



Modélisation

Introduction générale

1

Ouassila Labbani-Narsis
ouassila.labbani@u-bourgogne.fr

Avant de commencer ...

- Supports de cours disponibles sur



<https://plubel-prod.u-bourgogne.fr/>

- Participer aux cours



- Pas de bavardage



- Contact : ouassila.narsis@u-bourgogne.fr

Plan du cours

14h CM, 24h TD, 08h TP

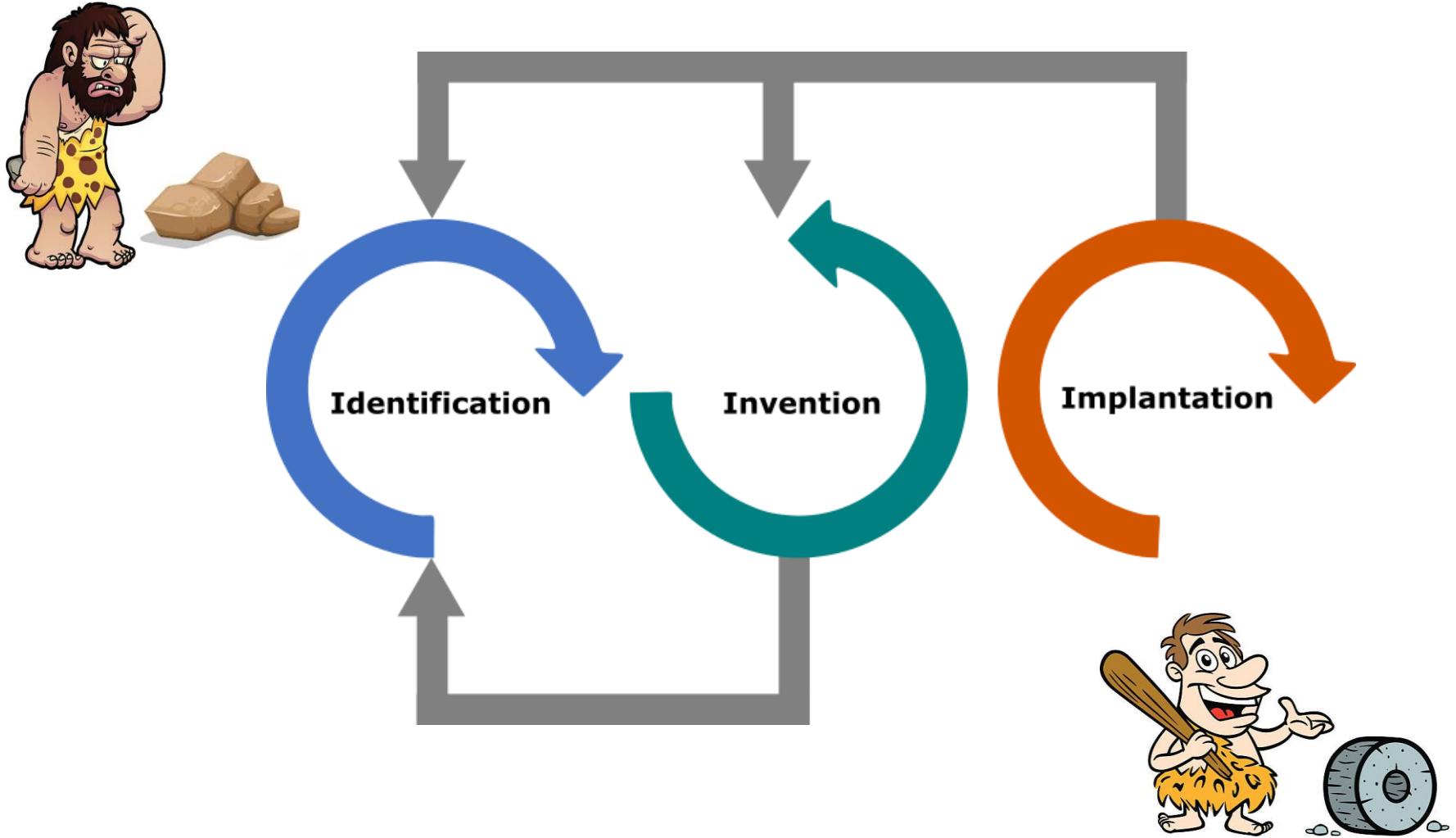
- Introduction générale à la modélisation
- Modélisation **UML** (Unified Modeling Language)
- Langage **OCL** (Object Constraints Language)



Modélisation ??



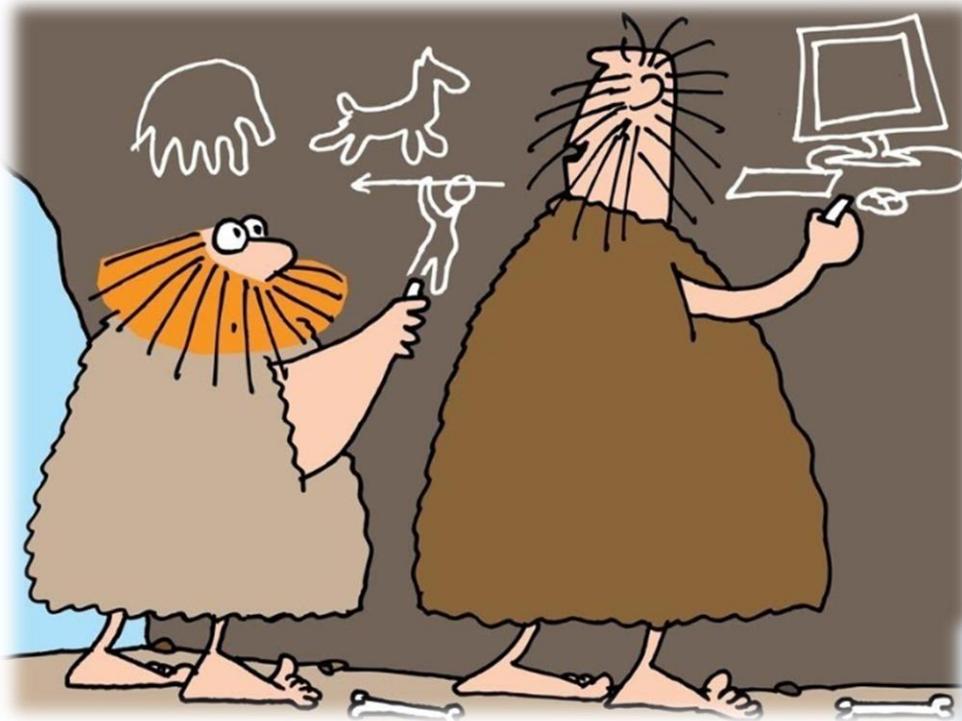
Besoin – Idée – Invention



Besoin – Idée – Invention



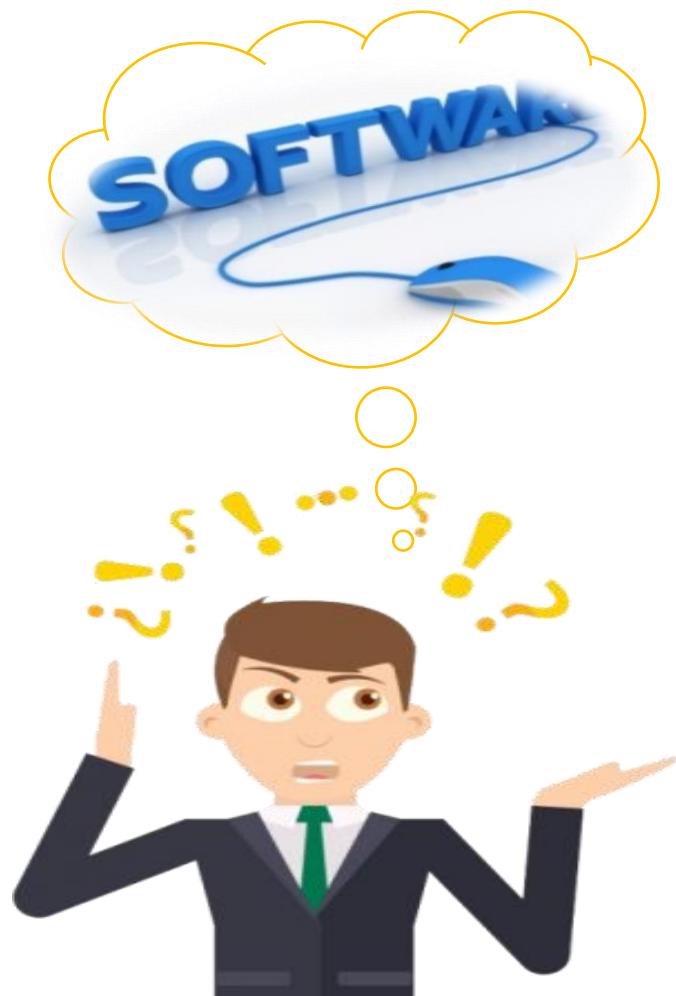
Modélisation Numérique



Développement logiciel



Qu'est ce qu'un logiciel ?



Qu'est ce qu'un logiciel?

 **logiciel**

nom masculin



Ensemble des programmes, procédés et règles, et éventuellement de la documentation, relatifs au fonctionnement d'un ensemble de traitement de données. (Par opposition au matériel.)

« *Ensemble des programmes, procédés et règles, et éventuellement de la documentation, relatifs au fonctionnement d'un ensemble de traitement de données* »

(J. O., Vocab. de l'informat., 17 janv. 1982)

Caractéristiques d'un logiciel

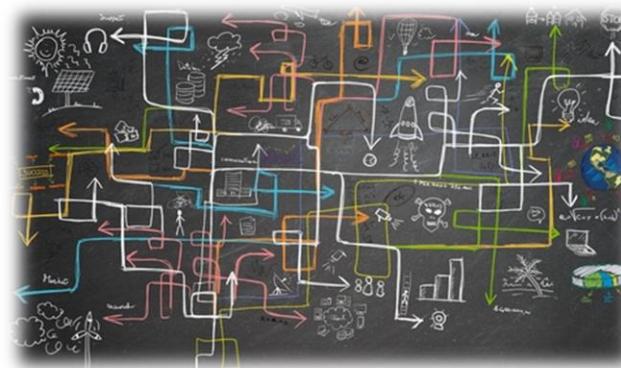
- Un logiciel est fait pour des **utilisateurs**

- Dialogue métier, IHM, production des documents, guides, ...



