




# Modélisation avec UML

Modélisation avec UML1


## ***Modélisation avec UML***

© Robert Ogor[www.enst-bretagne.fr](http://www.enst-bretagne.fr)

Modélisation avec UML2

## ***Vue générale du cours***

- 1) **Introduction au langage de modélisation UML**
  - points de vue et diagrammes
  - cas d'utilisation, analyse, conception, implémentation
- 2) **Le diagramme des cas d'utilisations**
  - acteur
  - cas d'utilisation et scénario
- 3) **Notion de classes et objets et leur diagramme**
  - introduction aux classes, aux objets
  - notion de relation, de composition et d'héritage
  - recherche d'un diagramme de classes à partir du cahier des charges
- 4) **Modèle dynamique**
  - diagramme de séquences, de collaboration, d'état et d'activité
  - réalisation des cas d'utilisation par les diagrammes de séquences
  - réalisation des cas d'utilisation par les diagrammes de collaboration
- 5) **Conception**
  - diagramme de déploiement et de composants
- 6) **Le langage de contrainte OCL**

© Robert Ogor[www.enst-bretagne.fr](http://www.enst-bretagne.fr)

# Modélisation avec UML

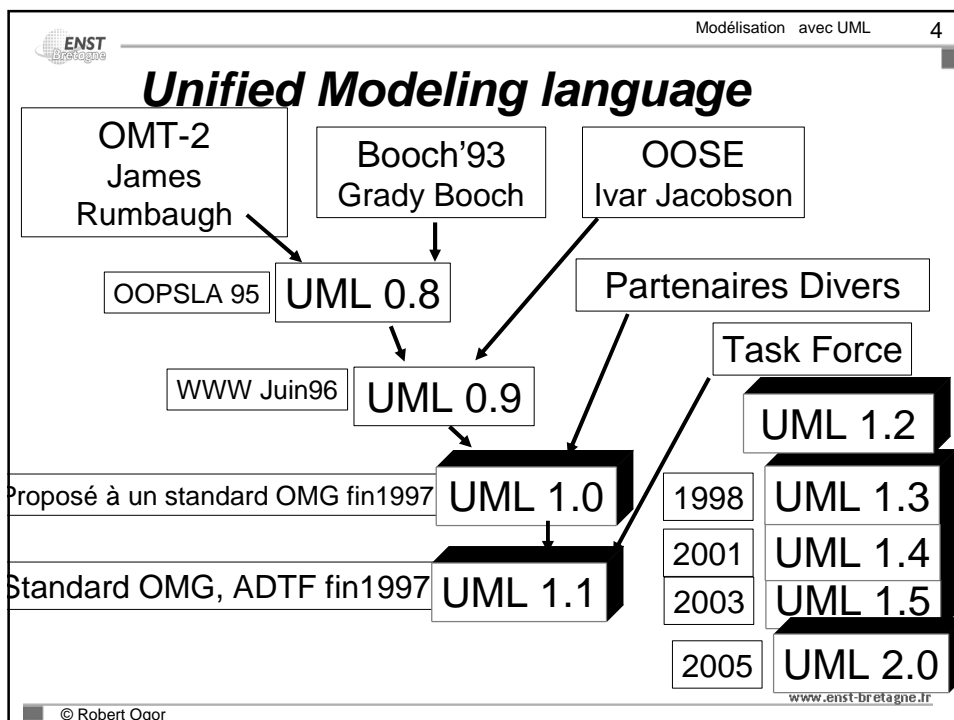
ENST Bretagne

Modélisation avec UML 3


## 1) Introduction au langage de modélisation UML

© Robert Ogor

www.enst-bretagne.fr




# Modélisation avec UML

Modélisation avec UML5

## ***Pourquoi modéliser***

- Un modèle est une simplification de la réalité qui permet de mieux comprendre le système à développer.
- Il permet
  - De visualiser le système comme il est ou comme il devrait l'être.
  - De valider le modèle vis à vis des clients
  - De spécifier les structures de données et le comportement du système.
  - De fournir un guide pour la construction du système.
  - De documenter le système et les décisions prises.

© Robert Ogor[www.enst-bretagne.fr](http://www.enst-bretagne.fr)


Modélisation avec UML6

## ***Les principes de la modélisation***

- 1) Le modèle doit être connecté au monde réel
- 2) Un modèle peut être exprimé avec différents niveaux de précision
- 3) Un simple modèle n'est pas suffisant, il y a plusieurs façons de voir un système.
  - plan de masse
  - vue de face, de côté, ...
  - plan des niveaux
  - plan électrique
  - plan de plomberie
  - plan des calculs de construction

© Robert Ogor[www.enst-bretagne.fr](http://www.enst-bretagne.fr)


# Modélisation avec UML

Modélisation avec UML7

## ***Qu'apporte la modélisation objet***

- Plus grande indépendance du modèle par rapport aux fonctionnalités demandées.
- Des fonctionnalités peuvent être rajoutées ou modifiées, le modèle objet ne change pas.
- Plus proche du monde réel.

© Robert Ogor[www.enst-bretagne.fr](http://www.enst-bretagne.fr)

Modélisation avec UML8

## ***Les objectifs d'UML***


- Représenter des systèmes entiers
- Etablir un couplage explicite entre les concepts et les artefacts exécutables
- Prendre en compte les facteurs d'échelle
- Créer un langage de modélisation utilisable à la fois par les humains et les machines

Recherche d'un langage commun :

- Utilisable par toutes les méthodes
- Adapté à toutes les phases du développement
- Compatible avec toutes les techniques de réalisation

© Robert Ogor[www.enst-bretagne.fr](http://www.enst-bretagne.fr)


# Modélisation avec UML

Modélisation avec UML9

## ***UML un langage***

- UML n'est pas une méthode
- UML est un langage de modélisation objet
- UML a été adopté par toutes les méthodes objet
- UML est dans le domaine public, c'est une norme

© Robert Ogor[www.enst-bretagne.fr](http://www.enst-bretagne.fr)


Modélisation avec UML10

## ***UML un langage pour***

- visualiser
  - chaque symbole graphique a une sémantique
- spécifier
  - de manière précise et complète, sans ambiguïté,
- construire
  - les classes, les relations SQL peuvent être générées automatiquement
- documenter
  - les différents diagrammes, notes, contraintes, exigences seront présentés dans un document.

© Robert Ogor[www.enst-bretagne.fr](http://www.enst-bretagne.fr)


# Modélisation avec UML

Modélisation avec UML11

## ***UML et les domaines d'utilisation***

- Systèmes d'information des entreprises
- Les Banques et les services financiers
- Télécommunications
- Transport
- Défense et aérospatiale
- Scientifique
- Applications distribuées par le WEB

© Robert Ogorwww.enst-bretagne.fr

Modélisation avec UML12

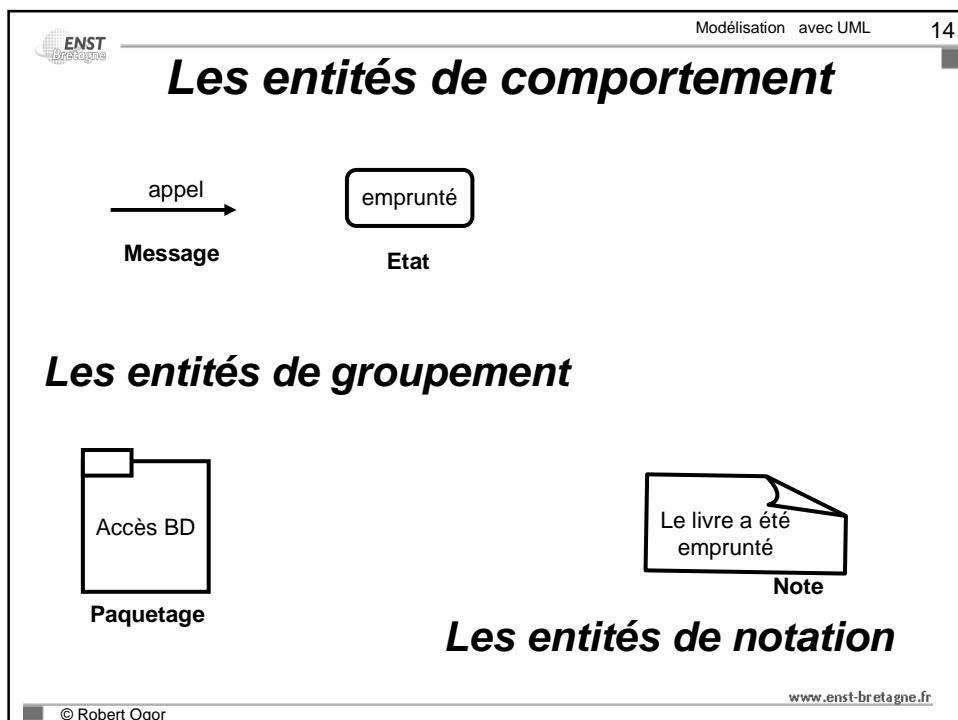
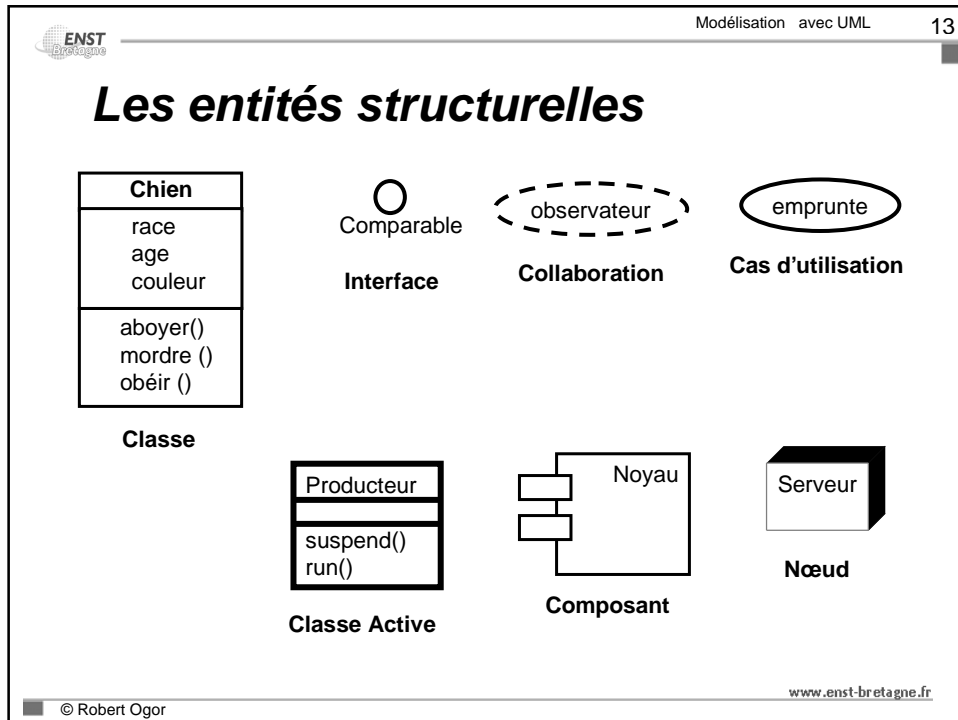
## ***Les trois éléments de base en UML***

- 1) les blocs de base pour construire
  - les entités utilisées → 


Entités structurales  
Entités de comportement  
Entités de regroupement  
Entité d'annotation
  - la notion de relation
  - les diagrammes
- 2) les règles à observer pour utiliser ces blocs de base
  - règles sémantiques
  - règles de présentation
- 3) les mécanismes communs
  - spécification
  - présentation
  - extension des modèles

© Robert Ogorwww.enst-bretagne.fr

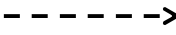

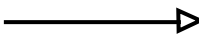
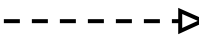
# Modélisation avec UML




# Modélisation avec UML

Modélisation avec UML15

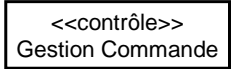
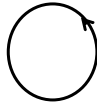

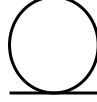
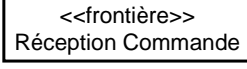
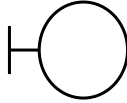
## Les relations

	dépendance
	association
	héritage
	réalisation

© Robert Ogor[www.enst-bretagne.fr](http://www.enst-bretagne.fr)

Modélisation avec UML16

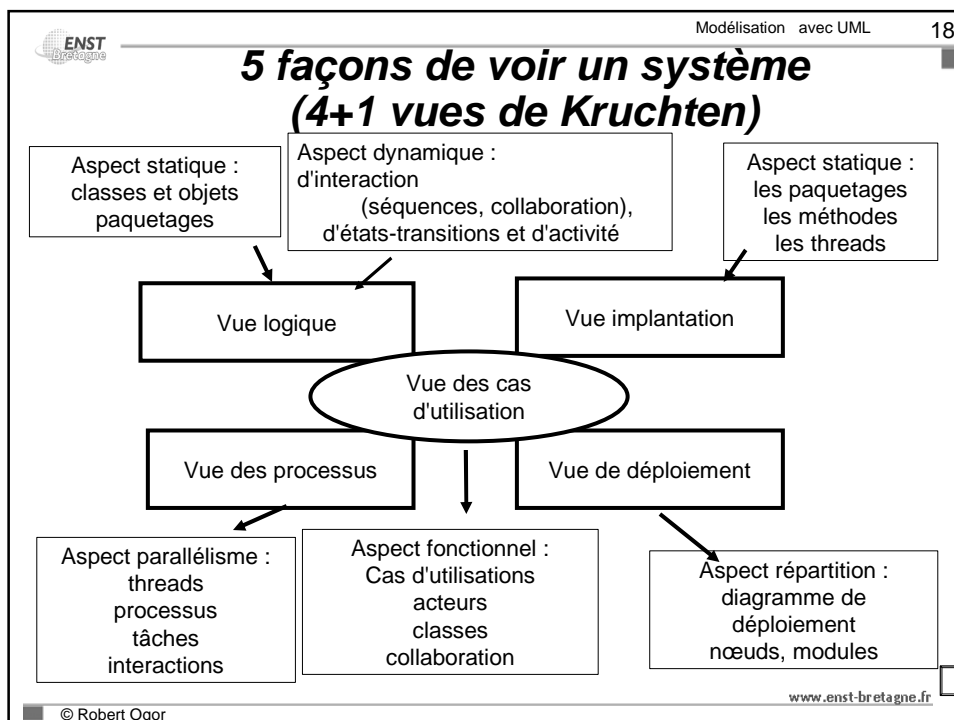
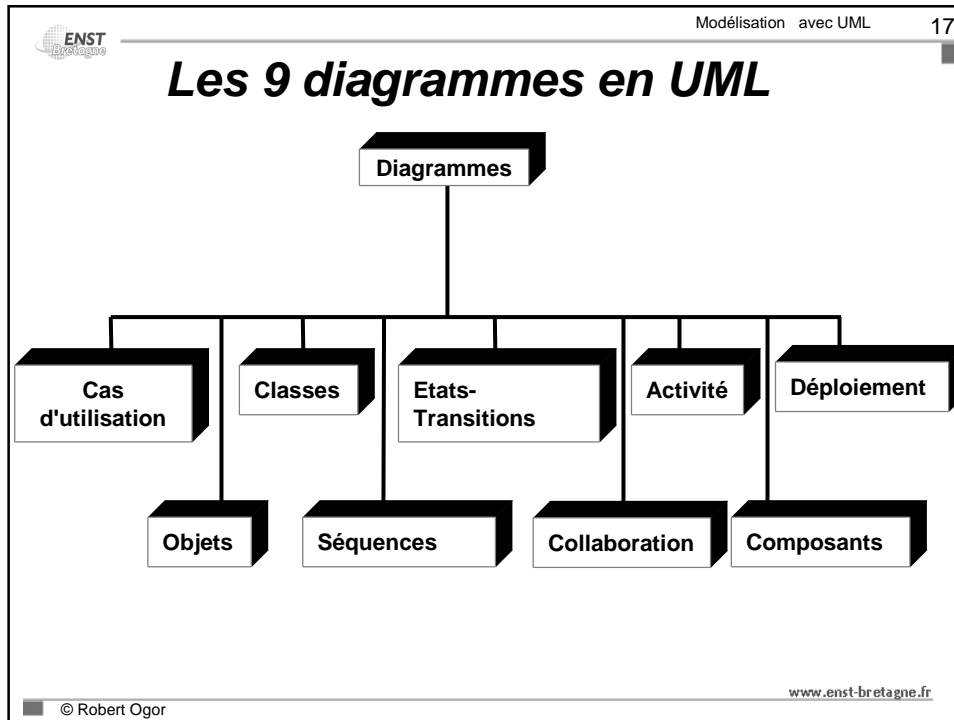
## Stéréotypes et icônes associées

	 gestion Commande
	 stockage Commande
	 réception Commande


© Robert Ogor[www.enst-bretagne.fr](http://www.enst-bretagne.fr)



# Modélisation avec UML





# Modélisation avec UML

Modélisation avec UML19

## ***Point de vue des cas d'utilisation***


- Vue du système par ses utilisateurs finaux
- Regroupe le comportement du système selon
  - Priorité: critique, important, accessoire
  - Risques à circonscrire
  - Options disponibles
  - Couverture de l'architecture
  - Autres objectifs tactiques et contraintes

  
© Robert Ogor[www.enst-bretagne.fr](http://www.enst-bretagne.fr)


Modélisation avec UML20

## ***Point de vue logique***

- Décomposition orientée-objet
  - Décomposition en objets et classes
  - Regroupement en paquetages.
  - Connexions par héritage, association, etc.
  - Accent sur l'abstraction, l'encapsulation, l'uniformité.
  - Réalisation des scénarios des cas d'utilisation.


  
© Robert Ogor[www.enst-bretagne.fr](http://www.enst-bretagne.fr)


# Modélisation avec UML

Modélisation avec UML21

## ***Point de vue processus***


- Décomposition en tâches et processus
- Regroupement des groupes de processus
- Communication
- Information sur les caractéristiques suivantes
  - Disponibilité, fiabilité
  - Intégrité, performance
  - Contrôle


  
© Robert Ogor[www.enst-bretagne.fr](http://www.enst-bretagne.fr)

Modélisation avec UML22

## ***Point de vue implantation***


- Décomposition en modules et niveaux
- Regroupement de modules en paquetages
- Organisation des sous-systèmes en niveaux pour :
  - Réduire le couplage et la visibilité
  - Augmenter la robustesse
- Information sur les caractéristiques suivantes :
  - Facilité de développement
  - Potentiel de réutilisation
  - Gestion de configuration

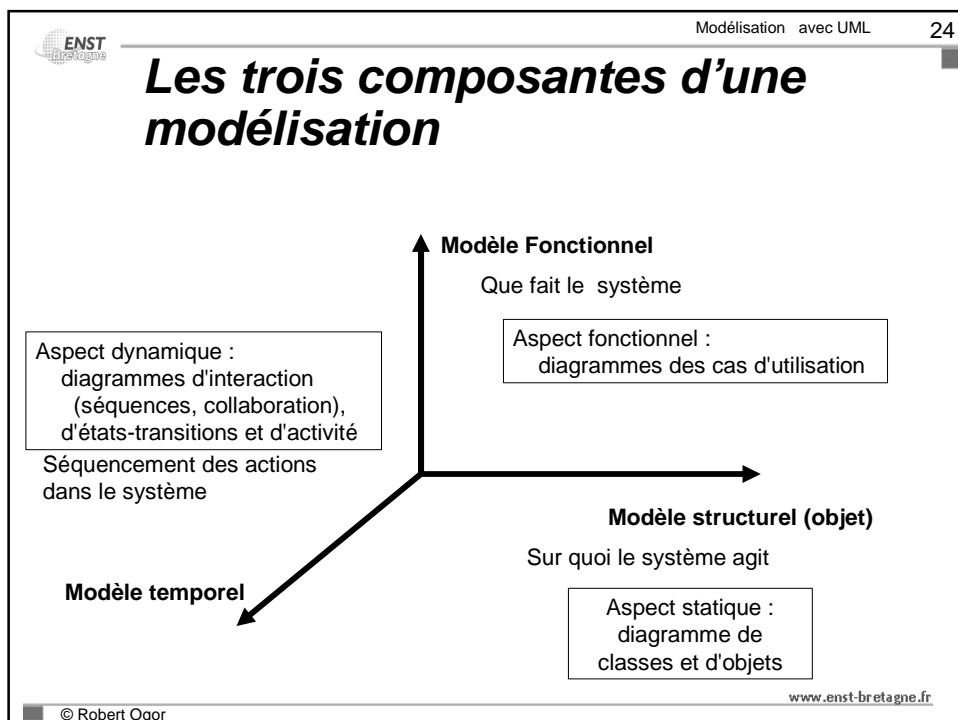
  
© Robert Ogor[www.enst-bretagne.fr](http://www.enst-bretagne.fr)

Modélisation avec UML23

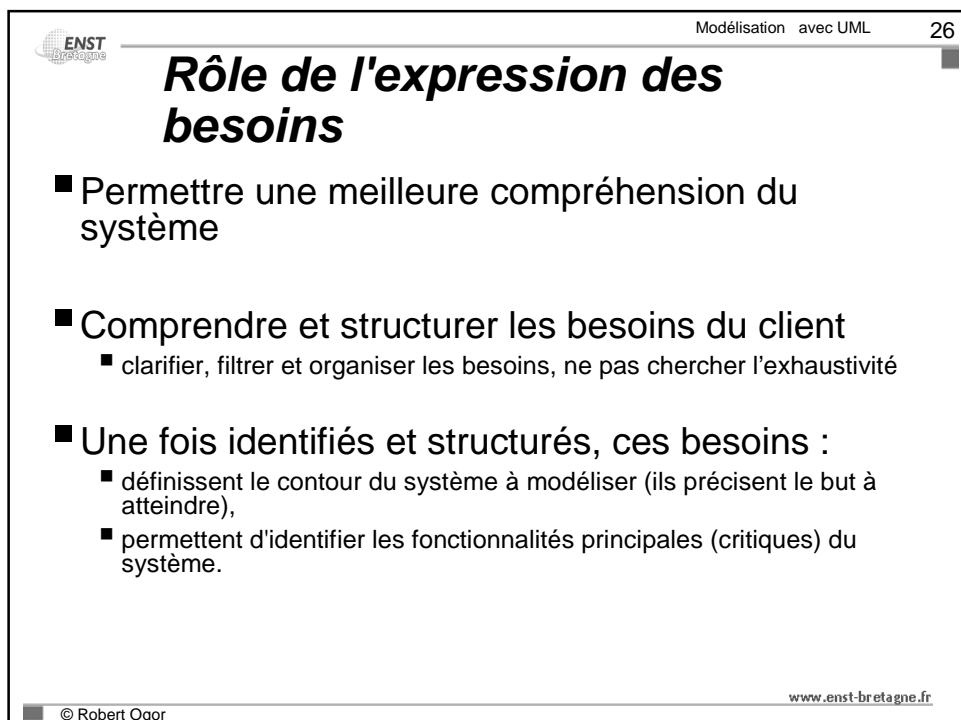
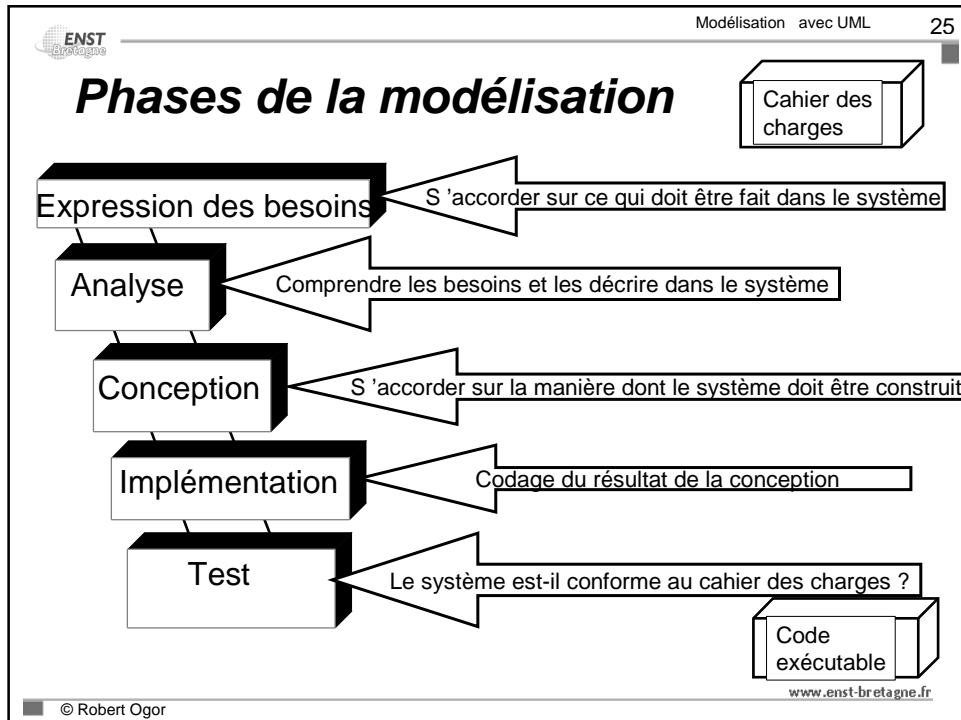
## *Point de vue déploiement*


- Décomposition en nœuds d'exécution
- Rôle d'un nœud
- Inter-connectivité, topologie
- Information sur les caractéristiques suivantes :
  - Performance, disponibilité
  - Installation, maintenance

  
© Robert Ogor  
[www.enst-bretagne.fr](http://www.enst-bretagne.fr)



# Modélisation avec UML




Modélisation avec UML27

## ***Expression des besoins***

- Comprendre le contexte du système en définissant un modèle du domaine et du métier
- Recenser les besoins fonctionnels et les définir par des cas d'utilisations
- Noter les contraintes, exigences non fonctionnelles.

