



**UFR SET**  
**Département Informatique**  
**Licence 3 en Informatique (Génie Logiciel)**  
**Introduction au Génie Logiciel**

**M. DIOUF**



# Chapitre 1

## Qu'est ce que le Génie Logiciel ?

Qu'est ce qu'un logiciel ?

Qu'est ce que le génie logiciel ?

Processus de développement

Documentation

# QU'EST CE QU'UN LOGICIEL ?

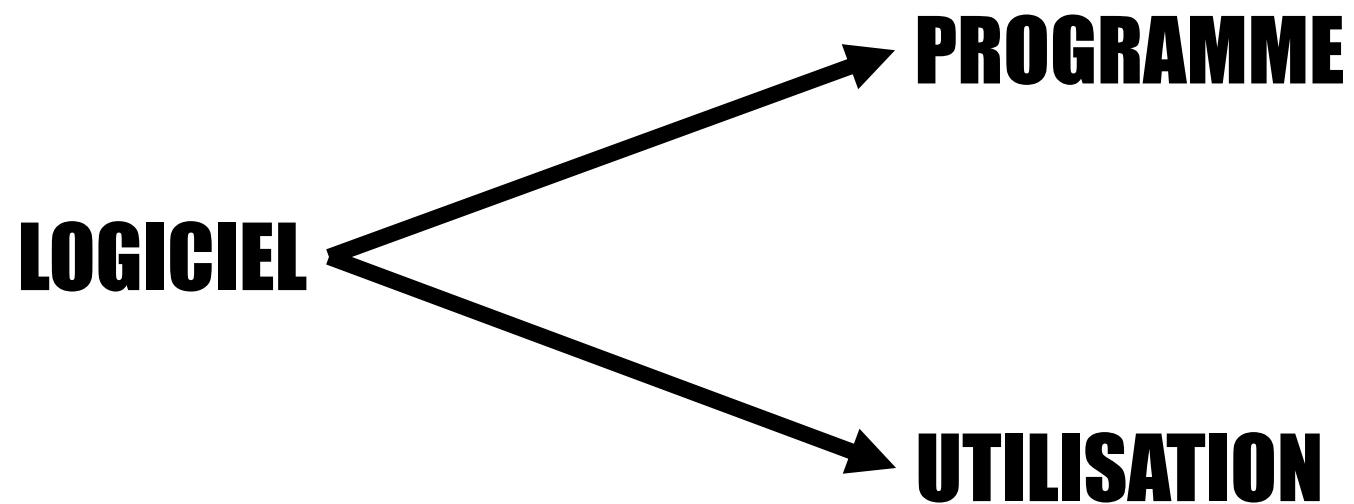
## IDÉES...



# **QU'EST CE QU'UN LOGICIEL ?**

**Ensemble de programmes qui permet à un système informatique d'assurer une tâche ou une fonction en particulier.**

- Programmes
- Données
- Documentation



# ENVIRONNEMENT D'USAGE VS SPECIFICATION



# ENVIRONNEMENT D'USAGE D'UN LOGICIEL

## Environnement : Qui utilise le logiciel ?

- **Utilisateurs :**

- Grand Public (Traitement de Texte, Chat...)
- Spécialistes (Calcul Météorologique, Logiciel d'infographie)
- Développeur (Compilateur, IDE)

- **Autres logiciels :**

- Packages
- Module

- **Matériel :**

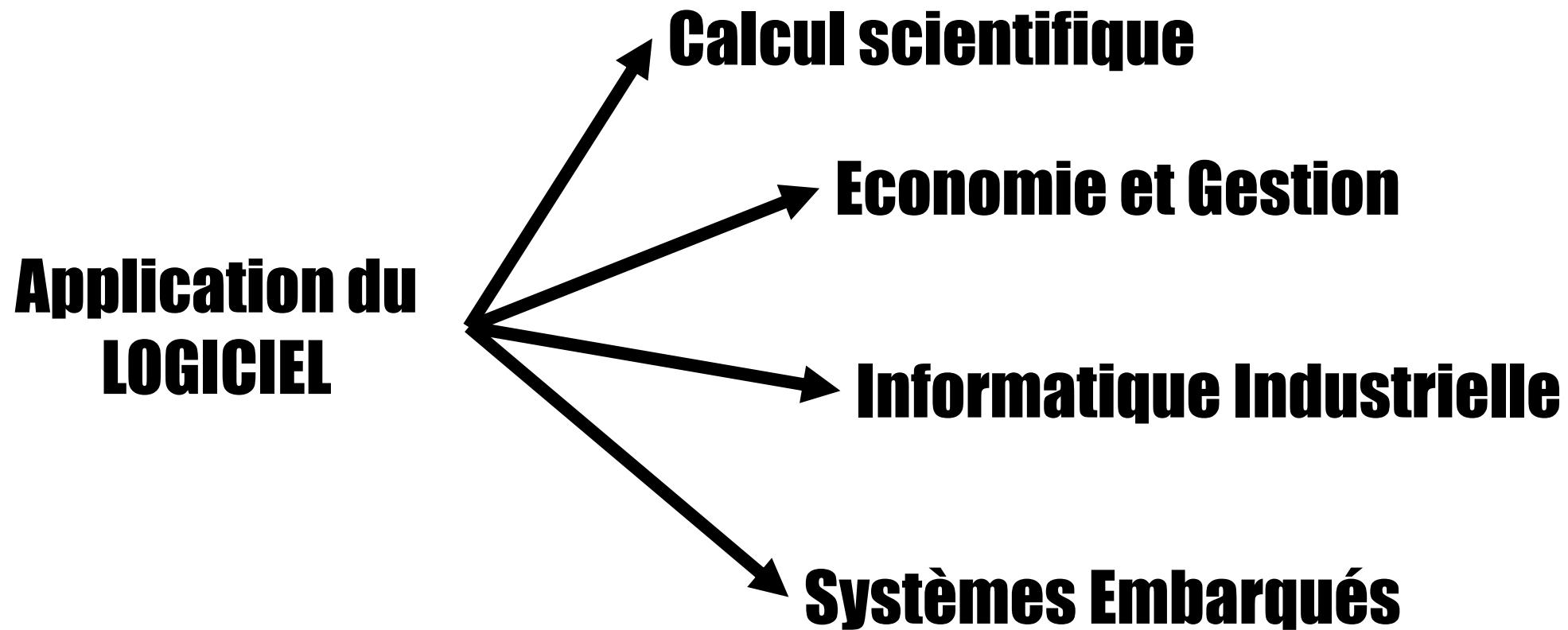
- Capteur (IOT)
- Réseau (Protocole de communication)

## Spécification : Ce que doit faire le logiciel ?

Ensemble de critères que doivent satisfaire son fonctionnement interne et externe

# CRISE DU LOGICIEL

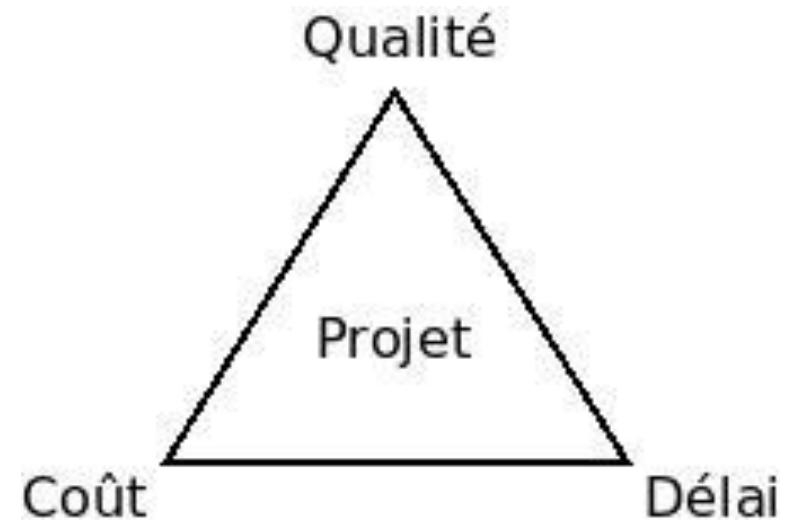
Dans les années 60...



# CRISE DU LOGICIEL

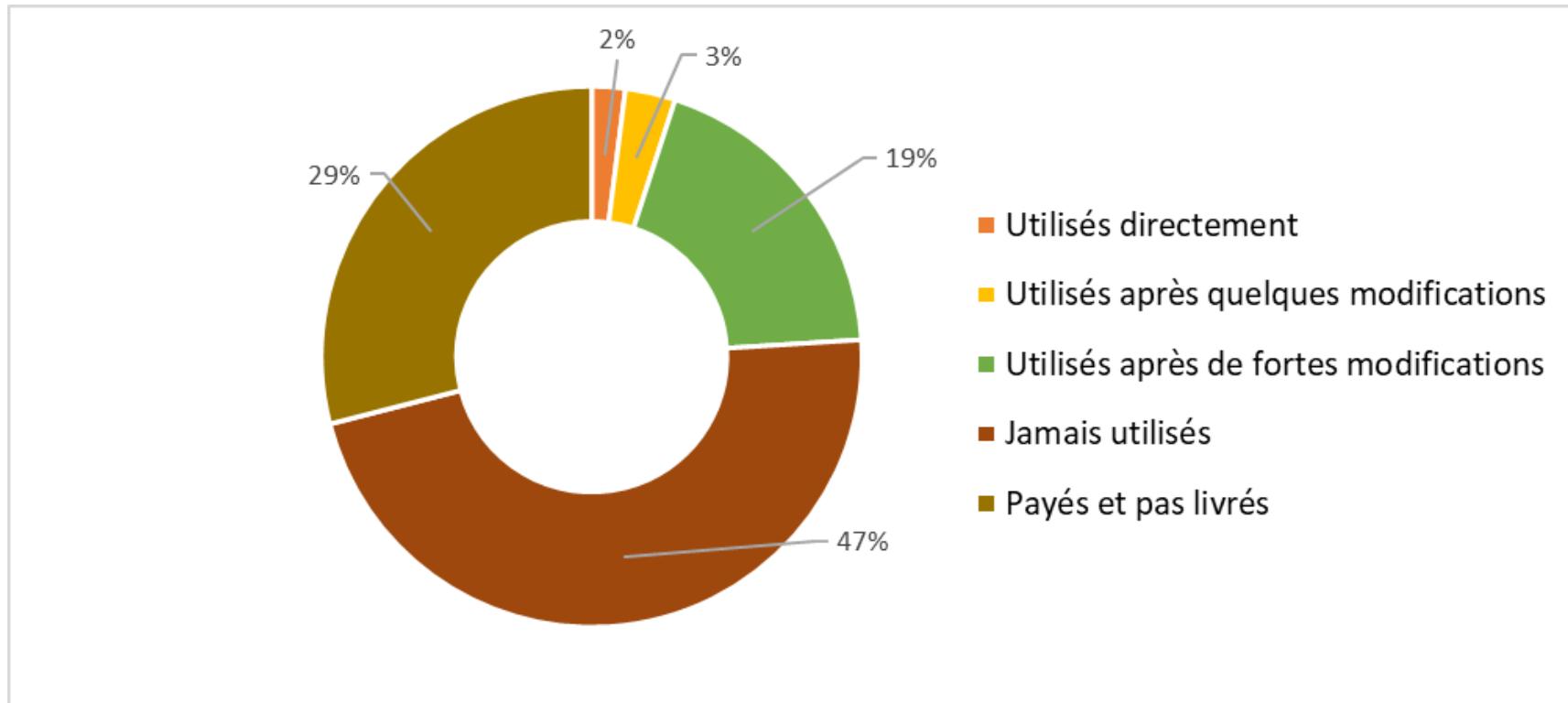
Dans les années 60...

- **délais de livraison non respectés**
- **budgets non respectés**
- **ne répond pas aux besoins de l'utilisateur ou du client**
- **difficile à utiliser, maintenir, et faire évoluer**



# CRISE DU LOGICIEL

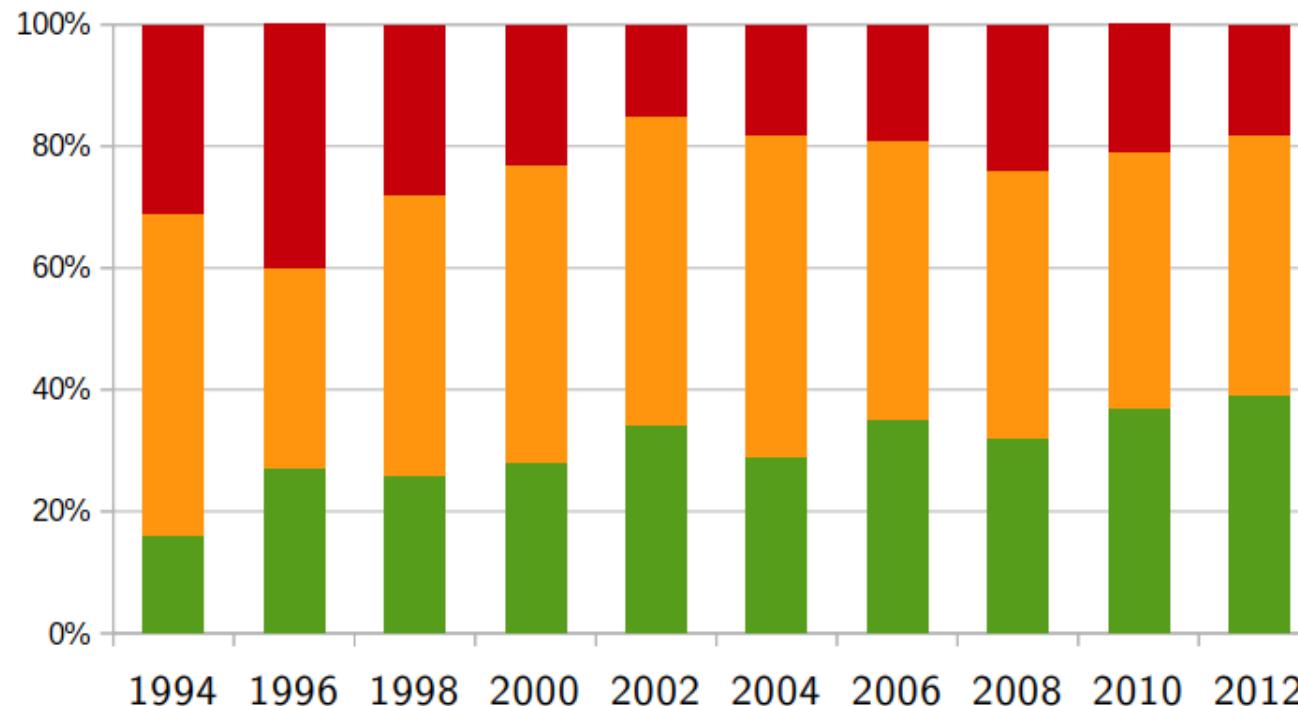
**Exemple : Etude du DoD des USA sur 9 gros projets de conception de logiciels**



Jarzombek, Stanley J., The 5th annual JAWS S3 proceedings, 1999

# CRISE DU LOGICIEL

**Exemple : Enquête sur des milliers de projets, de toutes tailles et de tous secteurs**

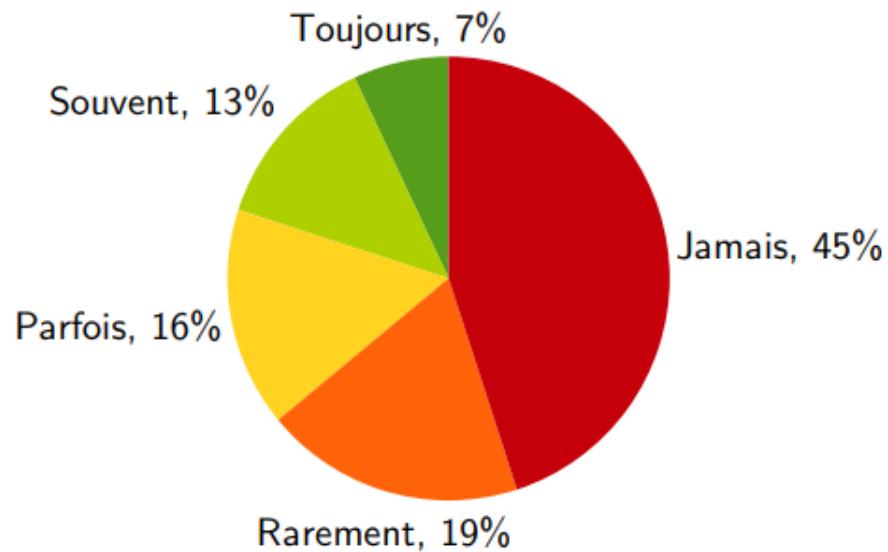


- Abandonnés avant la fin ou livrés mais jamais utilisés
- Achevés et opérationnels, mais livrés hors délais, hors budget ou sans toutes les fonctionnalités demandées.
- Achevés dans les délais et pour le budget impartis, avec toutes les fonctionnalités demandées.

Standish group, Chaos Manifesto 2013 - Think Big, Act Small, 2013

# CRISE DU LOGICIEL

## Fonctionnalité d'un logiciel



Standish group, Chaos Manifesto 2002, 2002

« La *satisfaction* du client et la *valeur* du produit sont plus grandes lorsque les *fonctionnalités* livrées sont bien *moins nombreuses* que demandé et ne remplissent que les *besoins évidents*. »

Standish group, Chaos Report 2015, 2015

# QU'EST CE QUI PEUT FAIRE QU'UN LOGICIEL SOIT DE FAIBLE QUALITE ?

- **Taches complexe**

- Forte complexité du logiciel
- Faible taille des équipes de conception / développement

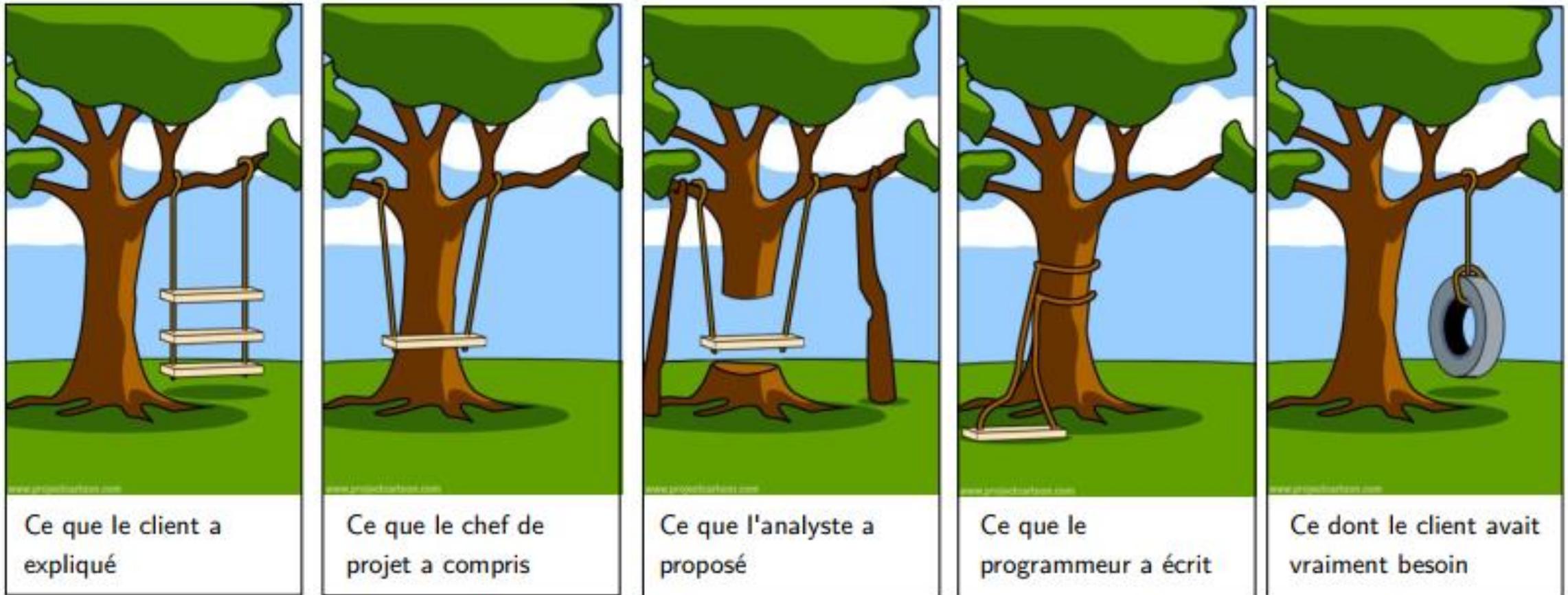
- **Manque de méthode conception et rigueur**

- Absence de stratégie de conception
- Manque d'outils et phase de validation

- **Mauvaise compréhension des besoins**

- Négligence de la phase d'analyse des besoins
- Manque d'implication du client dans le processus

# QU'EST CE QUI PEUT FAIRE QU'UN LOGICIEL SOIT DE FAIBLE QUALITE ? CHOC DES ACTEURS D'UN PROJET EN GL



# **QU'EST CE QUI PEUT FAIRE QU'UN LOGICIEL SOIT DE FAIBLE QUALITE ?**

- Difficultés spécifiques du logiciel**

- ✓ Produit invisible et immatériel
- ✓ Difficile de mesurer la qualité
- ✓ Conséquences critiques causées par modifications infimes
- ✓ Mises à jour et maintenance dues à l'évolution rapide de la technologie
- ✓ Difficile de raisonner sur des programmes
- ✓ Défaillances logicielles principalement humaines

# POURQUOI UN LOGICIEL DOIT ETRE DE BONNE QUALITE ?

- **Fiabilité, sûreté et sécurité des logiciels**



Le bug informatique qui a causé la mise hors service des systèmes de guidage à centrales inertielles de la fusée Ariane 5 en 1996 a eu lieu durant la procédure d'étalonnage de l'appareil.