- Il est **complexe** et de **très grande taille**

- Gérer les interactions,
 - Nombre homme-mois,
 - Nombre d'instructions



Caractéristiques d'un logiciel

- Il fait intervenir **plusieurs participants**

- Travail en équipe(s),
- Organisation, planification, ...



- Il est **long** et **coûteux** à développer

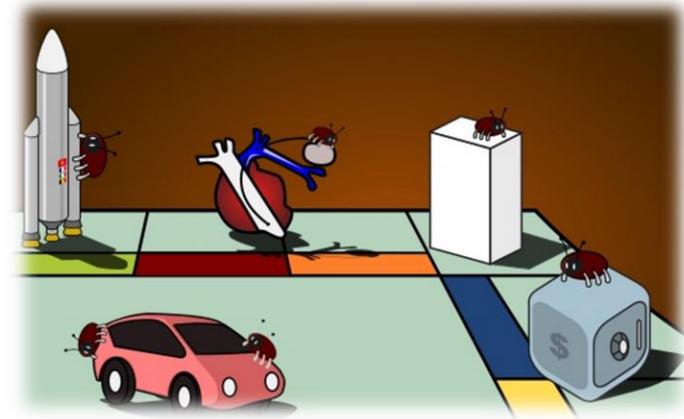
- Risques nombreux et importants : délais, coût, ...



Caractéristiques d'un logiciel

Le logiciel est immatériel et invisible

- La qualité n'est pas vraiment apparente
- On ne peut l'observer qu'en l'utilisant



Développement difficile à automatiser



- Beaucoup de main-d'œuvre nécessaire ...
- Difficile d'estimer l'effort de développement

Caractéristiques d'un logiciel

Le logiciel ne s'use pas, mais il **vieillit**

- Détérioration suite aux changements :
 - Duplication de code
 - Introduction d'erreurs
- Mal conçu au départ :
 - Inflexibilité
 - Manque de modularité
 - Documentation insuffisante
- Evolution du matériel



Principale problématique

Le logiciel n'est **pas fiable!**

- Tout système comporte des bugs...
- Délais et exigences non satisfaits ...



Besoin de **méthodes**, d'**outils** et de **principes** pour le développement de logiciel

→ Naissance d'une nouvelle discipline : le **Génie Logiciel**

Génie Logiciel - Historique

La notion de **génie logiciel** (*software engineering*), est due à l'informaticienne et mathématicienne **Margaret Hamilton**, la conceptrice du système embarqué du programme Apollo¹



- Le terme *software engineering* a été mentionnée pour la première fois en 1967 et repris l'année suivante à une conférence concernant la *crise du logiciel*

NATO Software Engineering Conference 1968

1. Le **programme Apollo** est le programme spatial de la [NASA](#) mené durant la période [1961 – 1975](#) qui a permis aux Etats-Unis d'envoyer pour la première fois des hommes sur la lune.

Génie Logiciel - Définition

Le **génie logiciel** est l'ensemble des activités de conception et de mise en œuvre des produits et des procédures tendant à rationaliser la production du logiciel et son suivi

Journal officiel du 19 février 1984

- Activité par laquelle le code source d'un logiciel est spécifié puis **produit**
- Techniques de fabrication assurant :
 - respect des exigences
 - respect de la qualité
 - respect des délais et des coûts



Génie Logiciel - Principe

Le génie logiciel est un processus visant la **résolution** de problèmes posés par un client

- **Identifier et comprendre** le problème à résoudre
- Proposer une solution **utile** résolvant un problème **concret**



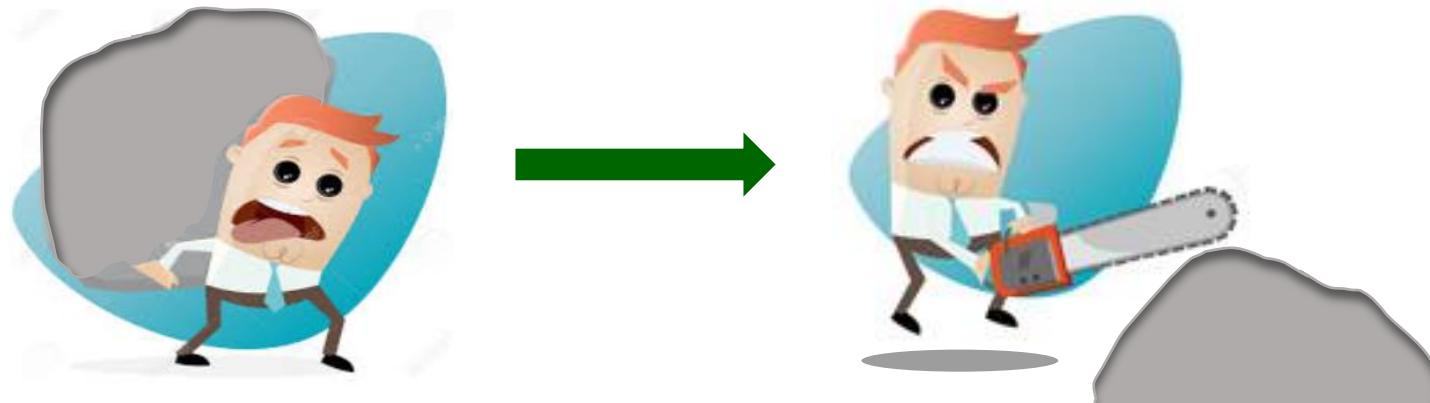
Développement **systématique** et/ou **évolution**

- Utilisation des techniques **maîtrisées, organisées, et rigoureuses**
- Bonnes pratiques **standardisées** (IEEE, ISO, ...)
- La plupart des projets consistent en une **évolution**

Génie Logiciel – Cycle de vie

Comme pour tout **produit complexe** :

- Il faut décomposer la production en « **phases** »
- L'ensemble des phases constitue un « **cycle de vie** »
- Les phases font apparaître des activités clés



Génie Logiciel – Cycle de vie

Phases de création d'un logiciel :

- analyse du besoin,
- élaboration des spécifications,
- conceptualisation du mécanisme interne au logiciel ainsi que des techniques de programmation,
- développement,
- intégration,
- validation et vérification,
- maintenance



Analyse des besoins

Objectif :

- déterminer **ce que doit faire** (et ne pas faire) **le logiciel**
- déterminer les **ressources**, les **contraintes**

Caractéristiques :

- parler **métier** et non informatique
- entretiens, questionnaires, ...
- **observation** de l'existant, **étude** de situations similaires



Résultat : ensemble de documents

- **rôle** du système
- futures **utilisations**
- aspects de l'**environnement**
- (parfois) un **manuel d'utilisation** préliminaire

Spécification

Objectif :

- établir une **1^{ère} description du futur système**
- consigner dans un document qui fait référence



Données :

- résultats de l'analyse des besoins + faisabilité informatique

Résultat : Spécification Technique de Besoin (STB)

- ce que fait le logiciel, mais pas comment

Remarques :

- pas de choix d'implémentation
- (parfois) un manuel de référence préliminaire



Conception

Objectif :

- décrire **comment** le logiciel est construit
- décider **comment** utiliser la techno. pour répondre aux besoins

Travail :

- enrichir la description de **détails d'implémentation**
- pour aboutir à une description **très proche** d'un programme

2 étapes :

- conception **architecturale**
- conception **détaillée**



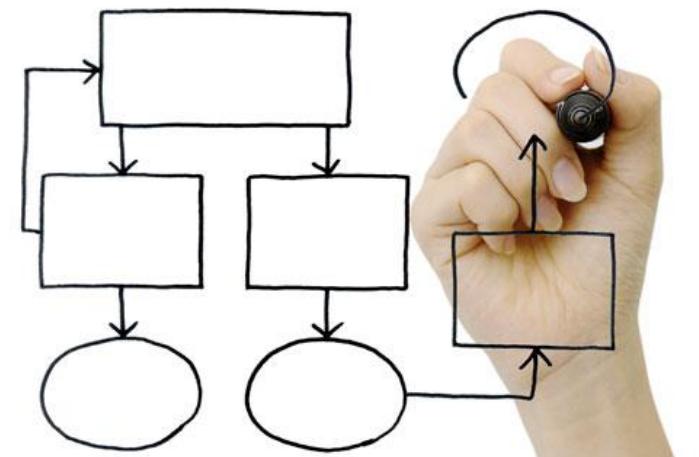
Conception architecturale

Objectif : décomposer le logiciel en composants

- mettre au point l'**architecture** du système
- définir les **sous-systèmes** et leurs **interactions**
- concevoir les **interfaces** entre composants

Résultat :

- **description** de l'architecture globale du logiciel
- **spécification** des divers composants



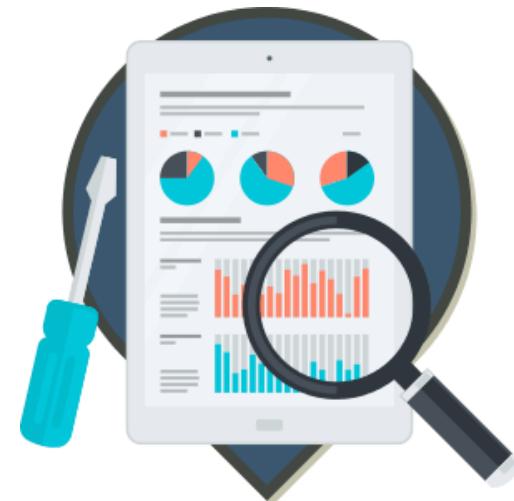
Conception détaillée

Objectif :

→ élaborer les éléments internes de chaque sous-système

Résultat :

→ pour **chaque composant**, description des structures de données, des algorithmes



Remarque :