Le modèle du domaine regroupe les objets qui se situent dans le contexte du système :


- entités existantes ou événements qui s'y produisent.

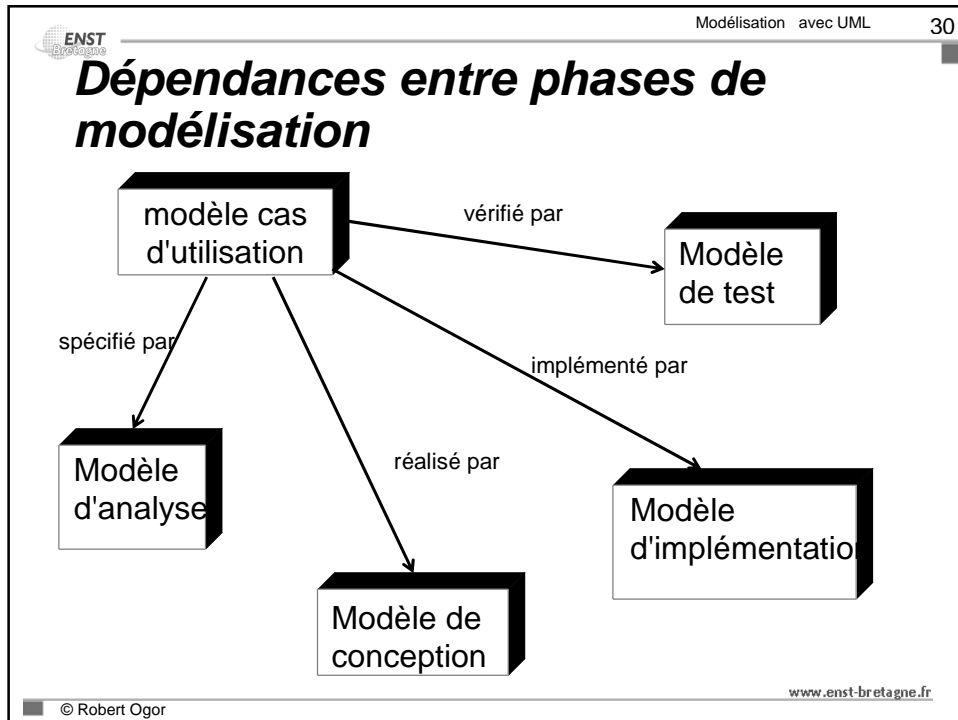
© Robert Ogor[www.enst-bretagne.fr](http://www.enst-bretagne.fr)

Modélisation avec UML28

## ***Exigences non fonctionnelles***

- Contraintes de concurrence
- Contraintes de temps de réponse
- Contraintes de distribution
- Contraintes de performance
- Contraintes de répartition

© Robert Ogor[www.enst-bretagne.fr](http://www.enst-bretagne.fr)



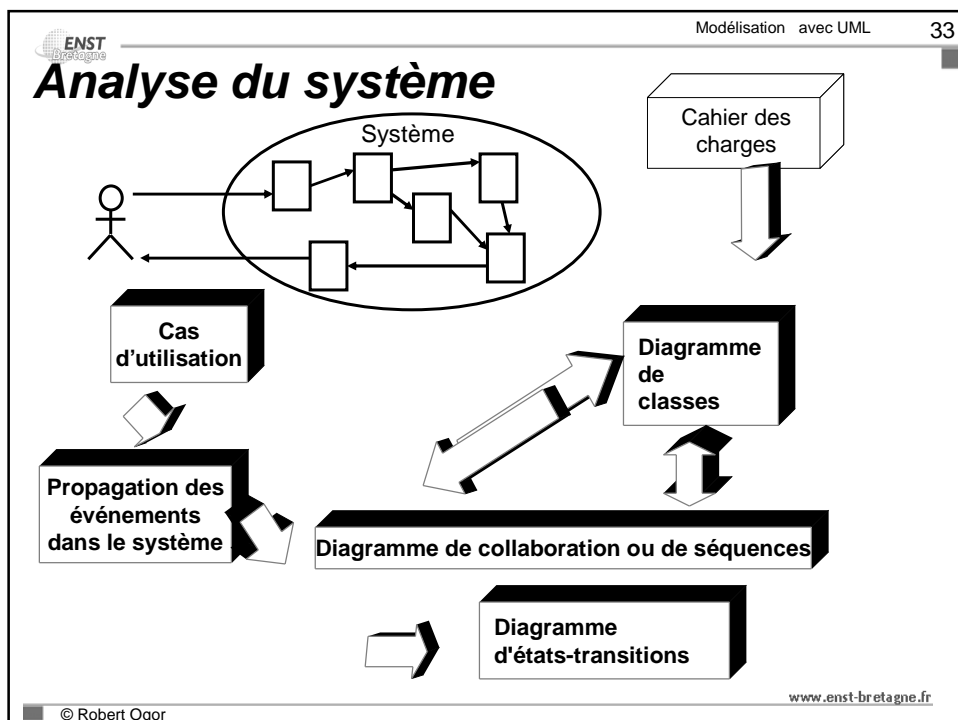
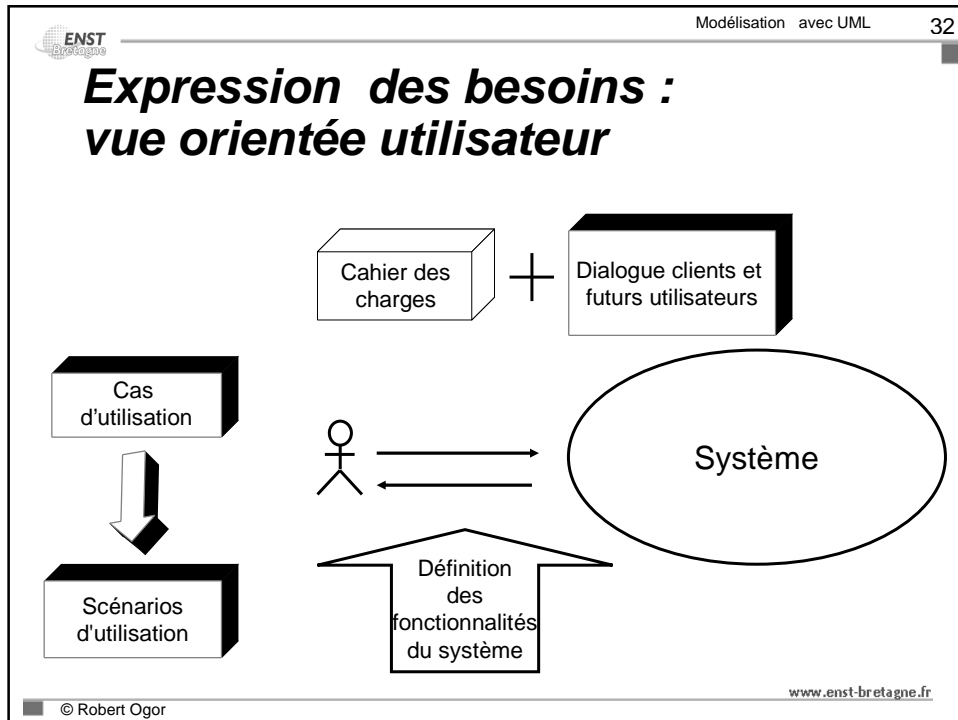
ENST Bretagne Modélisation avec UML 31

## L'analyse

- Le but de l'analyse est de traduire dans un langage proche de celui des informaticiens les modèles exprimés dans l'expression des besoins.
- Cependant pour rester compréhensible par les clients ou utilisateurs, il n'intervient que des entités du domaine (métier) considéré.
- Elle sert d'interface, avec l'expression des besoins, aux dialogues et aux contrats avec les clients et les utilisateurs.
- L'analyse doit servir de support pour la conception, l'implémentation et la maintenance.


© Robert Ogor [www.enst-bretagne.fr](http://www.enst-bretagne.fr)

# Modélisation avec UML





# Modélisation avec UML


Modélisation avec UML34

## ***La notation graphique***

BUT :

- Modéliser les objets, les relations entre objets, les interactions avec le système.
- Servir de support de communication entre l'analyste, le client, et les utilisateurs.


© Robert Ogor[www.enst-bretagne.fr](http://www.enst-bretagne.fr)

Modélisation avec UML35

## ***Cahier des charges : gestion de bibliothèque***

- Un gérant de bibliothèque désire automatiser la gestion des prêts.
- Il commande un logiciel permettant aux utilisateurs de connaître les livres présents, d'en réserver jusqu'à 2. L'adhérent peut connaître la liste des livres qu'il a empruntés ou réservés.
- L'adhérent possède un mot de passe qui lui est donné à son inscription.
- L'emprunt est toujours réalisé par les employés qui travaillent à la bibliothèque. Après avoir identifié l'emprunteur, ils savent si le prêt est possible (nombre max de prêts = 5), et s'il a la priorité (il est celui qui a réservé le livre).
- Ce sont les employés qui mettent en bibliothèque les livres rendus et les nouveaux livres. Il leur est possible de connaître l'ensemble des prêts réalisés dans la bibliothèque.


© Robert Ogor[www.enst-bretagne.fr](http://www.enst-bretagne.fr)

Modélisation avec UML36

## 2) Le diagramme des Use Cases ou des cas d'utilisation

Ce que doit faire le système sans spécifier comment il le fait

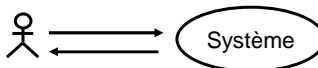
© Robert Ogorwww.enst-bretagne.fr

Modélisation avec UML37

## But des Use Cases

Les cas d'utilisation représentent les fonctionnalités que le système doit savoir faire.

Chaque cas d'utilisation peut être complété par un ensemble d'interactions successives d'une entité en dehors du système (l'utilisateur) avec le système lui même.



Les Uses Cases permettent :

- De connaître le comportement du système sans spécifier comment ce comportement sera réalisé.
- De définir les limites précises du système
- Au développeur de bien comprendre l'attente des utilisateurs et les experts du domaine.

De plus les Use Cases sont :

- Des instruments de validation et de test du système en cours et en fin de construction.

© Robert Ogorwww.enst-bretagne.fr

ENST  
Bretagne

Modélisation avec UML

38

## Modèle des cas d'utilisations

- Un diagramme de cas d'utilisation définit :
  - le système
  - les acteurs
  - les cas d'utilisations
  - les liens entre acteurs et cas d'utilisations
- Un modèle de cas d'utilisation se définit par :
  - des diagrammes de cas d'utilisation
  - une description textuelle des scénarios d'utilisation
  - une description de ces scénarios par :
    - les diagrammes de séquences
    - les diagrammes de collaboration

© Robert Ogor

www.enst-bretagne.fr

ENST  
Bretagne

Modélisation avec UML

39

## Les Acteurs

- Un acteur représente une personne ou un périphérique qui joue un rôle (interagit) avec le système.
- Relation entre acteurs : généralisation (héritage)

Héritage  
relation entre acteurs  
un bibliothécaire est un abonné  
un administrateur est un bibliothécaire  
un administrateur est un abonné

© Robert Ogor

www.enst-bretagne.fr

ENST  
Bretagne

Modélisation avec UML

40

## Les cas d'utilisation (use-case)

- Un cas d'utilisation est un moyen de représenter les différentes possibilités d'utiliser un système.
- Il exprime toujours une suite d'interactions entre un acteur et l'application.
- Il définit une fonctionnalité utilisable par un acteur.

```
graph TD; Emprunter([Emprunter]); Réserver([Réserver]); Regarder([Regarder la liste des livres]);
```

© Robert Ogor

www.enst-bretagne.fr

ENST  
Bretagne

Modélisation avec UML

41

## Organisation des Use Cases : include

- La relation "include" précise qu'un cas d'utilisation contient le comportement défini dans un autre cas d'utilisation.
- Cette relation permet de mettre en commun des comportements communs à plusieurs cas d'utilisation..

```
graph TD; Emprunt([Emprunt]) -.->|<<include>>| Identification([Identification abonné]); Réserve([Réserve]) -.->|<<include>>| Identification;
```

© Robert Ogor

www.enst-bretagne.fr

ENST  
Bretagne

Modélisation avec UML

42

## Organisation des Use Cases : extend

- La relation "extend" précise qu'un cas d'utilisation peut dans certains cas augmenter le comportement d'un autre cas d'utilisation.
- Une condition devra valider cette augmentation.
- Le point d'utilisation de cette augmentation peut être défini dans un "point d'extension".

```
graph LR; R([Réservation]) -- "<<extend>>  
avant le choix du livre" --- RL([Regarder liste des livres]);
```

- Dans cet exemple, le cas d'utilisation "Regarder la liste des livres" augmente le cas d'utilisation d'une réservation, avant le choix du livre, si l'utilisateur en fait la demande.

© Robert Ogor

www.enst-bretagne.fr

ENST  
Bretagne

Modélisation avec UML

43

## Organisation des Use Cases : généralisation

- Cette relation "est un" introduit la notion d'héritage.
- Les cas d'utilisation "Vérification par mot passe" et "Vérification par carte" sont des spécialisations du cas d'utilisation "Identification abonné".
- Autrement dit : si l'on dit que l'on fait une "Identification abonné", ce peut être une "Vérification par carte" ou une "Vérification par mot passe".

```
graph BT; VMP([Vérification par mot passe]) --> IA([Identification abonné]); VC([Vérification par carte]) --> IA;
```

© Robert Ogor


www.enst-bretagne.fr

Modélisation avec UML44

## **Modélisation d'un système: obtenir les cas d'utilisation**

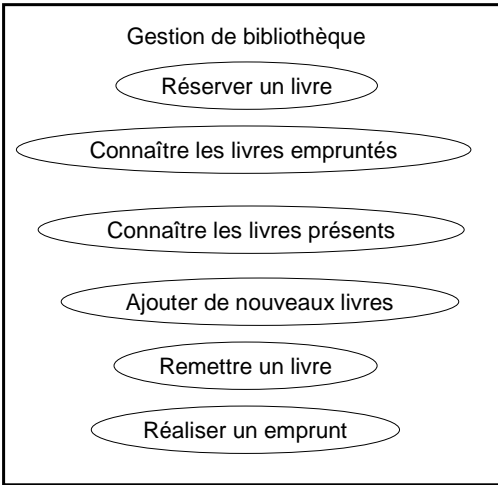
- Identifier les acteurs qui utilisent, qui gèrent, qui exécutent des fonctions spécifiques.
- Organiser les acteurs par relation d'héritage.
- Pour chaque acteur, rechercher les cas d'utilisation avec le système. En particulier, ceux qui modifient l'état du système ou qui attendent une réponse du système.
- Ne pas oublier les variantes d'interactions (cas d'erreur, cas interdits).
- Organiser ces interactions par héritage, par utilisation et par extension.

© Robert Ogor[www.enst-bretagne.fr](http://www.enst-bretagne.fr)


Modélisation avec UML45

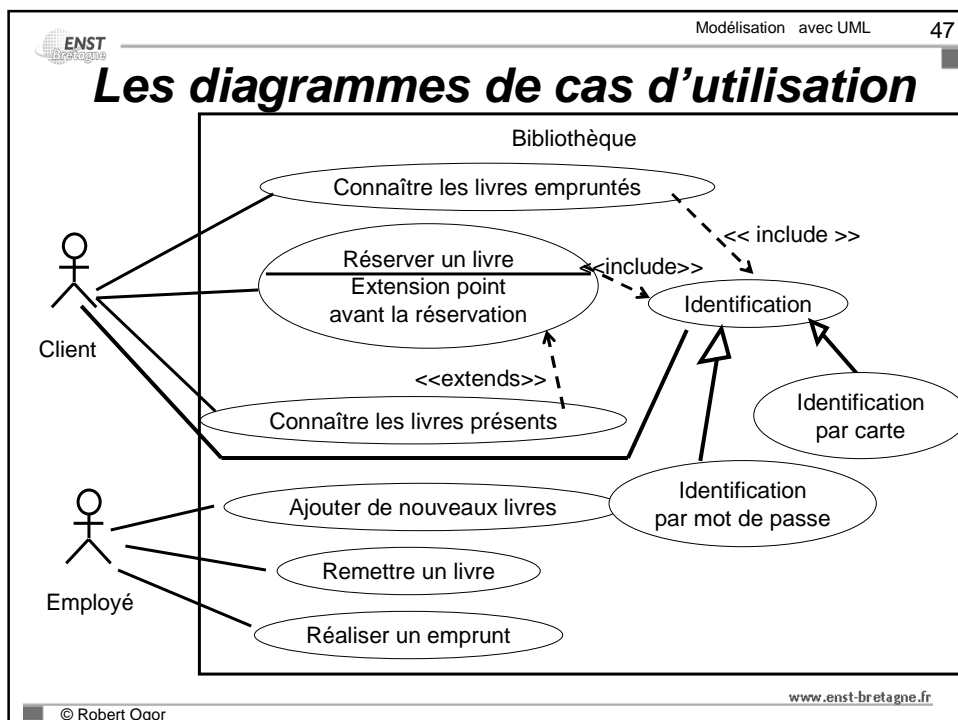
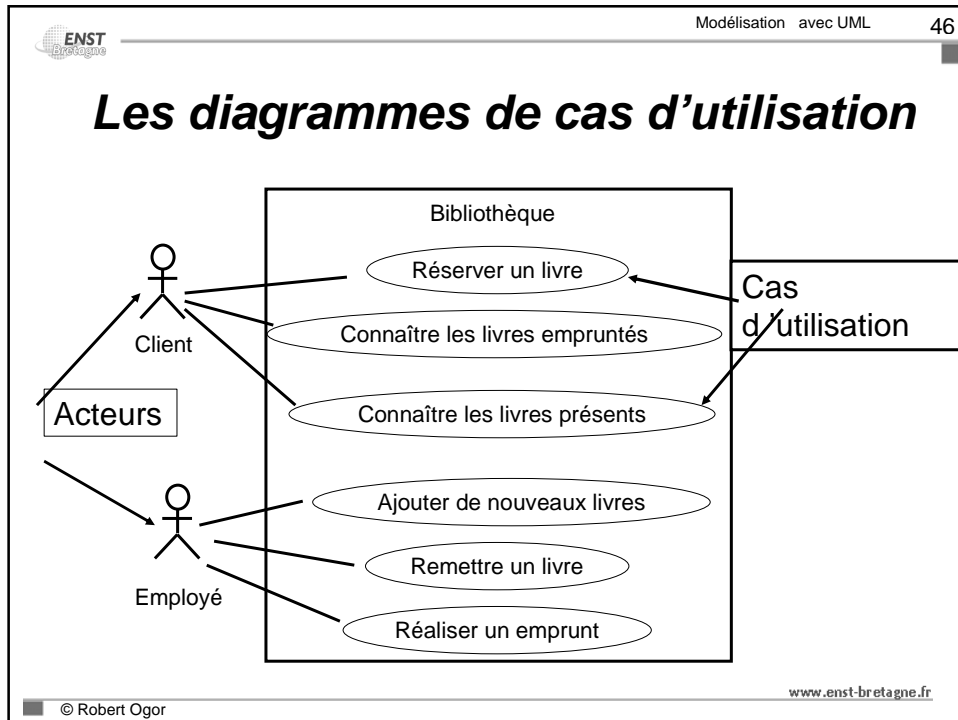
## **Le système**


- Le système définit l'application informatique, il ne contient donc pas les acteurs, mais les cas d'utilisation et leur associations



The diagram shows a rectangular box representing the system boundary. Inside the box, the title 'Gestion de bibliothèque' is at the top. Below it, seven use cases are listed vertically, each enclosed in an oval. The use cases are: 'Réserver un livre', 'Connaître les livres empruntés', 'Connaître les livres présents', 'Ajouter de nouveaux livres', 'Remettre un livre', and 'Réaliser un emprunt'.

© Robert Ogor[www.enst-bretagne.fr](http://www.enst-bretagne.fr)




Modélisation avec UML48

## Scénarios d'un cas d'utilisation

- La description d'un cas d'utilisation se fait par des scénarios qui définissent la suite logique des interactions qui constituent ce cas.
- On peut définir des scénarios simples ou des scénarios plus détaillés faisant intervenir les variantes, les cas d'erreurs, etc.
- Cette description se fait de manière simple, par un texte compréhensible par les personnes du domaine de l'application.
- Elle précise ce que fait l'acteur et ce que fait le système
- La description détaillée pourra préciser les contraintes de l'acteur et celles du système.

© Robert Ogorwww.enst-bretagne.fr

Modélisation avec UML49

## Scénarios d'un cas d'utilisation

Réservation d'un livre


description simplifiée

Le client se présente devant un terminal:

- (1) Le système affiche un message d'accueil.
- (2) Le client choisit l'opération réservation parmi les différentes opérations proposées.
- (3) Le système lui demande de s'authentifier.
- (4) Le client donne son identification (nom, mot de passe).
- (5) Le système lui demande de choisir un livre.
- (6) Le client précise le livre qu'il désire.
- (7) Le système lui précise si un exemplaire du livre lui est réservé.