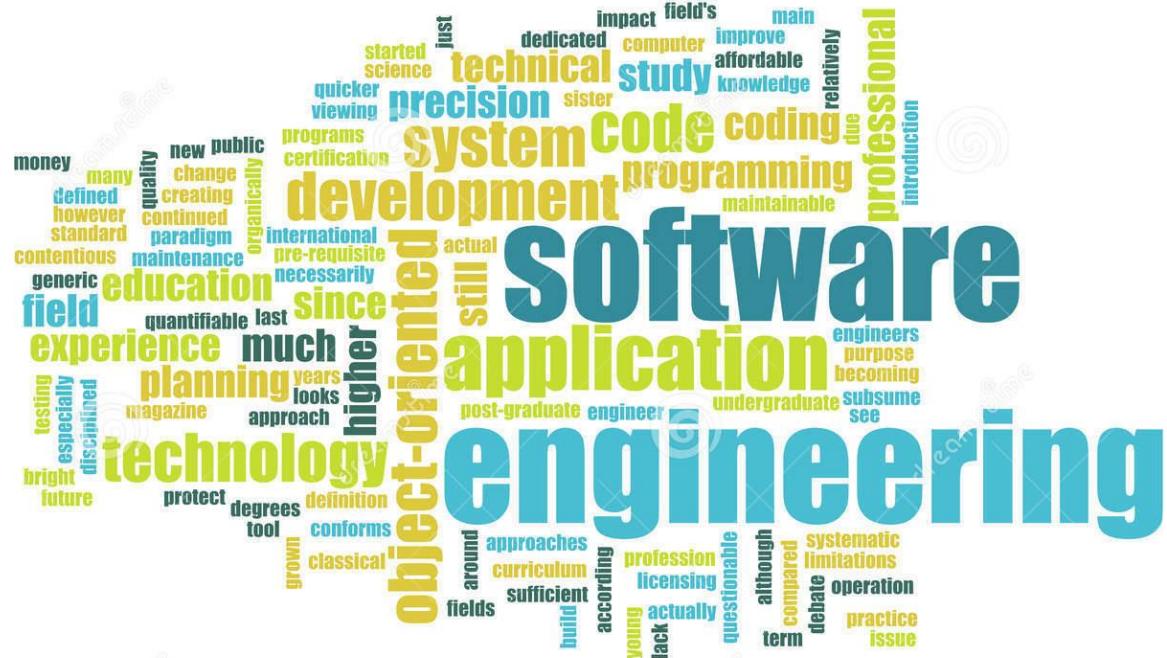
# POURQUOI UN LOGICIEL DOIT ETRE DE BONNE QUALITE ?

- **Raisons économiques (Dette technique)**



- Coût de la correction, du rappel des appareils défectueux
- Coût de l'impact sur l'image, de l'arrivée tardive sur le marché
- Coût en vies, coût de l'impact écologique

# GENIE LOGICIEL



Ensemble des méthodes, des techniques et des outils dédiés à la **conception**, au **développement** et à la **maintenance** des systèmes informatiques.

# GENIE LOGICIEL

**Objectif** : Avoir des procédures systématiques pour des logiciels de grande taille afin que

- La spécification corresponde aux besoins réels du client
- Le logiciel respecte sa spécification
- Les délais et les coûts alloués à la réalisation soient respectés

**Principe** : Appliquer les méthodes classiques d'ingénierie au domaine du logiciel

# GENIE LOGICIEL

## Ingénierie (ou génie) :

Ensemble des fonctions allant de la conception et des études à la **responsabilité de la construction** et au contrôle des équipements d'une installation technique ou industrielle. A l'instar du : génie civil, naval, aéronautique, mécanique, chimique...



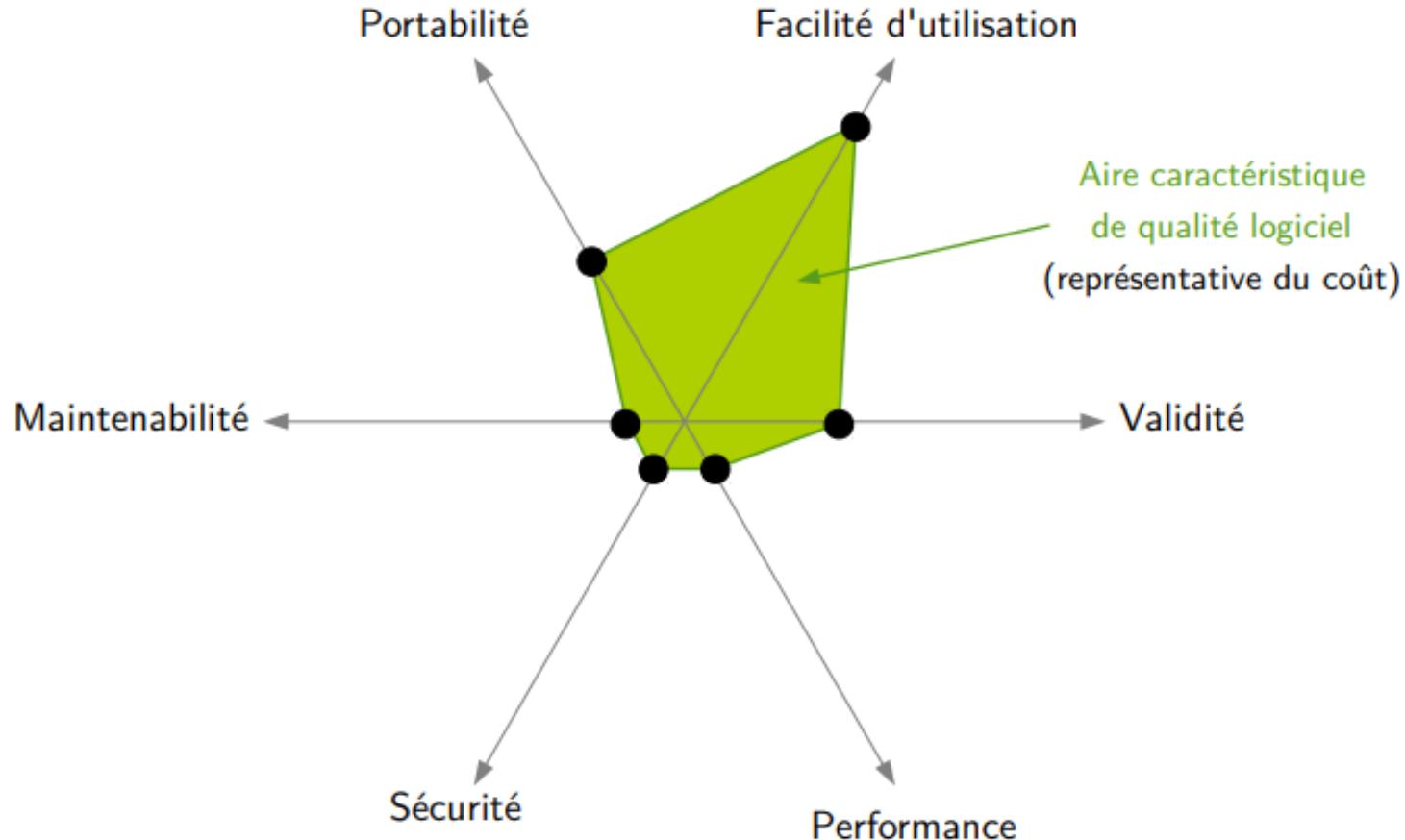
# **FINALITE DU GENIE LOGICIEL**

## Critères d'un bon logiciel

- **Validité** : réponse aux besoins des utilisateurs
- **Facilité d'utilisation** : prise en main et robustesse
- **Performance** : temps de réponse, débit, fluidité...
- **Fiabilité** : tolérance aux pannes
- **Sécurité** : intégrité des données et protection des accès
- **Maintenabilité** : facilité à corriger ou transformer le logiciel
- **Portabilité** : changement d'environnement matériel ou logiciel

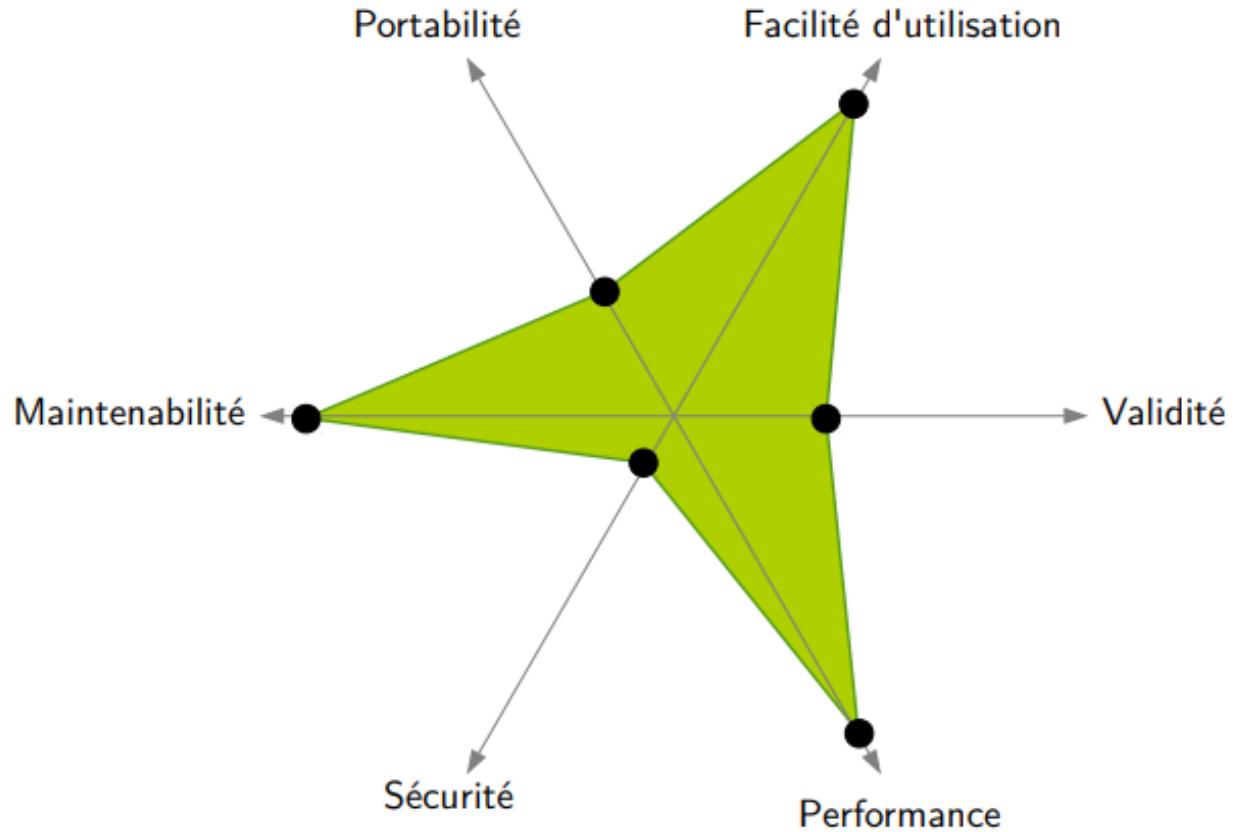
# FINALITE DU GENIE LOGICIEL

## Contrôleur de télécommande



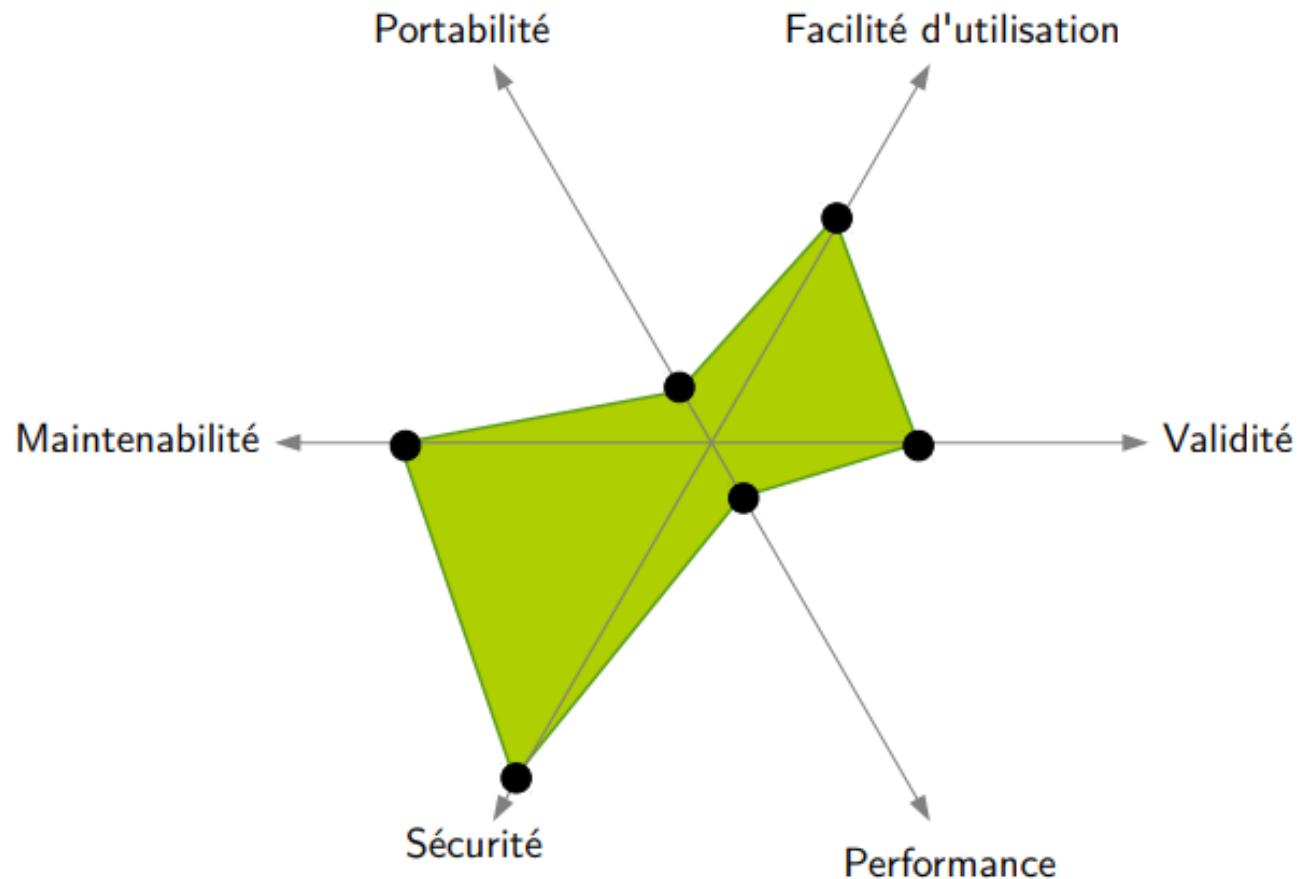
# FINALITE DU GENIE LOGICIEL

## Jeu vidéo



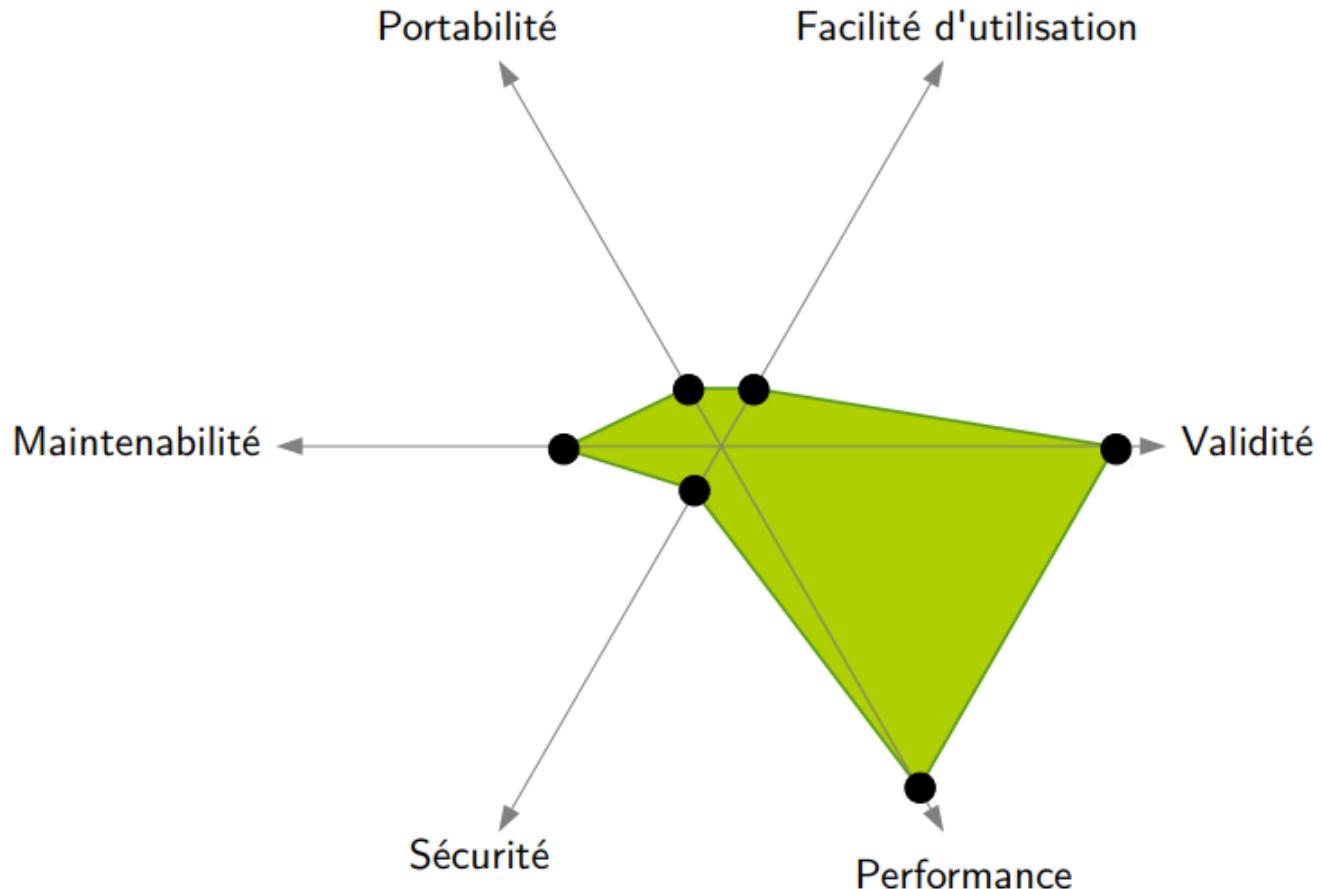
# FINALITE DU GENIE LOGICIEL

## E Mailing



# FINALITE DU GENIE LOGICIEL

## Simulateur de météo



# PROCESSUS DE DEVELOPPEMENT LOGICIEL

C'est l'ensemble des activités successives, organisées en vue de la production d'un logiciel

- Pas de processus idéal
- Choix du processus en fonction des contraintes (taille des équipes, temps, qualité...)
- Adaptation de « processus types » aux besoins réels

Activités du développement logiciel

1. Analyse des besoins
2. Spécification
3. Conception
4. Programmation
5. Validation et vérification (Tests)
6. Livraison (Déploiement)
7. Maintenance

Documentez  
chaque activité

# PROCESSUS DE DEVELOPPEMENT LOGICIEL

## 1. Analyse des besoins :

Comprendre les besoins du client

- Objectifs généraux, environnement du futur système, ressources disponibles, contraintes de performance...
- Fournie par le client (expert du domaine d'application, futur utilisateur...)

## 2. Spécification

- Établir une description claire de ce que doit faire le logiciel (fonctionnalités détaillées, exigences de qualité, interface...)
- Clarifier le cahier des charges en listant les exigences fonctionnelles et non fonctionnelles

# PROCESSUS DE DEVELOPPEMENT LOGICIEL

## 3. Conception :

Élaborer une solution concrète réalisant la spécification

- Description architecturale en composants (avec interface et fonctionnalités)
- Réalisation des fonctionnalités par les composants (algorithmes, organisation des données)
- Réalisation des exigences non fonctionnelles (performance, sécurité...)