→ si la **conception est possible**, la **faisabilité** est démontrée
→ sinon, la **spécification est remise en cause**

Programmation

Objectif :

- passer des structures de données et algorithmes à un **ensemble de programmes**

Résultat :

- ensemble de **programmes**
- ensemble de **bibliothèques / modules**
- **documentés** (commentaires)



Remarque :

- activité la mieux **maîtrisée** et **outilée** (parfois automatisée)

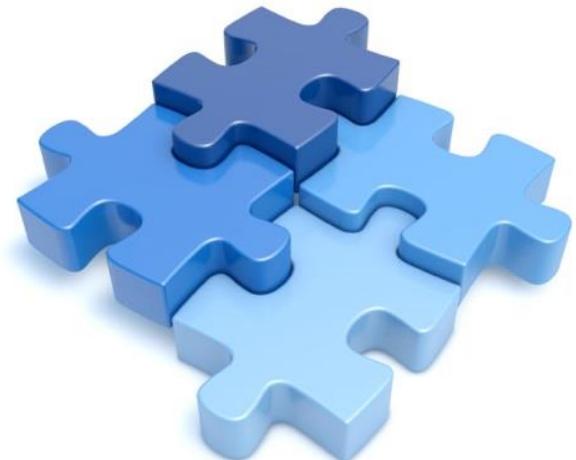
Gestion de configuration et intégration

Gestion de configurations :

- gérer les **composants** logiciels (programmes, bibliothèques, ...)
- maîtriser leur **évolution** et leurs mises à jour

Intégration :

- assembler les composants
- pour obtenir un système **exécutable**



Validation et vérification

Pourquoi ?

- Pour éviter les bugs
- Pour assurer la qualité
- Pour réduire le coût



Validation et vérification

Eviter les bugs

- en phase d'**implantation** (programmation)
- en phase d'**intégration** (conception)
- en phase de **recette** (spécification)
- en phase d'**exploitation**



Validation et vérification

Validation : assurer que le système fonctionne selon les attentes de l'utilisateur

« Are we building the right product ? »

Assurance d'un certain niveau de confiance dans le système

↳ test de logiciels

Vérification : assurer que le système fonctionne correctement

« Are we building the product right ? »

Preuve formelle de propriétés sur un modèle du système

↳ model-checking, preuve

Maintenance



Objectifs :

- **corriger** des défauts
- **améliorer** certaines caractéristiques
- **suivre les évolutions** (besoin, environnement)

Remarques :

- peut remettre en cause les fonctions du système
- peut impliquer un re-développement (*re-ingeneering*)

Répartition des activités

Actuellement, dans un **projet logiciel** bien conduit :

45 % Analyse, Spécification, Conception

20 % Programmation

35 % Intégration, Vérification, Validation



Qualité logiciel

Concevoir des logiciels avec des exigences de qualité



Qualité logiciel

Concevoir des logiciels avec des exigences de qualité



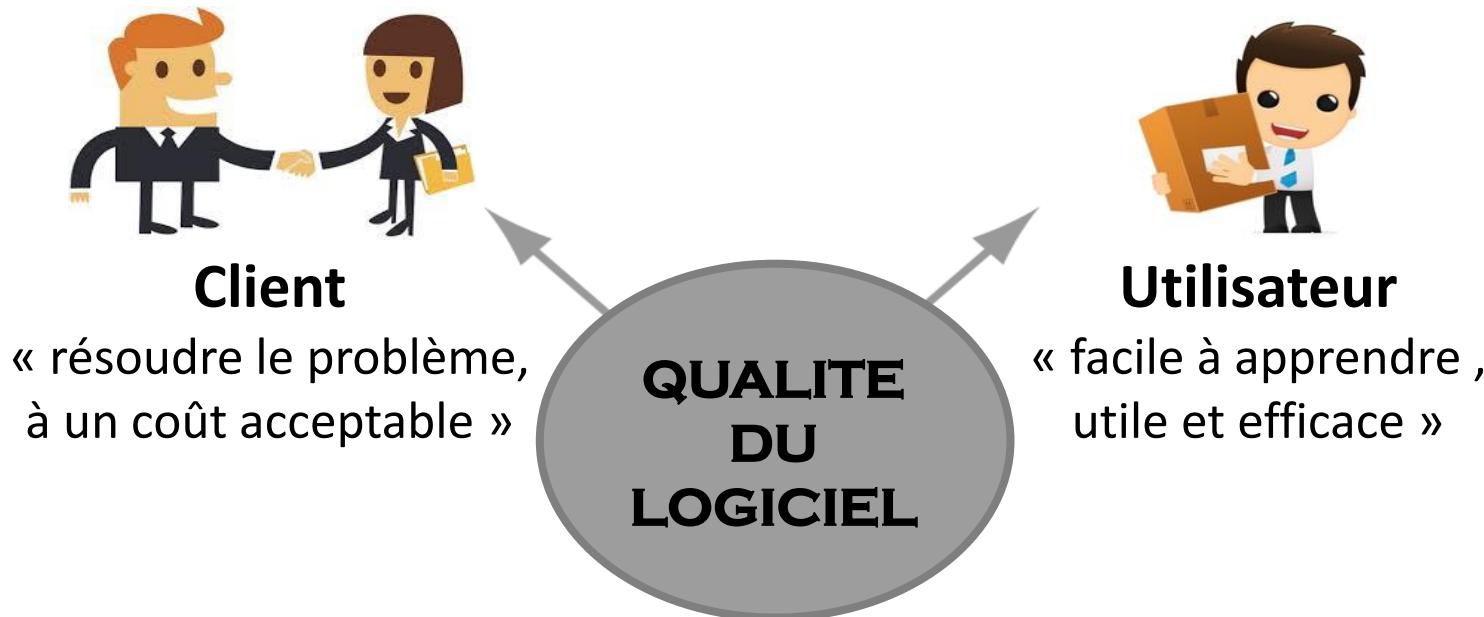
Client

« résoudre le problème,
à un coût acceptable »



Qualité logiciel

Concevoir des logiciels avec des exigences de qualité



Qualité logiciel

Concevoir des logiciels avec des exigences de qualité



Client

« résoudre le problème,
à un coût acceptable »



Utilisateur

« facile à apprendre ,
utile et efficace »



**QUALITE
DU
LOGICIEL**

Développeur

« facile à concevoir,
à maintenir, à réutiliser»

Qualité logiciel

Concevoir des logiciels avec des exigences de qualité



Client

« résoudre le problème,
à un coût acceptable »



Utilisateur

« facile à apprendre ,
utile et efficace »



Développeur

« facile à concevoir,
à maintenir, à réutiliser»



Gestionnaire

« se vend bien, satisfait les clients,
peu coûteux à développer»

Résumé

- La qualité du logiciel est **fondamentale**
- Elle est perçue différemment selon les **points de vue** :
 - **Qualité externe** : client, utilisateurs
 - **Qualité interne** : développeurs, gestionnaires
- Pour l'atteindre, on adopte des **principes**
 - approches standards et modèles de développement
 - approche Orientée Objets



IDM : Ingénierie Dirigée par les Modèles



Le développement logiciel

Comment concevoir efficacement

et intelligemment un système

informatique complexe et

hétérogène ?



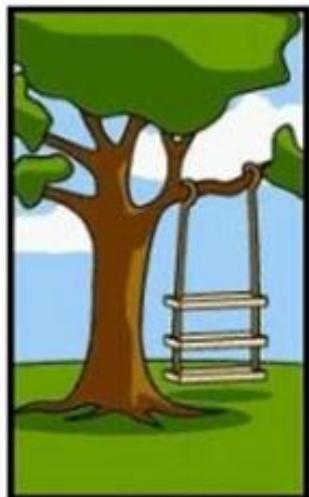
Problématique

Besoin de maîtriser la complexité

- Comprendre les exigences du client et des utilisateurs
- Gérer le développement entre différents membres d'une ou de plusieurs équipes
- Améliorer la production et réduire les coûts et les temps de développement
- Gérer la réutilisation et l'interopérabilité entre les différentes parties du système



Problème de communication !?



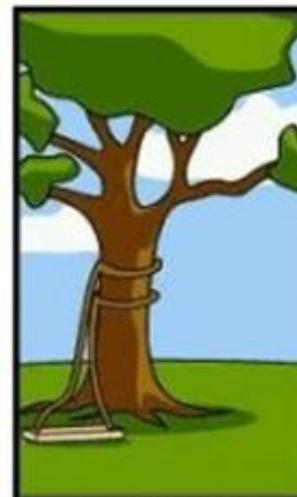
Comment le client
a exprimé son besoin



Comment le chef de
projet l'a compris



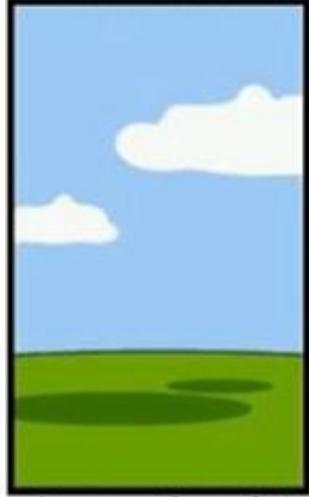
Comment l'ingénieur
l'a conçu



Comment le
programmeur l'a écrit



Comment le responsable
des ventes l'a décrit



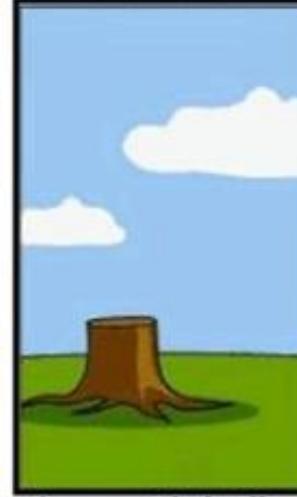
Comment le projet
a été documenté



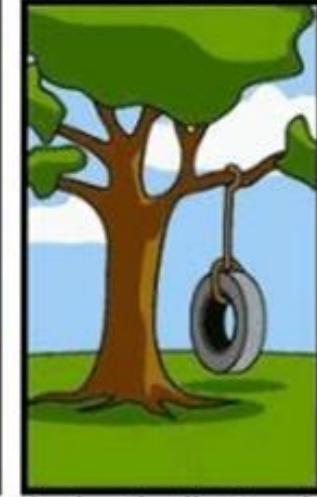
Ce qui a finalement
été installé



Comment le client
a été facturé



Comment la hotline
répond aux demandes



Ce dont le client avait
réellement besoin

Bien comprendre le client...