© Robert Ogorwww.enst-bretagne.fr



Modélisation avec UML50


## Scénarios d'un cas d'utilisation


**Réservation d'un livre**

description détaillée

Pré-conditions: Le client doit être inscrit à la bibliothèque  
Le client ne doit pas avoir atteint le nombre maximum de réservation  
Un exemplaire du livre doit être enregistré

Post-conditions: (Si l'opération s'est bien déroulée)  
Le client a une réservation supplémentaire  
Le nombre d'exemplaires disponibles du livre est décrétementé de 1

© Robert Ogorwww.enst-bretagne.fr

Modélisation avec UML51

## Scénarios d'un cas d'utilisation

**Réservation d'un livre**

Cas normaldescription détaillée

- (1) Le système affiche un message d'accueil sur le terminal avec un choix d'opérations
- (2) Le client choisit l'opération réservation parmi les différentes opérations proposées.
- (3) Le système lui demande de s'authentifier.
- (4) Le client donne son identification (nom, mot de passe).
- (5) Le système demande le titre du livre en donnant la possibilité de choisir dans une liste.
- (6) Le client précise le livre qu'il désire.
- (7) Le système lui précise que un exemplaire du livre lui est réservé.

variante 1

en (6) le client demande à connaître les livres présents

variante 2

en (4) le client n'est pas reconnu, dans ce cas, tant qu'il n'est pas reconnu, on lui redemande de s'authentifier

variante 3


en (4) le client est reconnu, mais le mot de passe est incorrect, il peut avoir 5 essais, par la suite le client sera interdit pendant 1 heure.

variante 4

en (4) le client n'a plus le droit de réserver

variante 5

en (7) le livre n'est pas libre

© Robert Ogorwww.enst-bretagne.fr

ENST Bretagne Modélisation avec UML 52

## Scénario par diagramme de séquences

- Suite aux descriptions textuelles, le scénario peut être représenté en utilisant un diagramme de séquences. Le diagramme de séquences permet :
  - de visualiser l'aspect temporel des interactions
  - de connaître le sens des interactions (acteur vers système ou contraire)

```

sequenceDiagram
    actor Client
    participant System as :Système de prêts
    Note left of Client: Réserver un livre
    System->>Client: Afficher message d'accueil
    Client->>System: Choix de l'opération réservation
    System->>Client: Demande d'identification du client
    Client->>System: Identification du client
    System->>Client: Demande identification du livre
    Client->>System: Identification du livre
    System->>Client: Message « le livre est réservé »
    
```

© Robert Ogor [www.enst-bretagne.fr](http://www.enst-bretagne.fr)

ENST Bretagne Modélisation avec UML 53

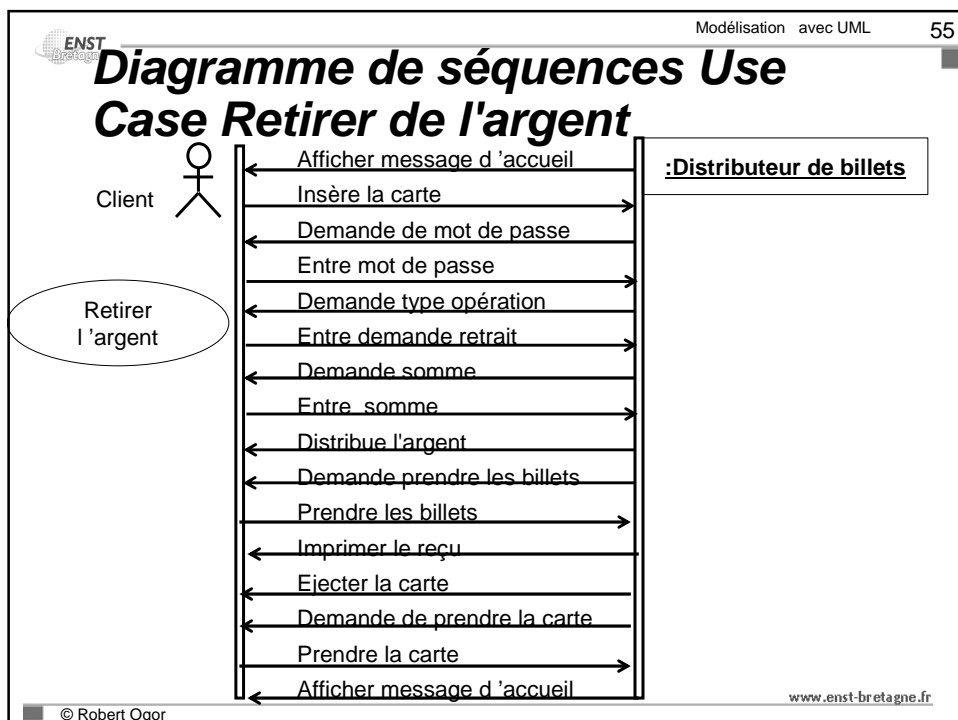
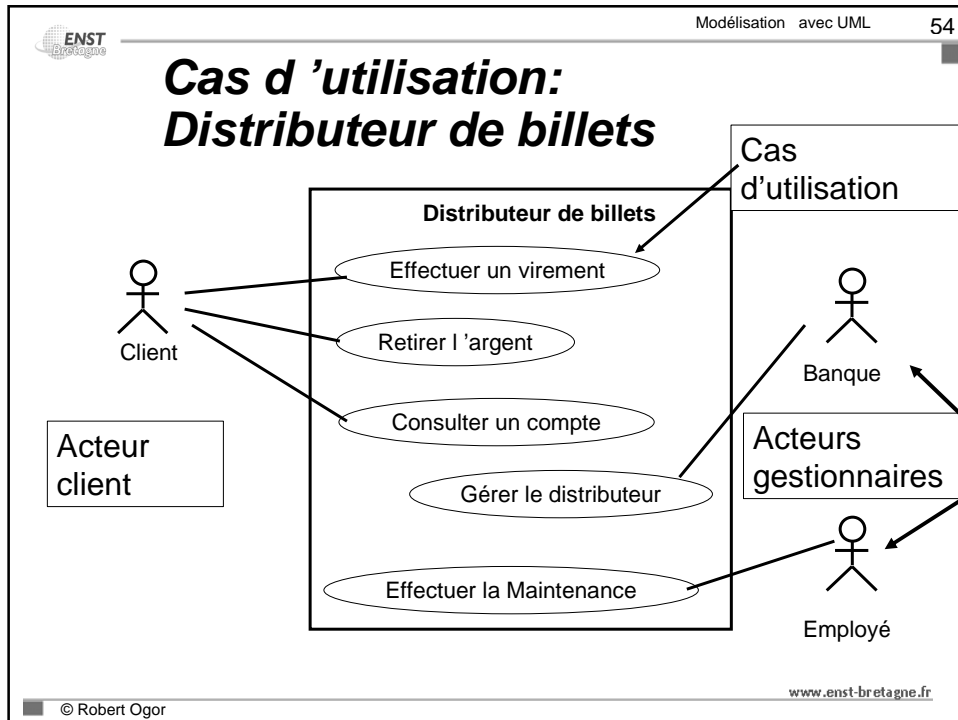
## Variation du scénario


```

sequenceDiagram
    actor Client
    participant System as :Système de prêts
    Note left of Client: Réserver un livre
    System->>Client: Afficher message d'accueil
    Client->>System: Choix de l'opération réservation
    System->>Client: Demande d'identification du client
    Client->>System: Identification du client
    System->>Client: Refus trop de livres réservés
    
```

© Robert Ogor [www.enst-bretagne.fr](http://www.enst-bretagne.fr)


# Modélisation avec UML



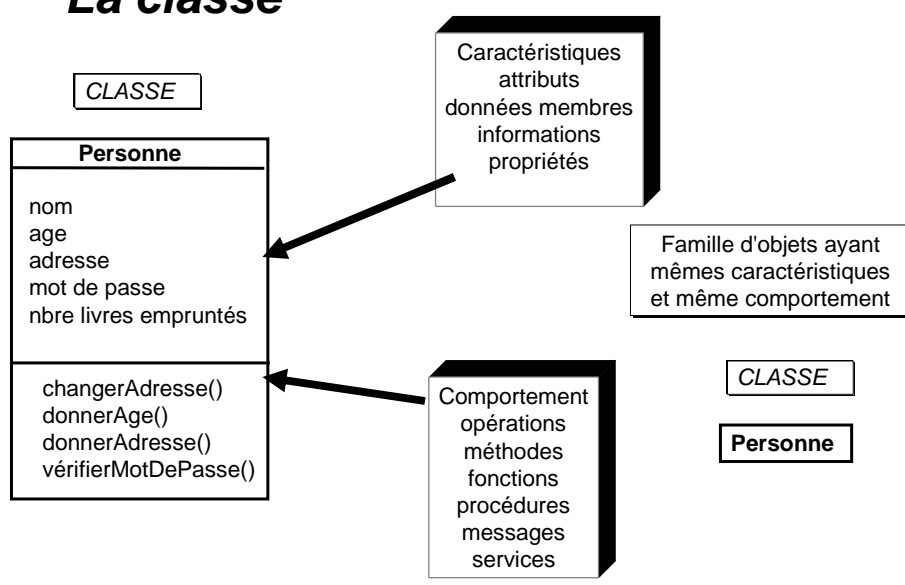
Modélisation avec UML56

## 3) Concepts objets et diagrammes de classes

© Robert Ogorwww.enst-bretagne.fr

Modélisation avec UML57

### La classe



**CLASSE**

**Personne**

nom  
age  
adresse  
mot de passe  
nbre livres empruntés

changerAdresse()  
donnerAge()  
donnerAdresse()  
vérifierMotDePasse()

Caractéristiques  
attributs  
données membres  
informations  
propriétés

Comportement  
opérations  
méthodes  
fonctions  
procédures  
messages  
services

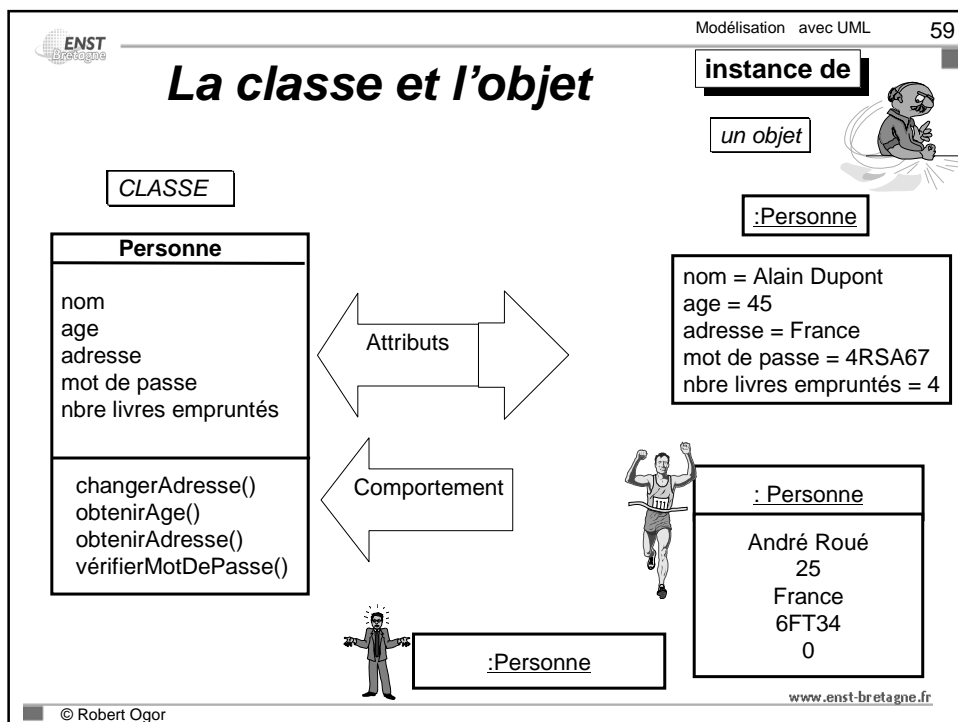
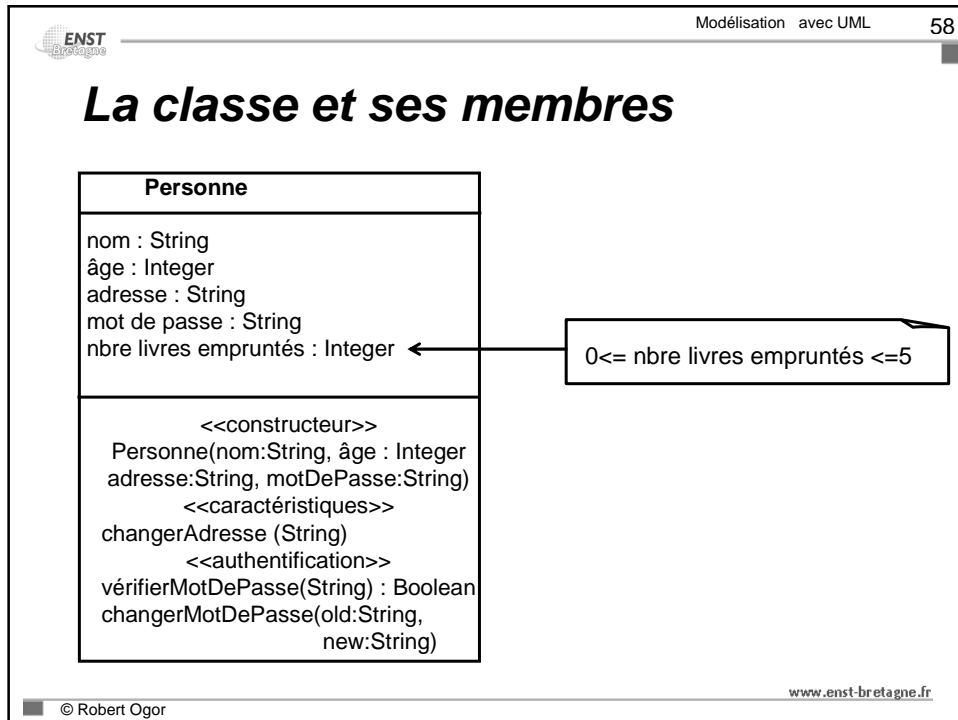
Famille d'objets ayant  
mêmes caractéristiques  
et même comportement


**CLASSE**

**Personne**

© Robert Ogorwww.enst-bretagne.fr

# Modélisation avec UML



Modélisation avec UML60


## ***Protection des attributs et des opérations : principe de l'encapsulation***

- Peut-on accéder à tous les attributs ou à toutes les méthodes d'un objet ? NON
  - La classe définit ce qui est accessible.  
C'est le principe de l'encapsulation.  
Un objet complexe ne peut être utilisé qu'au travers de ce qui est accessible.

Exemple :

- On ne peut utiliser une *voiture* qu'à travers son *volant*, son *frein*, son *accélérateur*, etc.
- L'accès au *carburateur* est impossible sauf par les méthodes qui le font de manière cohérente (méthode *accélérer* de l'*accélérateur*).


© Robert Ogor[www.enst-bretagne.fr](http://www.enst-bretagne.fr)

Modélisation avec UML61

## ***Protection des attributs et des opérations : usage et notation***

- Les attributs sont en général inaccessibles (secret).
  - Ils sont alors qualifiés de :
    - "**protected**" (notation UML : # )
    - ou "**private**" (notation UML: - )
  - Leur lecture ou modification n'est possible qu'au travers de certaines opérations (accesseurs : *changerAdresse()*, *obtenirAge()*, etc.)
- Les opérations sont en général accessibles (publiques) :
  - Elles sont alors qualifiées de :
    - "**public**" (notation UML: + )
- Certaines opérations peuvent cependant être privées et certains attributs peuvent être publics (non souhaitable / principe d'encapsulation)

© Robert Ogor[www.enst-bretagne.fr](http://www.enst-bretagne.fr)



Modélisation avec UML

62

## Protection des attributs et des opérations : notation UML

CLASSE

**Personne**

---

- nom  
- âge  
- adresse

---


# changerAdresse()  
# obtenirRue()  
+ obtenirAge()

+ : attribut public  
# : attribut protected  
- : attribut private

+ : opération public  
# : opération protected  
- : opération private

© Robert Ogor

www.enst-bretagne.fr



Modélisation avec UML

63

## Attributs et opérations de classe

**Personne**


---

- nom  
- age  
- adresse  
- mot de passe  
- nbreLivresEmpruntés  
- nbreLivresEmpruntables = 5

---

+ changerAdresse()  
+ obtenirAge()  
+ obtenirAdresse()  
+ vérifierMotDePasse()  
+ getNbLivresEmpruntables()

Le nombre de livres empruntables n'est pas une caractéristique d' Alain Dupont (objet). C'est une caractéristique valable pour l'ensemble des personnes (la classe).



une\_personne:Personne

nom = Alain Dupont  
age = 45  
adresse = France  
mot de passe = 4RSA67  
nbreLivresEmpruntés= 4

La méthode getNbLivresEmpruntables() utilise la valeur nbreLivresEmpruntables connue par la classe. Cette méthode peut être appliquée directement à la classe Personne et bien sûr aussi aux objets instances de Personne.

© Robert Ogor

www.enst-bretagne.fr

# Modélisation avec UML

ENST Bretagne Modélisation avec UML 64

## Attributs et opérations de classe Notation UML

**CLASSE**

Personne
- nom - âge - adresse - <b><u>nombrePersonne</u></b>
# changerAdresse() # obtenirRue() + obtenirAge() + <b><u>obtenirNombrePersonne()</u></b>

+ : attribut public  
 # : attribut protected  
 - : attribut private  
 \_\_ : attribut de classe

+ : opération public  
 # : opération protected  
 - : opération private  
 \_\_ : opération de classe


© Robert Ogor [www.enst-bretagne.fr](http://www.enst-bretagne.fr)

ENST Bretagne Modélisation avec UML 65

## La réification

**entité physique**

Chien
race age couleur
aboyer() mordre () obéir ()



**événement**

Mariage
époux épouse date
seMarier () divorcer ()

**relation**

Appartenir
propriétaire date voiture
vendre () acheter () prêter ()

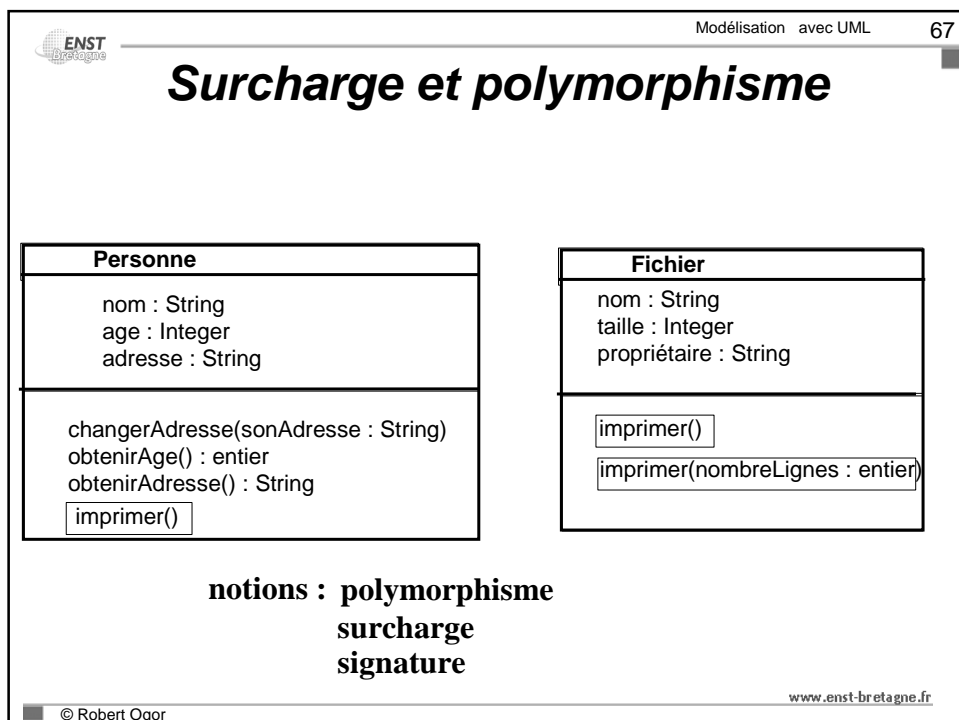
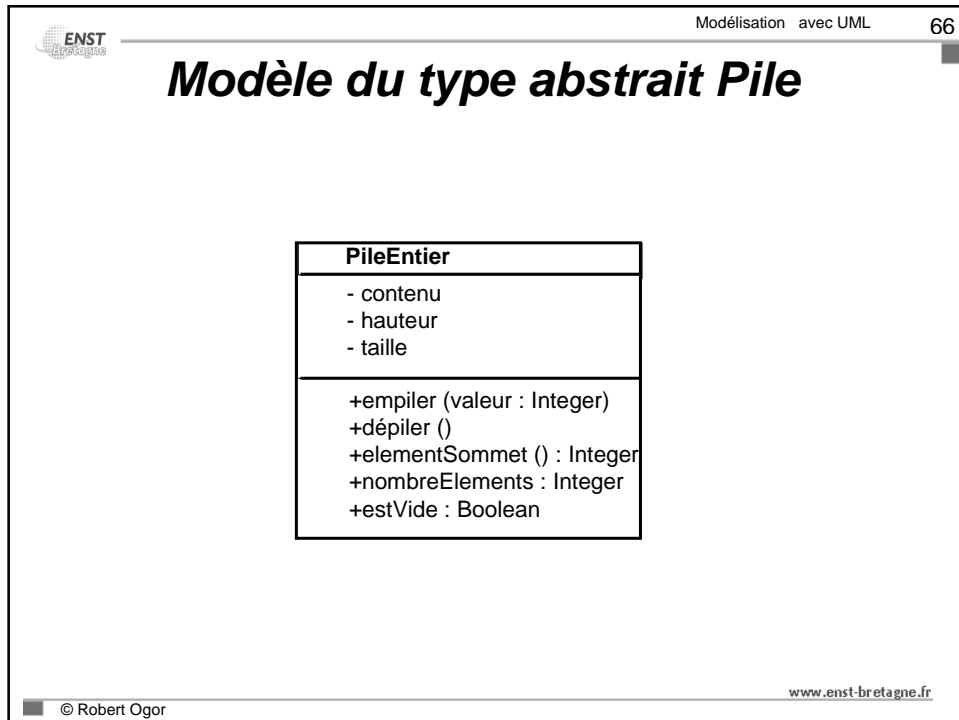
**situation**

Cours
professeur salle élèves
assister () quitter ()

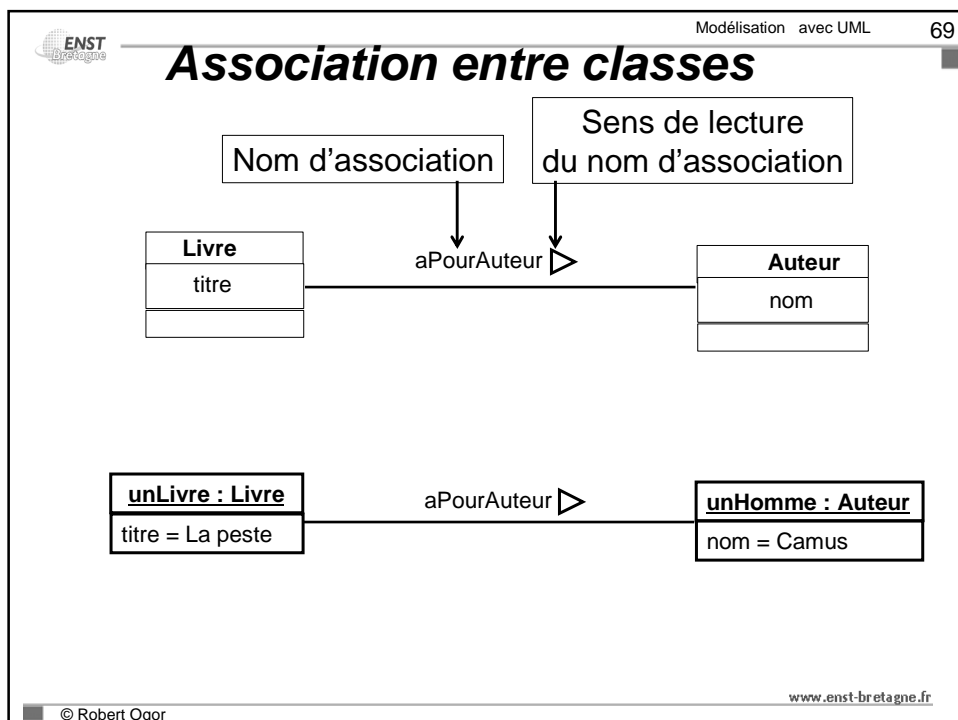
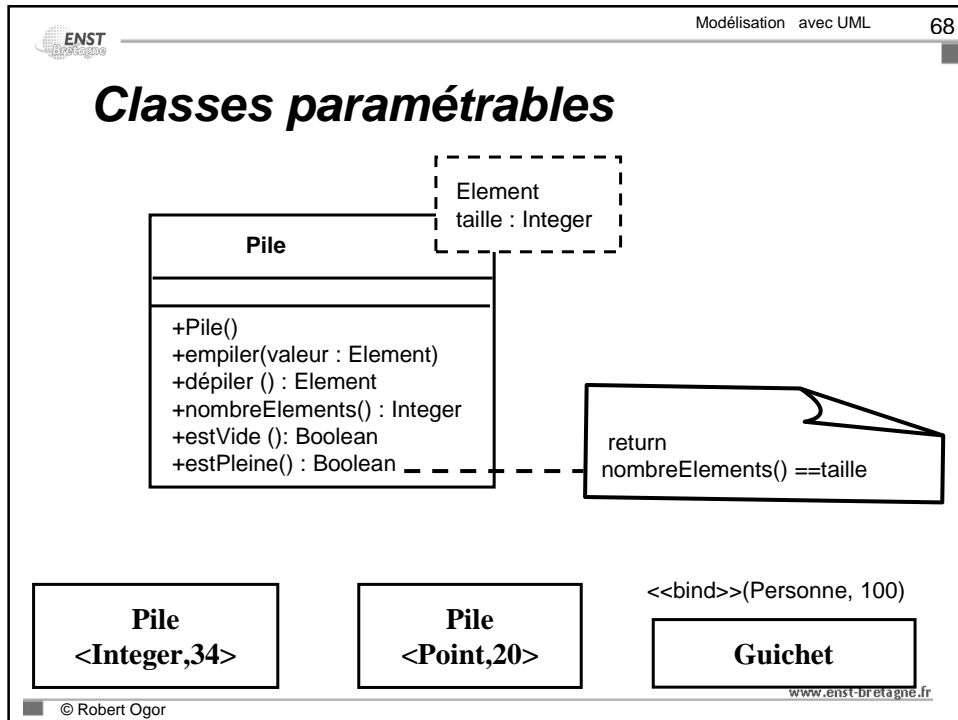
© Robert Ogor [www.enst-bretagne.fr](http://www.enst-bretagne.fr)