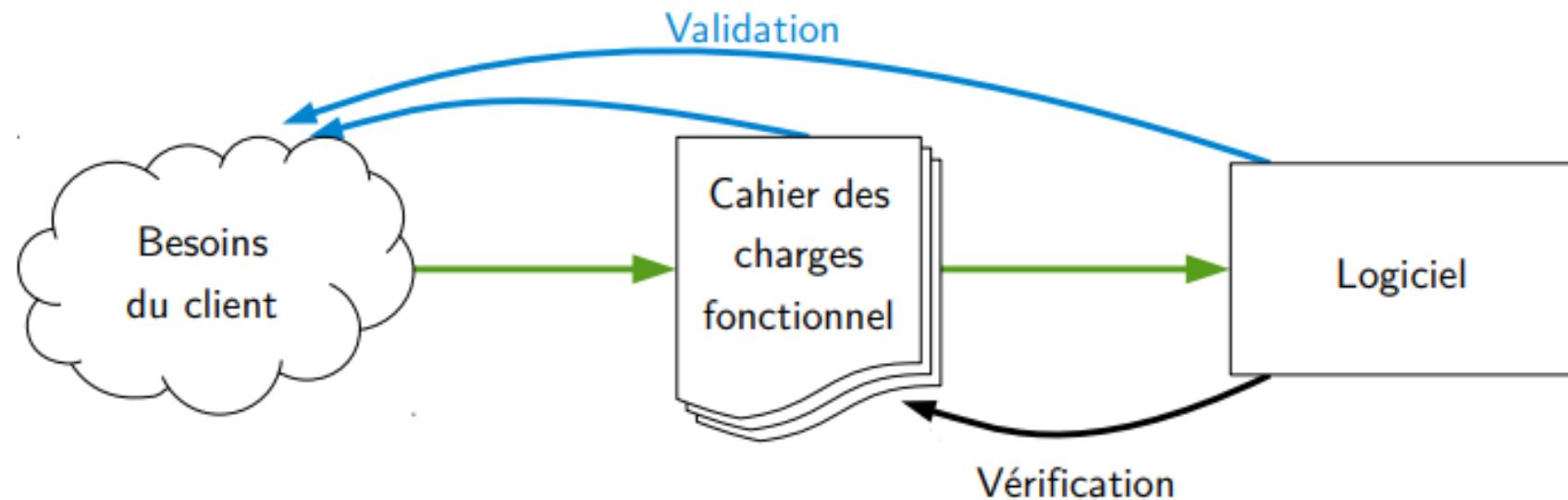
## 4. Programmation :

Implémentation de la solution conçue

- Choix de l'environnement de développement, du/des langage(s) de programmation, de normes de développement...

# PROCESSUS DE DEVELOPPEMENT LOGICIEL

- **Validation** : assurer que les besoins du client sont satisfaits (au niveau de la spécification, du produit fini...)
- **Vérification** : assurer que le logiciel satisfait sa spécification



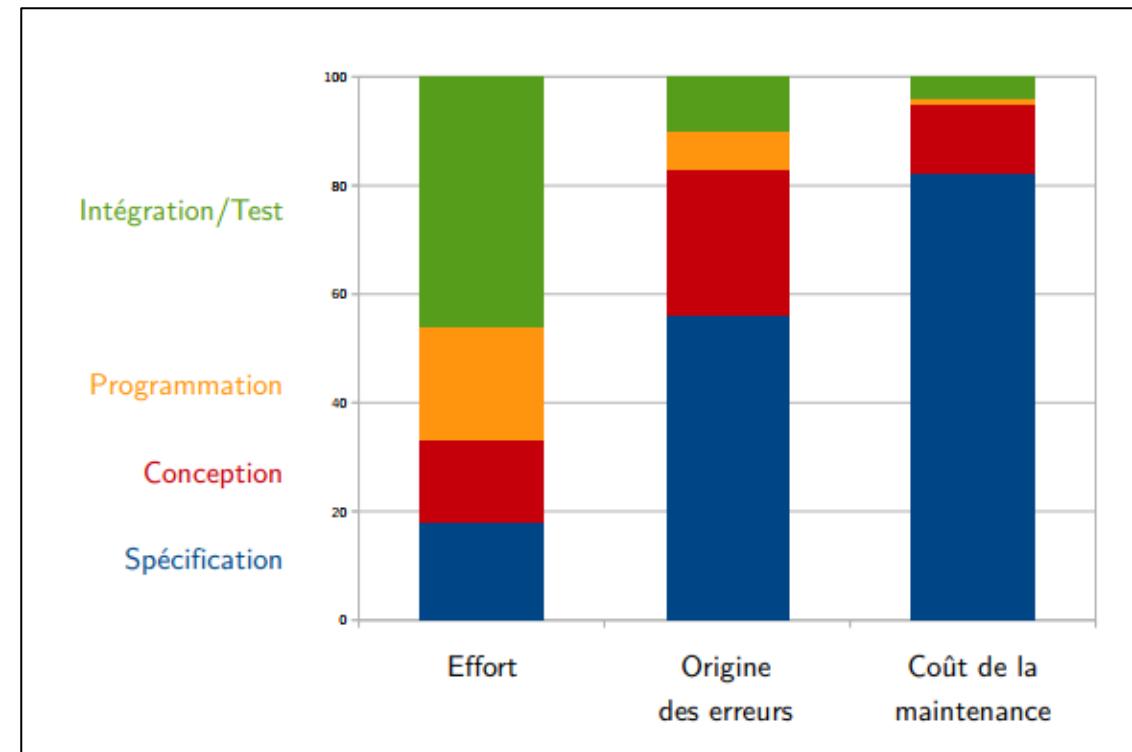
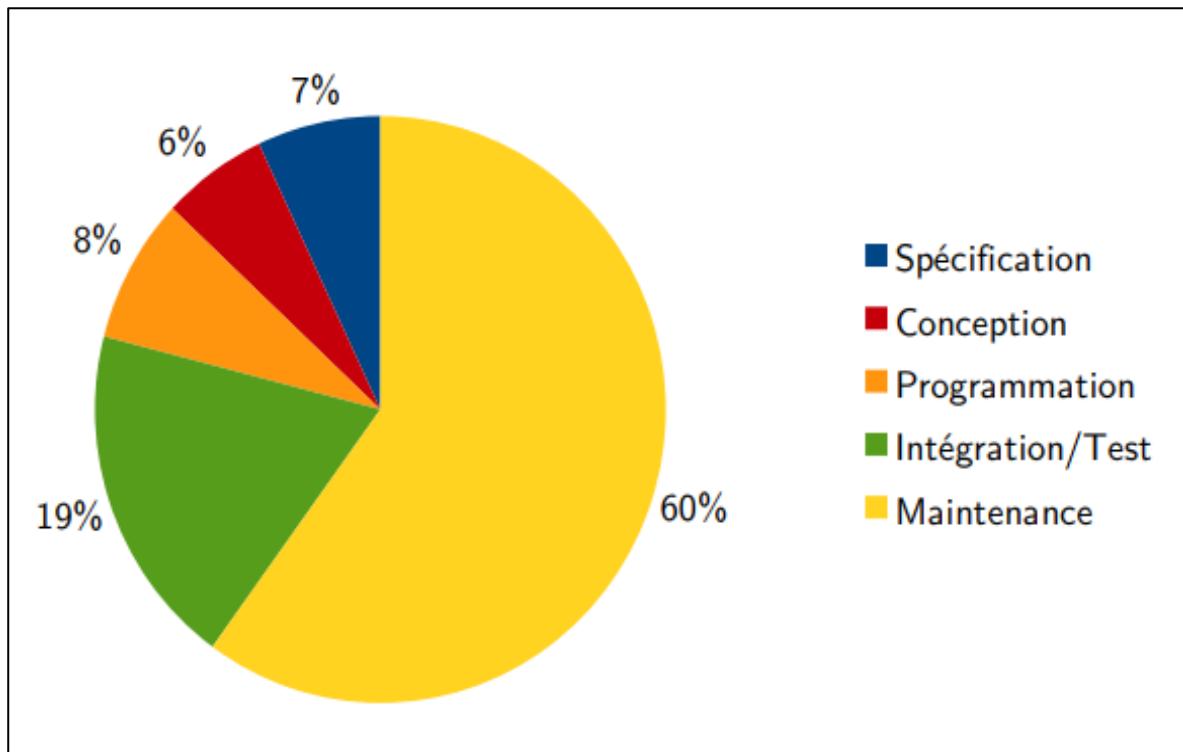
# PROCESSUS DE DEVELOPPEMENT LOGICIEL

## 6. Maintenance :

- **Correction** : identifier et corriger des erreurs trouvées après la livraison
- **Adaptation** : adapter le logiciel aux changements dans l'environnement (format des données, environnement d'exécution...)
- **Perfection** : améliorer la performance, ajouter des fonctionnalités, améliorer la maintenabilité du logiciel

# PROCESSUS DE DEVELOPPEMENT LOGICIEL

Répartition de l'effort dans un projet de développement de logiciel

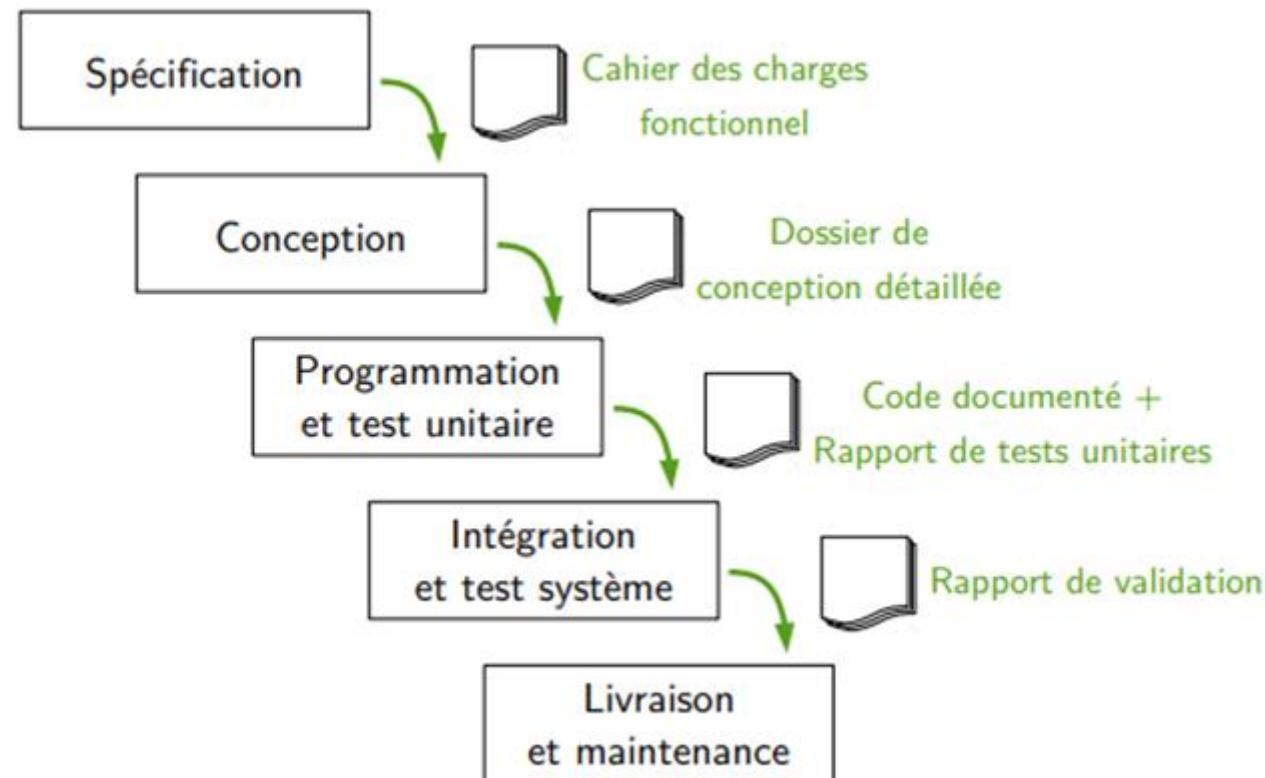


# PROCESSUS DE DEVELOPPEMENT LOGICIEL

## Processus en cascade :

Chaque étape doit être terminée avant que ne commence la suivante.

À chaque étape, la production d'un document étaye l'étape suivante



# PROCESSUS DE DEVELOPPEMENT LOGICIEL

**Processus en cascade :**

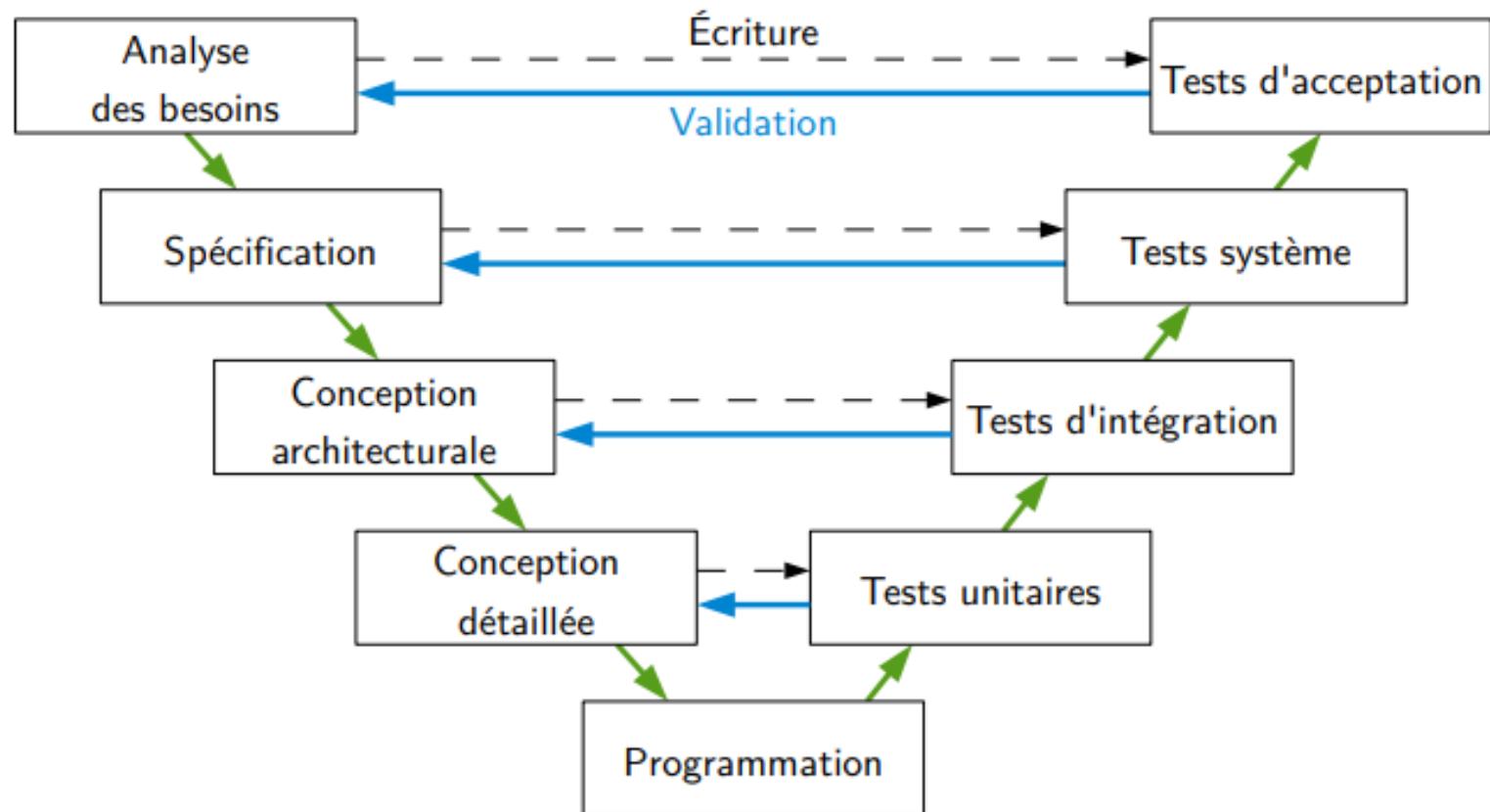
**Spécificités :**

- Hérité des méthodes classiques d'ingénierie et d'architecture
- Découverte d'une erreur entraîne un retour à la phase à l'origine de l'erreur et nouvelle cascade, avec de nouveaux documents...
- Coût de modification d'une erreur important, donc choix en amont cruciaux

**Pas toujours adapté si les besoins du client changeants ou difficiles à spécifier**

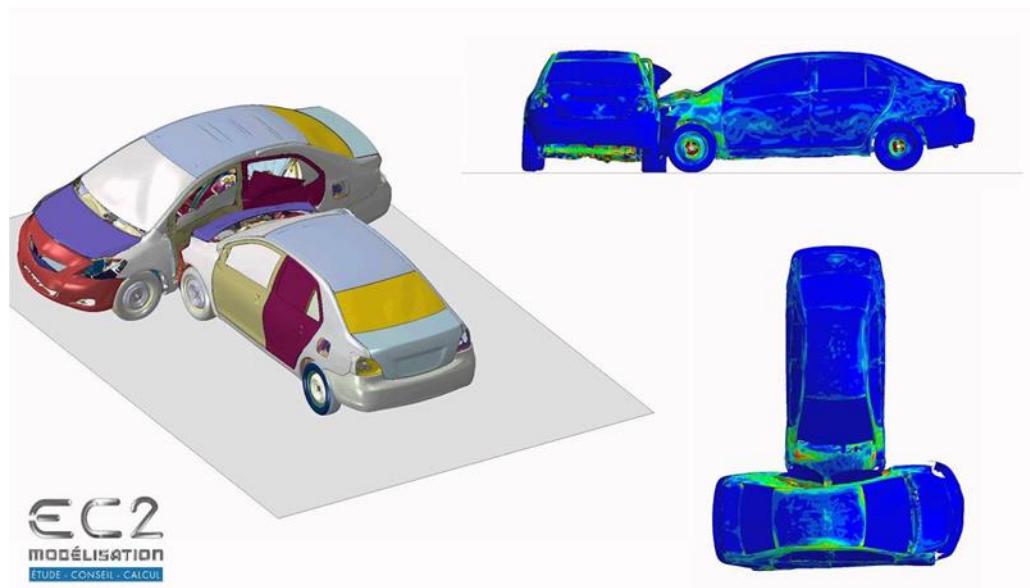
# PROCESSUS DE DEVELOPPEMENT LOGICIEL

Processus en V :



# PROCESSUS DE DEVELOPPEMENT LOGICIEL

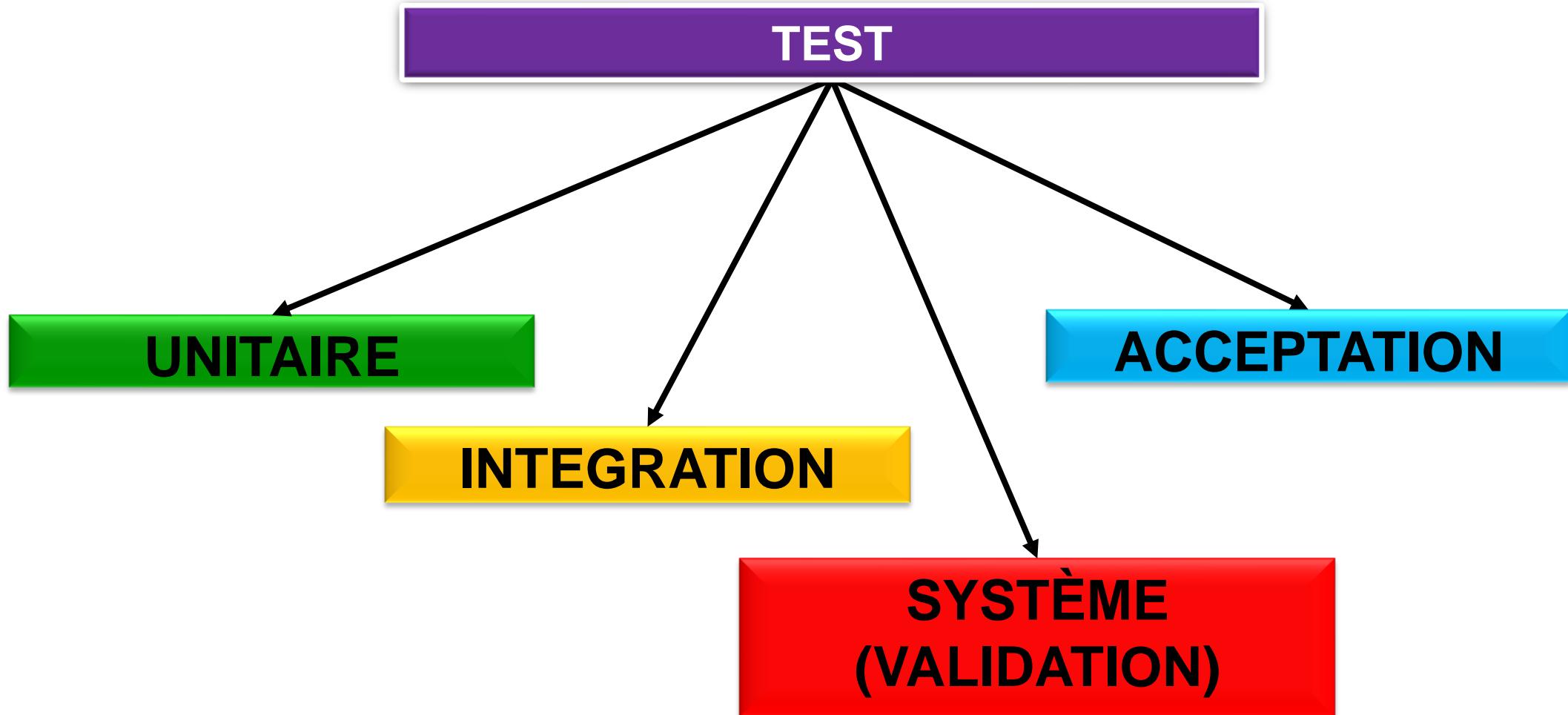
## Test



- En informatique, un **test** désigne une procédure de vérification partielle d'un système. Son objectif principal est d'identifier un nombre maximum de comportements problématiques du logiciel. Il permet ainsi, dès lors que les problèmes identifiés seront corrigés, d'en augmenter la qualité.
- D'une manière plus générale, le test désigne toutes les activités qui consistent à rechercher des informations quant à la qualité du système afin de permettre la prise de décisions.
- Un test ressemble à une expérience scientifique, il examine une hypothèse exprimée en fonction de trois éléments : les données en entrée, l'objet à tester et les observations attendues. Cet examen est effectué sous conditions contrôlées pour pouvoir tirer des conclusions dans l'idéal, être reproduit.