Comprendre la représentation qu'a le
client de son propre projet



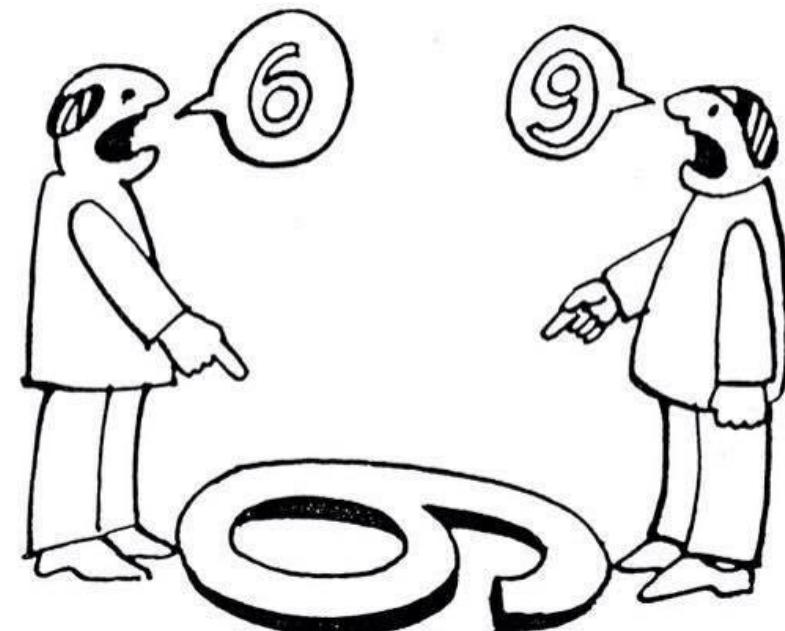
Cette **perception mentale** va se construire avec plusieurs mécanismes:

- **Interprétation:** comprend plusieurs processus cognitifs permettant de filtrer l'information
- **Généralisation:** étendre à une catégorie entière de situations ce qui a été appris dans une, ou plusieurs, situations
- **Distorsion:** modifier nos perceptions ou nos représentations et en construire de nouvelles

Bien comprendre le client...

Conséquence directe : une mauvaise communication qui s'amplifiera si aucune régulation n'est mise en place. C'est ainsi que naissent les mauvais départs dans les projets générant des conflits et conduisant à l'échec du projet

Il faut construire une vision du projet identique à celle du client pour bien maîtriser la complexité



Maîtriser la complexité ...

Comprendre les besoins

Adapter son vocabulaire et se faire sa propre opinion en allant observer le client sur le terrain

- Il faut parler comme le client
- Il faut parler aux spécialistes
- Il faut gérer l'hétérogénéité des langages et des domaines



Répondre à la question : « *sommes-nous en train de construire le bon produit ?* »

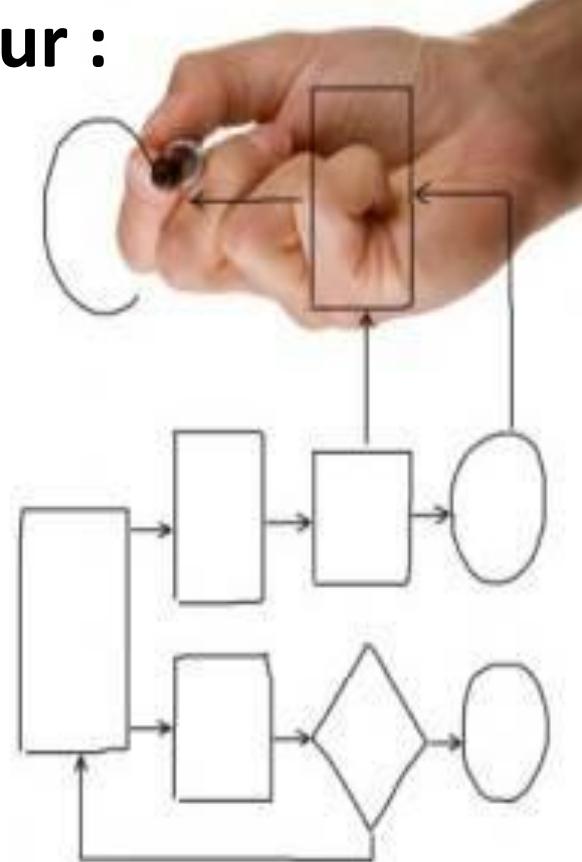
Suivre une méthodologie claire de développement

Besoin de modèles

Modéliser avant de fabriquer...

S'appuyer sur la notion de modèle pour :

- Maîtriser la complexité
- Réutiliser des parties
- Séparer les préoccupations
- Fusionner les préoccupations dans le produit final



Pourquoi modéliser ?

- Grand gap entre le monde du concepteur et celui du client
- Le développement est généralement effectué à plusieurs
- Les différentes parties d'un système sont indépendantes
- Nécessite différents domaines de compétences

Objectifs:

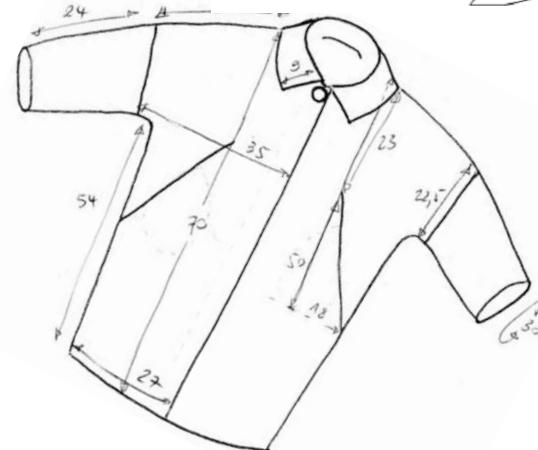
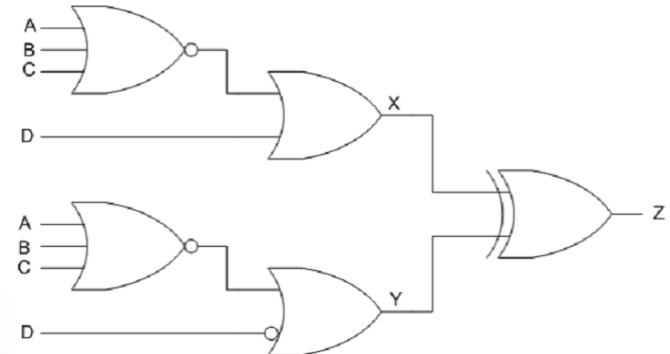
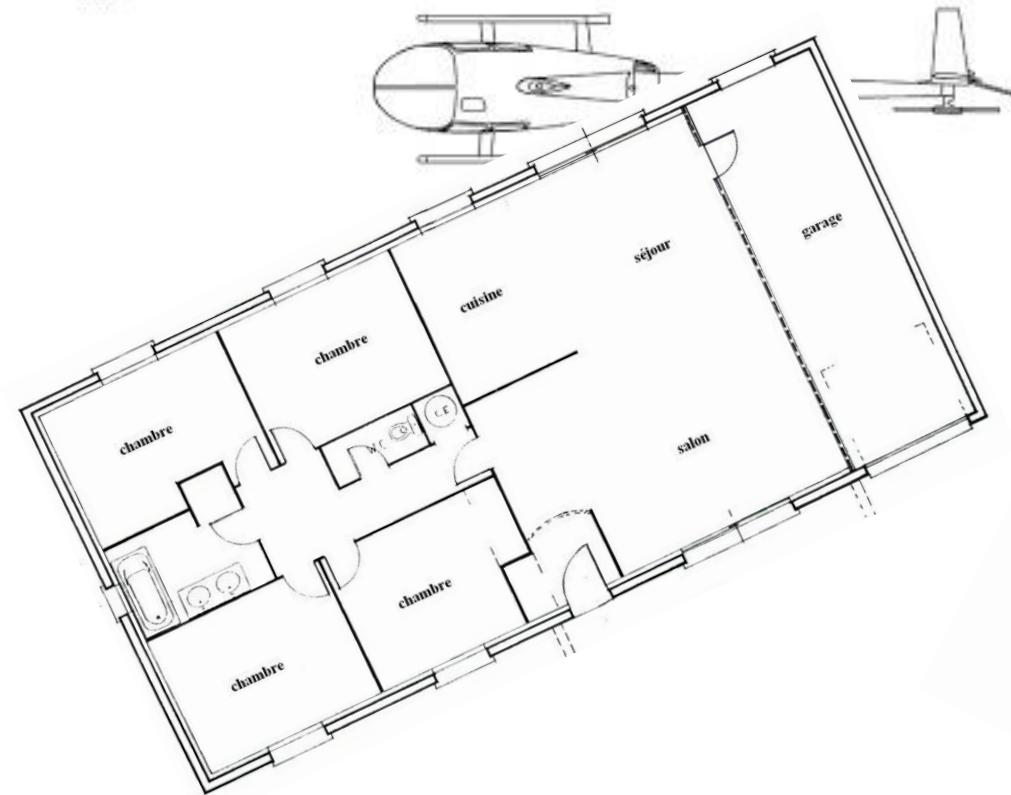
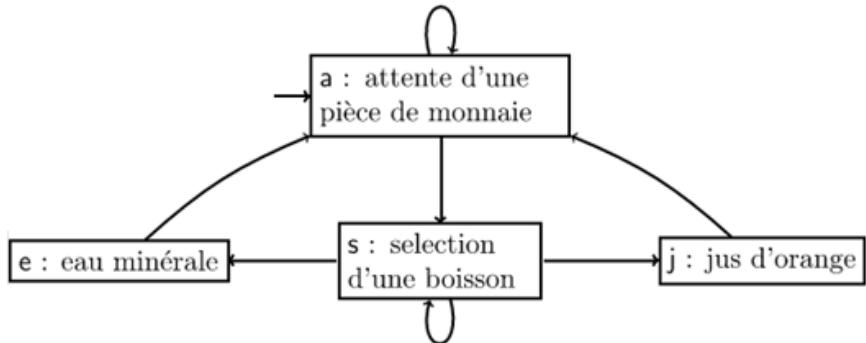
- Visualiser le système comme il est ou comme il devrait être
- Spécifier les structures de données et le comportement du système
- Fournir un guide pour la construction du système
- Documenter les systèmes et les décisions prises