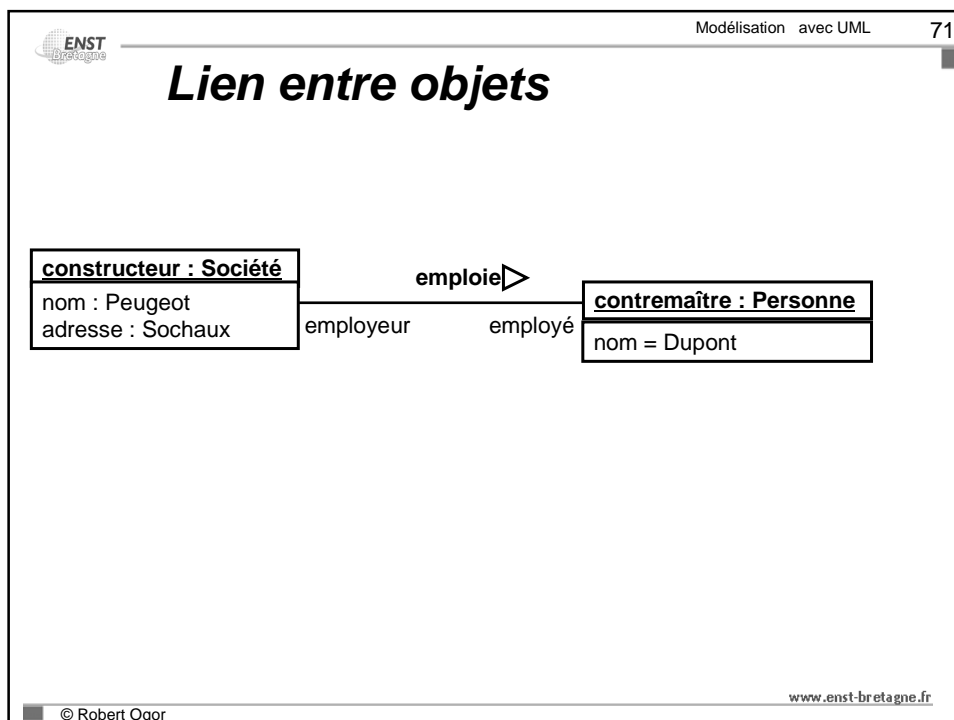
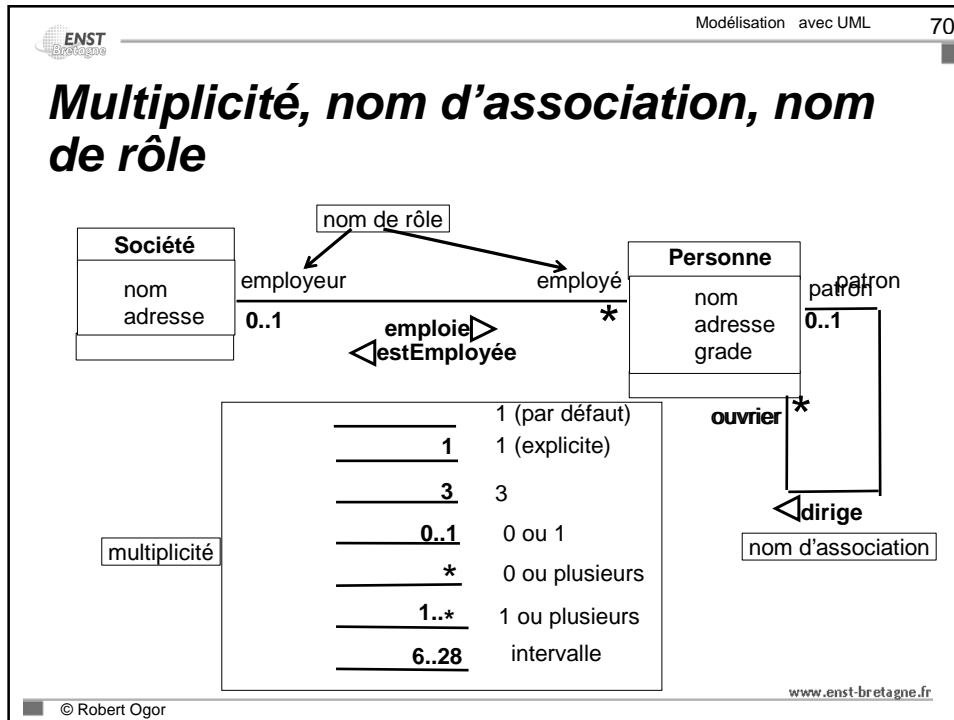
# Modélisation avec UML



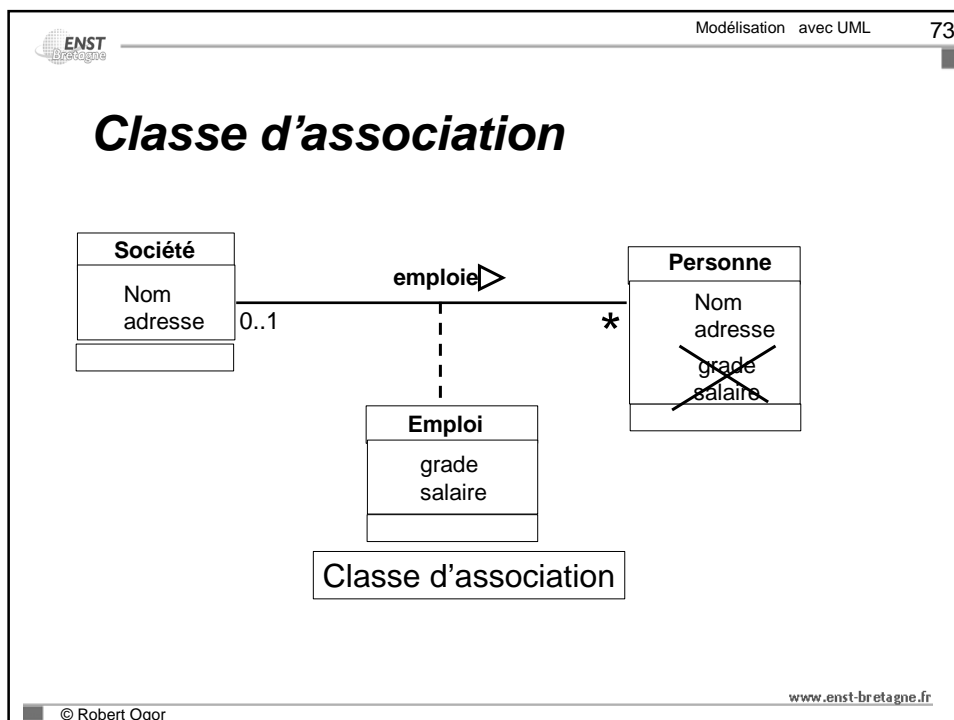
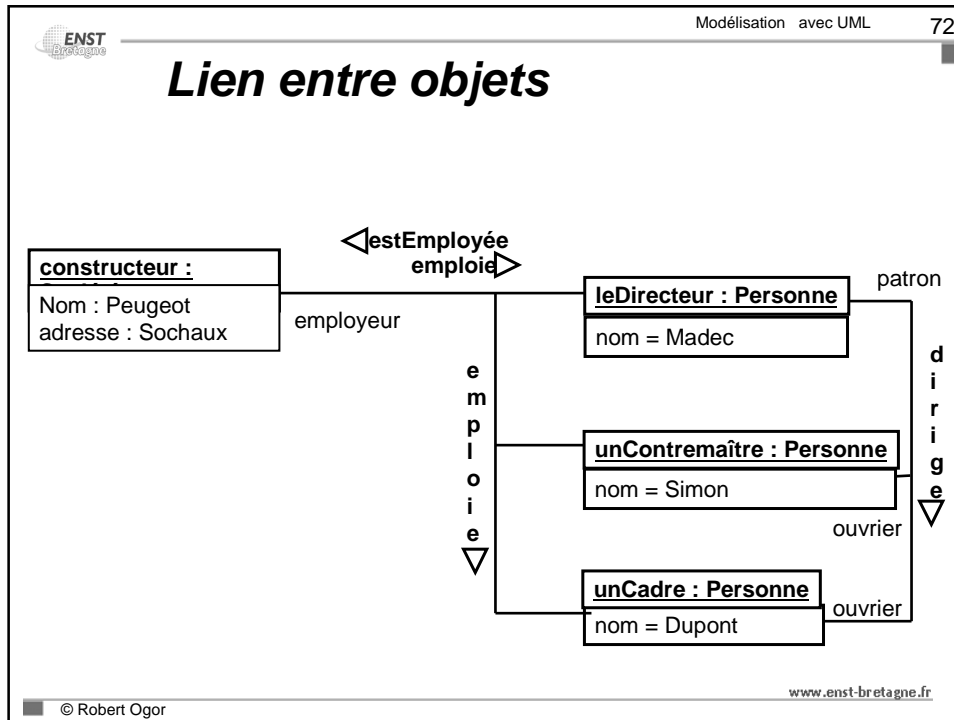
# Modélisation avec UML



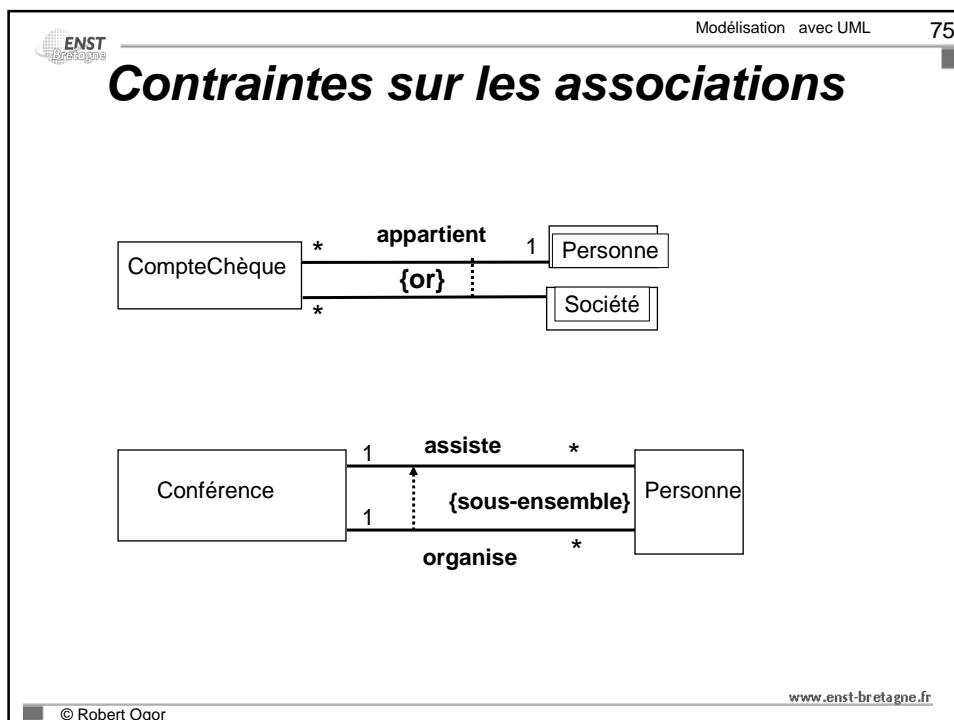
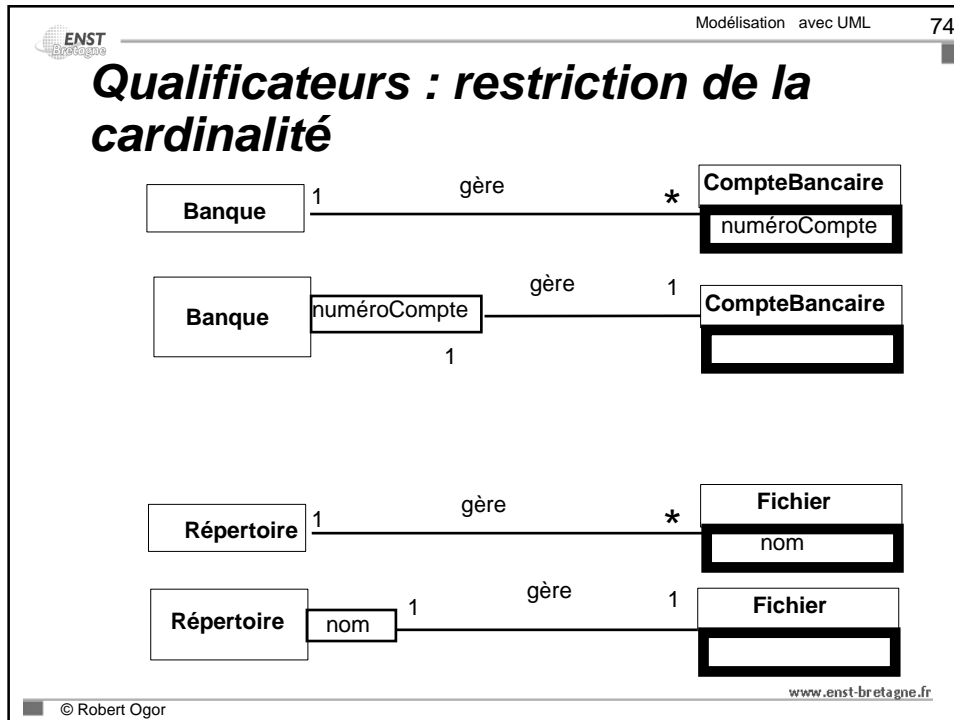
# Modélisation avec UML



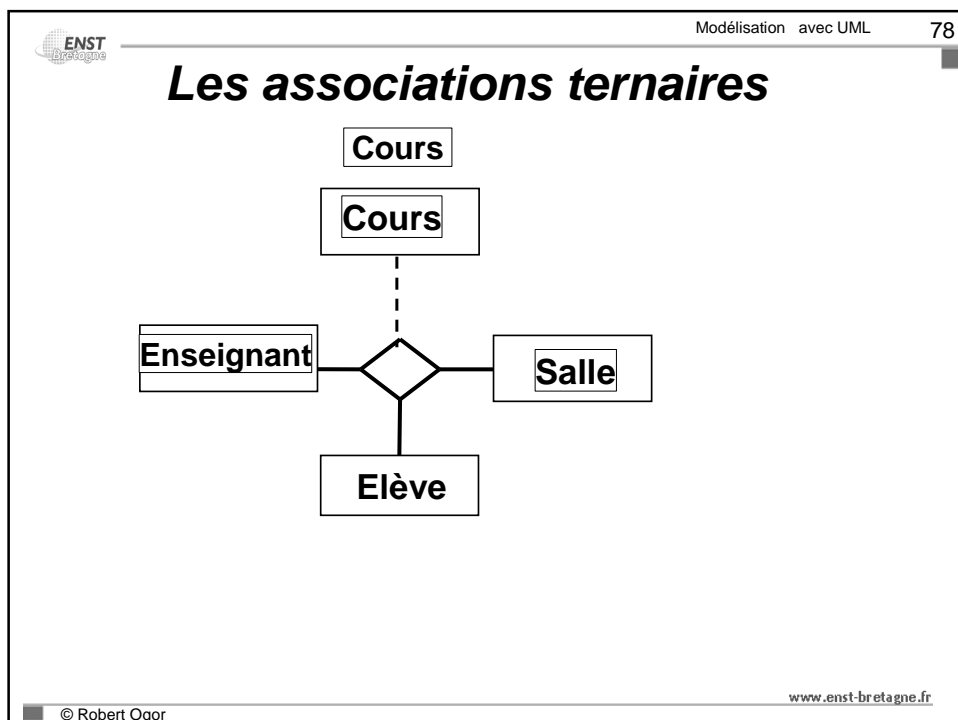
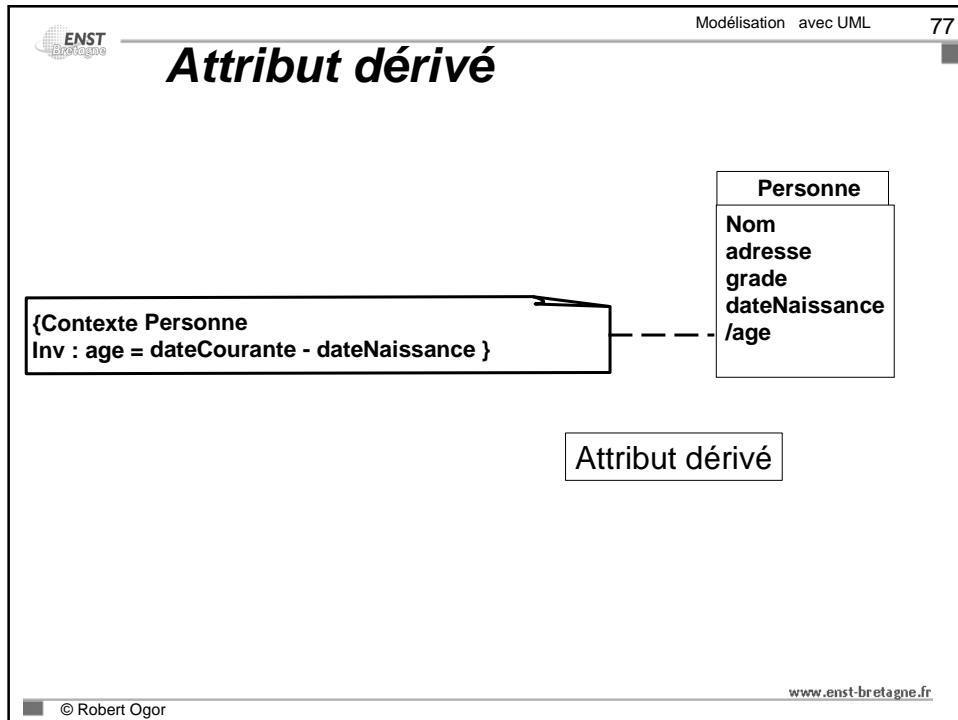
# Modélisation avec UML



# Modélisation avec UML



# Modélisation avec UML



# Modélisation avec UML

ENST Bretagne Modélisation avec UML 79

## Agrégation

C'est une association qui exprime un **couplage fort** lié à une **relation de subordination**, elle est **asymétrique** du type **ensemble/élément**.

Règles permettant de choisir une agrégation :

- Est ce une partie de?
- Les opérations appliquées à l'ensemble sont elles appliquées à l'élément?
- Les changements d'états sont-ils liés ?

Mais

- un élément agrégé peut être lié à d'autres classes
- la suppression de l'ensemble n'entraîne pas celle de l'élément

www.enst-bretagne.fr

© Robert Ogor

ENST Bretagne Modélisation avec UML 80


## Composition : agrégation forte

La composition est une **agrégation forte** qui **lie les cycles de vie** entre le composé (ensemble) et les composants (éléments).

Le choix entre composition et agrégation peut être laissé à la phase de conception.

www.enst-bretagne.fr

© Robert Ogor

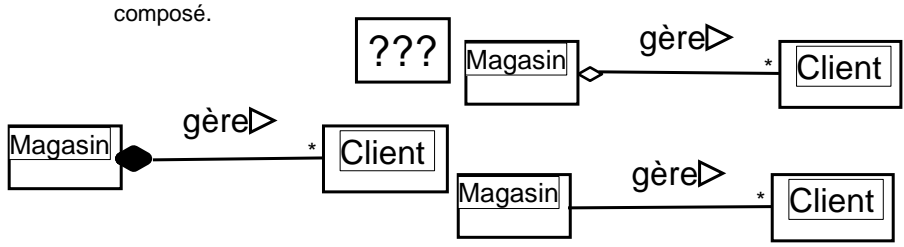


Modélisation avec UML

81

## Association, agrégation ou composition ?

- Règles obligatoires pour l'agrégation :
  - Est ce une partie de?
  - Les opérations appliquées au composé sont elles appliquées au composant?
  - Les changements d'états sont-ils liés ?
- Règles obligatoires pour la composition :
  - La suppression du composé entraîne t-elle la suppression des composants ?
  - Les attributs du composé sont-ils utilisés dans les composants ?
  - Les composants sont des instances du composé ?
  - Un composant ne peut pas être en relation avec d'autres classes externes au composé.




The diagrams show three types of relationships between **Magasin** and **Client** classes:

- Association:** A simple line with an open arrowhead labeled "gère" connecting **Magasin** to **Client** (multiplicity \*).
- Aggregation:** A line with an open diamond at the **Magasin** end and an open arrowhead at the **Client** end, labeled "gère" (multiplicity \*).
- Composition:** A line with a filled diamond at the **Magasin** end and an open arrowhead at the **Client** end, labeled "gère" (multiplicity \*).

www.enst-bretagne.fr

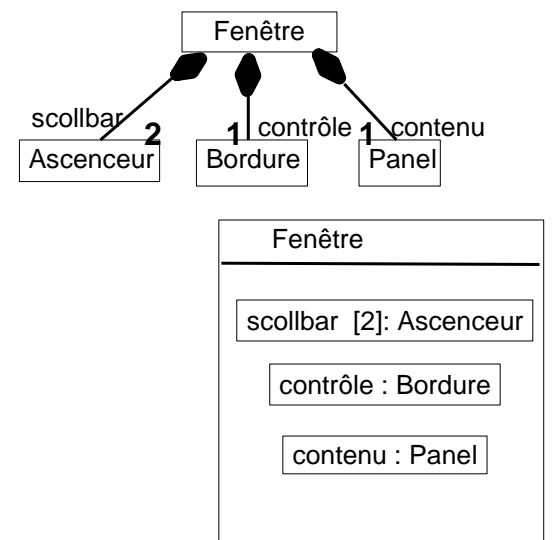
© Robert Ogor



Modélisation avec UML

82

## Composition



The diagram illustrates the composition of a **Fenêtre** class into three components:

- scrollbar** (multiplicity 2) of type **Ascenseur**
- contrôle** (multiplicity 1) of type **Bordure**
- contenu** (multiplicity 1) of type **Panel**

Below the diagram, a detailed view of the **Fenêtre** class structure is shown, listing the components and their multiplicities:

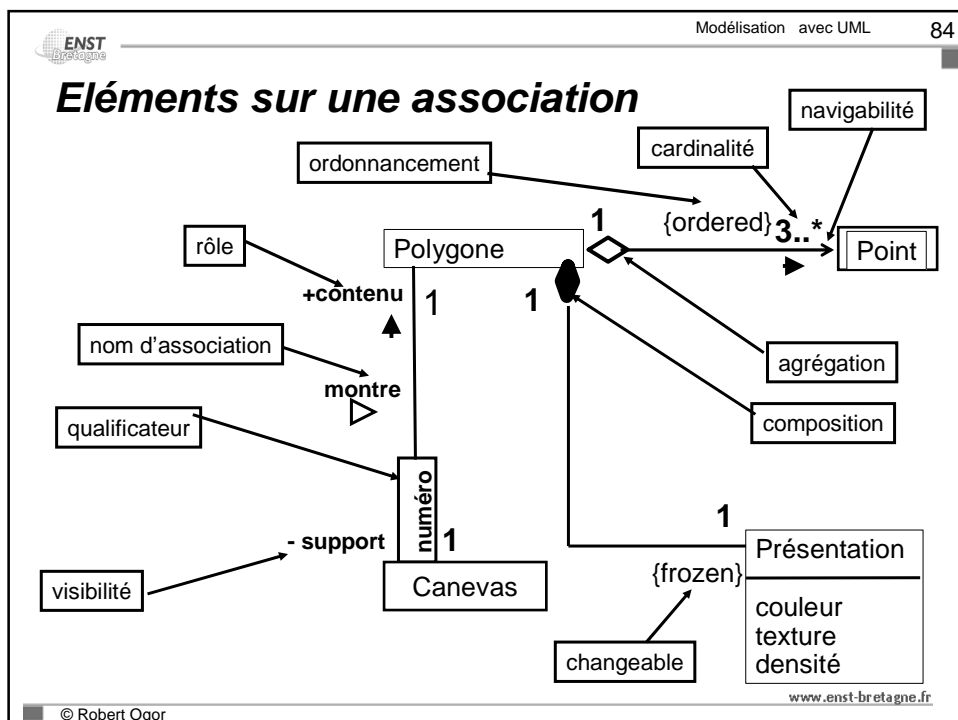
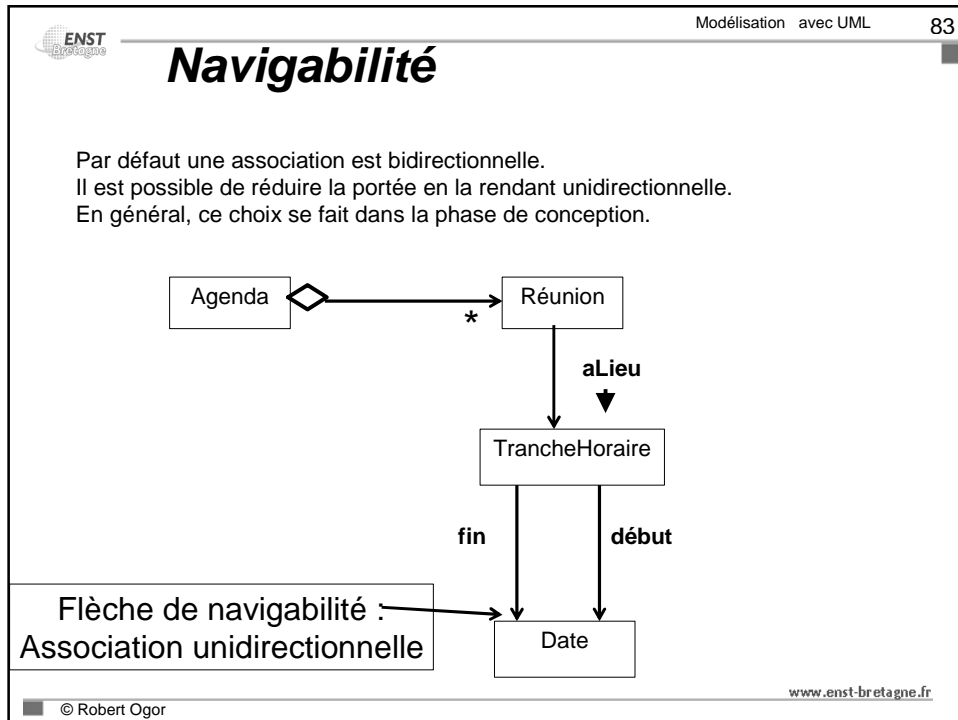
- scrollbar [2]: Ascenseur
- contrôle : Bordure
- contenu : Panel

www.enst-bretagne.fr

© Robert Ogor



# Modélisation avec UML




Modélisation avec UML85

## 3) Concepts objets et diagrammes de classes

### → Héritage

© Robert Ogor[www.enst-bretagne.fr](http://www.enst-bretagne.fr)

Modélisation avec UML86

## ***Héritage : buts et principes 1/3***

### ***Nouvelles classes dérivées de classes existantes***

**BUT**


Permettre une réutilisation optimale des classes déjà écrites, utilisées et validées

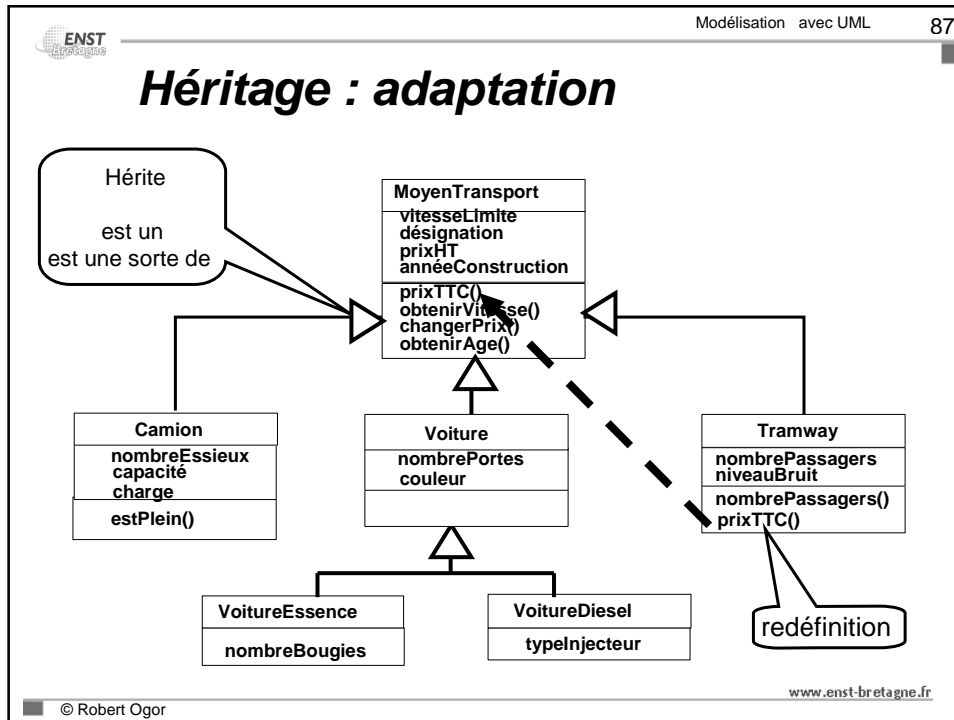
- réutilisation de la structure des données héritées
- réutilisation du code des services hérités

**PRINCIPE**

Ne pas modifier les classes déjà écrites cela modifierait l'utilisation qui en est faite.