# PROCESSUS DE DEVELOPPEMENT LOGICIEL

Test



# PROCESSUS DE DEVELOPPEMENT LOGICIEL

## Test

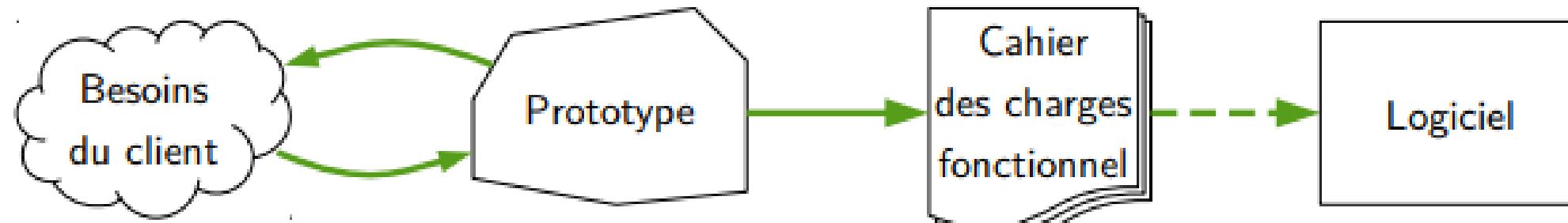
- **Test unitaire** : test de chaque unité d'un programme (méthode, classe, composant) indépendamment du reste du système ;
- **Test d'intégration** : test des interactions entre composants (interfaces et composants compatibles) ;
- **Test système** : test du système complet par rapport à son cahier des charges ;
- **Test d'acceptation (recette)** : fait par le client, validation par rapport aux besoins initiaux.

# QUELQUES STYLES DE DEVELOPPEMENT

## Développement par prototypage

### Principe :

- Développement rapide d'un prototype avec le client pour valider ses besoins
- Écriture de la spécification à partir du prototype, puis processus de développement linéaire



### Avantage :

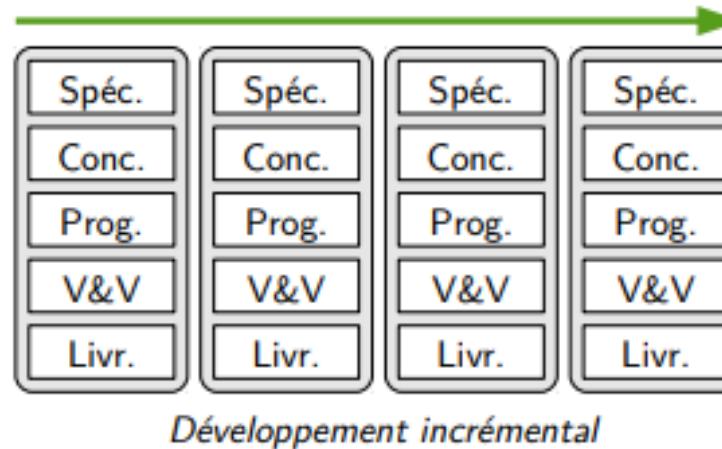
Validation concrète des besoins, moins de risques d'erreur de spécification

# QUELQUES STYLES DE DEVELOPPEMENT

## Développement incrémental

### Principe :

- Hiérarchiser les besoins du client
- Concevoir et livrer au client un produit implantant un sous - ensemble de fonctionnalités par ordre de priorité



**Avantage :** Minimiser le risque d'inadéquation aux besoins

**Difficulté :** Intégration fonctionnalités secondaires non pensées en amont

# QUELQUES STYLES DE DEVELOPPEMENT

## Méthode Agiles

### Principe :

- Implication constante du client
- Programmation en binôme (revue de code permanente)
- Développement dirigé par les tests
- Cycles de développement rapides pour réagir aux changements

### Avantages :

- Développement rapide en adéquation avec les besoins

### Inconvénients :

- Pas de spécification, documentation = tests, maintenance ?

# **DOCUMENTATION**

## **Objectif : Traçabilité du projet**

### **1. Pour l'équipe :**

- Regrouper et structurer les décisions prises
- Faire référence pour les décisions futures
- Garantir la cohérence entre les modèles et le produit

### **2. Pour le client :**

- Donner une vision claire de l'état d'avancement du projet

### **3. Base commune de référence :**

- Personne quittant le projet : pas de perte d'informations
- Personne rejoignant le projet : intégration rapide



# Chapitre 2

# Modélisation

Concepts Généraux

Modélisation Orientée Objet

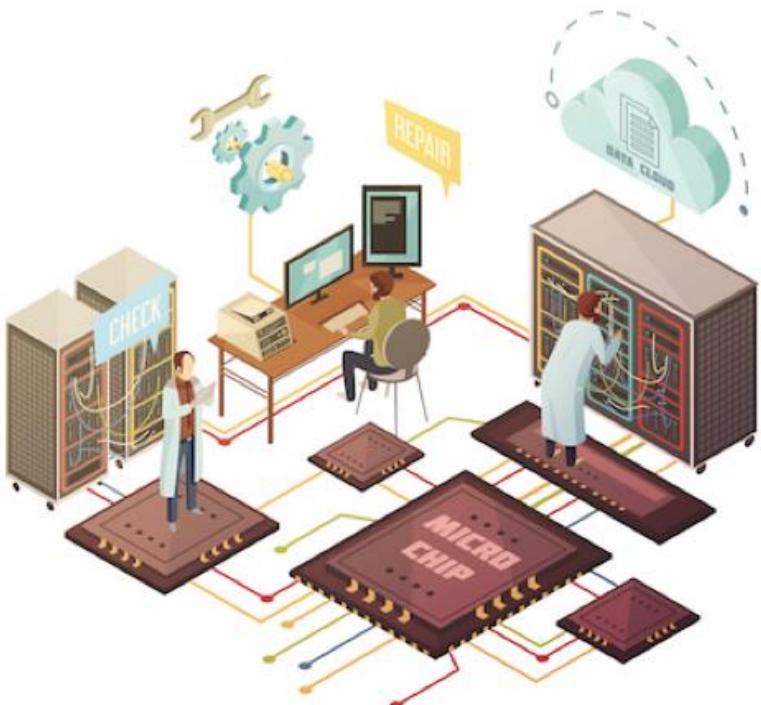
Conception de modèles avec UML

Processus de modélisation

# CONCEPTS GENERAUX

Qu'est ce qu'un Système d'Information ?

Un SI est un ensemble de ressources physiques et / ou logiques inter reliées permettant de recueillir des **données**, de les stocker et de les traiter avant de les diffuser sur un format compréhensible appelé **Information**.



- **Donnée** : Élément qui sert de base à un raisonnement, de point de départ pour une recherche.
- **Information** : C'est un élément de connaissance susceptible d'être conservé, traité ou transmis à l'aide d'un support et d'un mode de codification normalisé.

# CONCEPTS GENERAUX

Qu'est ce qu'un Modèle ?

C'est l'abstraction de la réalité



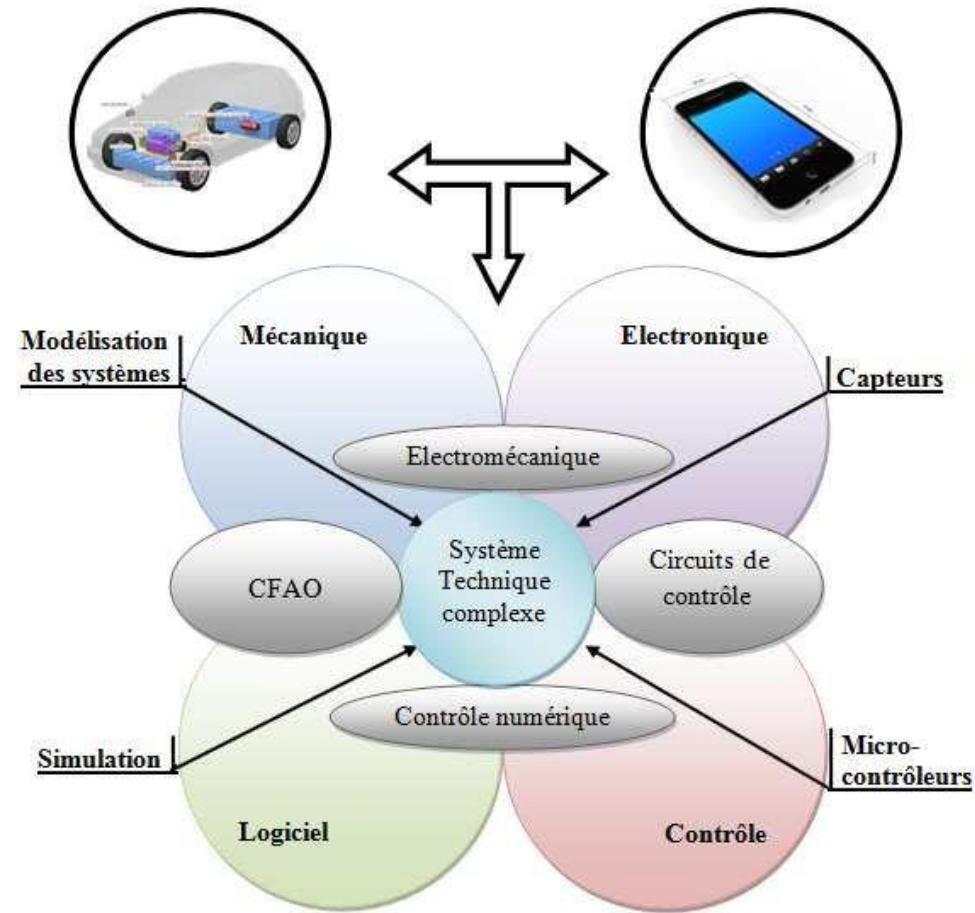
En philosophie, la notion d'abstraction renvoie à l'opération de l'esprit par laquelle les propriétés générales, universelles et nécessaires d'un objet sont distinguées de ses propriétés particulières et contingentes.

**Exemple : Le corps humain est composé de la tête, du tronc et de quatre membres.**

# CONCEPTS GENERAUX

## A quoi sert l'abstraction dans un modèle ?

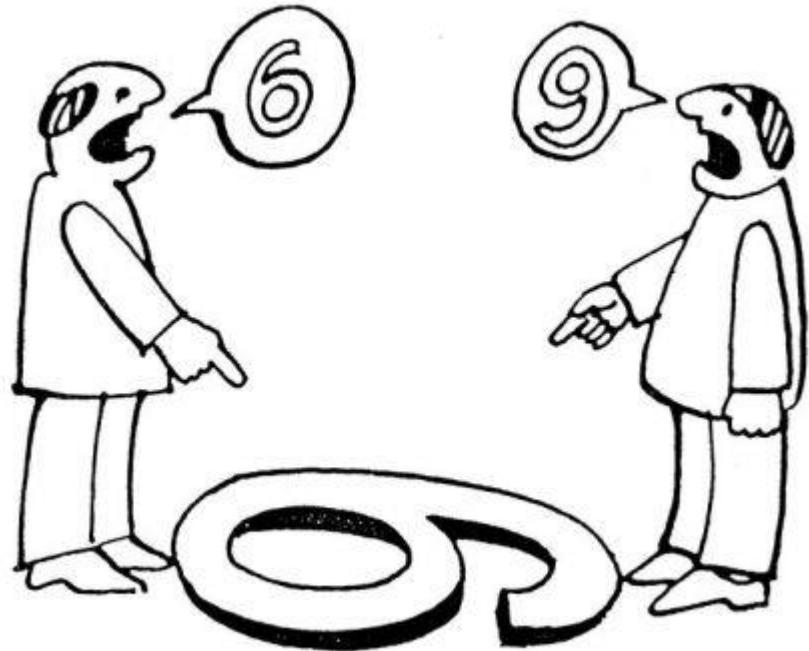
1. De faciliter la compréhension du système étudié par la réduction de sa complexité.
2. De simuler le système étudié à partir de sa représentation pour prendre connaissance de la reproduction de ses comportements.



# CONCEPTS GENERAUX

Qu'est ce qu'un modèle ?

Représente une vue **subjective** mais **pertinente** de la réalité.



Un modèle définit une frontière entre la réalité et la perspective de l'observateur. Ce n'est pas "la réalité", mais une vue personnelle et subjective de la réalité.

Bien qu'un modèle ne représente pas une réalité absolue, un modèle doit refléter des aspects importants de la réalité, il en donne donc une vue juste et pertinente

# CONCEPTS GENERAUX

## Exemples de modèle

- **Modèle météorologique :**

A partir de données satellitaire permet de prévoir les conditions climatiques pour les jours à venir.

- **Modèle économique :**

Peut par exemple permettre de simuler l'évolution de cours boursiers en fonction d'hypothèses macro-économiques (évolution du chômage, taux de croissance...).

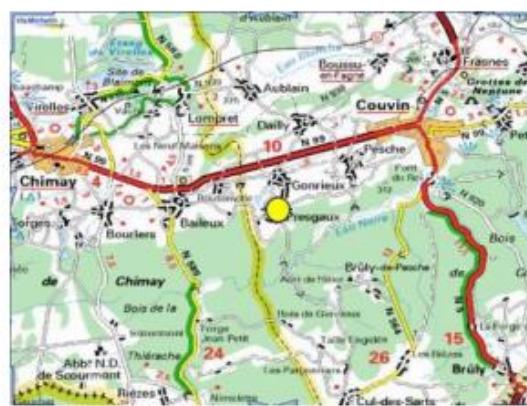
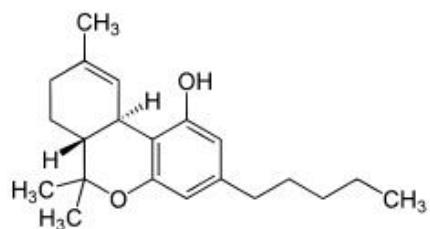
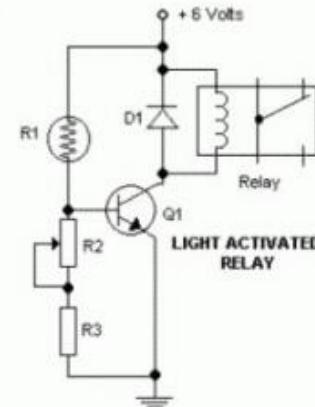
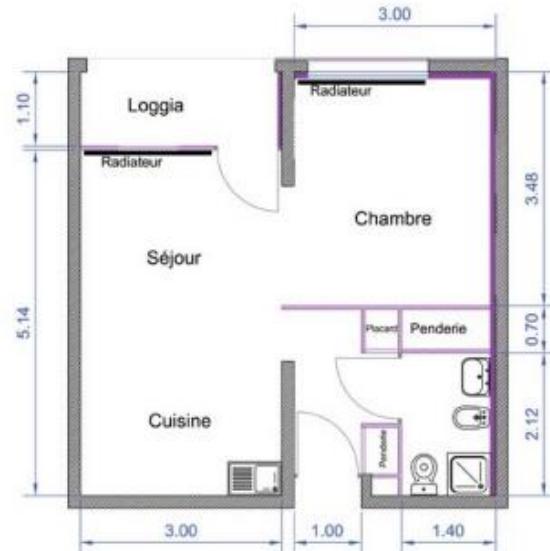
- **Modèle démographique :**

Définit la composition d'un panel d'une population et son comportement, dans le but de fiabiliser des études statistiques, d'augmenter l'impact de démarches commerciales, etc...

# CONCEPTS GENERAUX

## Qu'est ce que la modélisation ?

C'est tout processus de mise en œuvre d'un modèle d'un système donné.



# CONCEPTS GENERAUX

## Qu'est ce que la modélisation ?

**Modèle :**

***Simplification de la réalité, abstraction, vue subjective***

Modéliser un concept pour :

- Mieux le comprendre (modélisation en physique)
- Mieux le construire (modélisation en ingénierie)

En génie logiciel :

- Modélisation = spécification + conception
- Aider la réalisation d'un logiciel à partir des besoins du client

# **CONCEPTS GENERAUX**

## **Objectifs de la modélisation**

1. Visualiser un système tel qu'il est ou tel que nous voudrions qu'il soit.
2. Préciser la structure ou le comportement d'un système.
3. Fournir un canevas qui guide la construction d'un système.
4. Documenter les décisions prises.

# CONCEPTS GENERAUX

## Méthode de modélisation

Une méthode de modélisation, c'est le regroupement de trois aspects :

- **Démarche :**

Explique la marche à suivre en exploitant au mieux les principes de modularité, d'abstraction, de réutilisation, etc.

- **Formalisme de représentation :**

Facilite la communication, l'organisation et la vérification.

- **Modèles :**

Facilitent les retours sur la conception et l'évolution des applications.

# CONCEPTS GENERAUX

## Il existe trois méthodes de modélisation

- **Méthodes fonctionnelles :**

Diviser pour régner

- **Méthodes systémiques :**

Séparation des données et des traitements.

- **Méthodes objets :**

Intégration des données et des traitements dans une entité unique

# CONCEPTS GENERAUX

## Modélisation fonctionnelle

Elle est centrée sur L'identification :

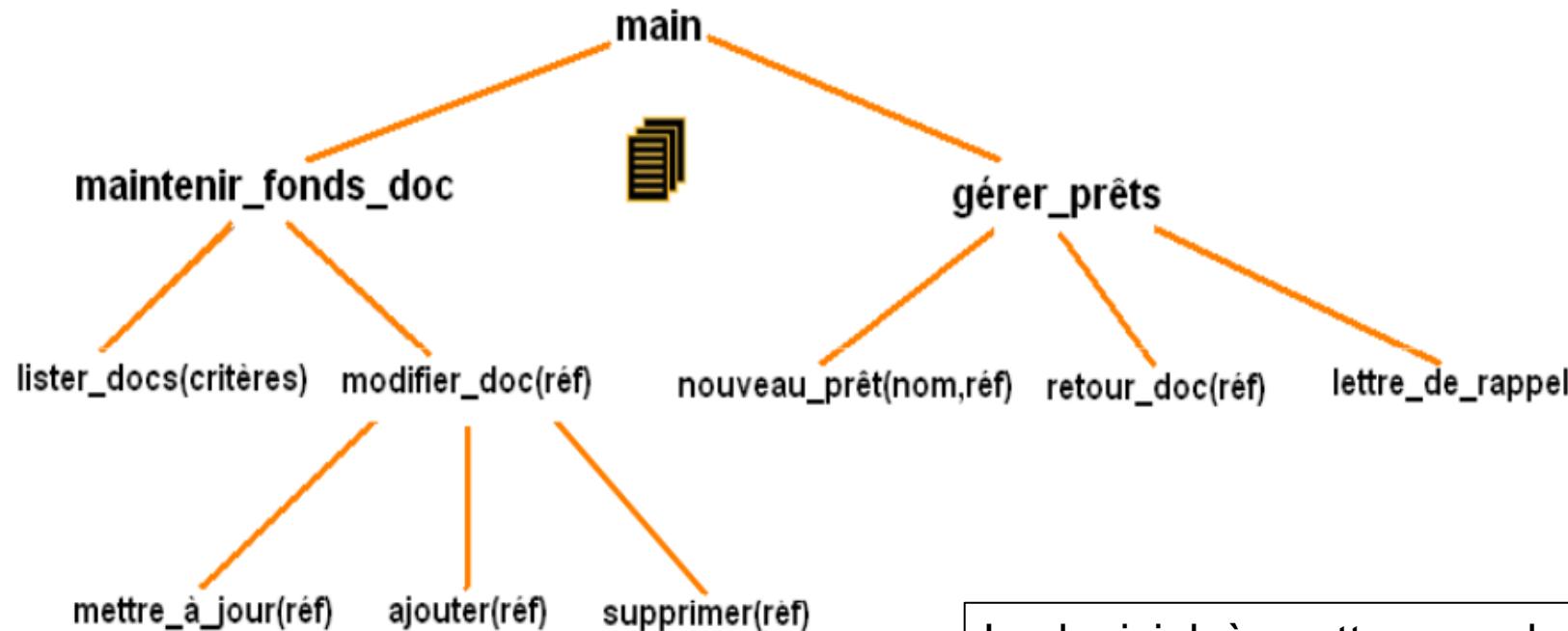
- Des **données** (attributs ou variables) du système
- Des **fonctions** (services services ou comportements) du système
- Et de l'**interdépendance** des fonctions.