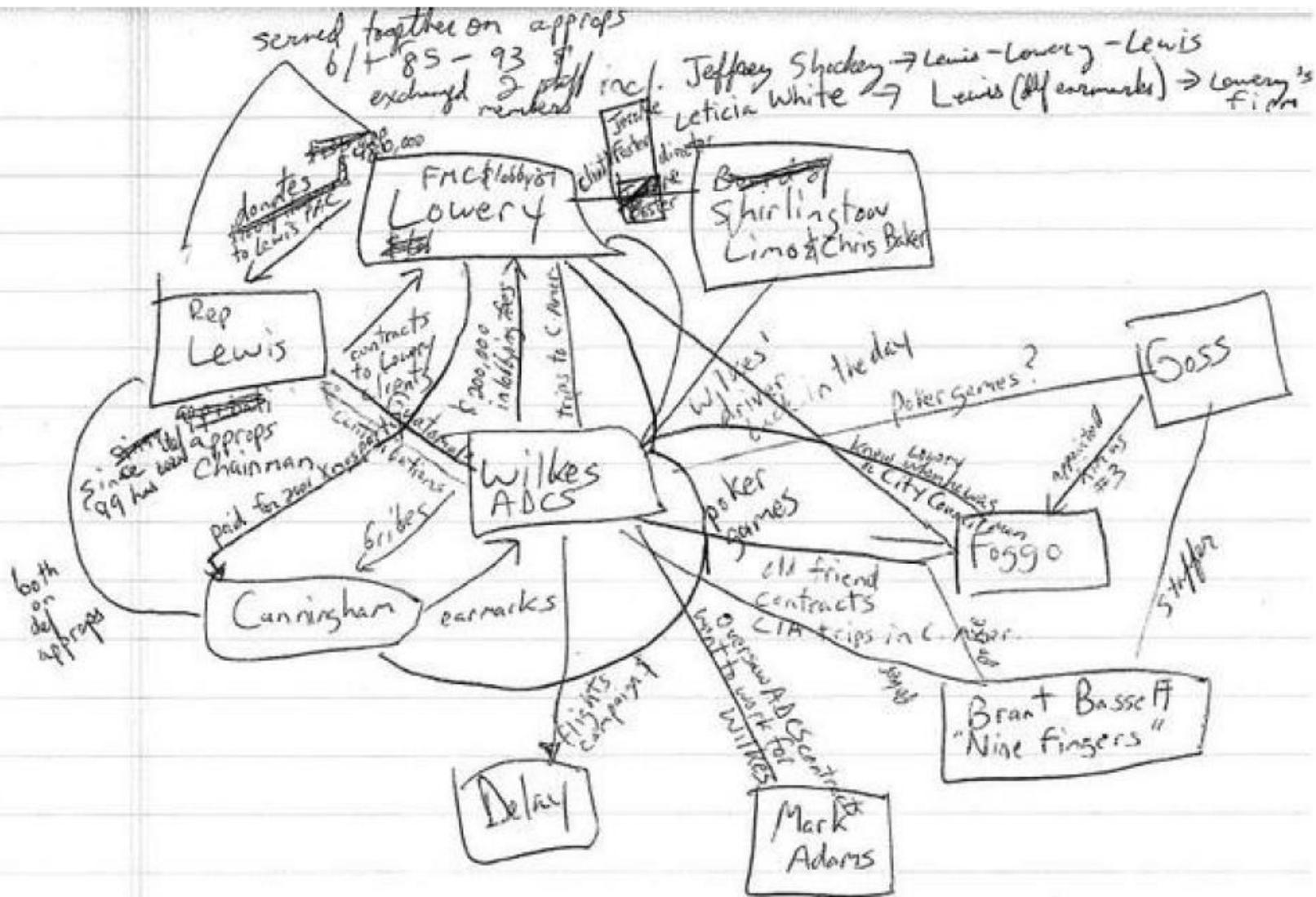
Qu'est ce qu'un modèle?



Qu'est ce qu'un modèle?



Qu'est ce qu'un modèle?



Comment modéliser ?

- La manière de modéliser influence fortement :
 - **la compréhension du problème**
 - **la solution**

Il n'existe pas de modèle universel

- Multiplicité des niveaux d'abstraction
- Multiplicité des points de vue
- Le modèle doit être en prise avec le réel
 - Bien choisir les niveaux d'abstraction
 - Réduire les risques financiers et techniques
 - Produire Mieux et Plus Vite

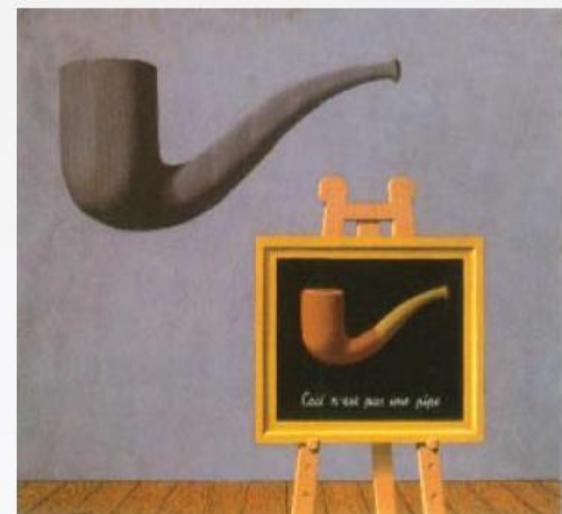


Modèle et réalité



Ceci n'est pas une pipe.

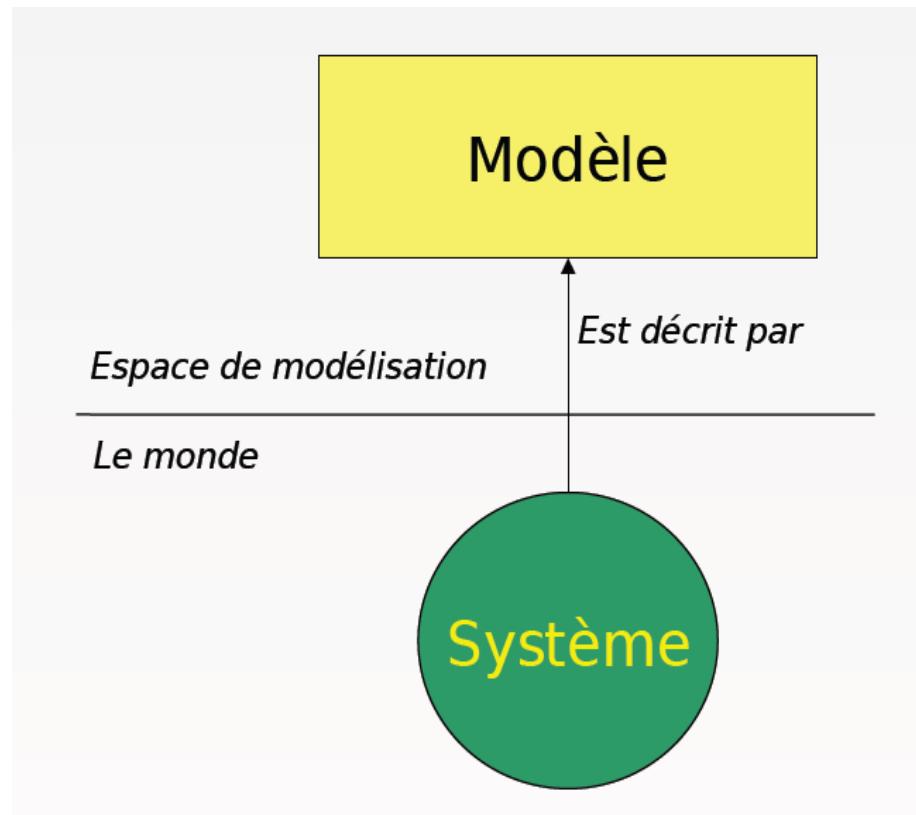
René Magritte,
« La trahison des images »
1928-29



René Magritte,
« Les deux mystères »
1966

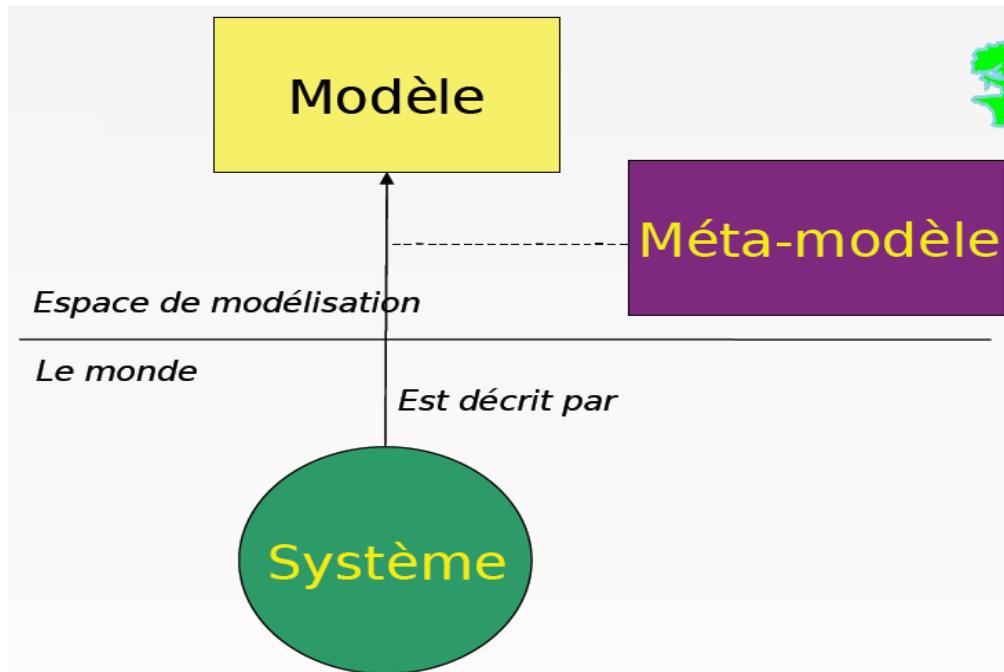
Modèle et système

- Un *modèle* est une représentation simplifiée d'une partie du monde appelée le *système*
- Un modèle doit répondre exactement de la même manière que le système