- ne pas hésiter à créer des classes, extensions d'autres déjà validées

© Robert Ogor[www.enst-bretagne.fr](http://www.enst-bretagne.fr)



ENST Bretagne Modélisation avec UML 88

## Héritage : buts et principes 2/3

### Ajout d'une classe de base (analyse)

**BUT 2**

Permettre une factorisation des caractéristiques et des comportements communs à plusieurs classes

- mise en commun des structure des données
- mise en commun du code des services

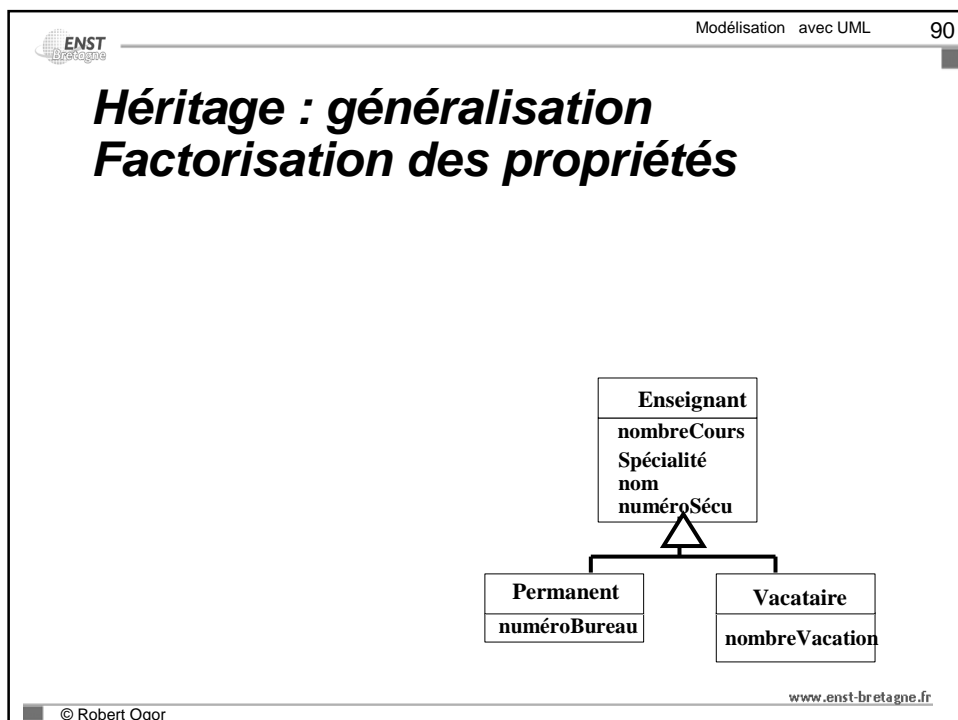
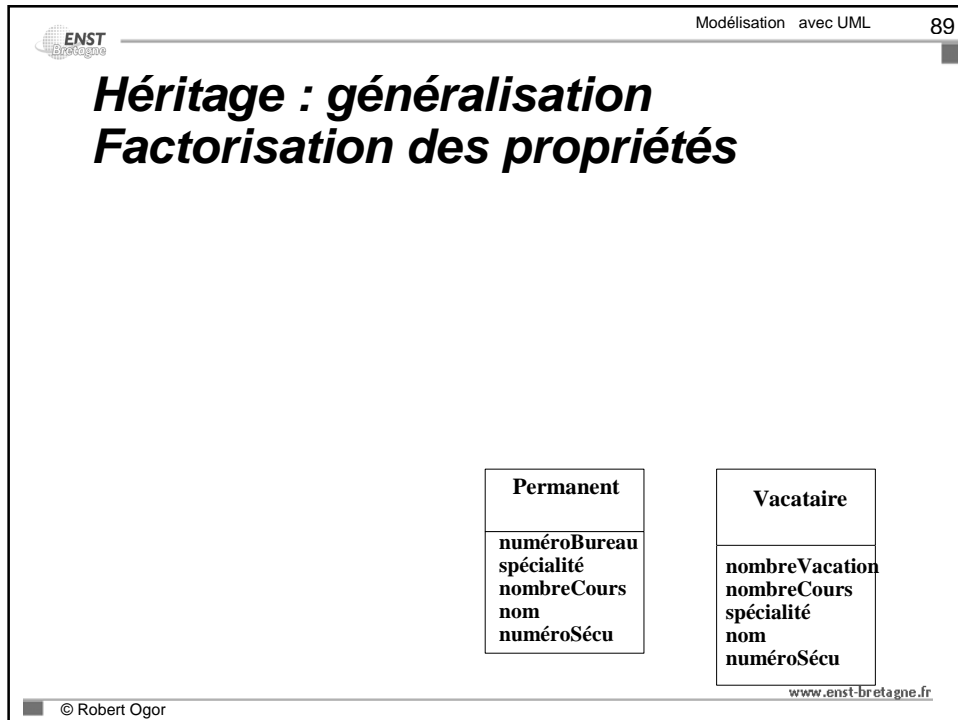
**PRINCIPE**

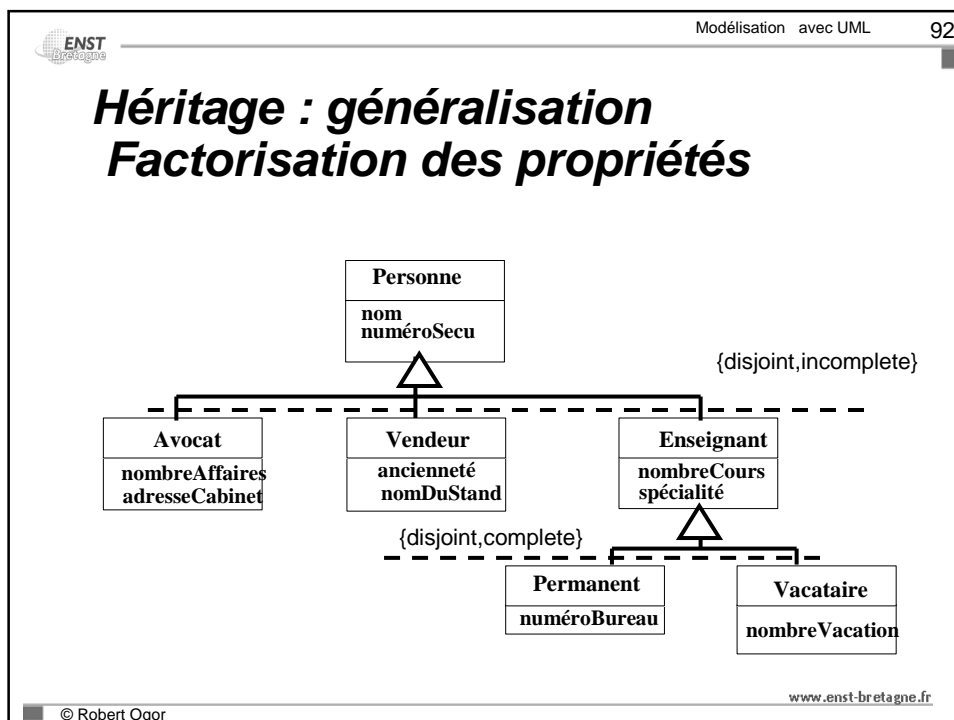
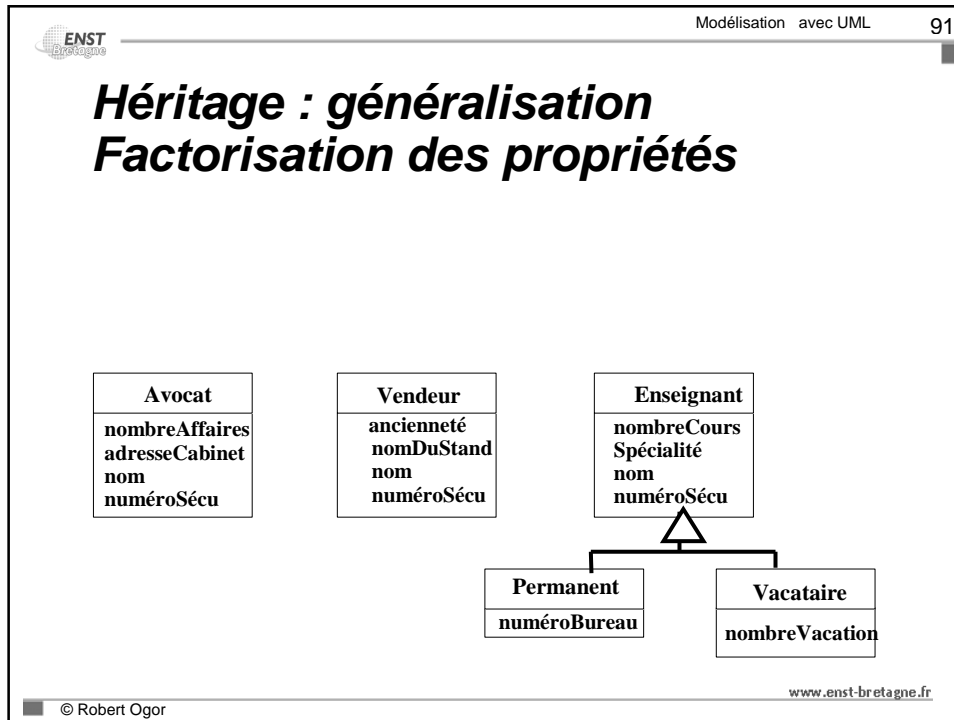
Lorsque plusieurs classes ont des caractéristiques et des comportements communs la création d'une classe ancêtre permet de regrouper ce qui est commun.

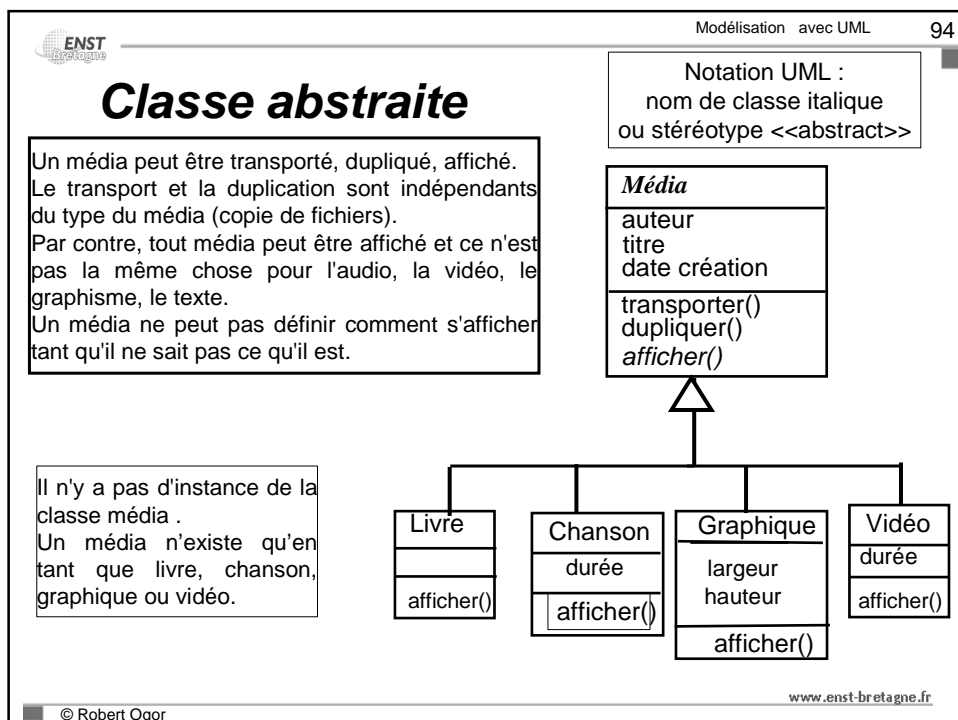
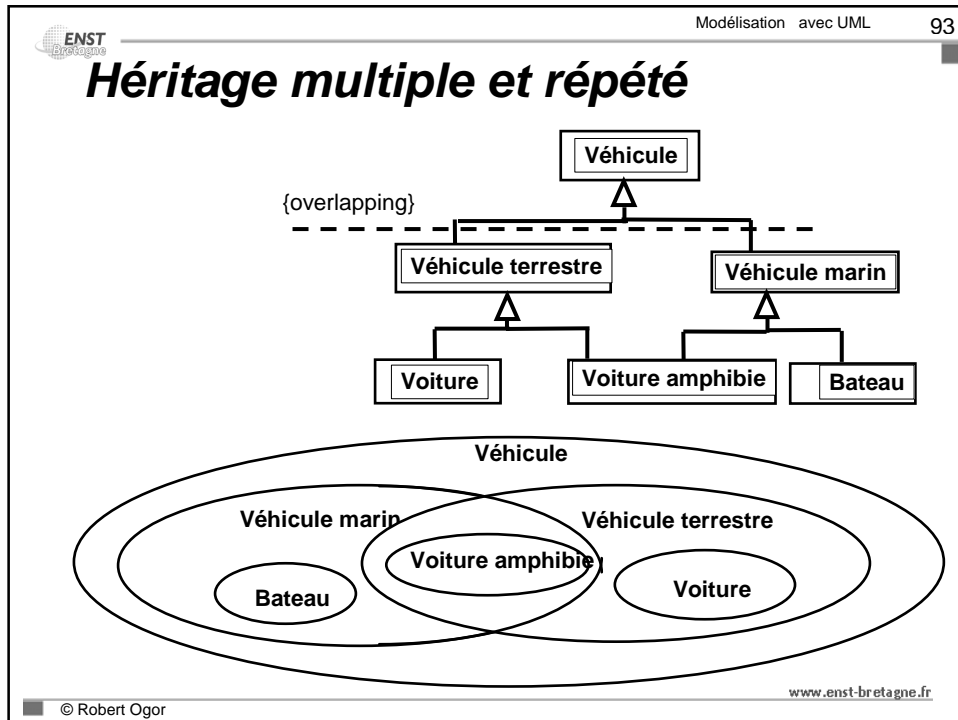
Cette classe ancêtre peut correspondre à une classe concrète ou à une classe abstraite

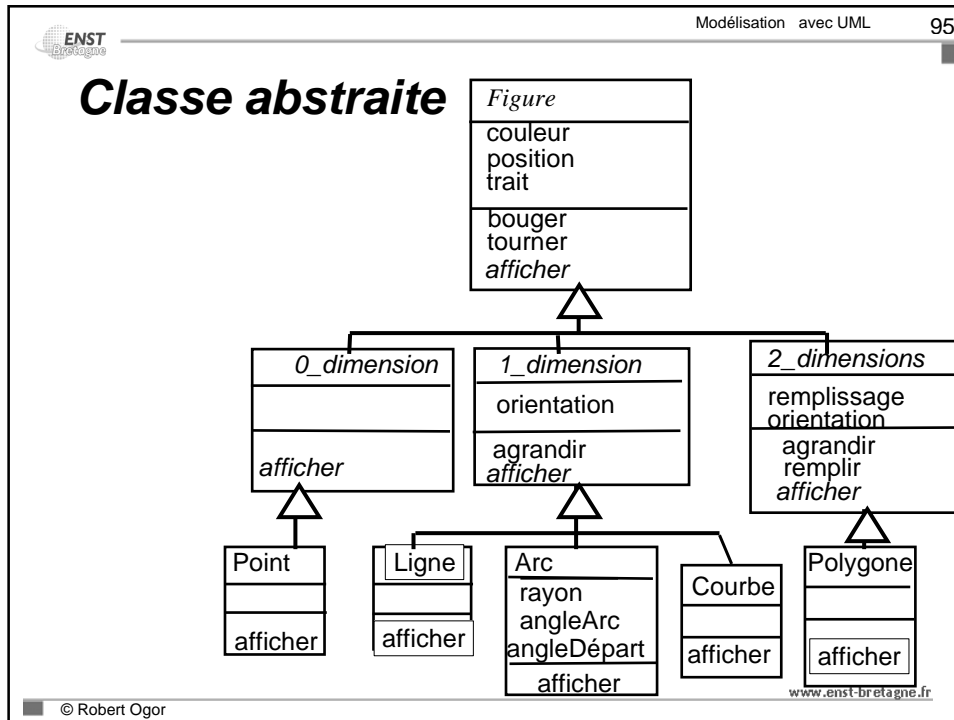
© Robert Ogor [www.enst-bretagne.fr](http://www.enst-bretagne.fr)

# Modélisation avec UML









ENST Bretagne Modélisation avec UML 96

## Héritage : buts et principes 3/3

### Polymorphisme

**BUT 3**

Créer des sous-types (sous-classes). Une sous-classe sera du même type que la classe dont elle hérite (super-classe).

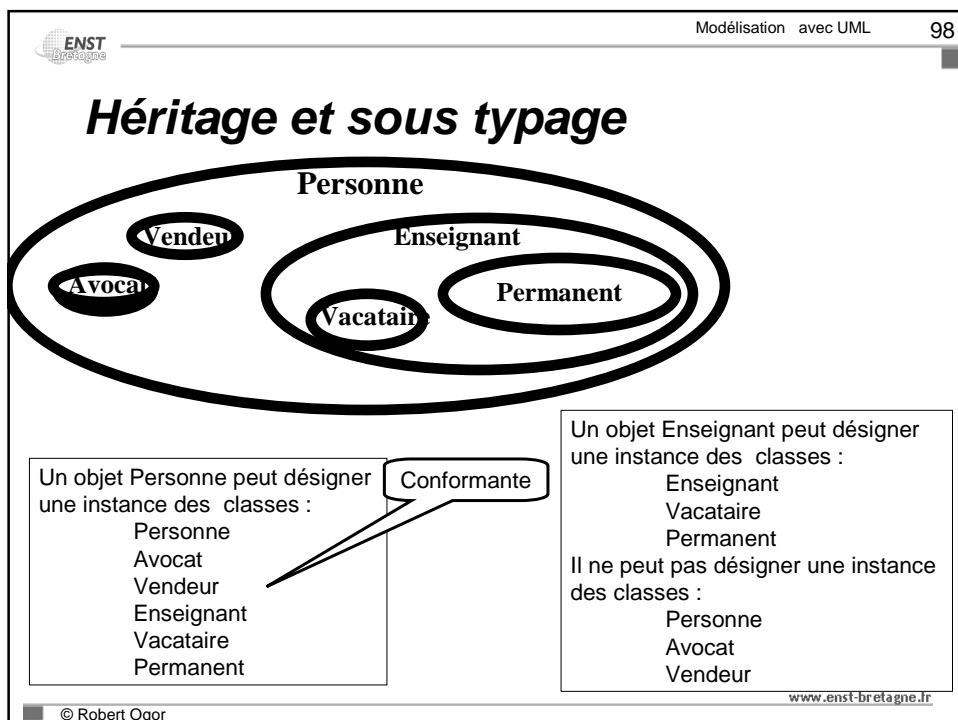
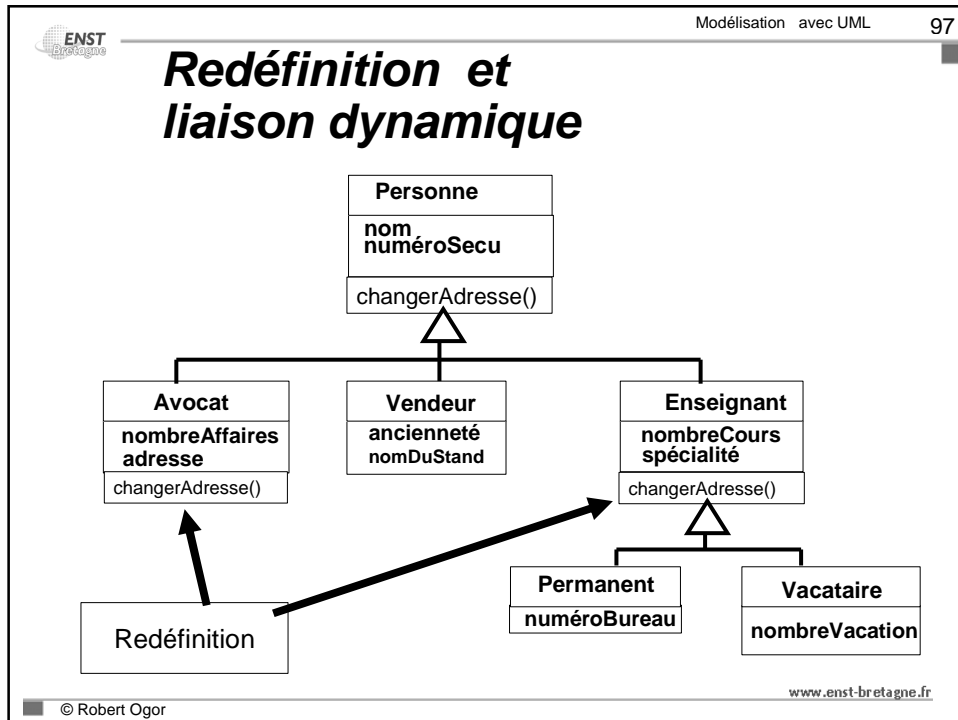
Ceci permet de mettre en œuvre le polymorphisme et la liaison dynamique

**PRINCIPE**


Un objet d'une classe donnée pourra toujours faire référence à des objets de ses sous classes (polymorphisme ).

Une opération exécutée par un objet sera celle que connaît l'objet dont il fait référence (liaison dynamique).

© Robert Ogor [www.enst-bretagne.fr](http://www.enst-bretagne.fr)






Modélisation avec UML99

## Conformante

- Première définition :
  - Se dit d'une classe plus spécialisée
- Exemple
  - Un Enseignant est conformant à une Personne
- Utilisation
  - Un objet conformant à un autre peut être utilisé à sa place sans pour autant déclencher d'erreur de type
  - étant plus spécialisé, il hérite de tous les services fournis par sa super-classe...