NB : Les fonctions manipulent les données du système

# CONCEPTS GENERAUX

## Modélisation fonctionnelle

Exemple :



Le logiciel à mettre en place va comporter une hiérarchie de fonctions, qui fournissent les services désirés, ainsi que de données qui représentent les éléments manipulés (livres, etc...).

# CONCEPTS GENERAUX

## Modélisation systémique

Séparation des données et des traitements.



# CONCEPTS GENERAUX

## Modélisation orientée objet

**Objectif principal** : Réduire et gérer la complexité des logiciels.

- Décomposition modulaire ;
- Regroupement des fonctions et des propriétés concernant un type donné dans un module ;
- Cacher la complexité des fonctions et celle de leurs actions ;
- Fournir une interface qui sera la partie visible du module ;
- Communication par envoi de message

# CONCEPTS GENERAUX

## Concepts clés de la modélisation orientée objet

### Objet

L'objet est une entité formée de la réunion d'un état et d'un comportement.  
Il est désigné par un identificateur (son nom).

#### Exemple d'objet voiture :

**Etat :**

**Marque = Peugeot206;**

**Couleur = bleue;**

**Nombreportes = 4;**

**Immatriculation = DK – 1233 - AA**

**Comportement :**

**rouler, freiner, s'arrêter.**

Objets :



BM



Renault



Peugeot



Audi

# CONCEPTS GENERAUX

## Concepts clés de la modélisation orientée objet

### L'état d'un objet

- Est défini par l'ensemble des valeurs prises par les caractéristiques déterminantes (attributs) de l'objet.
- Il est variable car les attributs peuvent prendre de nouvelles valeurs en fonction du comportement du système.

# **CONCEPTS GENERAUX**

## **Concepts clés de la modélisation orientée objet**

### **Le comportement d'un objet**

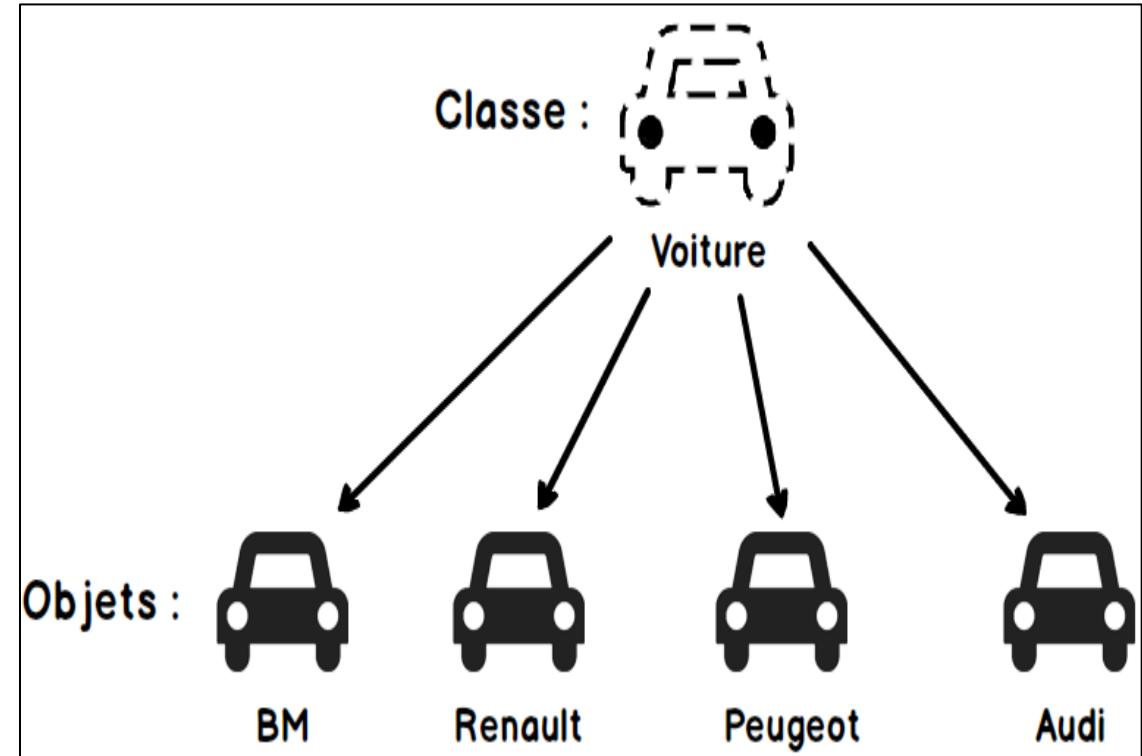
- Il correspond à la mise en œuvre d'une des compétences (fonctions, méthodes ou opérations) de l'objet.
- Le comportement est déclenché par un stimuli (message) interne ou externe à l'objet.

# CONCEPTS GENERAUX

## Concepts clés de la modélisation orientée objet

### Classe

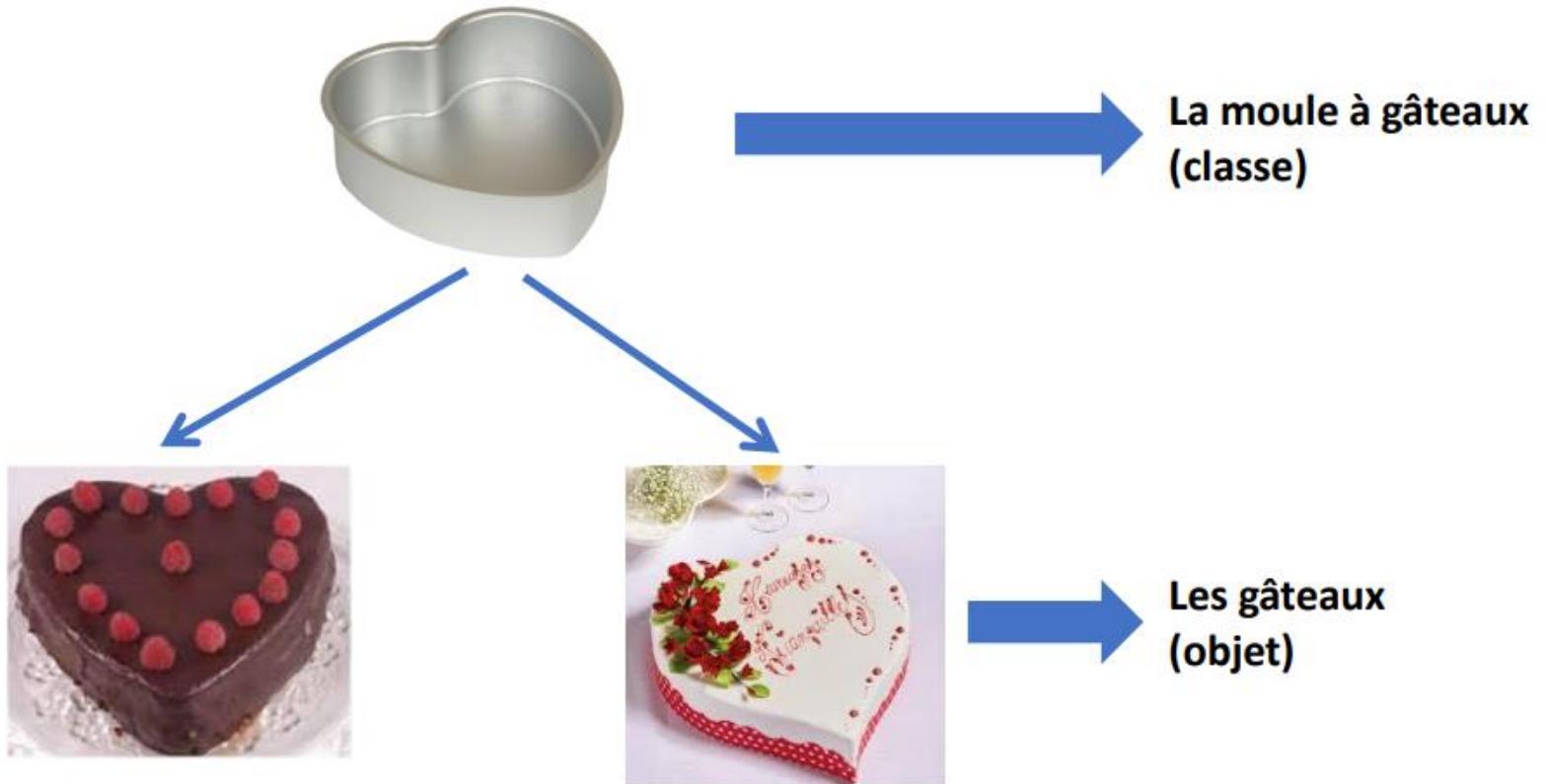
- Une classe est une factorisation d'attributs et de comportements d'un ensemble d'objets ;
- C'est donc un type abstrait caractérisé par des propriétés (attributs et méthodes) communes à un ensemble d'objets ;
- Un objet n'est alors rien d'autre qu'une instance de classe.



# CONCEPTS GENERAUX

## Concepts clés de la modélisation orientée objet

### Analogie entre classe et objet



# CONCEPTS GENERAUX

## Concepts clés de la modélisation orientée objet

### Visibilité

#### PUBLIC (+)

- *Attribut ou Méthode visible partout*

#### PRIVE (-)

- Attribut ou Méthode visible à l'intérieur de la classe

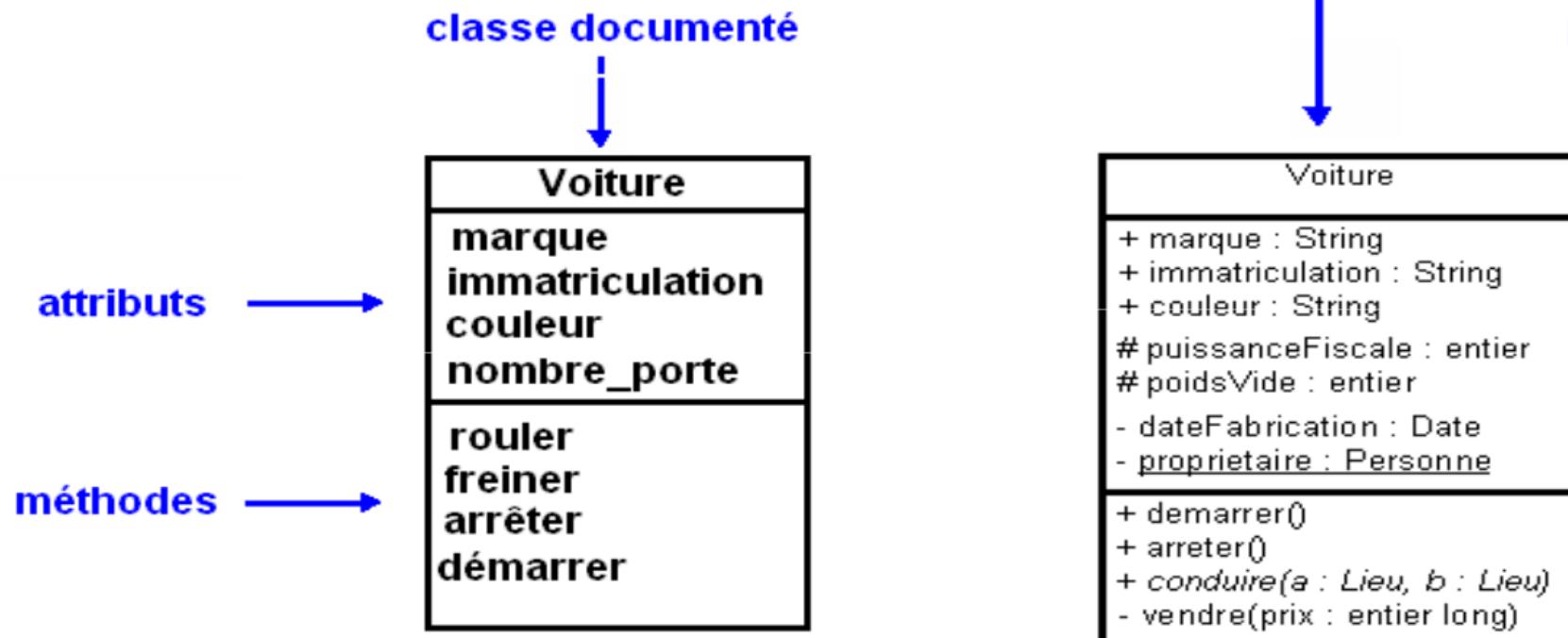
#### PROTÉGÉ (#)

- Attribut ou méthode visible juste à l'intérieur d'une classe et de par ses sous classes

# CONCEPTS GENERAUX

## Concepts clés de la modélisation orientée objet

### Conception d'une classe



# CONCEPTS GENERAUX

## Concepts clés de la modélisation orientée objet

### Encapsulation / Interface

- L'encapsulation consiste à masquer les détails d'implémentation d'un objet, en définissant une interface
- L'interface est la vue externe d'un objet, elle définit les services accessibles (offerts) aux utilisateurs de l'objet.
- L'encapsulation facilite l'évolution d'une application car elle stabilise l'utilisation des objets
- L'encapsulation garantit l'intégrité des données, car elle permet d'interdire l'accès direct aux attributs des objets (utilisation d'accesseurs et de mutateurs).

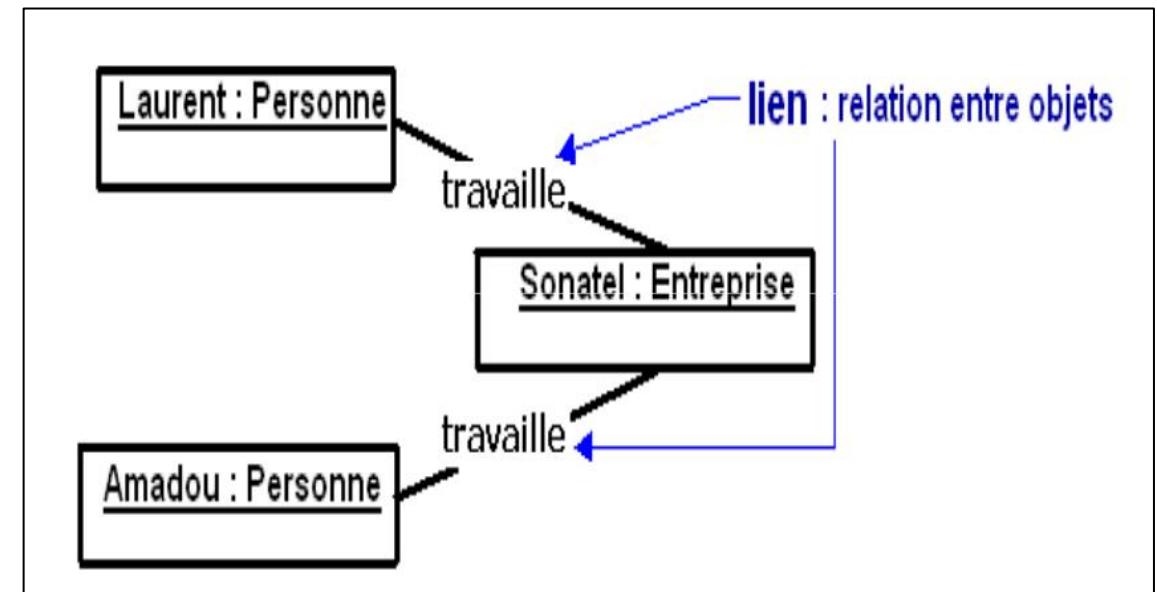
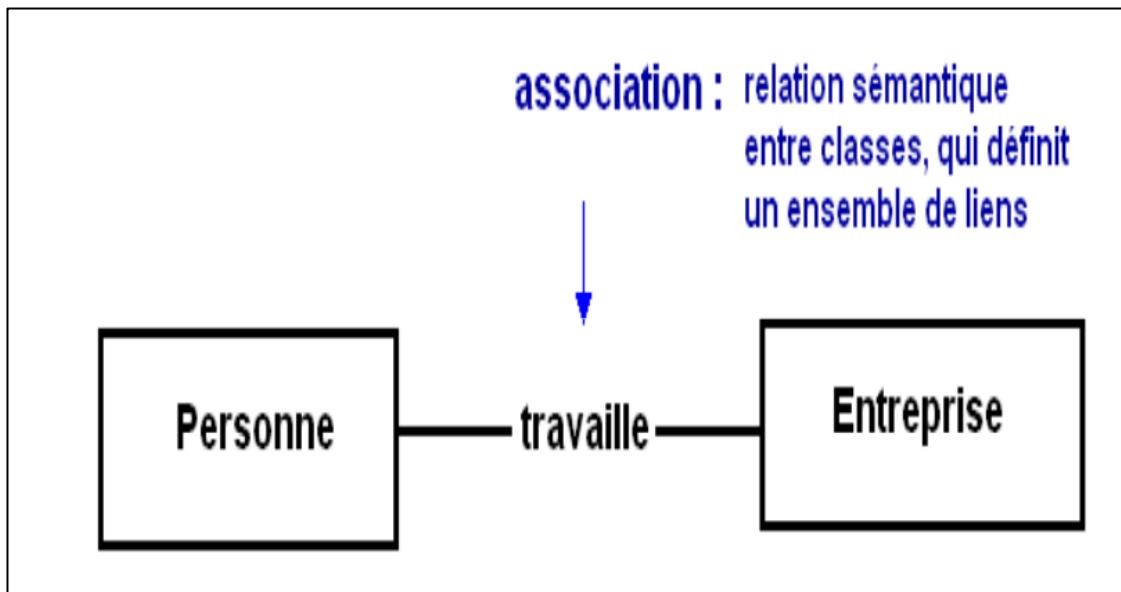
# CONCEPTS GENERAUX

## Concepts clés de la modélisation orientée objet

### Liens entre les classes

#### Association

- Une association exprime une connexion sémantique bidirectionnelle entre deux classes.
- L'association est instanciable, sous forme de liens entre objets issus de classes associées.



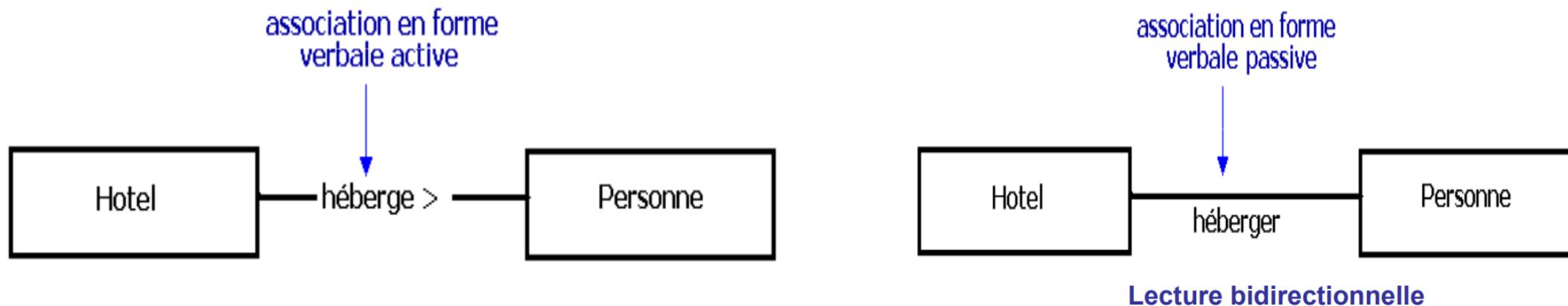
# CONCEPTS GENERAUX

## Concepts clés de la modélisation orientée objet

### Liens entre les classes

#### Association

- Précise le sens de lecture principal d'une association.