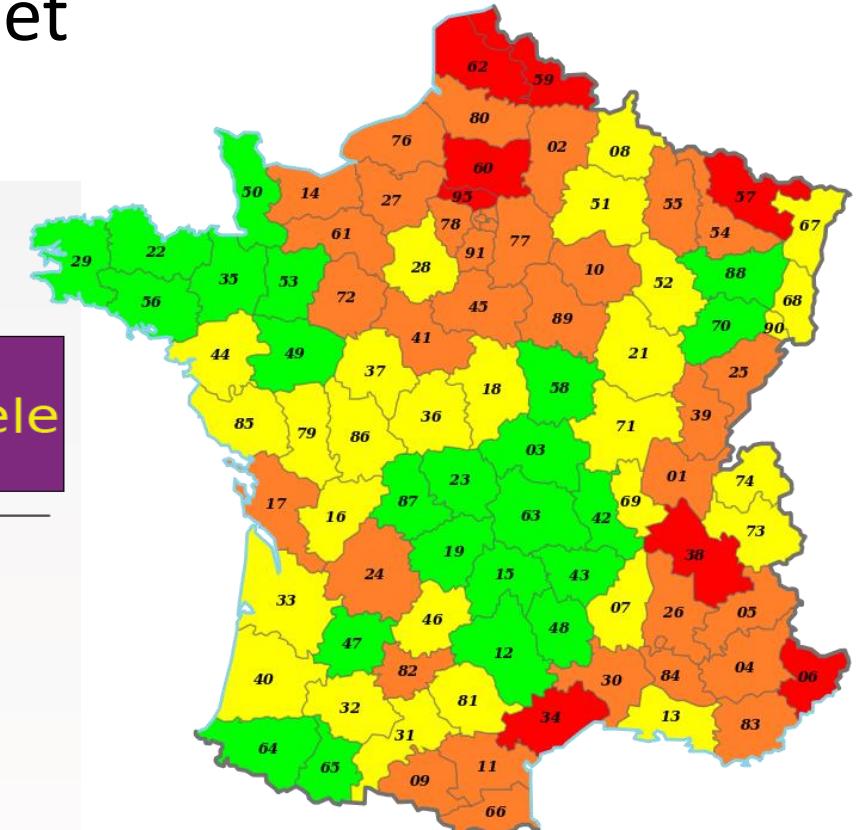


Modèle et méta-modèle

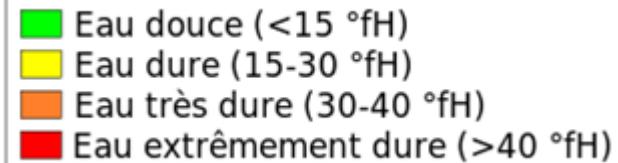
Un méta-modèle définit la relation entre un modèle et un système



Dureté de l'eau en France



- Le méta-modèle est la **légende** de la carte



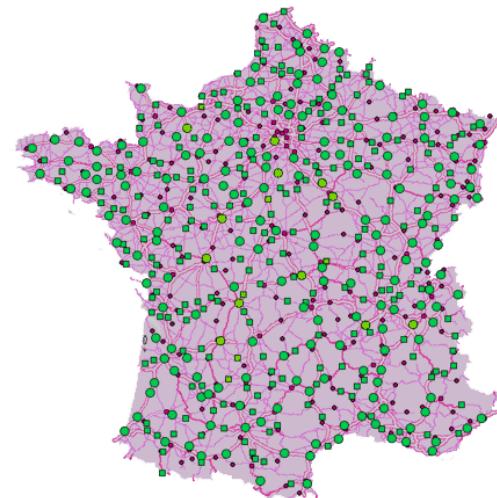
Un modèle, un point de vue



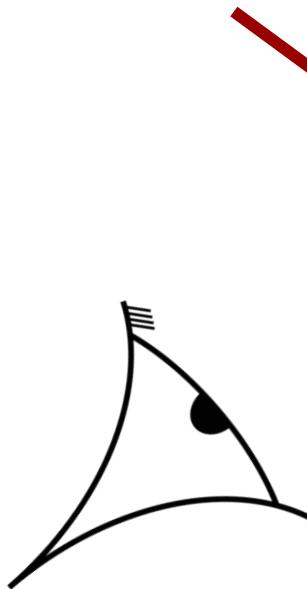
Carte indicatifs téléphoniques



Carte administrative



Carte des radars



Attention !

**Les activités d'abstraction, de modélisation,
de séparation et de fusion existent
depuis toujours**



Mais ...

- les modèles restent parfois implicites
- le processus est manuel
- de nouveaux noms, de nouveaux concepts, . . .

Les méthodes sont nombreuses. Elles sont plus ou moins définies et plus ou moins universelles

Solution : standardisation



Besoin de ...

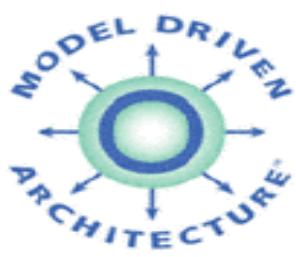
- faire communiquer des outils autour d'un modèle
- explorer/modifier un modèle d'un certain type de manière standard

Standard **OMG** (Object Management Group)

- **MDA** (Model Driven Architecture)
- **MDE** (Model Driven Engineering)



About The Object Management Group™ (OMG™)



The [Object Management Group \(OMG\)](#) is an [open membership](#), not-for-profit consortium that produces and maintains computer industry specifications for [interoperable enterprise applications](#). Our [membership](#) includes virtually every large company in the computer industry, and hundreds of smaller ones. Most of the companies that shape enterprise and Internet computing today are represented on our [Board of Directors](#).

Our flagship specification is the multi-platform [Model Driven Architecture \(MDA\)](#), recently underway but already well known in the industry. It is based on the modeling specifications the [MOF](#), the [UML](#), [XMI](#), and [CWM](#). OMG's own middleware platform is [CORBA](#), which includes the Interface Definition Language [OMG IDL](#), and protocol [IIOP](#). The [Object Management Architecture \(OMA\)](#) defines standard services that will carry over into MDA work shortly. OMG Task Forces standardize [Domain Facilities](#) in industries such as healthcare, manufacturing, telecommunications, and others.

All of our specifications may be [downloaded without charge](#) from our website. [Products implementing OMG specifications](#) are available from hundreds of sources. Our success-stories website documents [hundreds of mission-critical CORBA applications](#) in use today. And, [these companies and organizations](#) have committed to CORBA as their interoperability architecture.

[Any company may join OMG](#) and [participate in our standards-setting process](#). Our [one-company-one-vote policy](#) ensures that every company, large and small, has an effective voice in our process.

[Requests for Proposals](#) (the requirements document that initiates each OMG standard-setting activity) and other key documents are [available for viewing by anyone](#), member or not. Email discussion, meeting attendance, and voting are restricted to our members, except that prospective members are invited to [attend a meeting or two as a guest observer](#) while making their decision to join.

Standards organizations and other consortia [maintain liaison relationships with OMG](#). OMG is an ISO PAS submitter, able to submit our specifications directly into ISO's fast-track adoption process. [OMG IDL is already an ISO standard and ITU-T recommendation](#).

IDM : Ingénierie dirigée par les Modèles

Pourquoi IDM ?

- Toujours en quête de d'interopérabilité, de portabilité, de séparation des aspects fonctionnels et non fonctionnels, ...
- **Idée** : Monter en niveau d'abstraction !



L'initiative MDA de l'OMG vise à promouvoir l'idée de la mise en œuvre de systèmes distribués dirigée par les **modèles**

