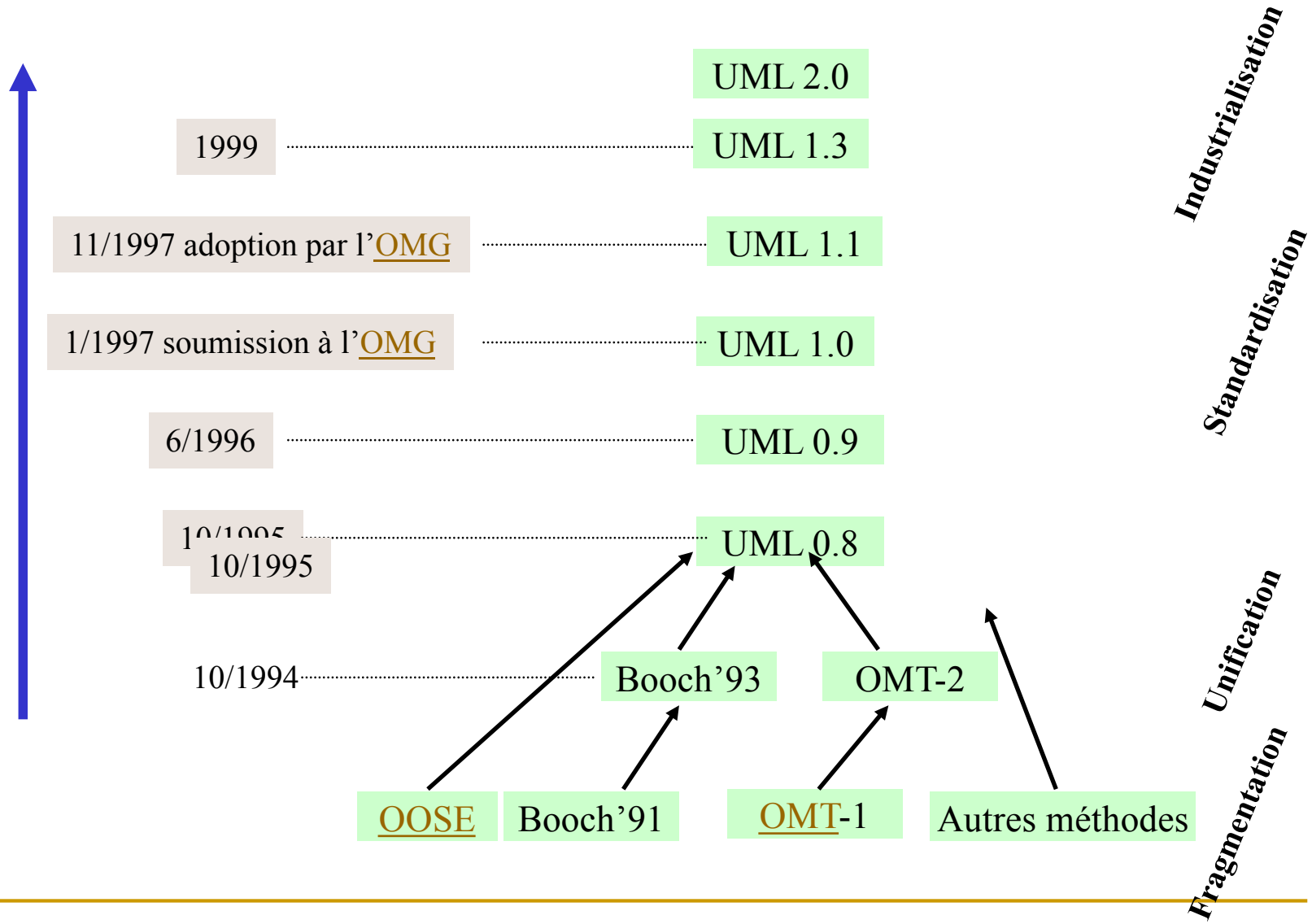
Langages de modélisation et de programmation objet

- Les langages OO sont apparus à la fin des années 60.
  - SIMULA(1967), Smalltalk(1972), C++(1985), JAVA (1995)
- Entre 89 et 94, le nombre de LOO est passé de moins de 10 à une cinquantaine
  - Choisir un langage adéquat relève d'un parcours de combattant ;
  - Des méthodes de genre nouveau sont apparues (Booch, OOSE, …). OMT-2 est la méthode qui était la plus utilisée pour l'analyse de Systèmes d'informations ;
- Au milieu des années 90, les auteurs de Booch, OOSE et OMT ont décidé de créer un langage unifié pour
  - Mettre au point un langage de modélisation utilisable par l'homme et la machine
  - pour modéliser un système des concepts à l'exécutable, e utilisant les techniques orientée objet
  - réduire la complexité de la modélisation
- Officiellement UML est née en 1994

Unification des méthodes de modélisation objets (source H. Panetto)

Origines	objet
Booch	Catégories et sous systèmes
Embley	Classes singletons et objets composites
fusion	Opération, numérotation des messages
D. Harel	Automates (statecharts)
Jacobson	Cas d'utilisation
Mayer	Pré et post conditions
J. Odell	Dynamique, éclairage sur les événements, classification
OMT	associations
R. Wirfs-Brock	Responsabilités
E.Gamma	notes,
Shlaer-Mellor	Cycle de vie des objets

Unification des méthodes de modélisation objet



Exemple de modèles et principe « diviser pour régner »