# CONCEPTS GENERAUX

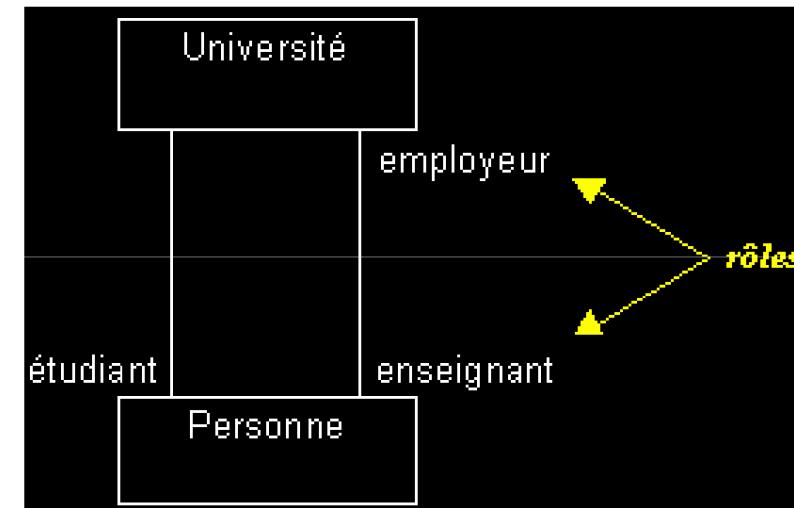
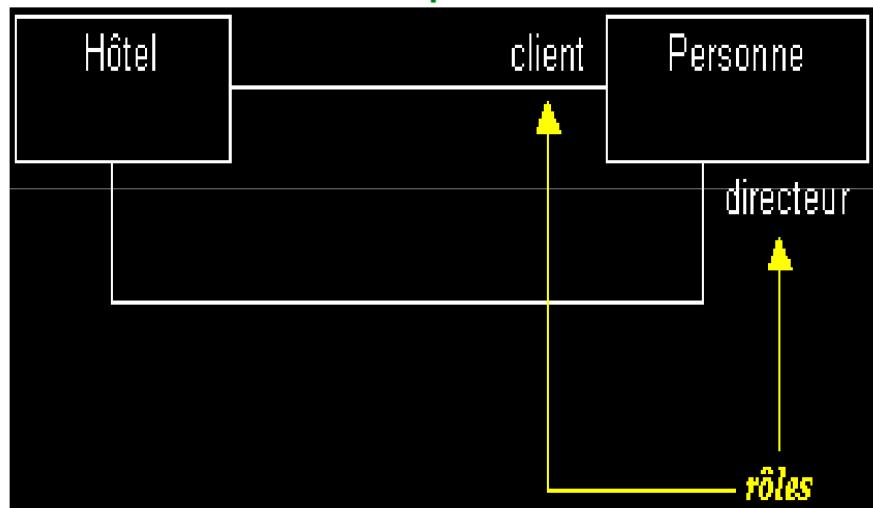
## Concepts clés de la modélisation orientée objet

### Liens entre les classes

#### Association

- **Rôle :**

Un rôle spécifie la fonction d'une classe pour une association donnée  
(indispensable pour les associations réflexives)



# CONCEPTS GENERAUX

## Concepts clés de la modélisation orientée objet

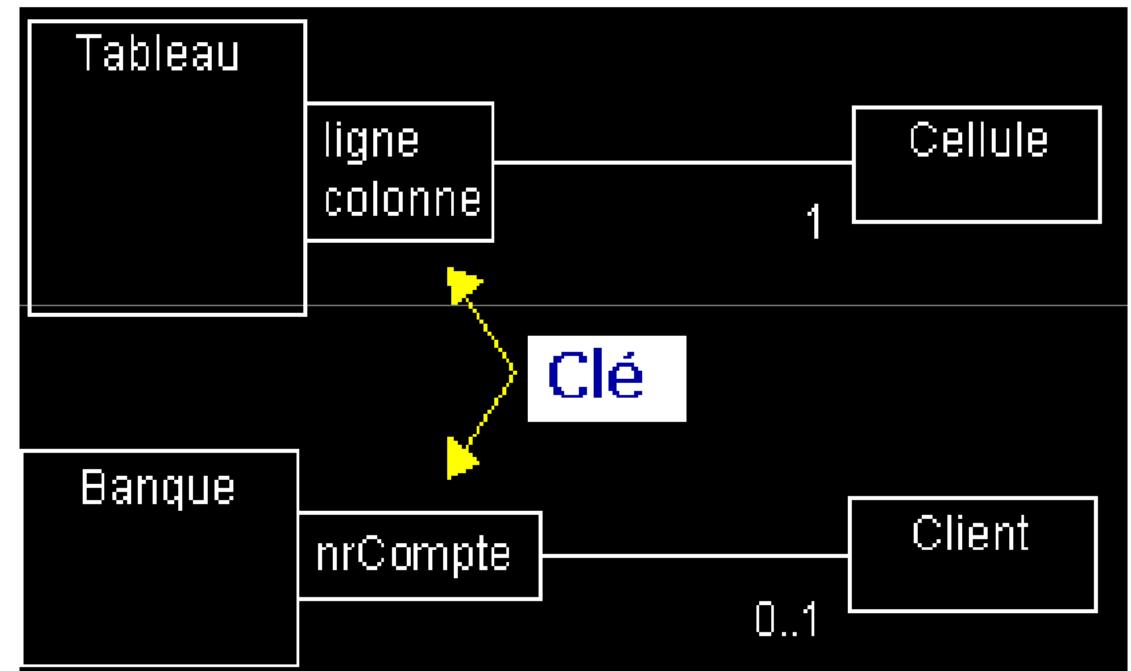
### Liens entre les classes

#### Association

- **Qualification :**

Permet de sélectionner un sous - ensemble d'objets, parmi l'ensemble des objets qui participent à une association.

La restriction de l'association est définie par une clé, qui permet de sélectionner les objets ciblés.



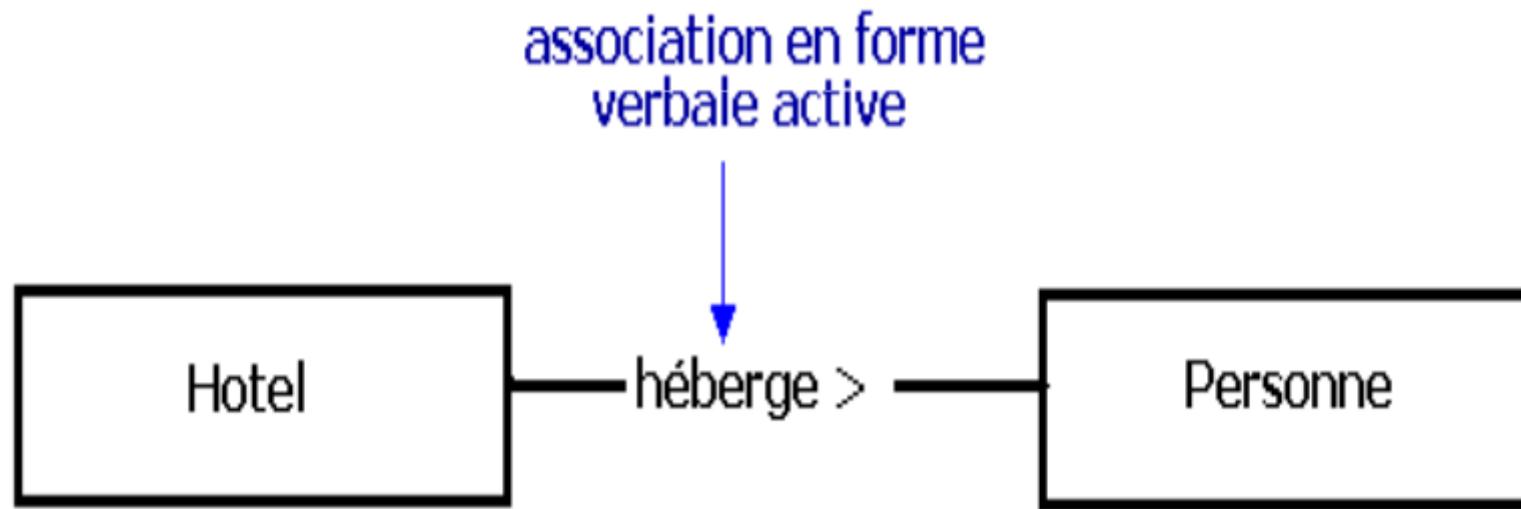
# CONCEPTS GENERAUX

**Concepts clés de la modélisation orientée objet**

**Liens entre les classes**

**Association en forme verbale active**

Précise le sens de lecture principal d'une association.

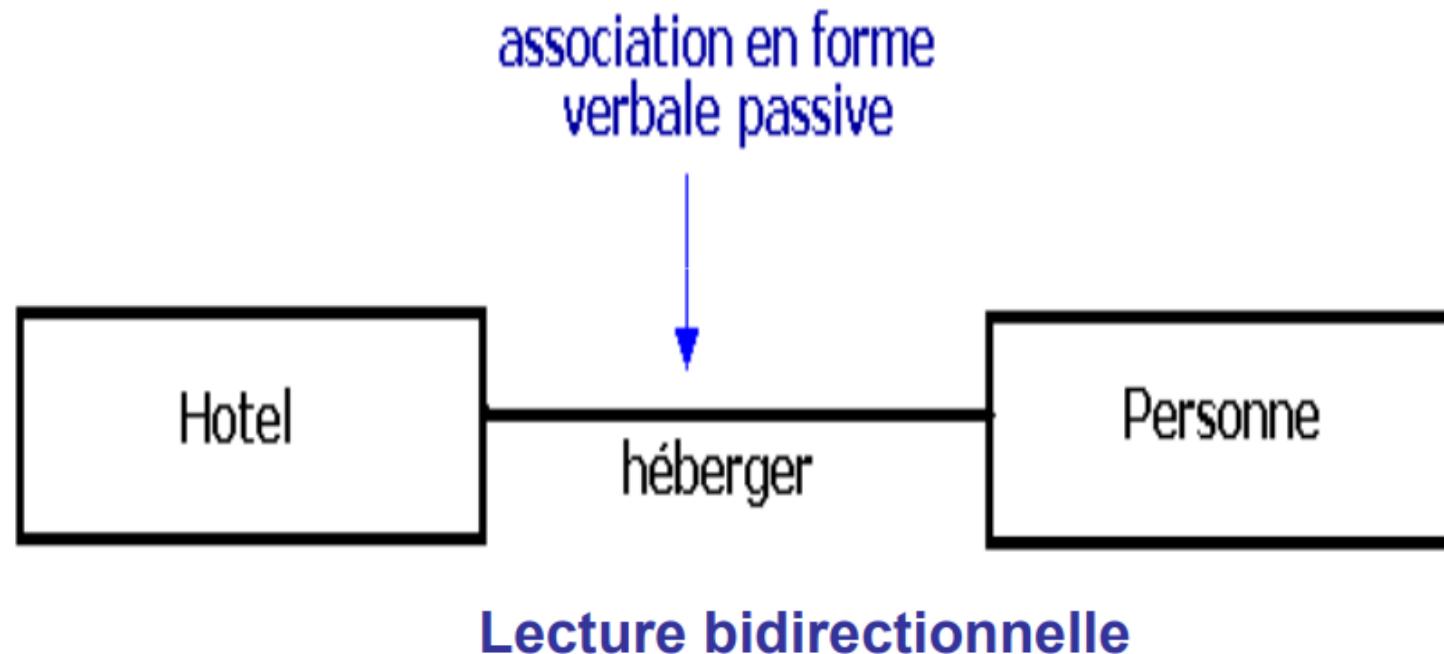


# CONCEPTS GENERAUX

**Concepts clés de la modélisation orientée objet**

**Liens entre les classes**

**Association en forme verbale passive**



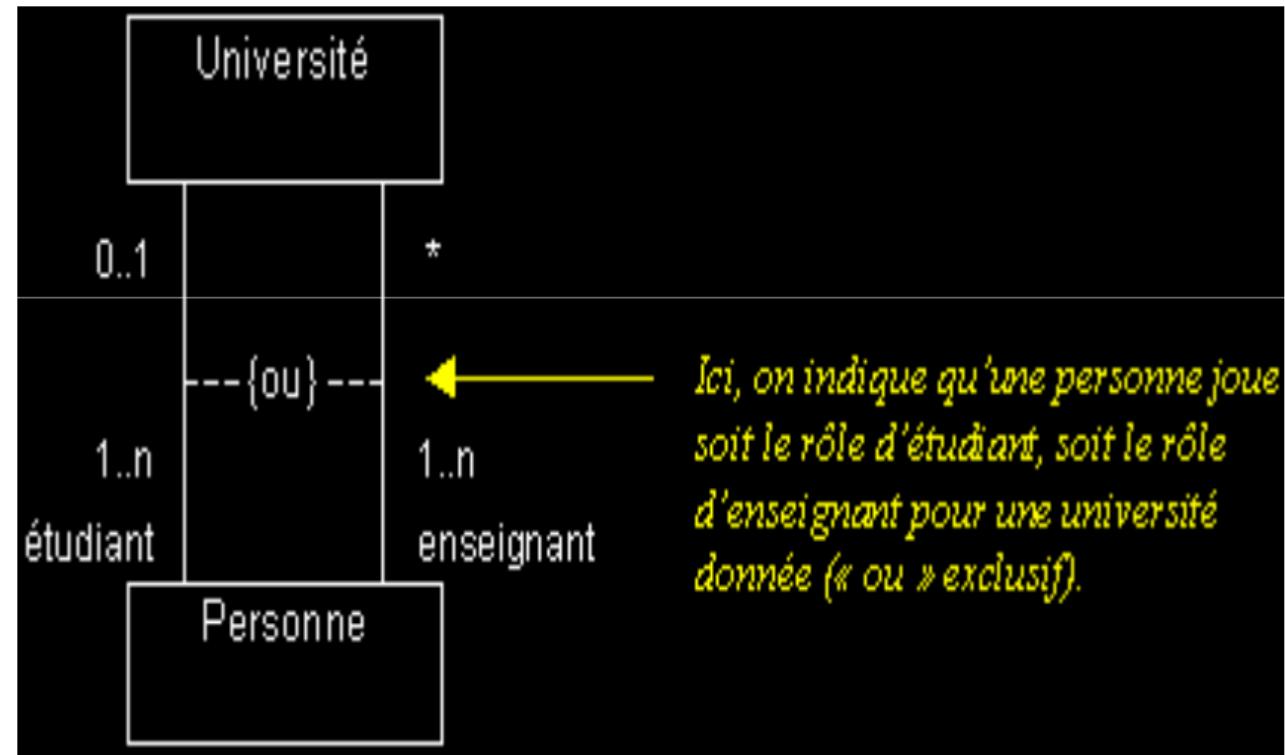
# CONCEPTS GENERAUX

## Concepts clés de la modélisation orientée objet

### Liens entre les classes

### Contrainte d'une association

Les contraintes sont des expressions qui précisent le rôle ou la portée d'un élément de modélisation (elles permettent d'étendre ou préciser sa sémantique).



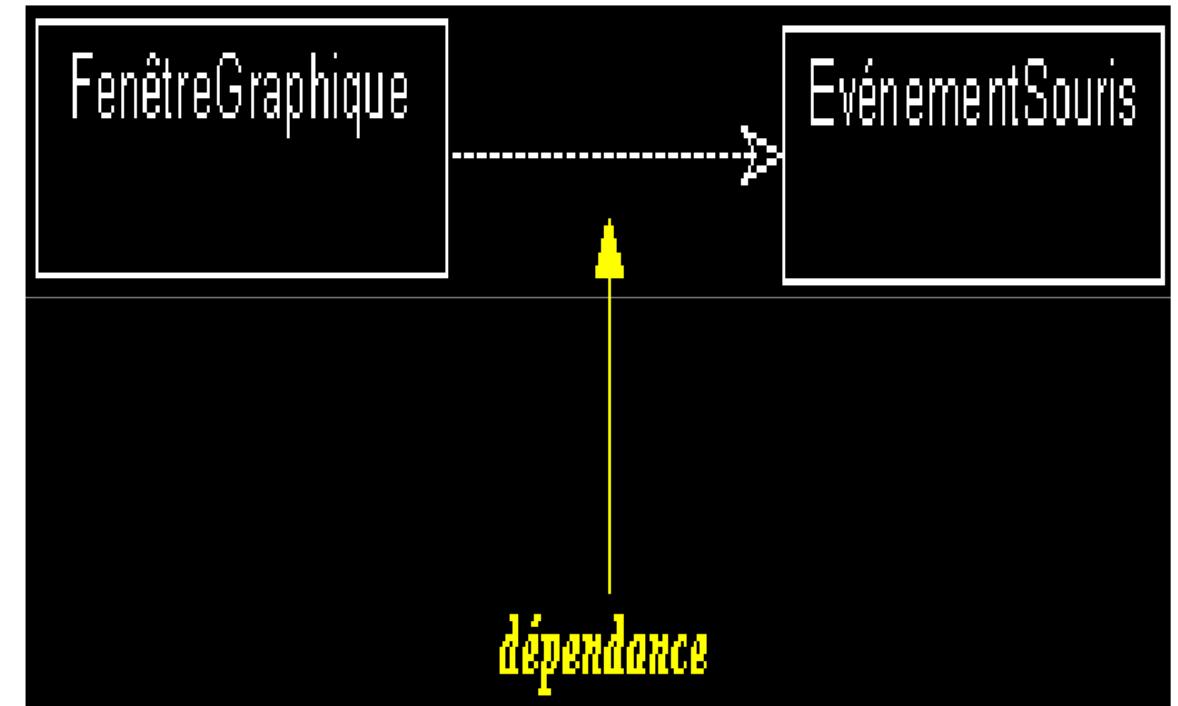
# CONCEPTS GENERAUX

## Concepts clés de la modélisation orientée objet

### Liens entre les classes

#### Relation de dépendance

Relation d'utilisation unidirectionnelle et d'obsolescence (une modification de l'élément dont on dépend, peut nécessiter une mise à jour de l'élément dépendant).



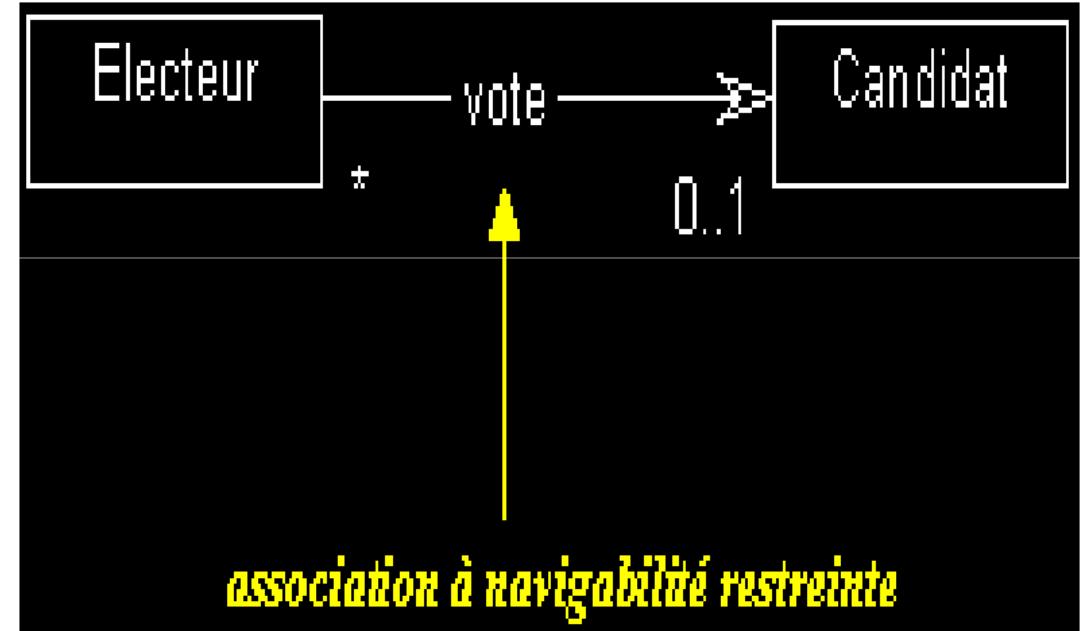
# CONCEPTS GENERAUX

## Concepts clés de la modélisation orientée objet

### Liens entre les classes

#### Association à navigabilité restreinte

- Par défaut, une association est navigable dans les deux sens. La réduction de la portée de l'association peut être réalisée en phase d'implémentation.
- Elle peut aussi être exprimée dans un modèle pour indiquer que les instances d'une classe ne "connaissent" pas les instances d'une autre classe .