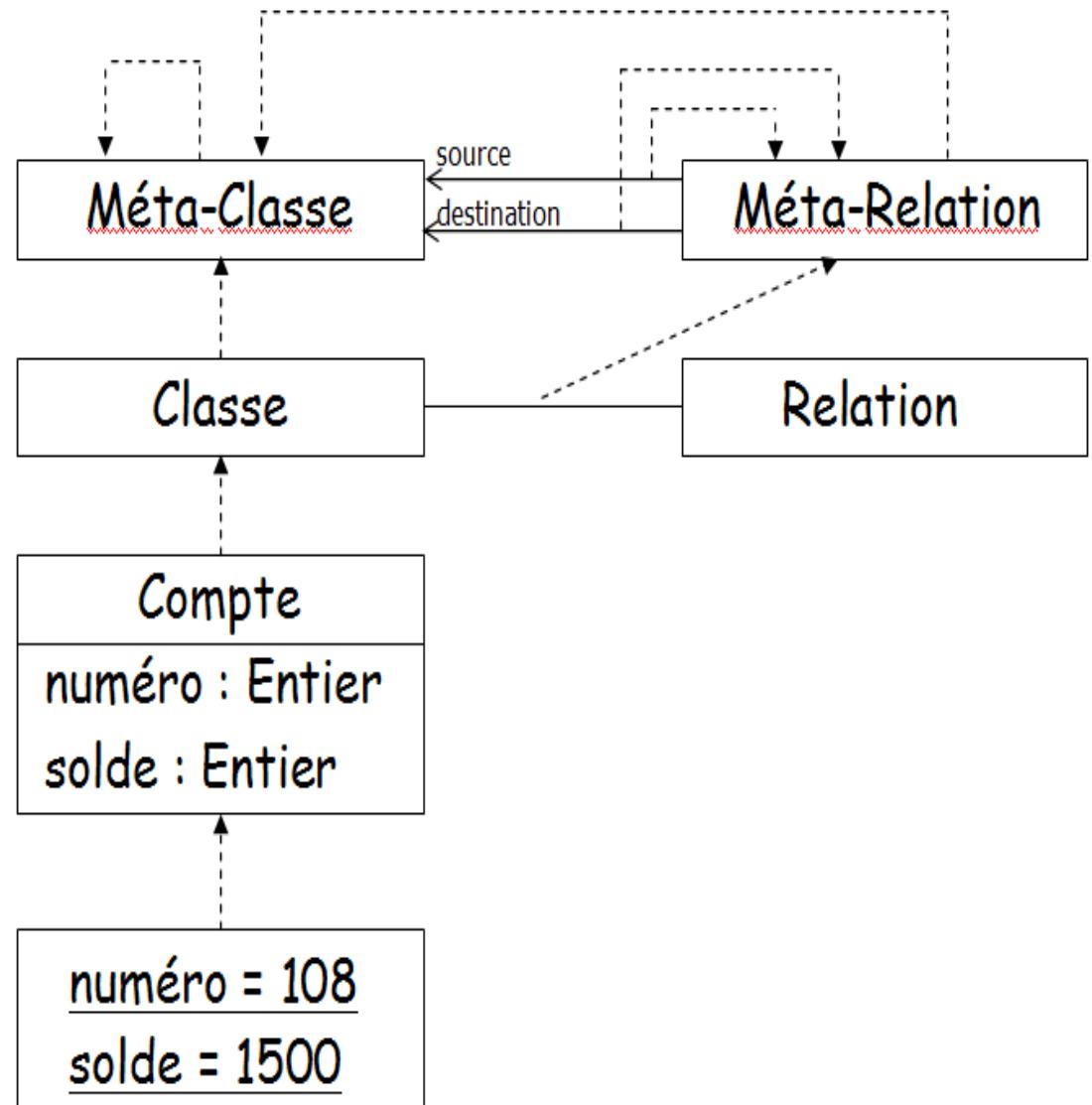
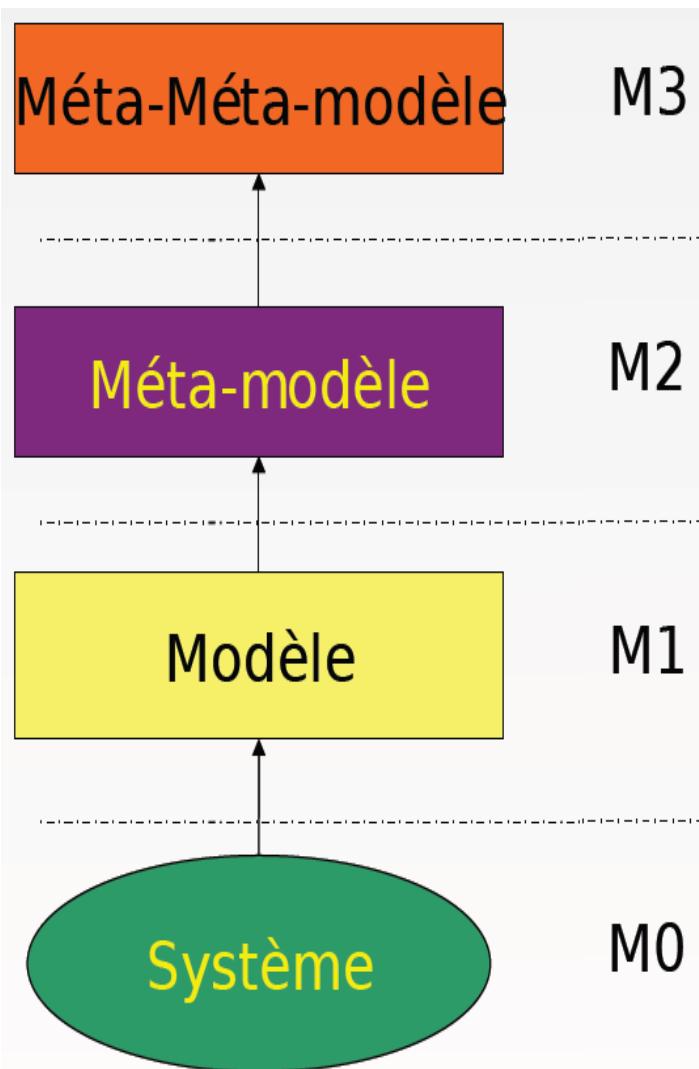
IDM : Ingénierie dirigée par les Modèles

«*Design once, build it on any platform*»

- Abstraire les parties métiers de leur mise en œuvre
- Basé sur des technologies et standards de l'OMG
 - UML, MOF, OCL, ...
- Définition de **méta-modèles**
 - Standardisation d'un domaine d'activité
 - Standardisation des technologies
- Définition d'une **méthodologie**
 - Démarche à suivre pour produire une application, dans une technologie, à partir d'un modèle abstrait



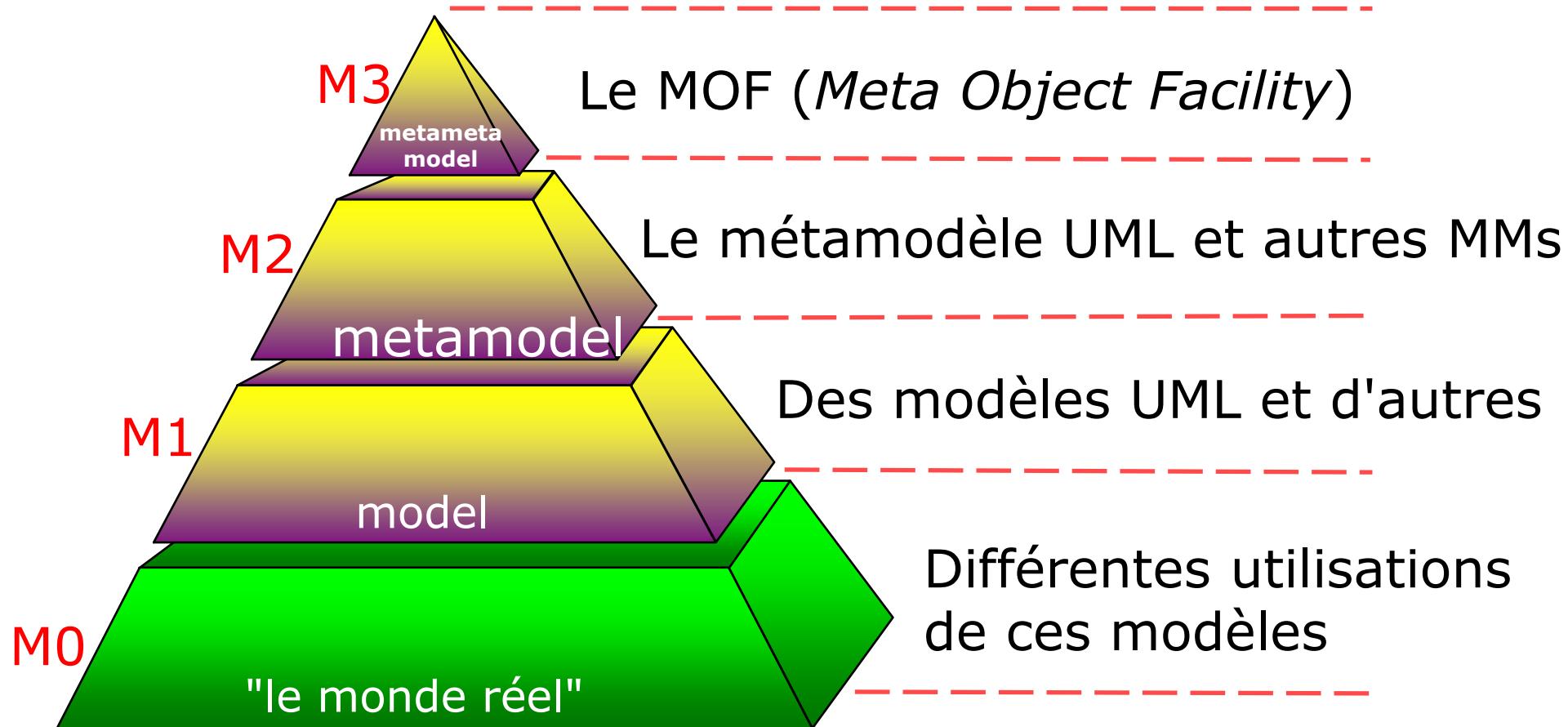
Niveaux de modélisation



Niveaux de modélisation

- **M0 : Instance**, programme Java, données
 - Représentation informatique du monde réel
- **M1: Modèle**, source Java, document XML
 - Descriptions d'éléments de niveau M0
- **M2 :Méta-modèle**, BNF Java, DTD XML
 - Concepts pour définir des éléments de niveau M1
- **M3 : Méta-métamodèle**, EBNF, XML
 - Concepts pour définir des éléments de niveau M2