© Robert Ogorwww.enst-bretagne.fr

Modélisation avec UML100

## Exemple de conformante

```
void acheterMaison(Personne acheteur)
{...
    acheteur.changerAdresse();
};
```

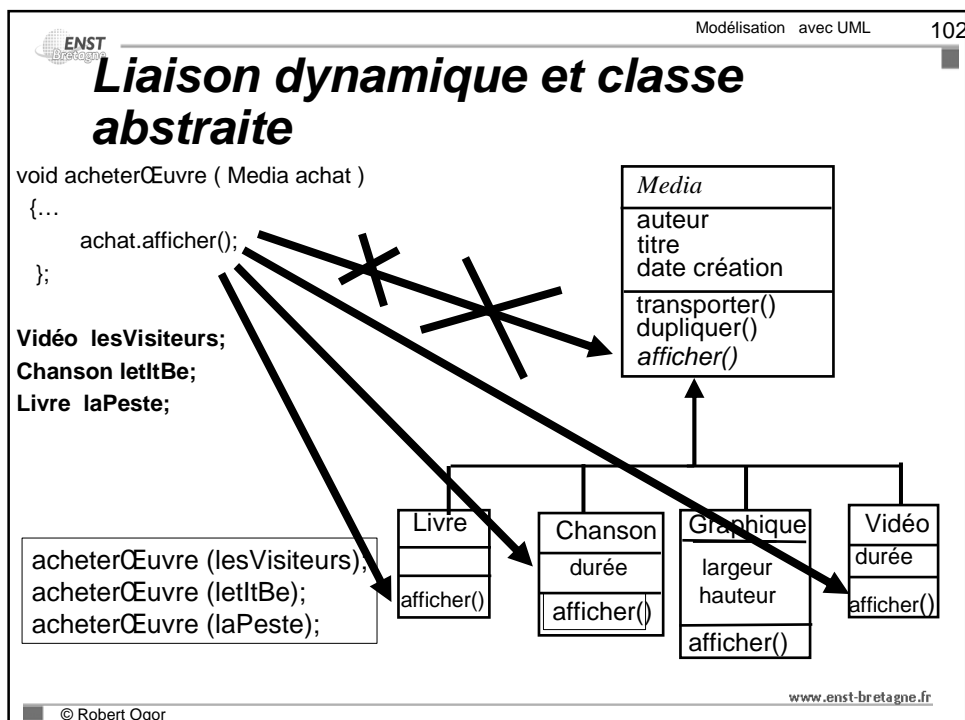
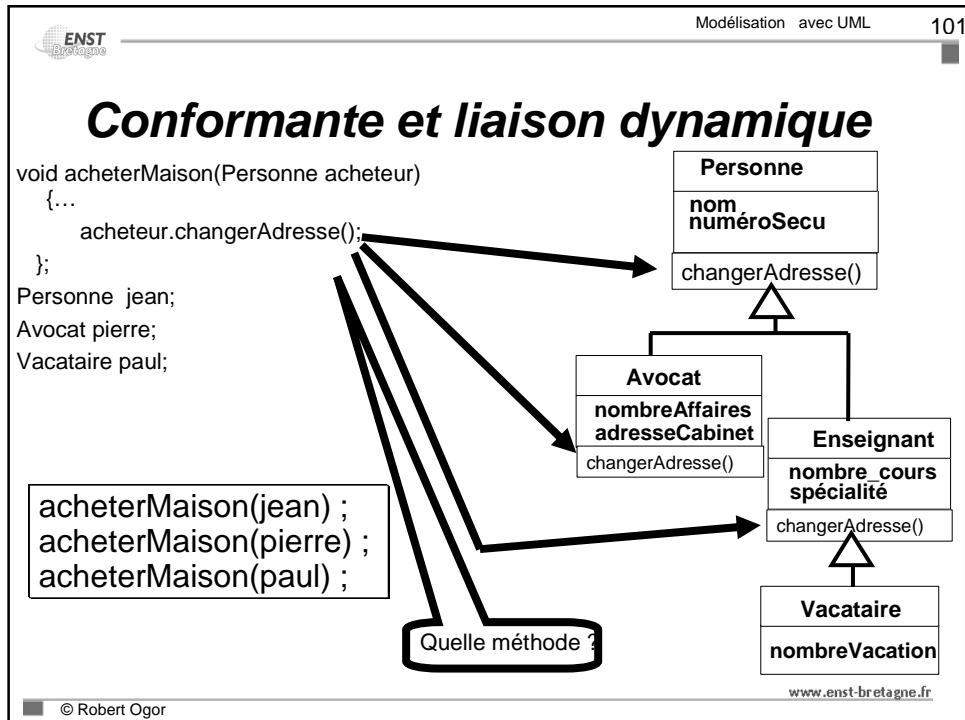
Le paramètre peut être abstrait ou concret

Personne jean;  
Avocat pierre;  
Vacataire paul;

acheterMaison(jean) ; // licite car jean est une Personne  
acheterMaison(pierre) ; // licite car pierre est conformant à Personne  
acheterMaison(paul) ; // licite car paul est conformant à Personne

© Robert Ogorwww.enst-bretagne.fr

# Modélisation avec UML



ENST  
Bretagne

Modélisation avec UML

103

## Les Interfaces

- Une **interface** permet de décrire le **comportement** d'une entité (classe, paquetage ou composant), c'est à dire un savoir faire sous la forme d'une **liste d'opérations**.
- Une interface ne peut donner lieu à **aucune implémentation**.
- Une interface est équivalente à une classe abstraite sans attributs où toutes les méthodes sont abstraites.
- Une **classe** peut déclarer qu'elle **implémente une interface**. Elle doit alors implémenter toutes les opérations de cette interface. Elle peut ensuite être utilisée partout où ce comportement est exigé.

© Robert Ogor

www.enst-bretagne.fr

ENST  
Bretagne

Modélisation avec UML

104

## Interface

opération,  
méthode,  
service  
sans corps

**<<interface>>  
Déplaçable**

saPlace ()  
avancer ()  
reculer ()  
monter ()  
descendre ()

Une interface n'est PAS une classe  
C'est une liste de services

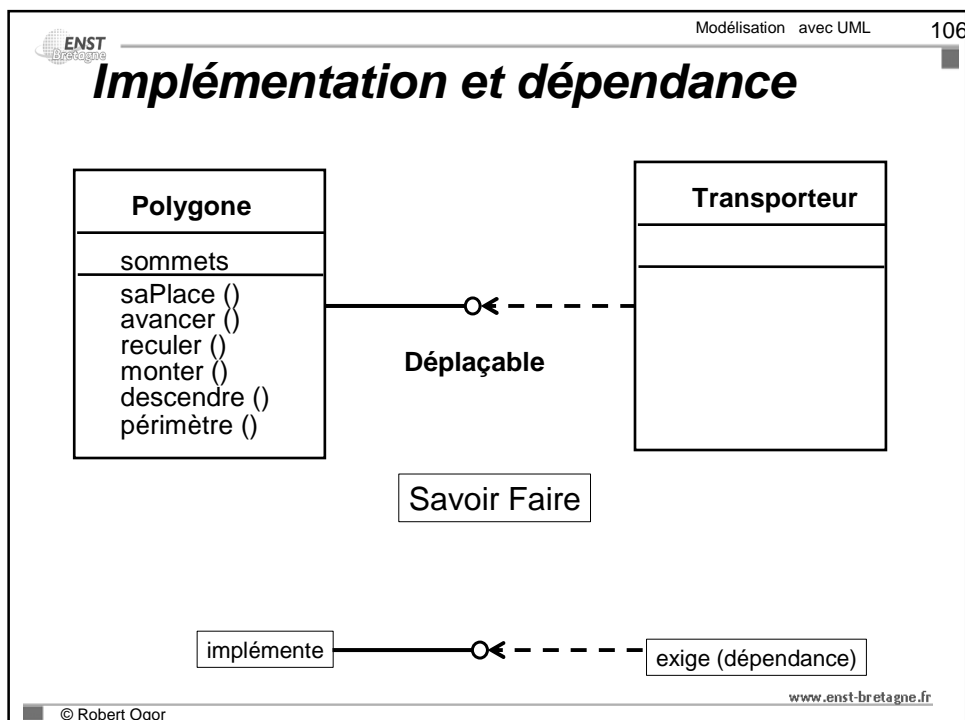
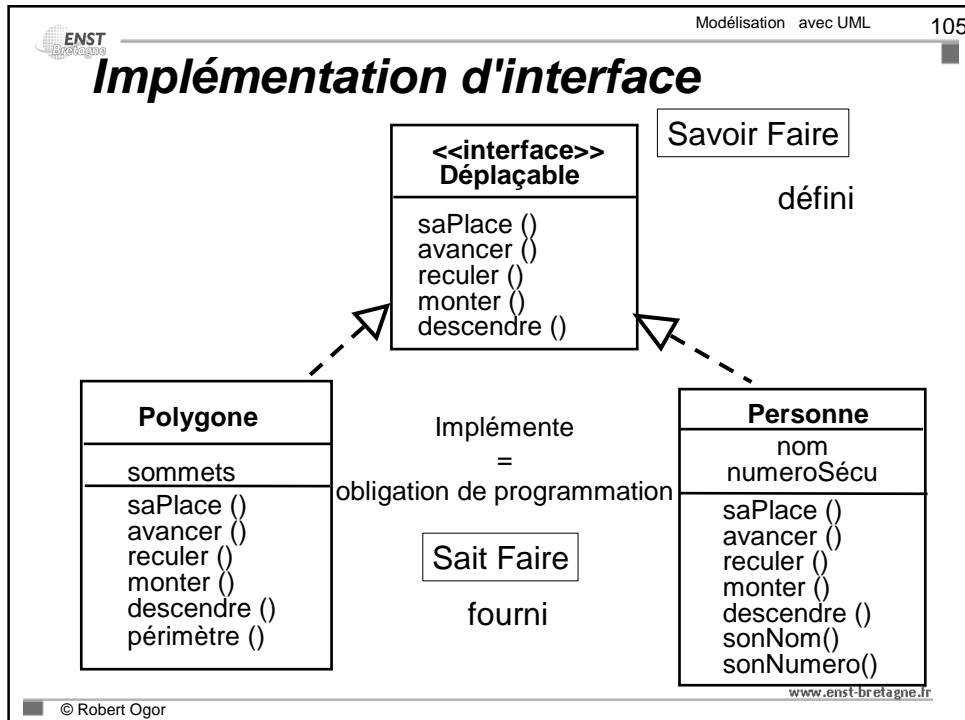
Elle ne peut pas servir à créer  
un objet

Une interface exprime un savoir faire

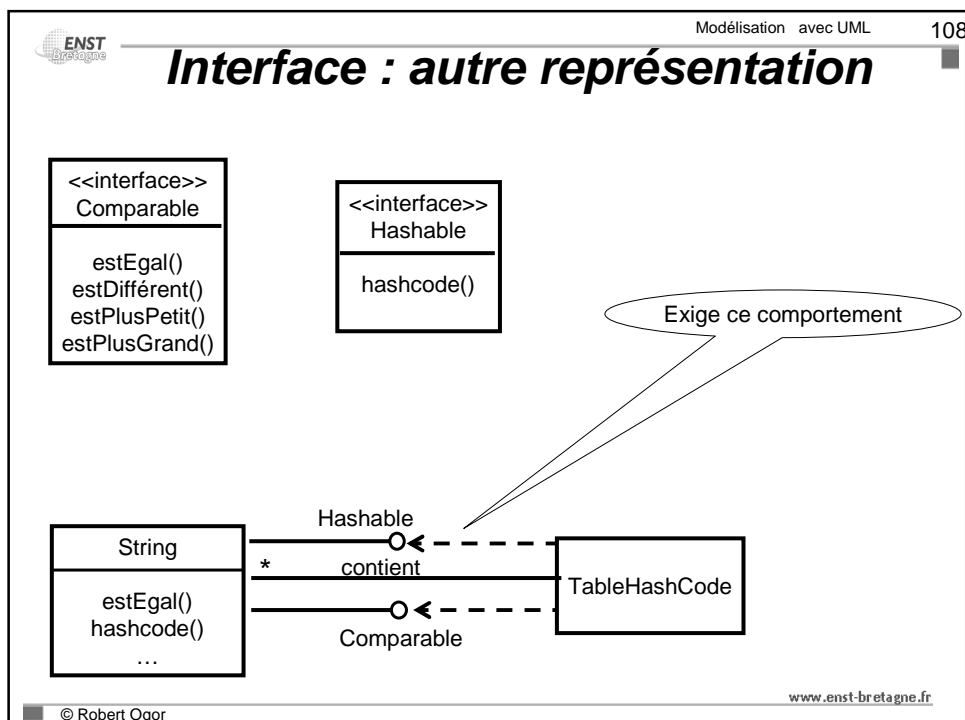
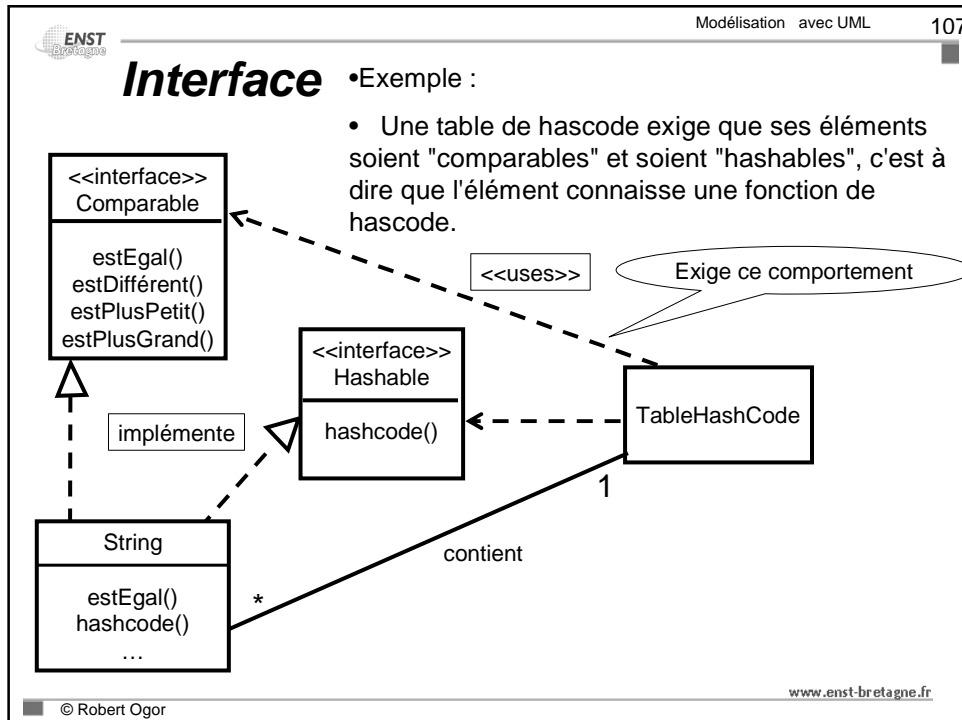
TYPE = CLASSE + INTERFACE

© Robert Ogor

www.enst-bretagne.fr



# Modélisation avec UML



# Modélisation avec UML

ENST  
Bretagne

Modélisation avec UML

109

## Utilisation d'interface

Une interface s'utilise pour désigner un objet.  
Il définit l'ensemble des services que doit savoir cet objet.

```
void manipuler(Déplaçable element)
{ ...
    element.avancer(8);
    element.monter(4);
... };
```

**<<interface>>  
Déplaçable**

Polygone

Personne

Personne et Polygone savent faire les opérations définies dans Déplaçable

Personne jean;  
Polygone poly;

obj. manipuler(jean) ; // licite car Personne implémente Déplaçable

obj. manipuler(poly) ; // licite car Polygone implémente Déplaçable

© Robert Ogor

www.enst-bretagne.fr

ENST  
Bretagne

Modélisation avec UML

110

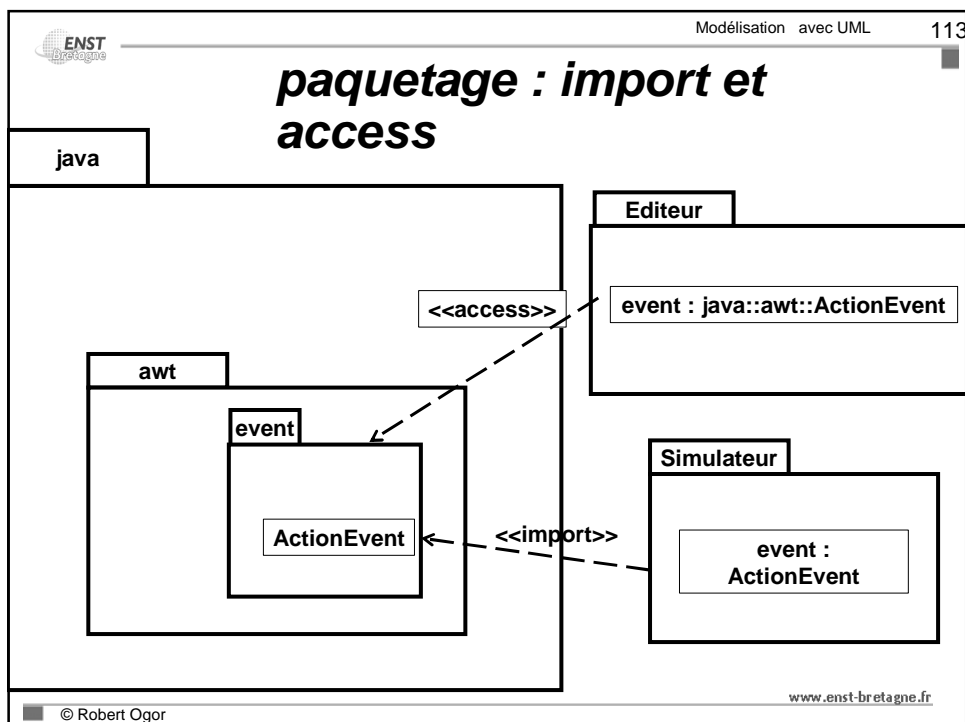
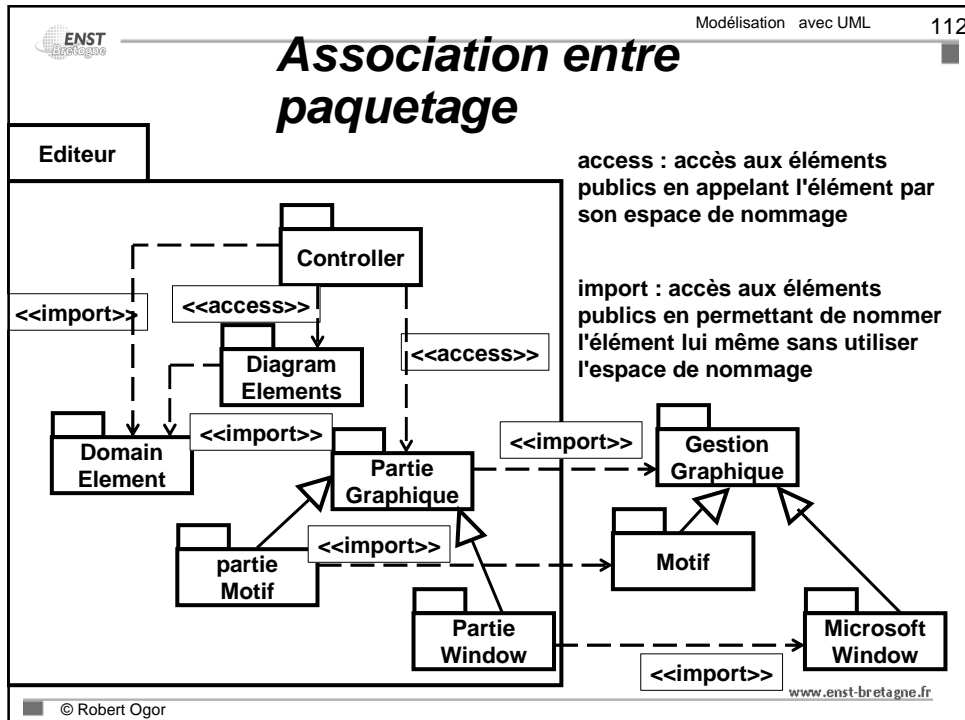
## Les Paquetages


- Une application est constituée de plusieurs classes, des dizaines ou des centaines. Il est important de les organiser en groupes (en fonction de certains critères surtout logiques).
- C'est le paquetage (package) qui permet ce regroupement.
- Un paquetage regroupe des classes, des interfaces, des paquetages.
- Il permet d'encapsuler certains éléments de la modélisation. Un élément du paquetage peut être inaccessible de l'extérieur du paquetage, il n'est alors connu que par les éléments du même paquetage.
- Il met en œuvre un espace de nommage

© Robert Ogor

www.enst-bretagne.fr


# Modélisation avec UML




Modélisation avec UML115

***Diagrammes de classes de la gestion de bibliothèque, recherche à partir du cahier des charges.***


***Recherche par responsabilité***

© Robert Ogor[www.enst-bretagne.fr](http://www.enst-bretagne.fr)

Modélisation avec UML116


***Phases de la modélisation objet***

- Identifier les classes candidates.
- Préparer le dictionnaire de données : classes retenues.
- Identifier les associations entre classes (en incluant les agrégations).
- Identifier les attributs.
- Organiser et simplifier les classes en utilisant l'héritage.
- Supprimer les associations inutiles
- Vérifier que le diagramme inclut toutes les demandes du cahier des charges.
- Itérer et affiner le modèle.
- Grouper les classes en modules (paquetages).

© Robert Ogor[www.enst-bretagne.fr](http://www.enst-bretagne.fr)



# Modélisation avec UML



Modélisation avec UML


117

## Identifier les classes : les classes candidates

- Un gérant de bibliothèque désire automatiser la gestion des prêts.
- Il commande un logiciel permettant aux utilisateurs de connaître les livres présents, d'en réserver jusqu'à 2. L'adhérent peut connaître la liste des livres qu'il a empruntés ou réservés.
- L'adhérent possède un mot de passe qui lui est donné à son inscription.
- L'emprunt est toujours réalisé par les employés qui travaillent à la bibliothèque. Après avoir identifié l'emprunteur, ils savent si le prêt est possible (nombre max de prêts = 5), et s'il a la priorité (il est celui qui a réservé le livre).
- Ce sont les employés qui mettent en bibliothèque les livres rendus et les nouveaux livres. Il leur est possible de connaître l'ensemble des prêts réalisés dans la bibliothèque

Gérant bibliothèque gestion prêts logiciel utilisateurs  
 livres  
 adhérent liste mot de passe inscription emprunt  
 employés emprunteur ensemble

© Robert Ogor



Modélisation avec UML

118

## Les classes retenues

■ Gérant	non pertinente, n'intervient pas
■ <u>bibliothèque</u>	<u>oui</u> responsabilité : gérer les livres, adhérents, prêts
■ gestion	non vague
■ <u>prêts</u> les prêts	<u>oui</u> responsabilité : contenir les infos et actions sur
■ logiciel	non vague
■ utilisateurs	(choix entre utilisateur, adhérent, emprunteur )
■ <u>livres</u>	<u>oui</u> responsabilité : permettre de connaître son état
■ <u>adhérent</u> identifiée	<u>oui</u> responsabilité : permettre à la personne d'être
■ liste	non implémentation ou conception
■ mot de passe	non attribut
■ Inscription	non action
■ emprunt	non action
■ <u>employés</u>	<u>oui</u> responsabilité : reconnaître qui a fait un prêt, etc..
■ emprunteur	(choix entre utilisateur, adhérent, emprunteur )
■ Ensemble	non implémentation ou conception

© Robert Ogor
[www.enst-bretagne.fr](http://www.enst-bretagne.fr)

ENST  
Bretagne

Modélisation avec UML 119

## Dictionnaire des données

- bibliothèque : organisme gérant une collection de livres qui peuvent être empruntés par ses adhérents. Une bibliothèque est gérée par ses employés.
- prêt : un prêt est caractérisé par le numéro du livre, la date, la durée. Il ne peut être fait que par un adhérent.
- livre ouvrage pouvant être emprunté.
- adhérent personne inscrite à la bibliothèque.
- employé personne travaillant à la bibliothèque.