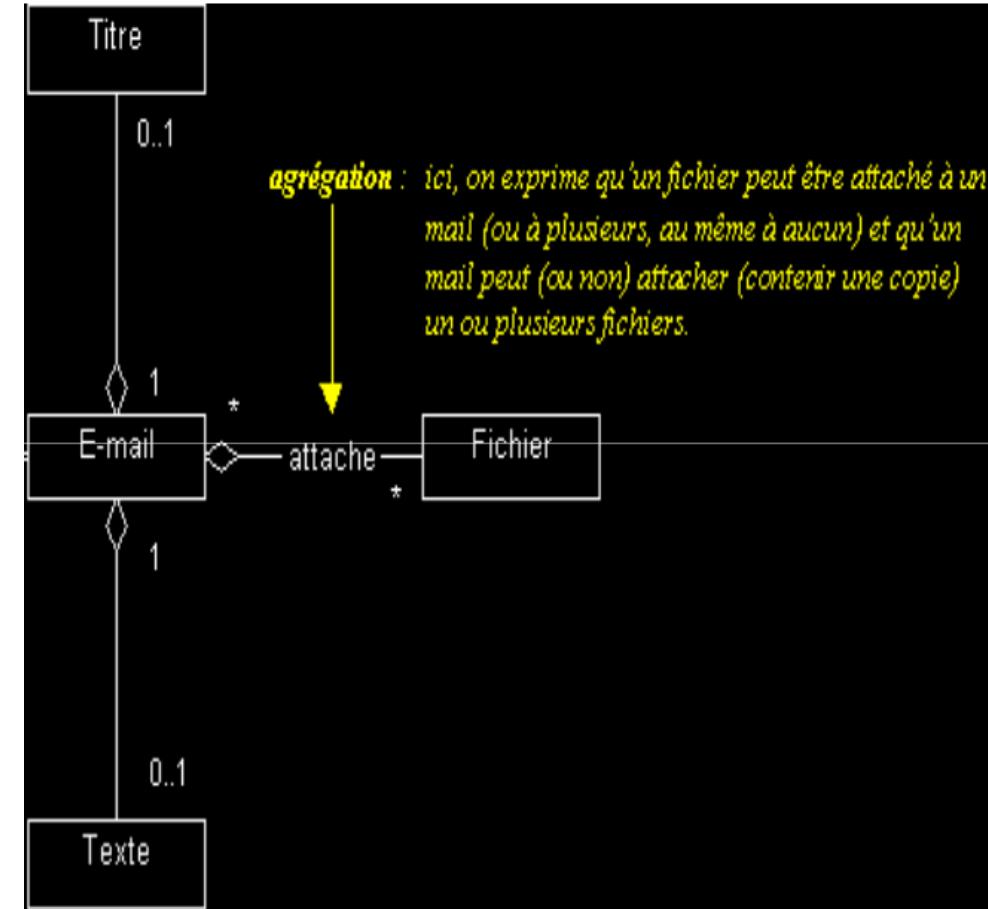
# CONCEPTS GENERAUX

## Concepts clés de la modélisation orientée objet

### Liens entre les classes

#### Agrégation

- Il s'agit d'une relation entre deux classes, spécifiant que les objets d'une classe sont des composants de l'autre classe.
- Il s'agit d'une relation entre deux classes, spécifiant que les objets d'une classe sont des composants de l'autre classe.
- L'agrégation permet d'assembler des objets de base, afin de construire des objets plus complexes



# CONCEPTS GENERAUX

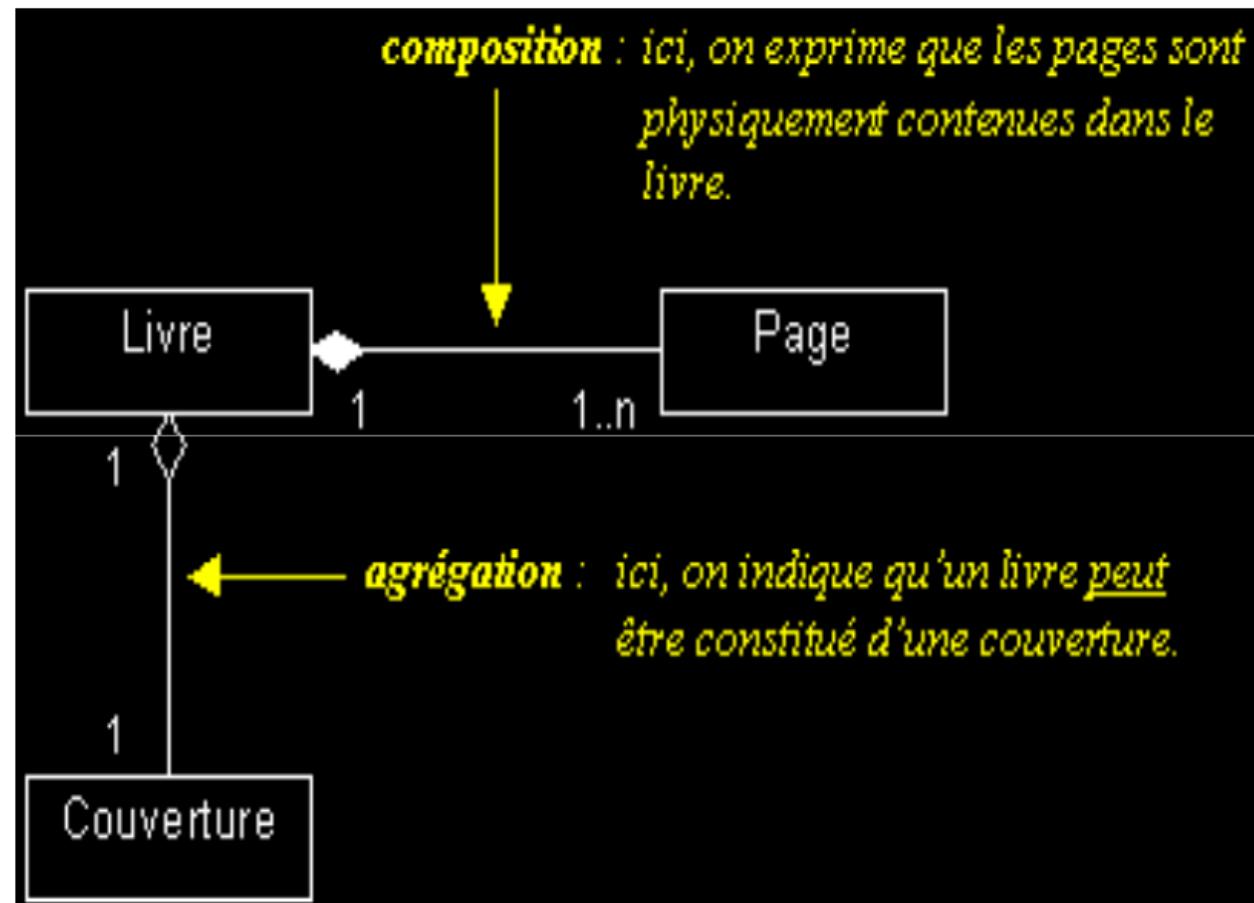
## Concepts clés de la modélisation orientée objet

### Liens entre les classes

#### Composition

La composition est une agrégation forte (agrégation par valeur).

- Les cycles de vies des éléments (les composants) et de l'agrégat (le composé) sont liés : si l'agrégat est détruit (ou copié), ses composants le sont aussi.

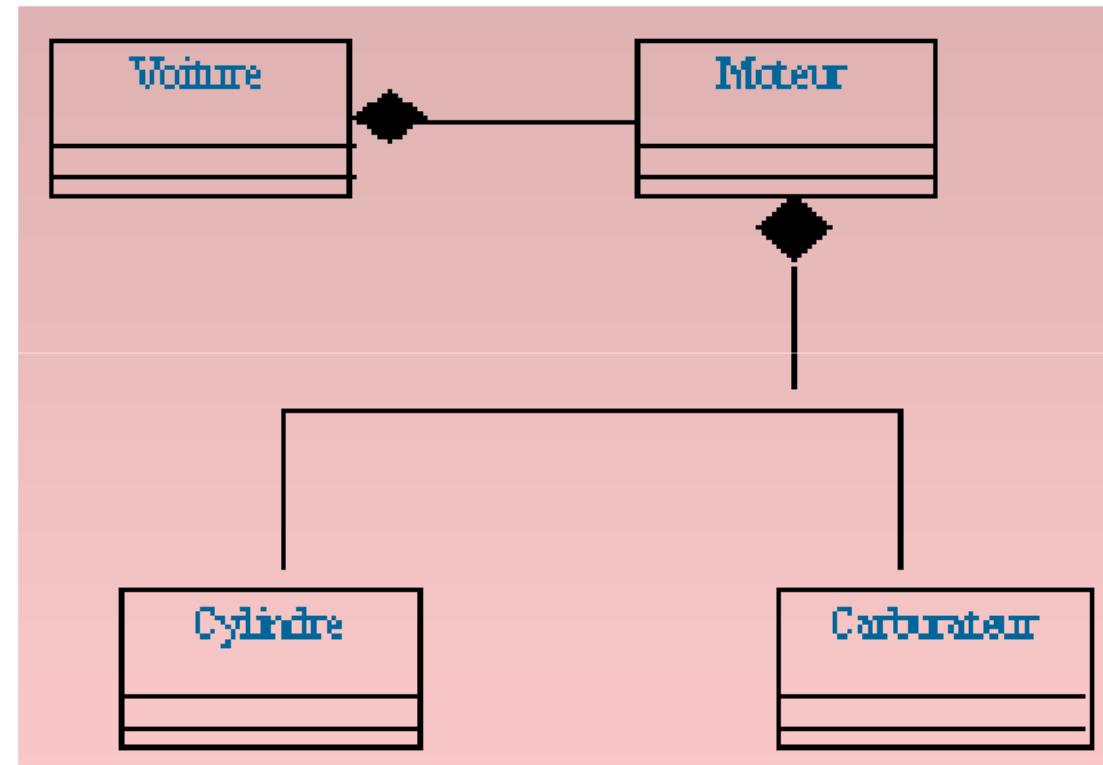


# CONCEPTS GENERAUX

Concepts clés de la modélisation orientée objet

Liens entre les classes

Composition



# CONCEPTS GENERAUX

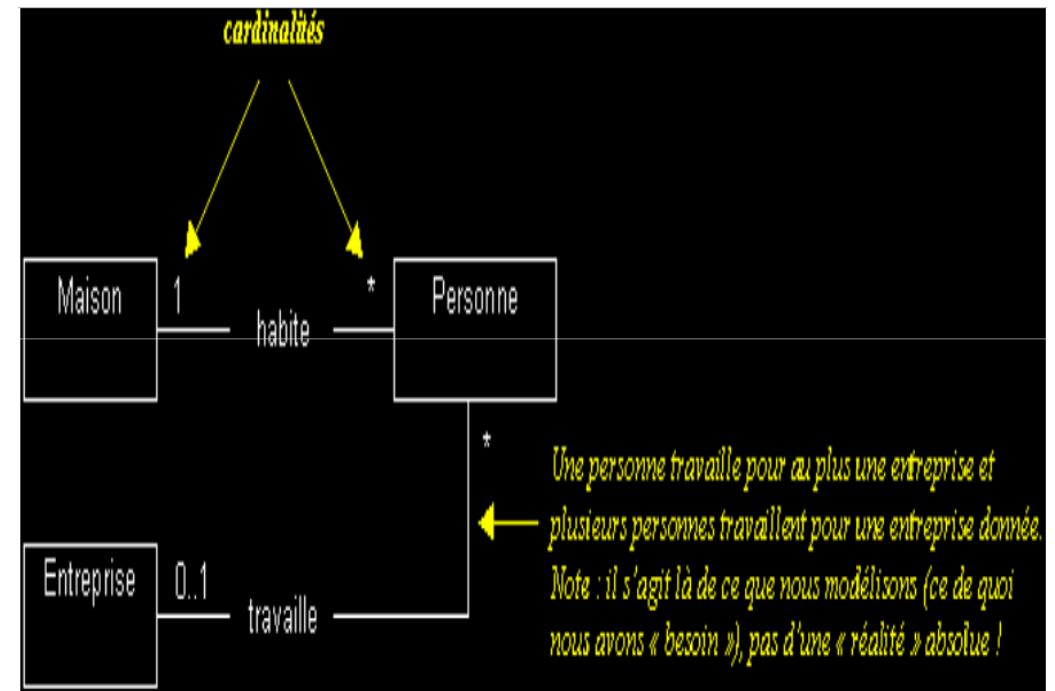
## Concepts clés de la modélisation orientée objet

### Liens entre les classes

### Multiplicité ou cardinalité

Précise le nombre d'instances qui participent à une relation.

- **n : exactement "n" (n, entier naturel > 0)**
- **n..m : de "n" à "m" (entiers naturels m > n)**
- **\* : plusieurs (équivalent à "0..n" et "0..\*")**
- **n..\* : "n" ou plus (n, entier naturel ou variable)**

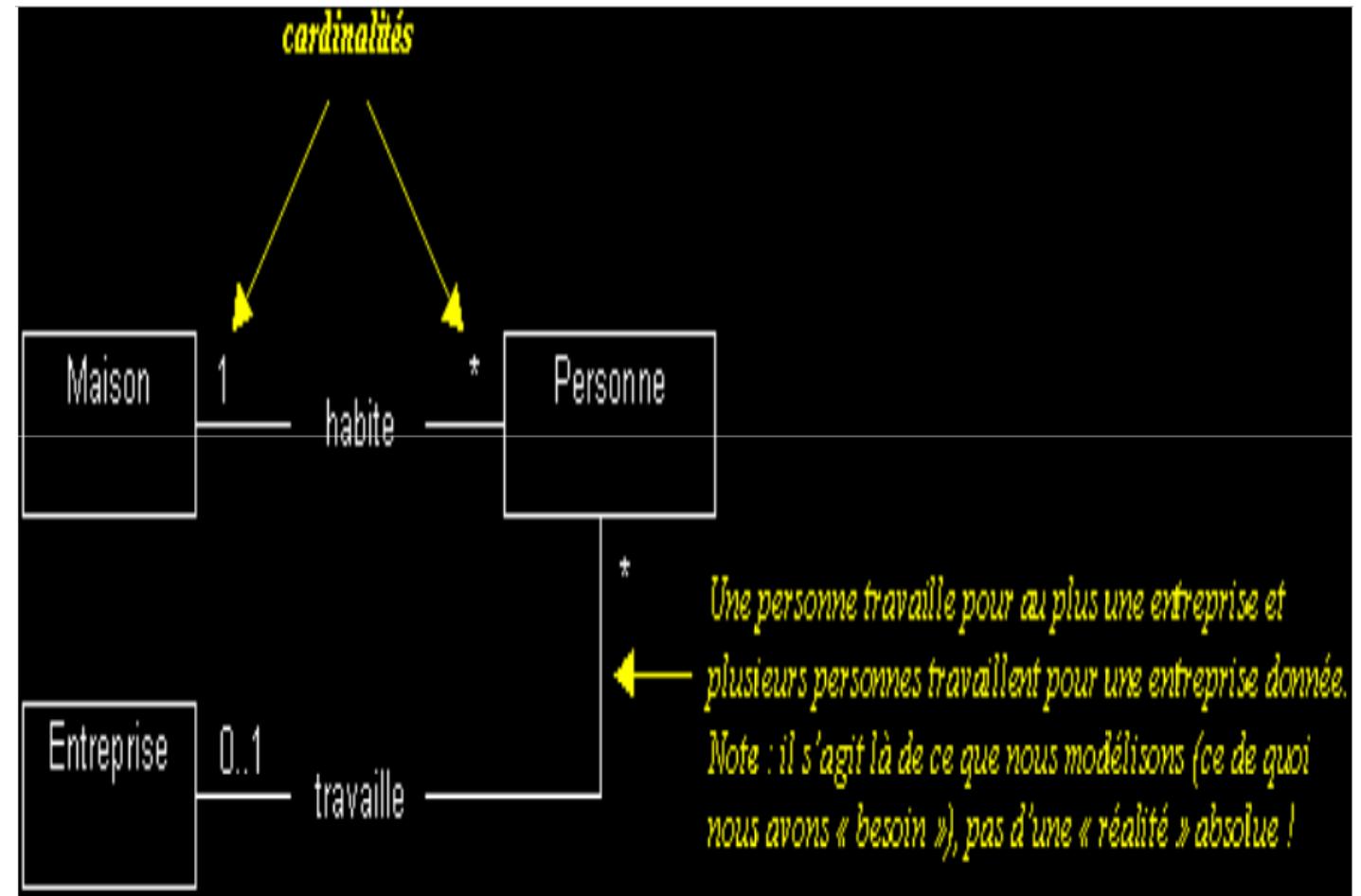


# CONCEPTS GENERAUX

## Concepts clés de la modélisation orientée objet

### Liens entre les classes

#### Multiplicité ou cardinalité



# **CONCEPTS GENERAUX**

## **Concepts clés de la modélisation orientée objet**

### **Liens entre les classes**

### **Héritage**

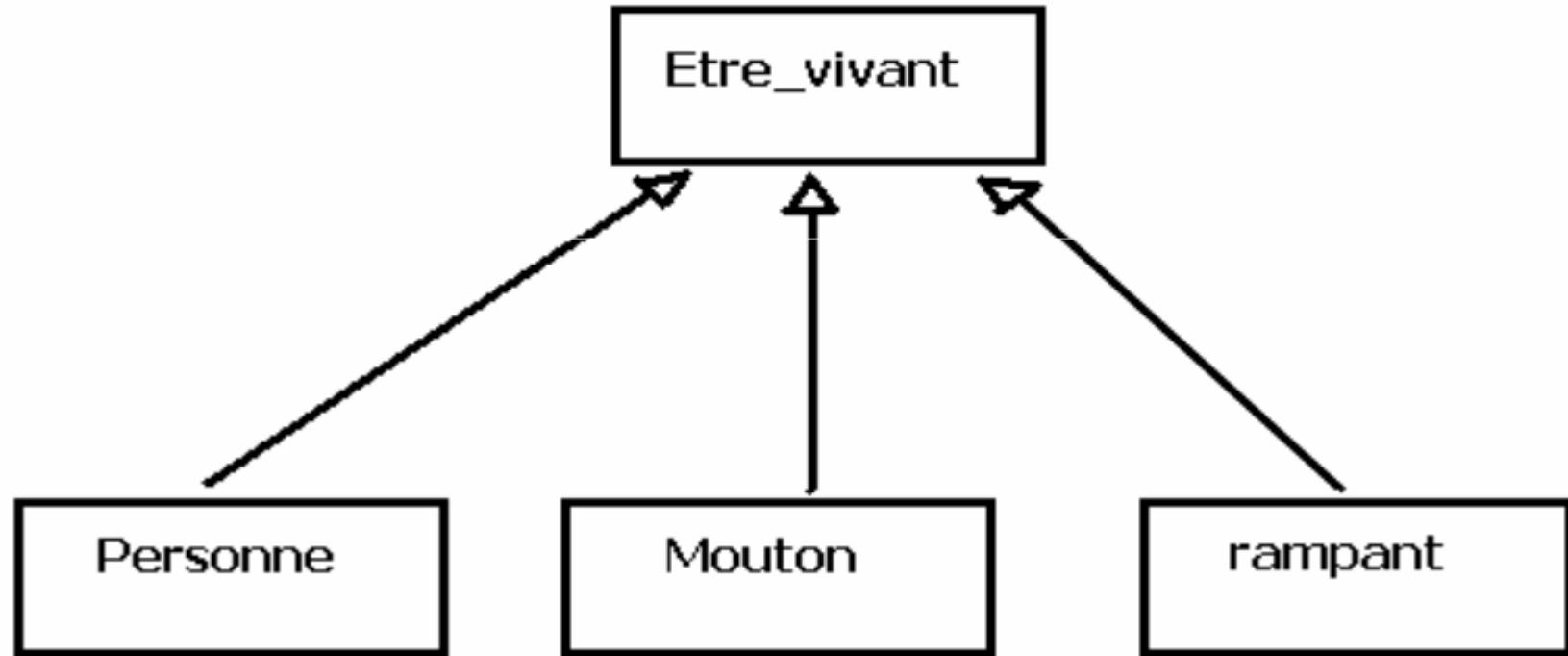
- L'héritage est un mécanisme de transmission des propriétés d'une classe (ses attributs et méthodes) vers une sous-classe.
- L'héritage peut être simple ou multiple.
- L'héritage évite la duplication et encourage la réutilisation.