Architecture à 4 niveaux



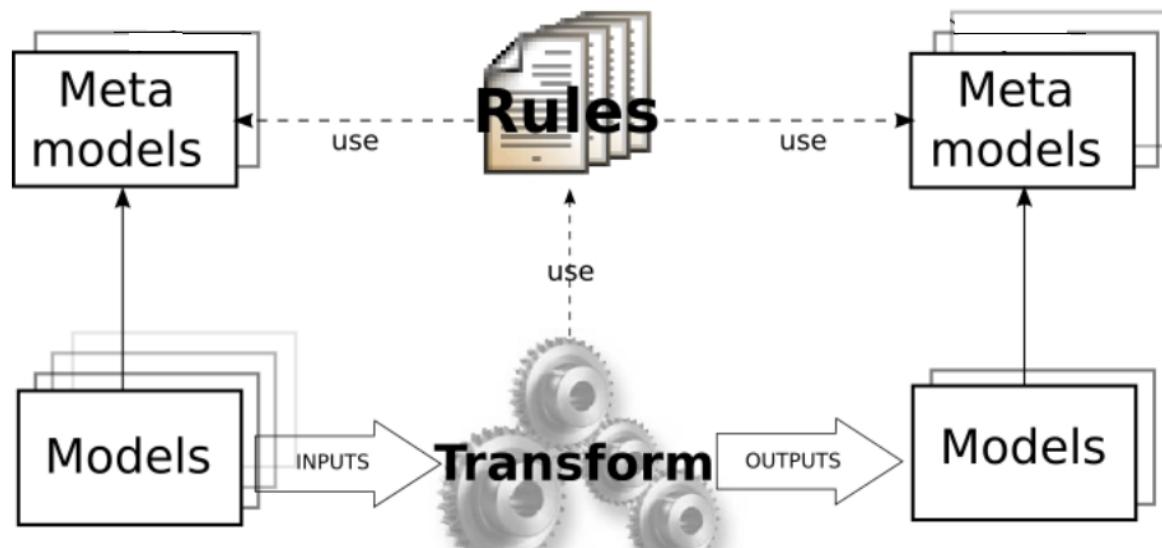
Pile de méta-modélisation (OMG)

- **Un unique méta-métamodèle**
 - Meta Object Facility (MOF)
 - Concepts de méta-modélisation
- **Un ensemble de méta-modèles**
 - Pour différents besoins / domaines
 - Compatibles puisque définis avec le MOF
- **Un grand nombre de modèles**
 - Décrivant un grand nombre de systèmes
 - Définis avec les concepts de leurs uniques méta-modèles

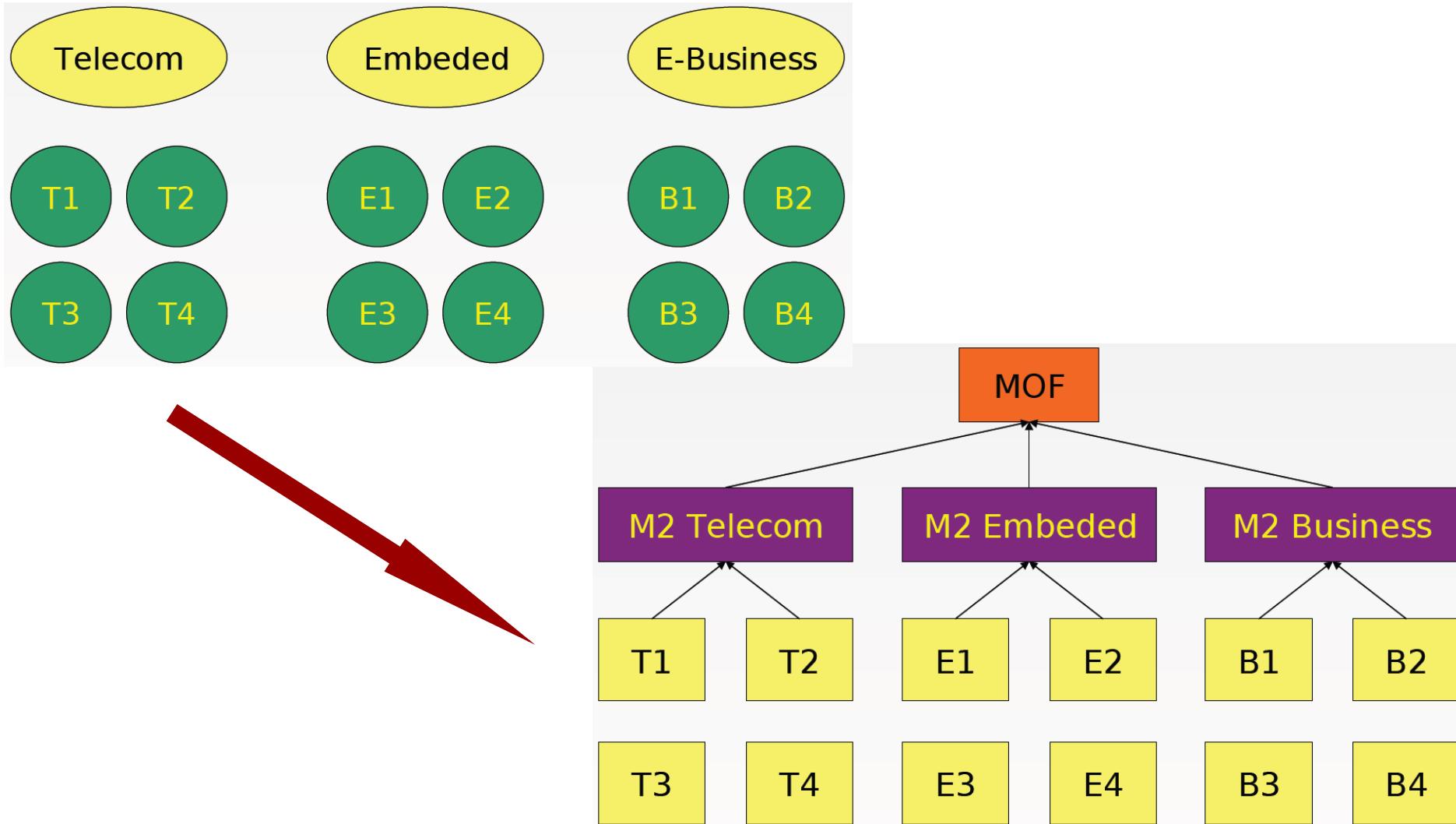


Transformation de modèles

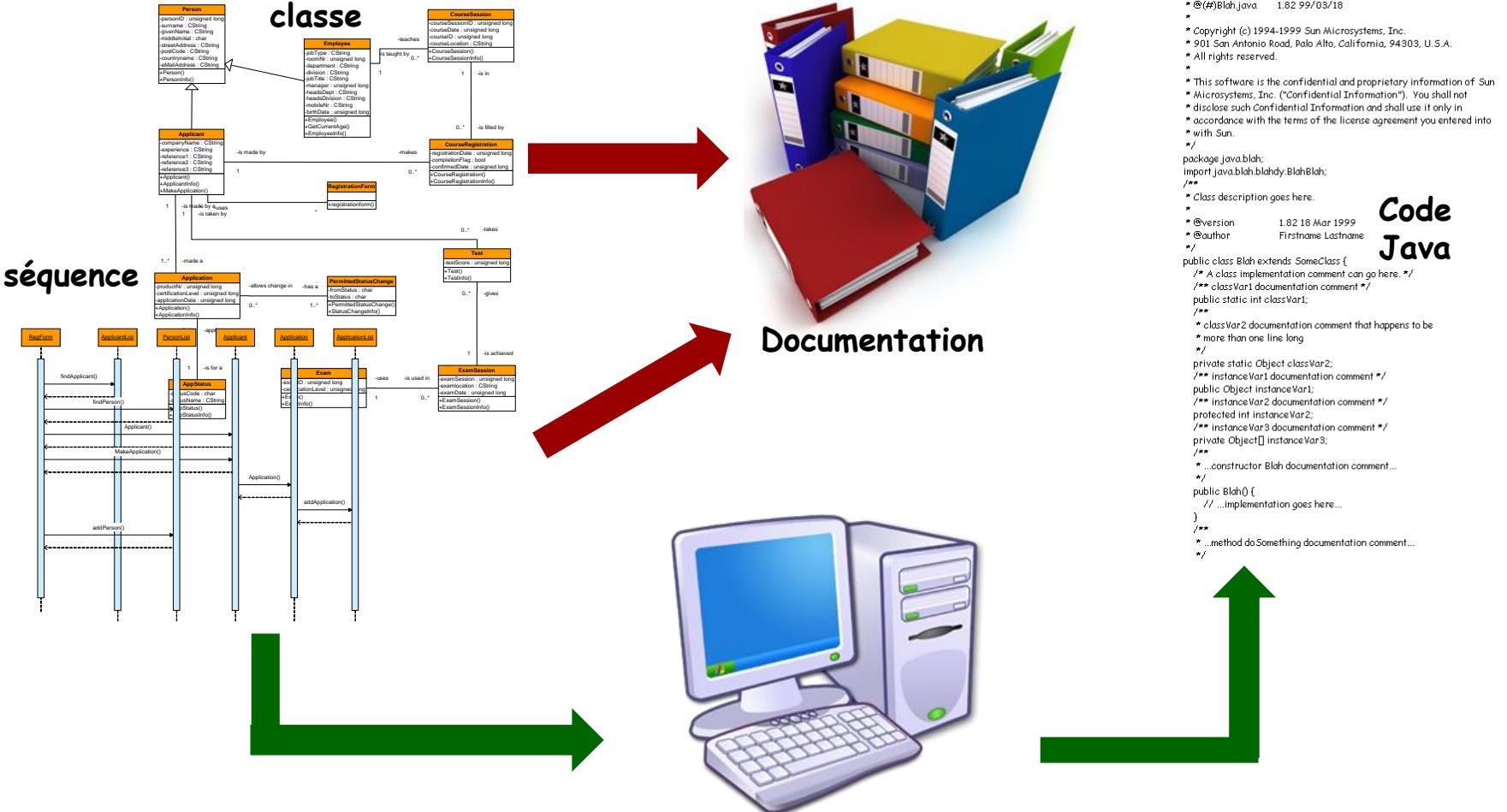
- Permet de passer d'un modèle *source* décrit à un certain niveau d'abstraction à un modèle *destination* décrit éventuellement à un autre niveau d'abstraction
- Ces modèles source et destination sont conformes à leur métamodèle respectif, le passage de l'un à l'autre est décrit par des **règles de transformation**



D'un paysage de systèmes... ...A un paysage de métamodèles



Du contemplatif au productif



Du compréhensible pour l'homme au compréhensible pour les machines

En résumé ...

Des Modèles plutôt que du Code...

- Un modèle est la simplification/abstraction de la réalité
- Le code ne permet pas de simplifier/abstraire la réalité
- Nous construisons des modèles afin de mieux comprendre les systèmes que nous développons
- Nous modélisons des systèmes complexes parce que nous sommes incapables de les comprendre dans leur totalité
- Nous générions automatiquement du code à partir des modèles