© Robert Ogor

www.enst-bretagne.fr

ENST  
Bretagne

Modélisation avec UML 120

## Chercher les associations

- Un gérant de bibliothèque désire automatiser la gestion des prêts.
- Il commande un logiciel permettant aux utilisateurs de connaître les livres présents, d'en réserver jusqu'à 2. L'adhérent peut connaître la liste des livres qu'il a empruntés ou réservés.
- L'adhérent possède un mot de passe qui lui est donné à son inscription.
- L'emprunt est toujours réalisé par les employés qui travaillent à la bibliothèque. Après avoir identifié l'emprunteur, ils savent si le prêt est possible (nombre max de prêts = 5), et s'il a la priorité (il est celui qui a réservé le livre).
- Ce sont les employés qui mettent en bibliothèque les livres rendus et les nouveaux livres. Il leur est possible de connaître l'ensemble des prêts réalisés dans la bibliothèque

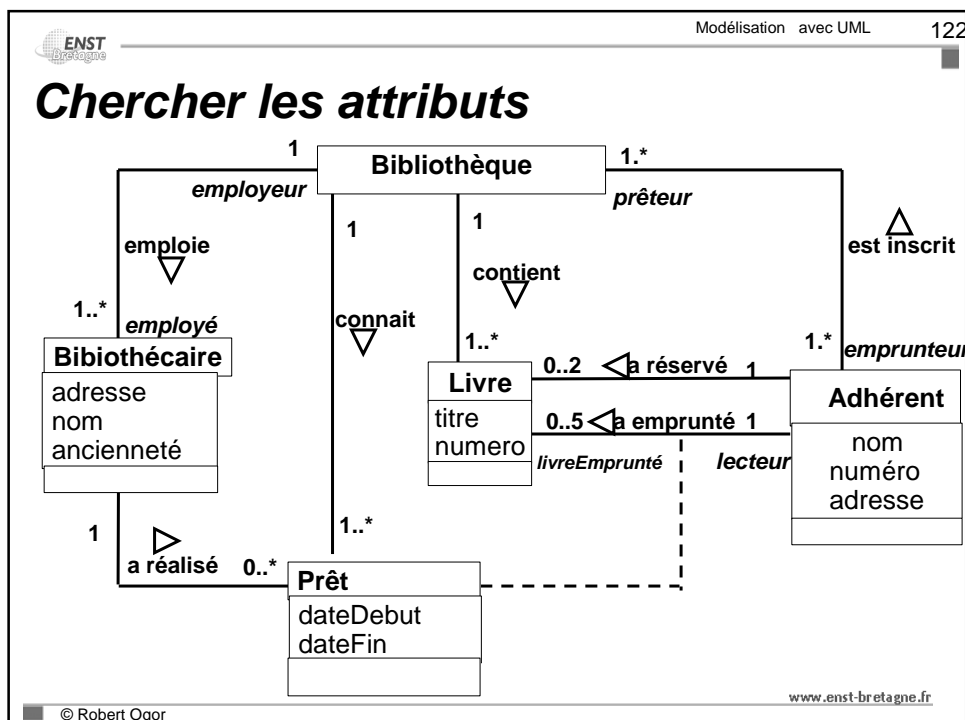
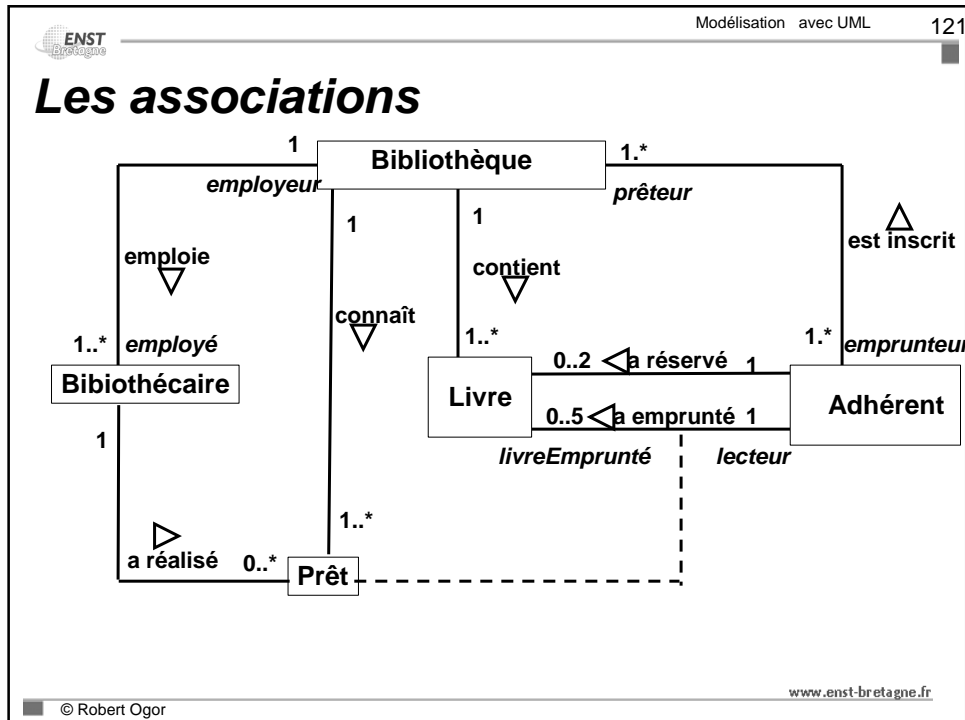
Associations sous entendues

Une adhérent est inscrit à la bibliothèque.  
La bibliothèque contient des livres

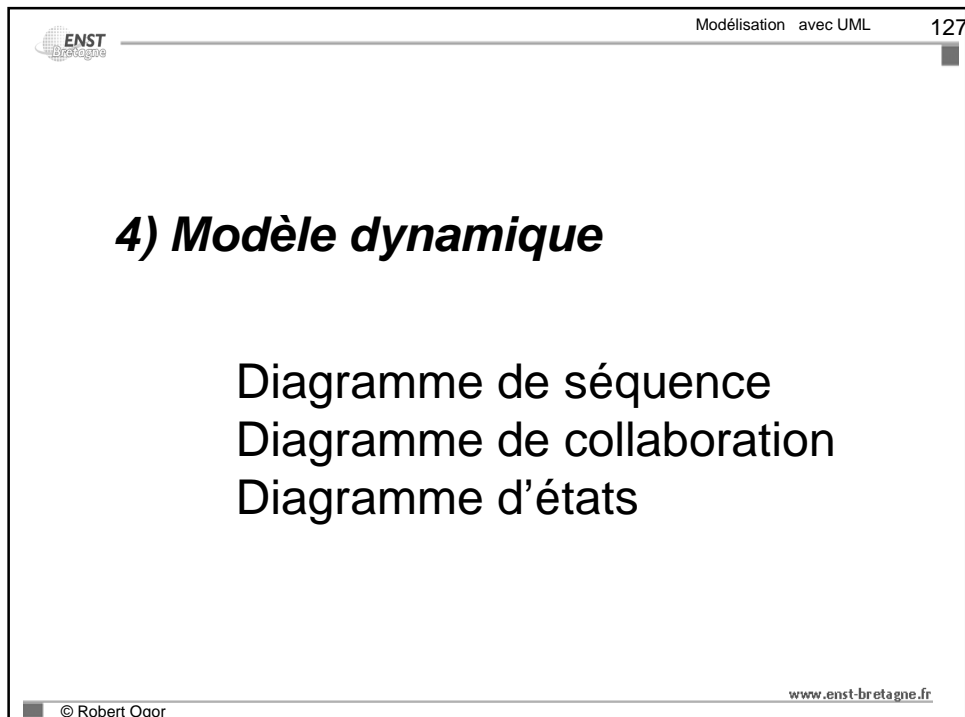
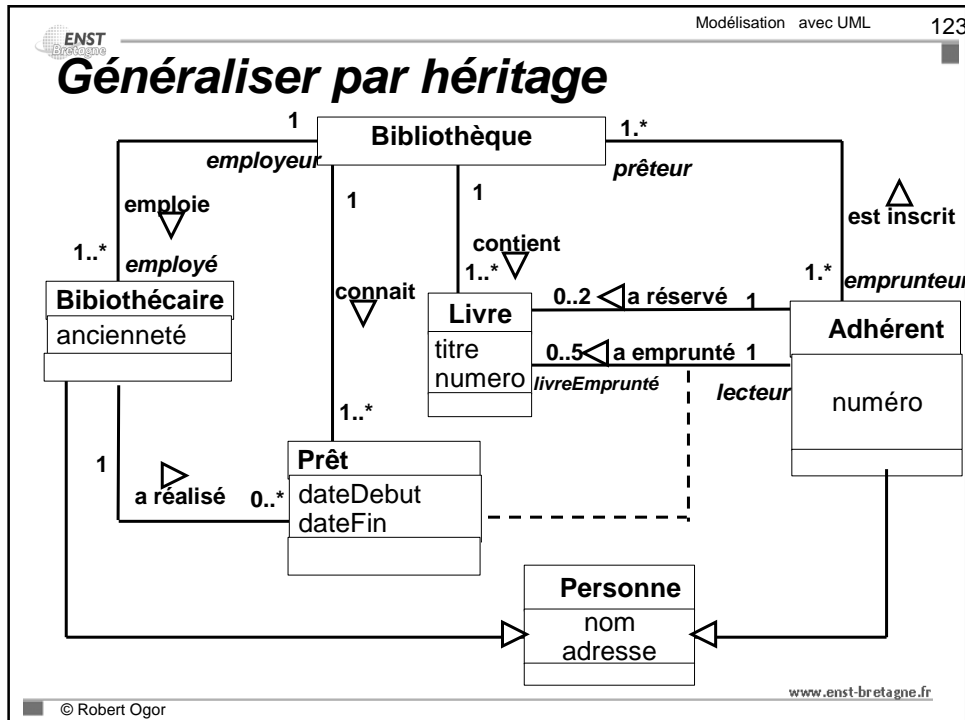
© Robert Ogor

www.enst-bretagne.fr

# Modélisation avec UML



# Modélisation avec UML



ENST  
Bretagne

Modélisation avec UML

128

## Modèle dynamique :

Le modèle dynamique montre le comportement du système et l'évolution des objets dans le temps.

Le diagramme illustre un modèle dynamique. À gauche, un acteur (figure humaine) est représenté. À droite, un système est représenté par un ovale contenant plusieurs objets (rectangles) reliés par des flèches indiquant des interactions ou des flux de données. Une flèche pointe de l'acteur vers le système, et une autre pointe du système vers l'acteur.

Le modèle dynamique va identifier les différents événements venant du monde externe et montrer l'enchaînement dans le système que provoquent ces événements externes.

**événement :**

Quelque chose qui se produit à un moment donné dans le temps, et qui n'a pas de durée.

**exemples :** *l'utilisateur décroche son téléphone, le conducteur appuie sur un bouton.*

© Robert Ogor

[www.enst-bretagne.fr](http://www.enst-bretagne.fr)

ENST  
Bretagne

Modélisation avec UML

129

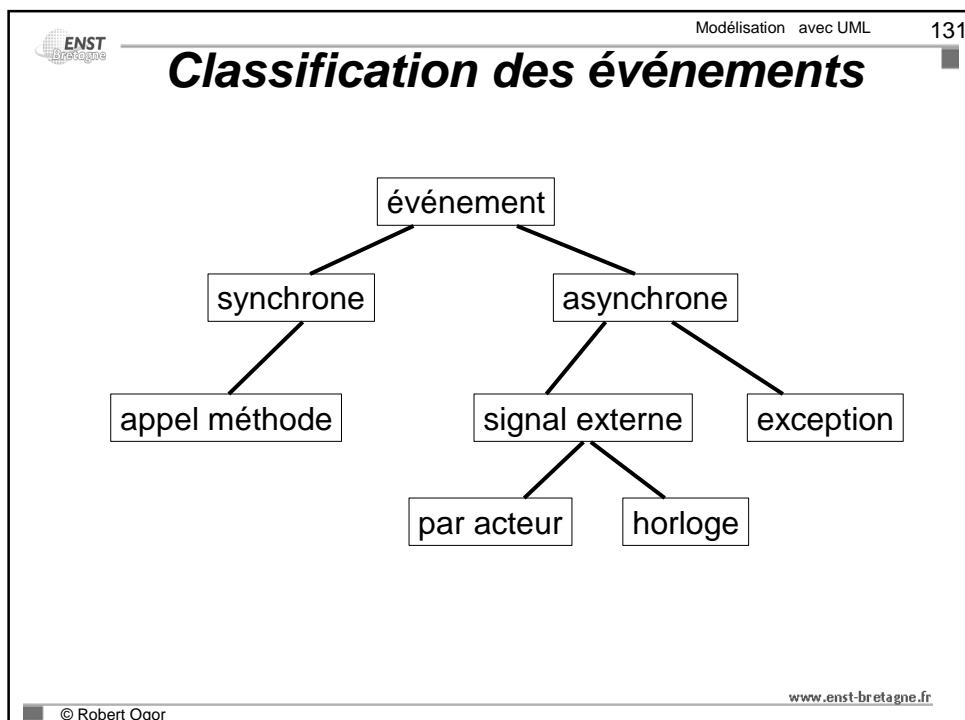
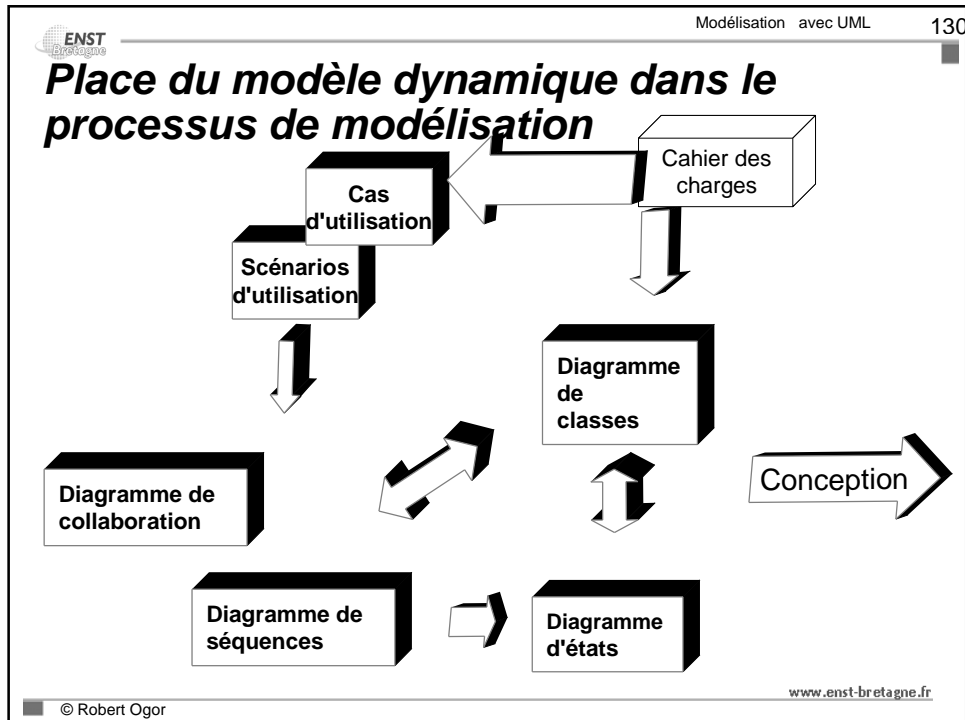
## But du modèle dynamique


- Trouver les relations temporelles et événementielles entre les objets.
- Définir les états des objets qui déterminent une réaction différente face à un événement.
- Trouver les actions effectuées par les objets suite à la réception d'événements

© Robert Ogor





[www.enst-bretagne.fr](http://www.enst-bretagne.fr)


# Modélisation avec UML




Modélisation avec UML132

## Les types de messages


synchrone 	L'expéditeur est bloqué pendant le traitement du message par l'expéditeur.
retour d'appel 	Un message synchrone peut être un appel de procédure, le retour peut être représenté (optionnel, le retour est implicite)
asynchrone  	L'expéditeur continue son exécution pendant le traitement du message
minuté	Comme le synchrone, mais un chien de garde est positionné, c'est à dire que l'expéditeur se réveille au bout d'un certain temps s'il ne reçoit pas de réponse.


© Robert Ogorwww.enst-bretagne.fr

Modélisation avec UML133

## Diagramme de collaboration ou diagramme de séquences

- Représente la **collaboration entre objets** pour la réalisation d'un cas d'utilisation ou la réalisation d'une opération.
- Met en évidence l'expéditeur et le récepteur de l'événement.
- Chaque **événement** reçu par un objet devra se traduire par une **opération** pour le gérer dans le diagramme de classes.

© Robert Ogorwww.enst-bretagne.fr



Modélisation avec UML

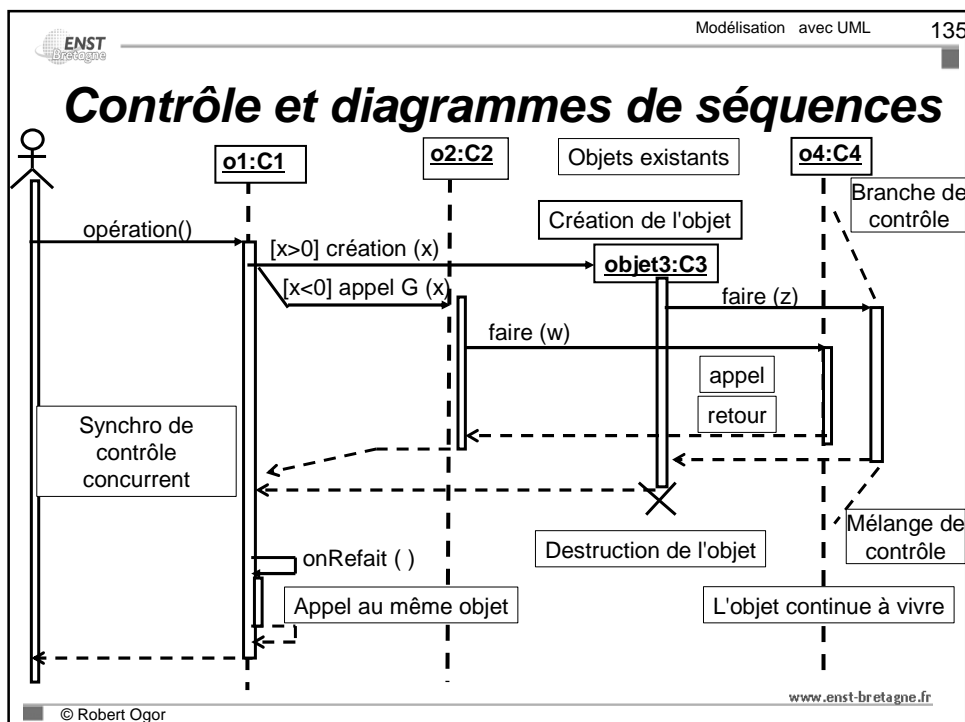
134

## Diagramme de séquence

- Met en évidence l'aspect temporel (haut vers le bas)
- Un objet a une ligne de vie représentée par une ligne verticale en pointillé.
- Une flèche reçue par un objet se traduit par l'exécution d'une opération.
- La durée de vie de l'opération est symbolisée par un rectangle.
- Certains objets vivent pendant tout le diagramme, d'autres sont activés et/ou meurent pendant la séquence.
- Il est possible de définir des choix et des itérations.

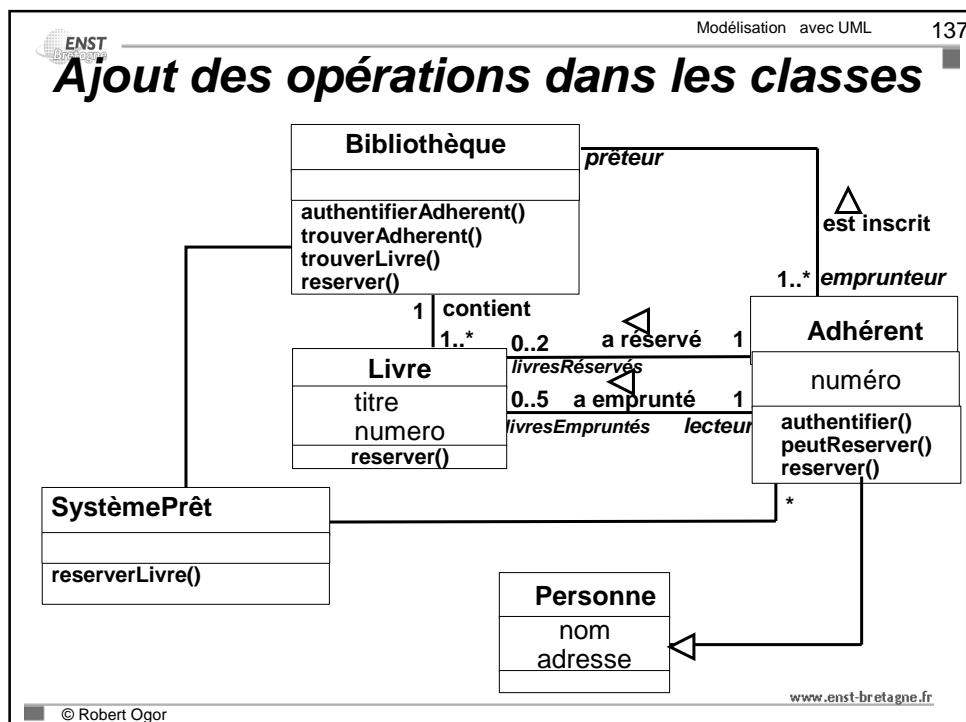
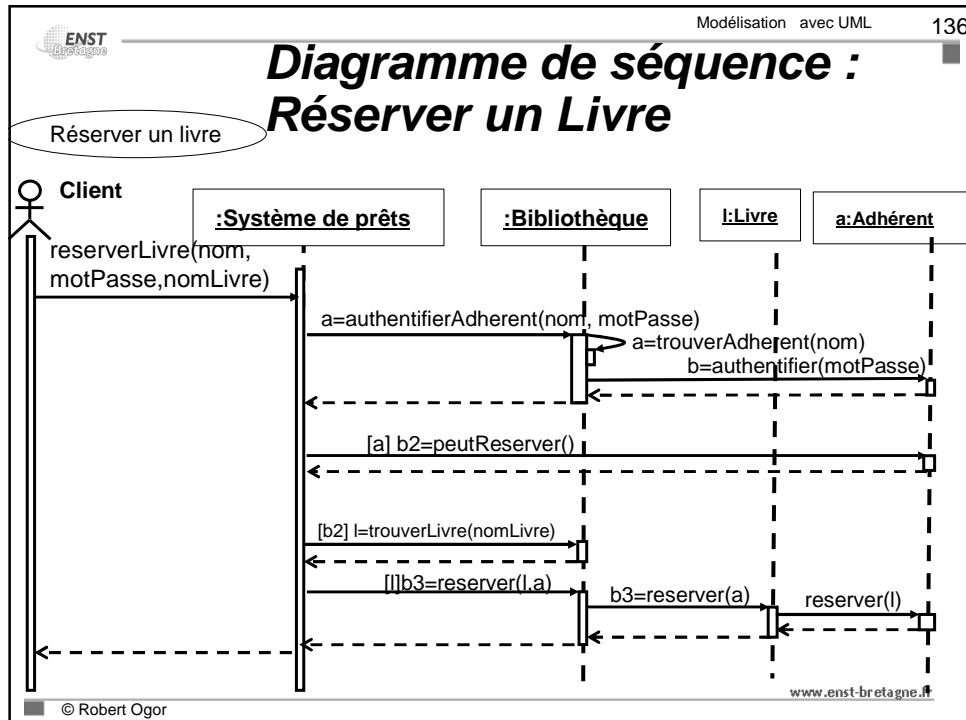
© Robert Ogor

www.enst-bretagne.fr





# Modélisation avec UML



ENST  
Bretagne

Modélisation avec UML

140

***Les Diagrammes de collaborations entre classes***

© Robert Ogor

www.enst-bretagne.fr

ENST  
Bretagne

Modélisation avec UML

141

***Diagramme de collaborations***

- Le diagramme de collaborations sous une forme distincte du diagramme de séquences représente les interactions entre classes en mettant moins en évidence l'aspect temporel.
- Ce modèle explique la coopération entre objets pour la réalisation d'une fonctionnalité, cette coopération implique les différents événements qui se propagent d'un objet à l'autre. Un objet doit avoir une méthode appropriée pour traiter chaque événement qu'il reçoit (message).
- L'aspect temporel n'est pas complètement caché car chaque message est numéroté.