# CONCEPTS GENERAUX

Concepts clés de la modélisation orientée objet

Liens entre les classes

Héritage simple

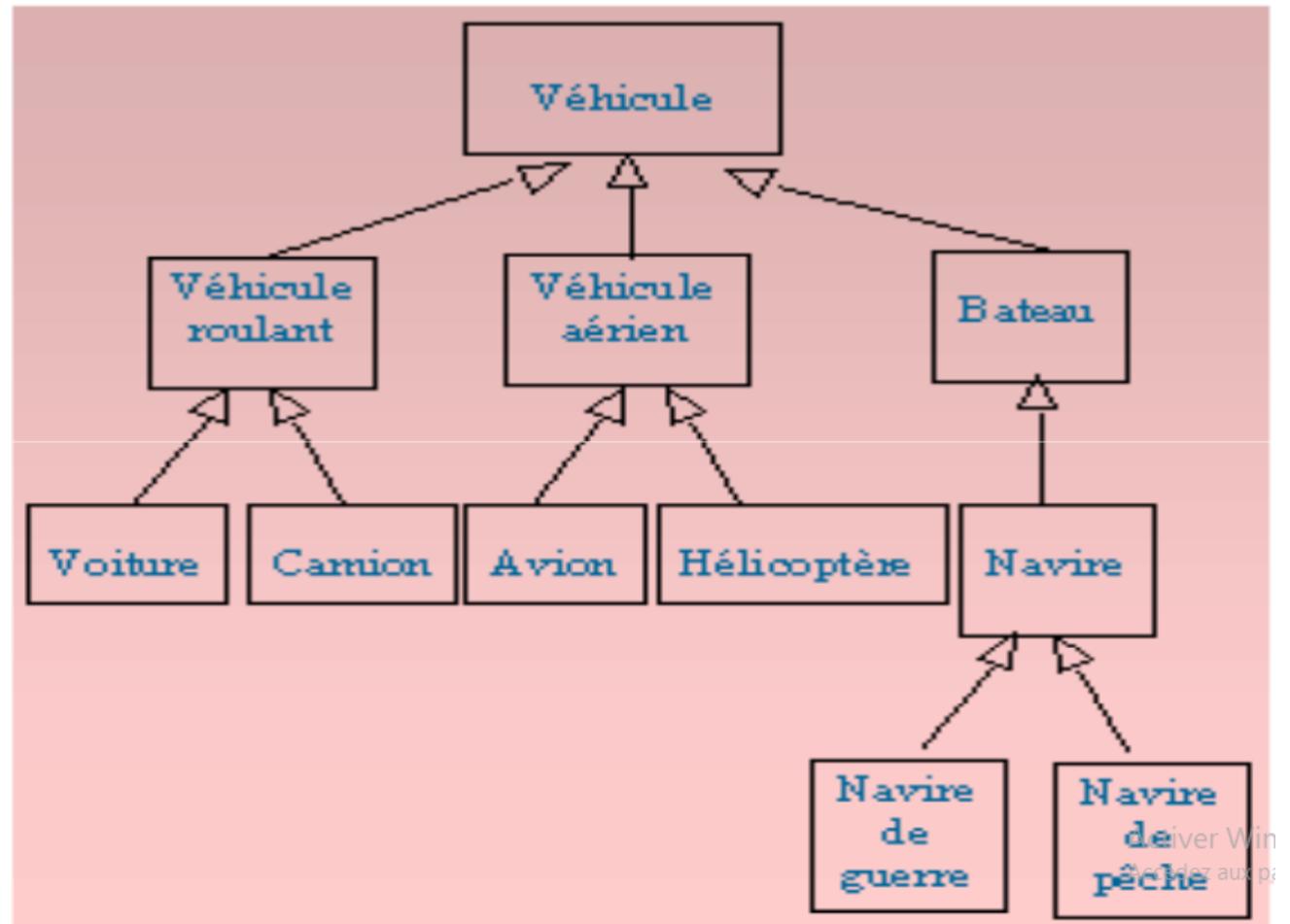


# CONCEPTS GENERAUX

## Concepts clés de la modélisation orientée objet

### Liens entre les classes

#### Héritage simple

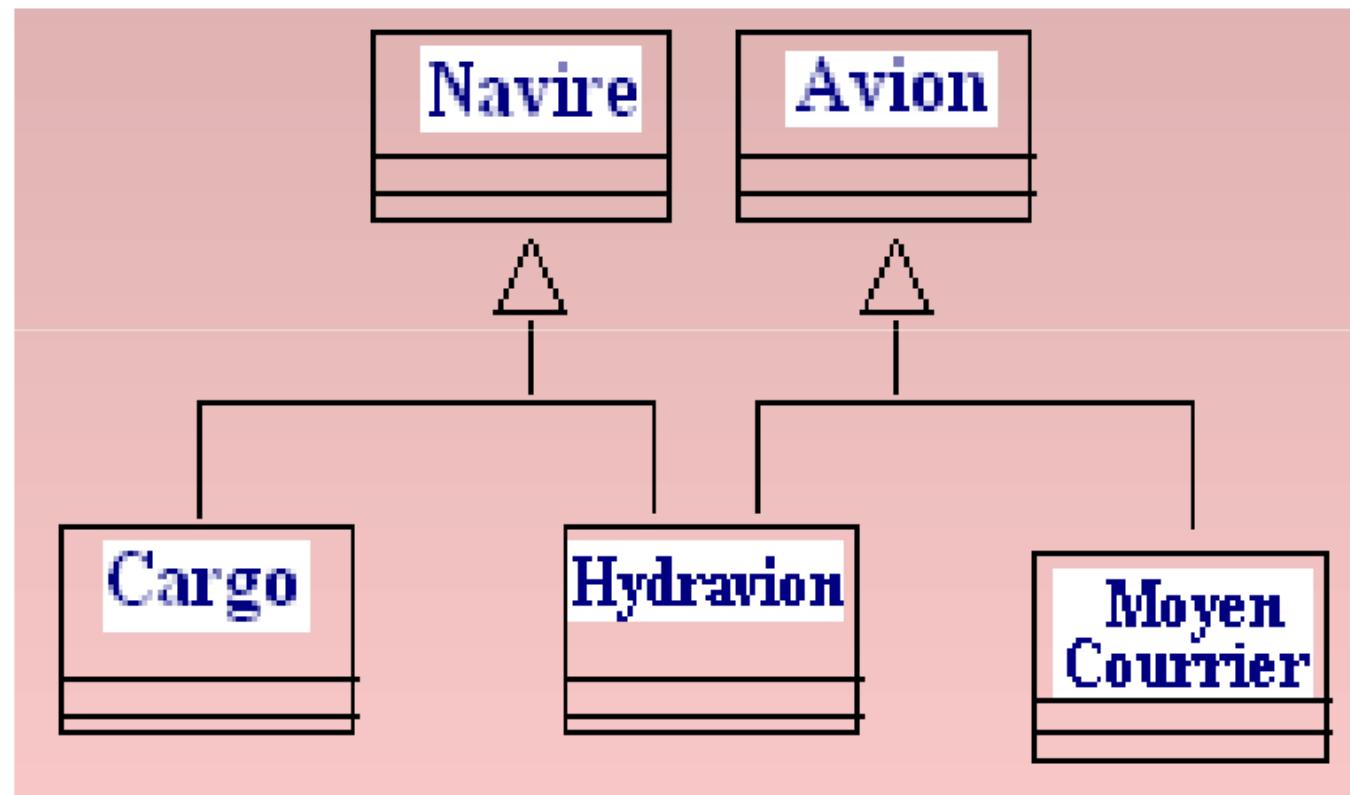


# CONCEPTS GENERAUX

Concepts clés de la modélisation orientée objet

Liens entre les classes

Héritage multiple



# CONCEPTS GENERAUX

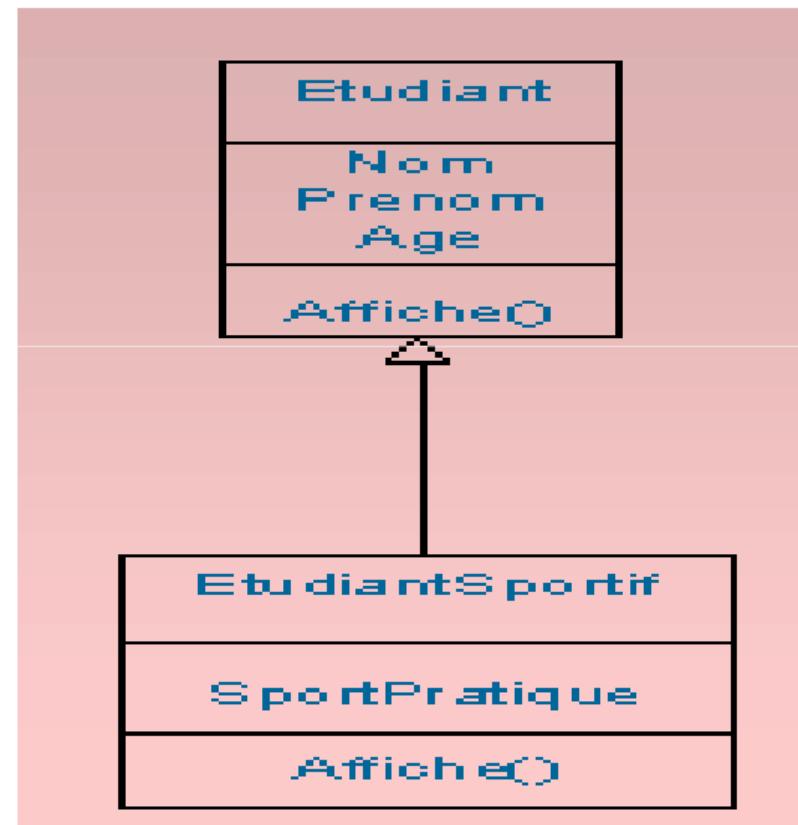
## Concepts clés de la modélisation orientée objet

### Liens entre les classes

### Héritage et redéfinition

- **Par réutilisation**

Spécialiser la méthode héritée en mettant une implémentation qui utilise celle héritée.



# CONCEPTS GENERAUX

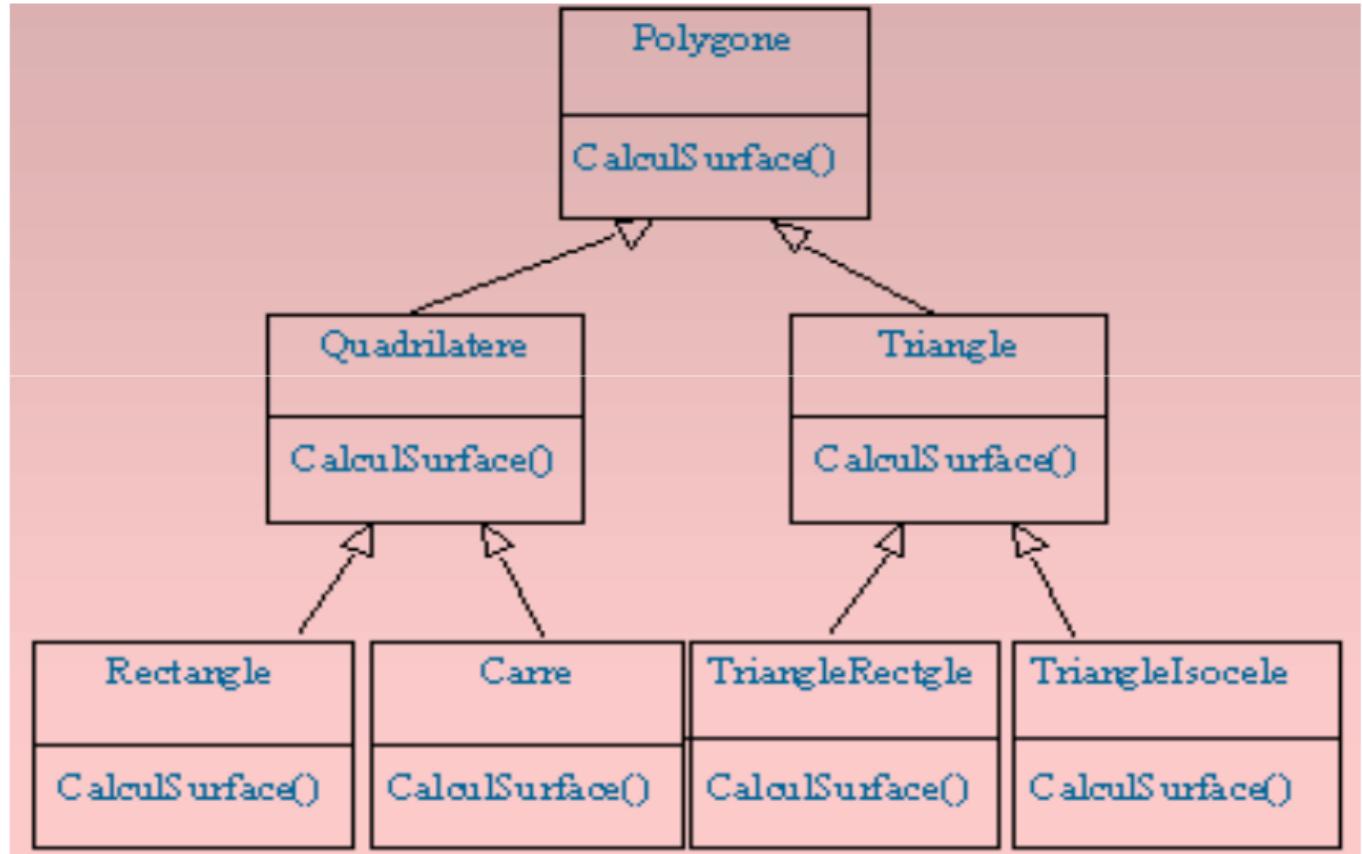
## Concepts clés de la modélisation orientée objet

### Liens entre les classes

### Héritage et redéfinition

- **Par substitution**

On remplace complètement la méthode héritée par une nouvelle implémentation.



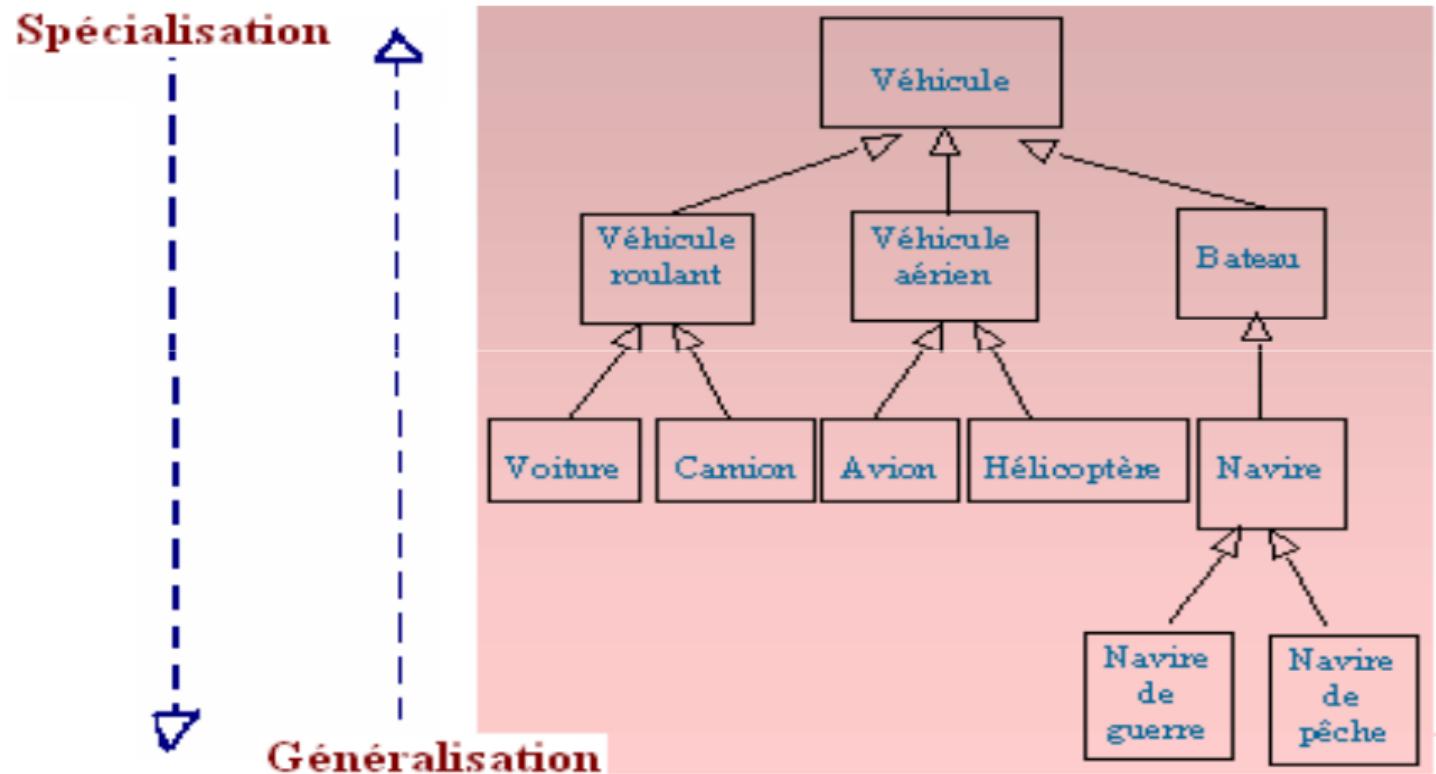
# CONCEPTS GENERAUX

## Concepts clés de la modélisation orientée objet

### Liens entre les classes

#### Héritage

La spécialisation et la généralisation permettent de construire des hiérarchies de classes



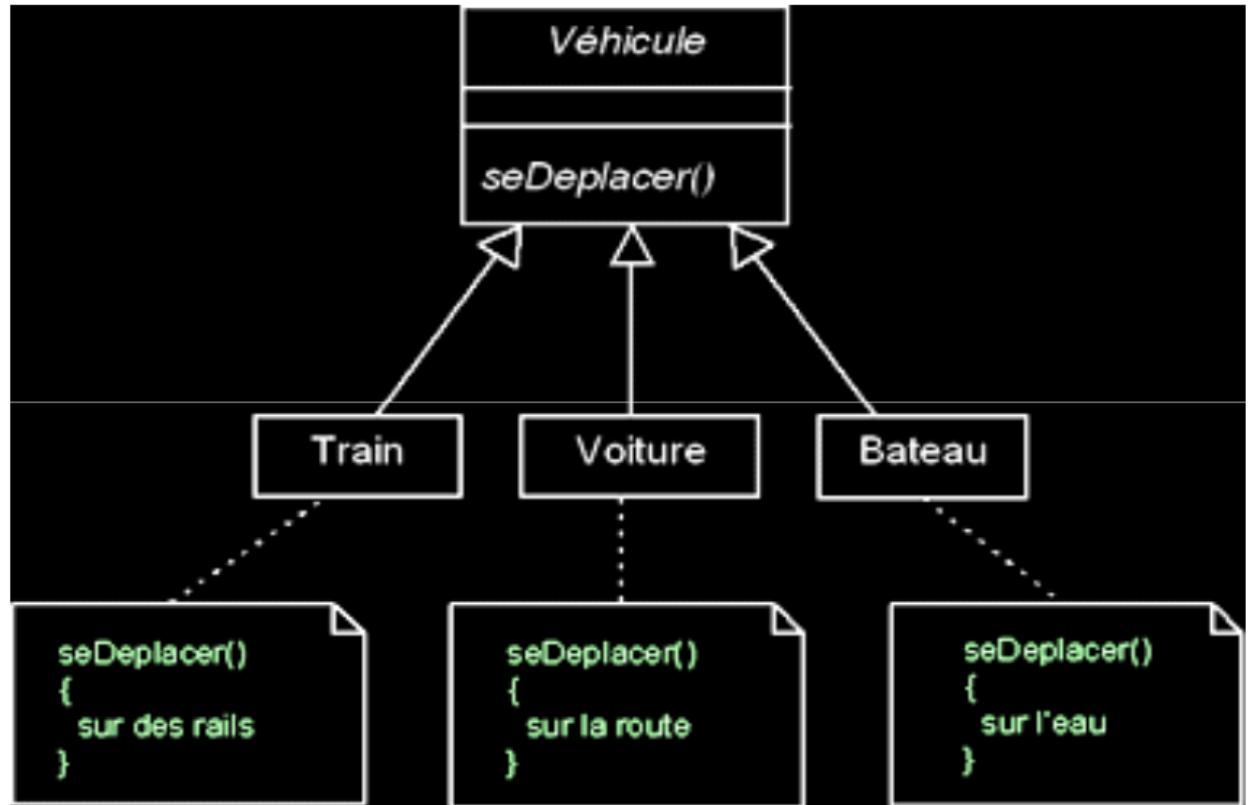
# CONCEPTS GENERAUX

## Concepts clés de la modélisation orientée objet

### Liens entre les classes

### Polymorphisme

- Le polymorphisme représente la faculté d'une méthode à pouvoir s'appliquer à des objets de classes différentes, qui héritent d'une même classe de base (super - classe).
- **Le polymorphisme augmente la généricité du code.**



# EXERCICE D'APPLICATION (A RENDRE PAR GITHUB AVANT LE 02/07/21)

Partie A :

Un cercle est défini par :

- Un point qui représente son centre
- Son rayon r

On peut créer un cercle en précisant son centre et son rayon.

Ici, nous commençons tout d'abord par définir la classe Point définie par :

- Les attributs : x et y de type réel
- Un constructeur qui permet de définir les valeurs de x et de y.
- Une méthode Afficher () qui affiche une chaîne de caractères POINT(x,y).

Les opérations que l'on souhaite exécuter sur un cercle sont :

- getPerimetre() : retourne le périmètre du cercle.
- getSurface() : retourne la surface du cercle.
- appartient (Point p) : retourne si un point p appartient ou non au cercle.
- Afficher () : Affiche une chaîne de caractères de type CERCLE(x,y,R)

Partie B : Un cylindre est tout d'abord un cercle en plus d'une hauteur.

- Définir la classe cylindre
- Attachez-lui une méthode getVolume() permettant de retourner le volume du cylindre.

Travail à faire :

- Faites les implémentations en Python

A déposer sur [mansour.diouf03@univ-thies.sn](mailto:mansour.diouf03@univ-thies.sn)