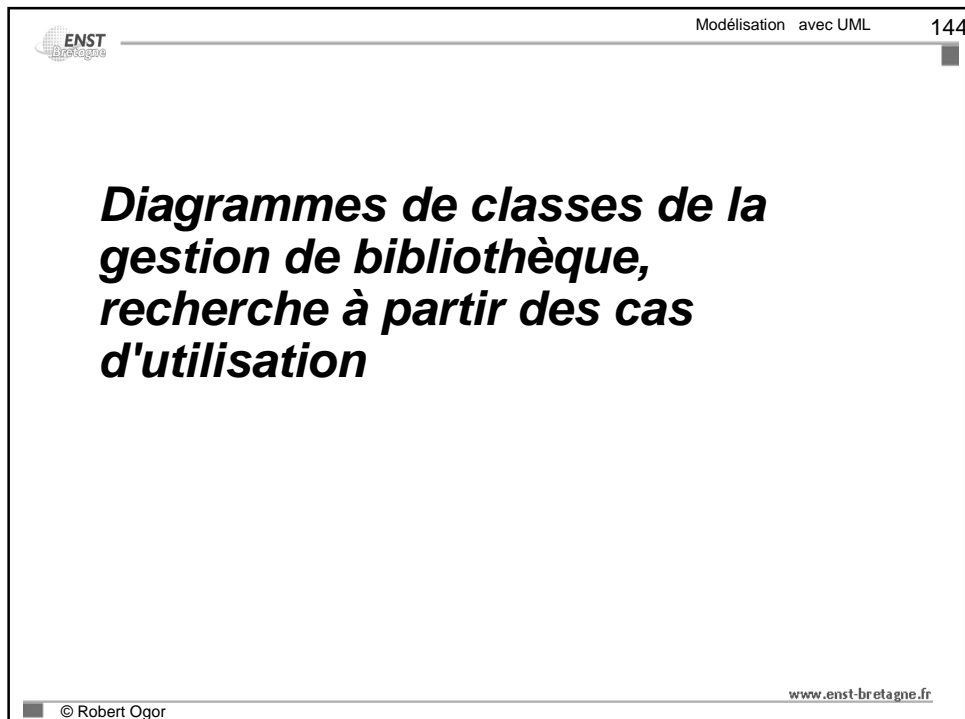
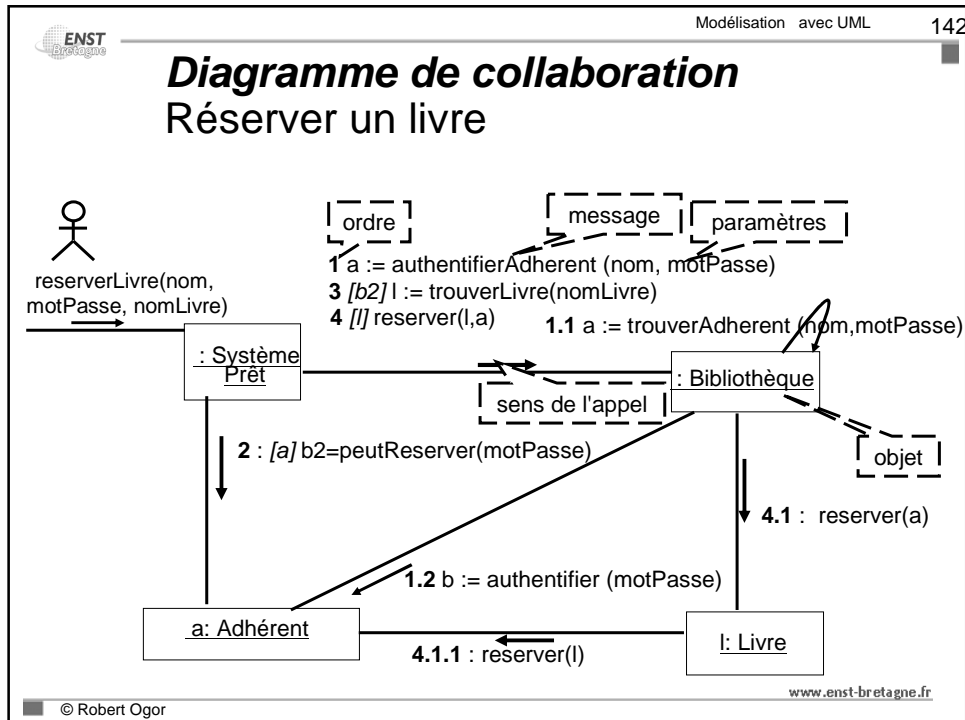
Le flot de données intervenant dans ces interactions peut être précisé.

```
graph LR
    Actor(( )) -- 1 --> O1[ ]
    O1 -- 2 --> O2[ ]
    O2 -- 3 --> O5[ ]
    O2 -- 4 --> O4[ ]
    O4 -- 5 --> O5
    O5 -- 6 --> Actor
```

© Robert Ogor

www.enst-bretagne.fr

# Modélisation avec UML



# Modélisation avec UML




ENST  
Bretagne

Modélisation avec UML

145

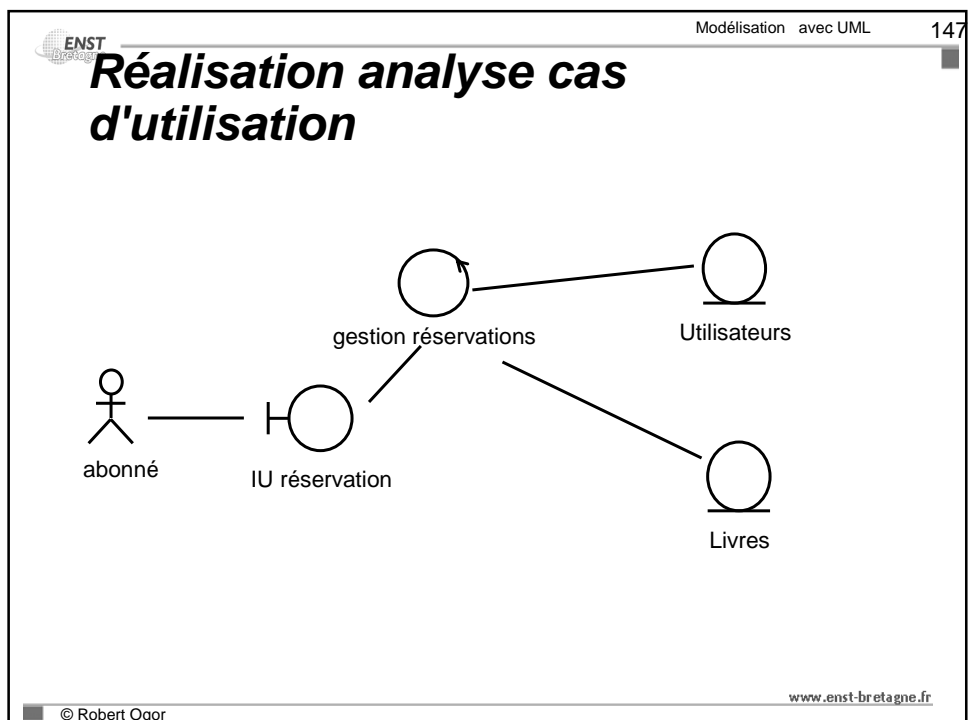
## Définition de classes d'analyses


On s'aperçoit que dans l'analyse d'un problèmes trois types de classes apparaissent couramment:

<b>classe frontière</b> La classe qui permet au système de communiquer avec le monde réel, elle est à la frontière du système, elle se conçoit en général par une interface graphique, nous la représentons par l'icône suivante		réception Commande
<b>classe entité</b> La classe qui mémorise et gère des données, par exemples les livres présents, les prêts effectuées, nous la représentons par par l'icône suivante		stockage Commande
<b>classe contrôle</b> La classe qui réalise le contrôle nécessaire pour interpréter le scénario décrivant un cas d'utilisation		gestion Commande

© Robert Ogor

[www.enst-bretagne.fr](http://www.enst-bretagne.fr)



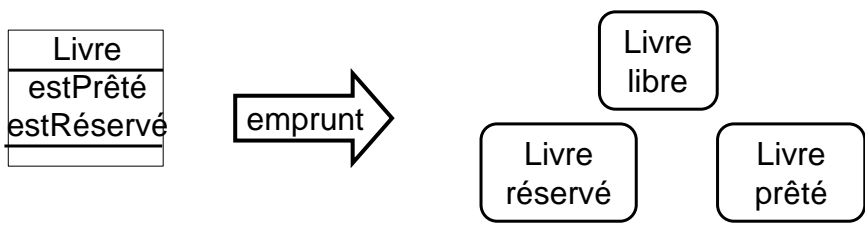


Modélisation avec UML

151

## Recherche des Etats


- L'état d'un objet est lié aux valeurs de ses variables d'instances et des interactions avec les autres objets. Il s'agit de ne conserver que les états significatifs.
- La réponse d'un objet à un événement dépend de l'état dans lequel cet objet se trouve.



Un objet passe dans un état donné par l'événement qui modifie ses variables, et quitte cet état par un autre événement qui les modifie à nouveau. Ce sont des transitions d'états.

[www.enst-bretagne.fr](http://www.enst-bretagne.fr)

© Robert Ogor

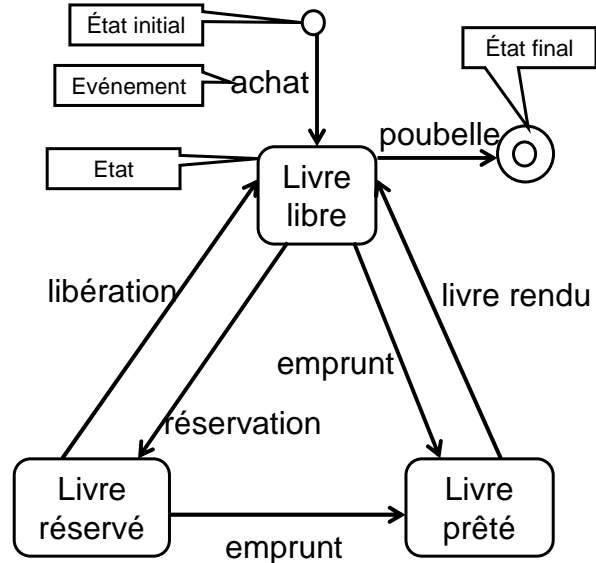


Modélisation avec UML

152

## Diagramme de transitions d'états

- Chaque transition est provoqué par un événement.
- Une transition n'a pas de durée.
- La durée d'un état est l'intervalle de temps qui s'écoule entre l'événement d'entrée et celui de départ.



[www.enst-bretagne.fr](http://www.enst-bretagne.fr)

© Robert Ogor

ENST Bretagne

Modélisation avec UML

153

## Activité liée à un état

Occupé

do : jouer tonalité occupé

Dans cet état, l'activité de la ligne téléphonique est d'émettre une tonalité.

Une activité est une opération associée à un état.

Une activité « do » se répète tant qu'on est dans l'état.

© Robert Ogor

www.enst-bretagne.fr

ENST Bretagne

Modélisation avec UML

154

## Action liée à un événement

```
graph TD
    etat[état] --> Connecté[Connecté]
    Connecté --> Déconnecté[Déconnecté]
    etat --> Déconnecté
    etat --> event[raccroche / déconnecter la ligne]
    event --> transition[transition]
    transition --> Déconnecté
    action[action] --> transition
```

La transition connecté vers déconnecté est déclenchée par l'événement extérieur « **le poste est raccroché** », une action est réalisée avant d'entrer dans l'état déconnecté : « **la ligne est déconnectée** ».

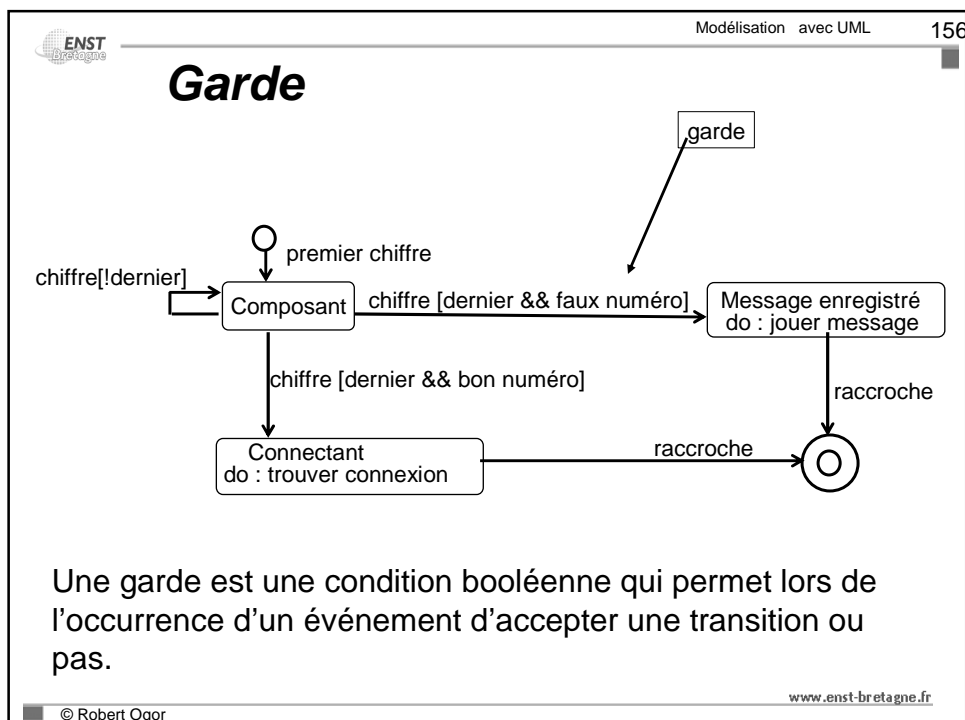
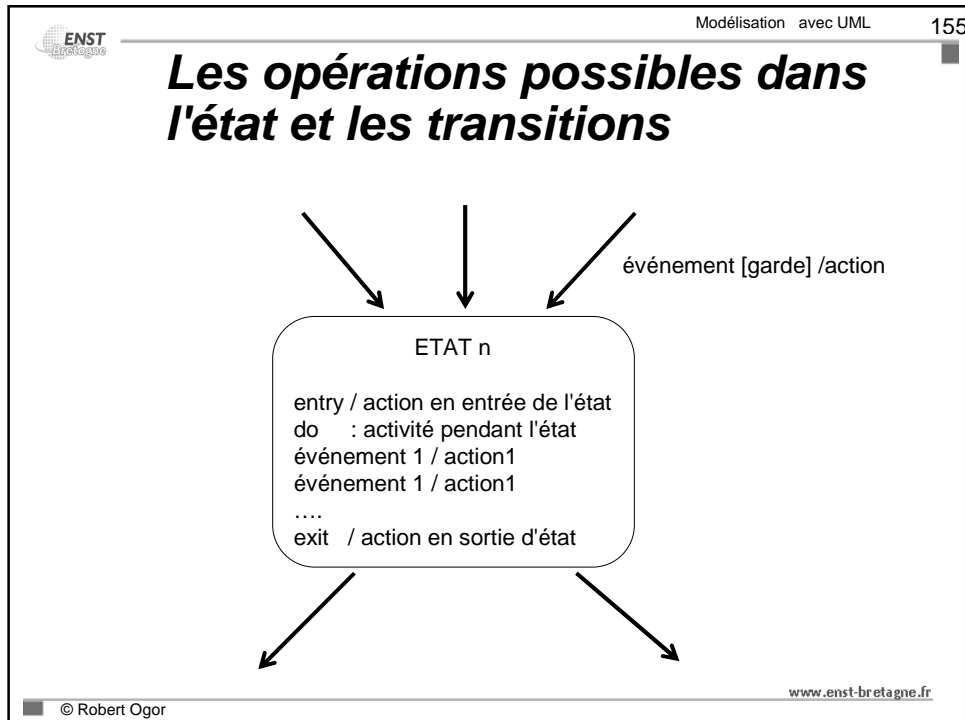
Une action n'a pas de durée, elle est associée à un événement.

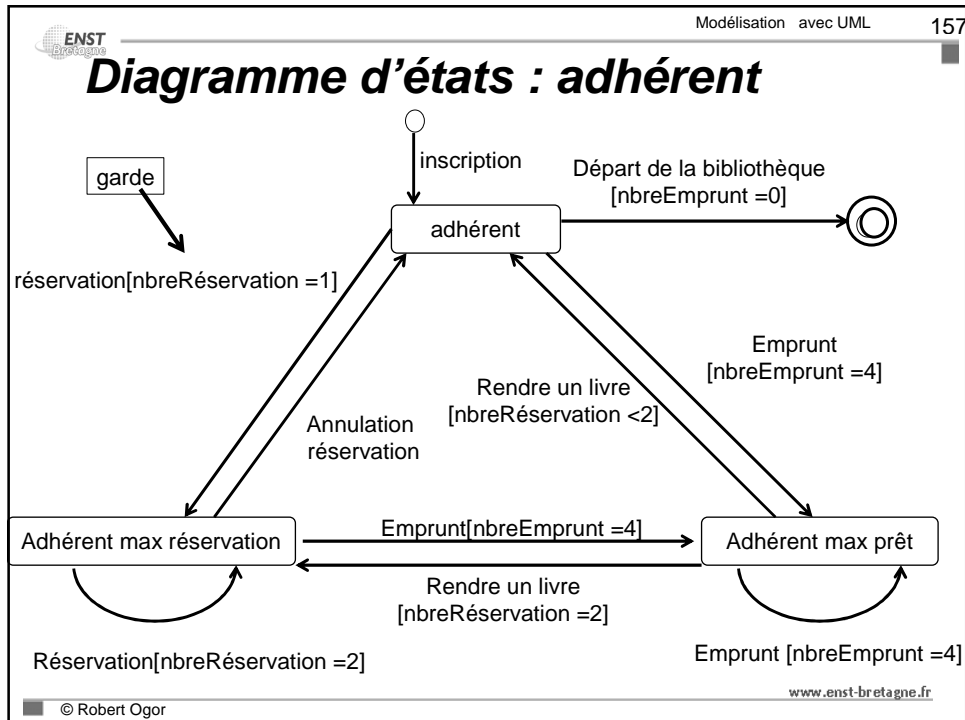
Le changement de valeur d'un attribut est un exemple d'action.

© Robert Ogor

www.enst-bretagne.fr

# Modélisation avec UML





ENST Bretagne Modélisation avec UML 173

## 5) Conception composants et déploiement

© Robert Ogor [www.enst-bretagne.fr](http://www.enst-bretagne.fr)



ENST  
Bretagne

Modélisation avec UML

174

## La notion de composant

Dans le monde du bâtiment, le modèle de l'architecte (logique) permet de visualiser, spécifier et documenter sur papier les caractéristiques de la future construction :

place des murs, des fenêtres, etc...

Lors de la construction, on utilise des composants fenêtres, portes, murs. Ce sont des choses physiques qui existent dans le monde réel.

Ils rendent des services mais définissent leurs exigences (taille, espace, etc.).

De même, dans un système informatique, le modèle logique dans une application permet de visualiser, spécifier et documenter la structure et le comportement des entités qui collaborent.

La construction va s'appuyer sur des composants qui existent dans le monde des ordinateurs : librairies, fichiers, tables, documents exécutables, etc

Un composant définit les services qu'il rend et aussi les services dont il est en attente pour pouvoir fonctionner.

© Robert Ogor

www.enst-bretagne.fr

ENST  
Bretagne

Modélisation avec UML

175

## Définition des composants

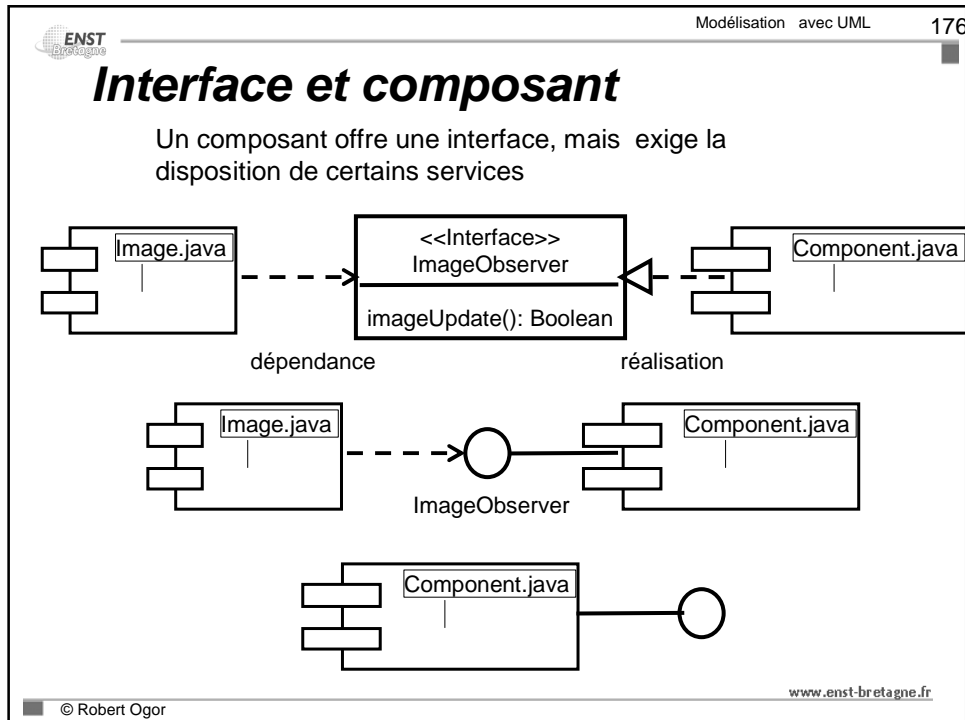
- Un composant est une partie physique et remplaçable d'un système qui sait faire et fournit la réalisation d'un ensemble d'interfaces.

Diagram illustrating components and their dependencies:

- Components: Noyau win32, composant COM+, beans Java.
- Files: Hello.html, Hello.class, Hello.java, Hello.jpg.
- Relationships: Hello.html depends on Hello.class (dashed arrow). Hello.class depends on Hello.jpg (dashed arrow, labeled 'dépendance'). Hello.class is associated with Hello.java (dashed arrow).

© Robert Ogor

www.enst-bretagne.fr



ENST  
Bretagne

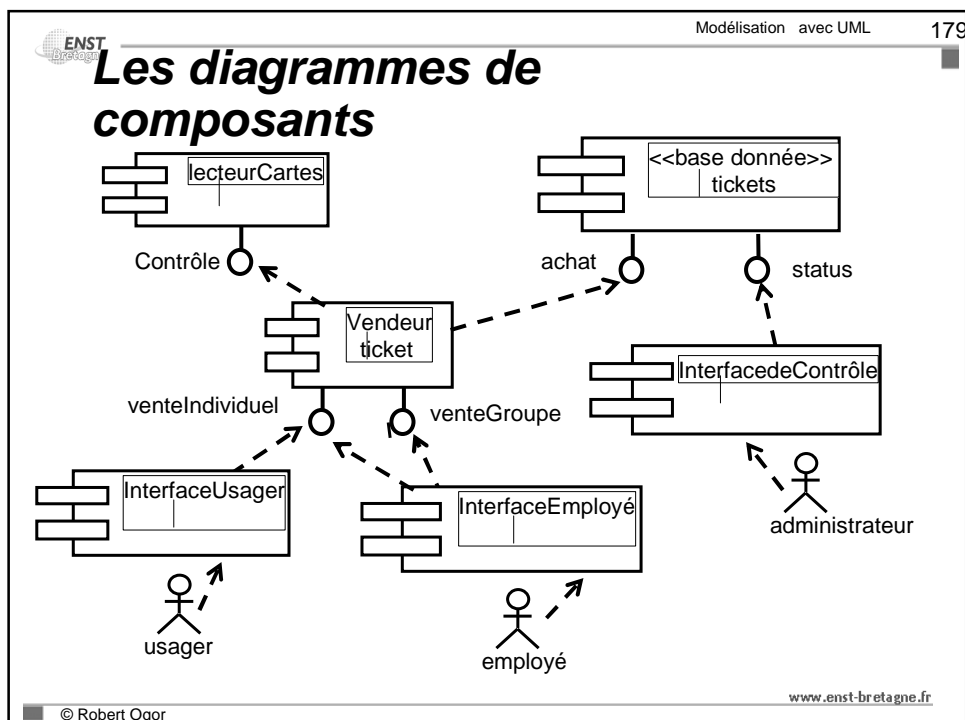
Modélisation avec UML 178

## Les diagrammes de composants


- Ce modèle définit l'architecture logicielle du système dans un environnement de développement donné.
- Il est issu de la conception et permet de représenter le système et les sous systèmes du modèle physique de l'architecture logicielle à réaliser.
- Un système ou un sous système définit un espace de visibilité et regroupe des classes.
- Tout les langages ne supporte pas cette notion de système mais elle existe sous forme de "package" en ADA.

© Robert Ogor


www.enst-bretagne.fr




# Modélisation avec UML

Modélisation avec UML180


## ***Les diagrammes de déploiement***

© Robert Ogor[www.enst-bretagne.fr](http://www.enst-bretagne.fr)

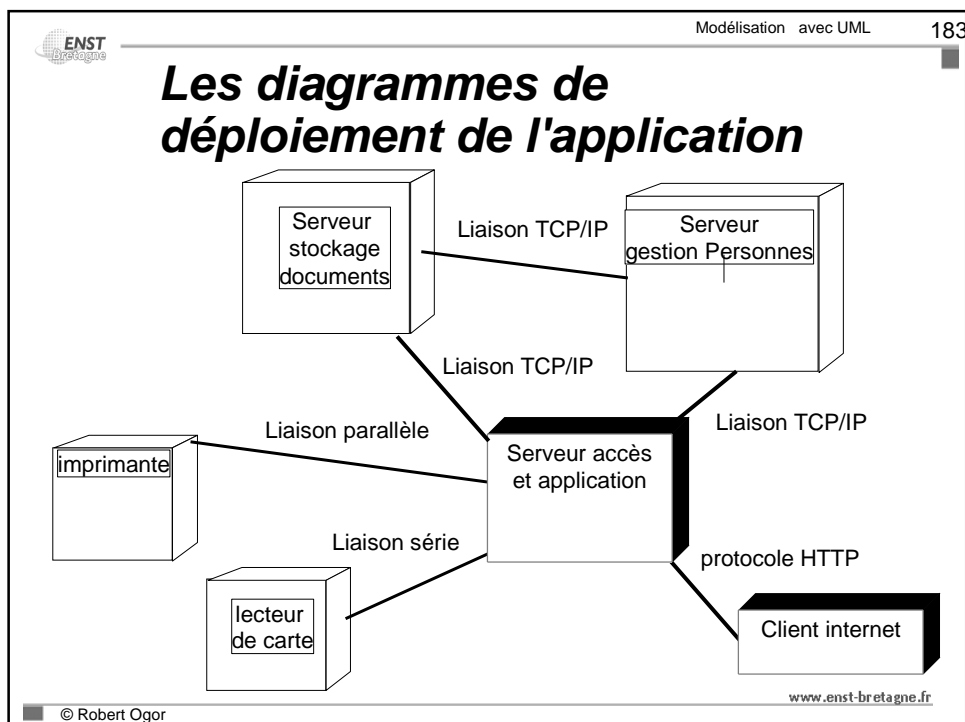
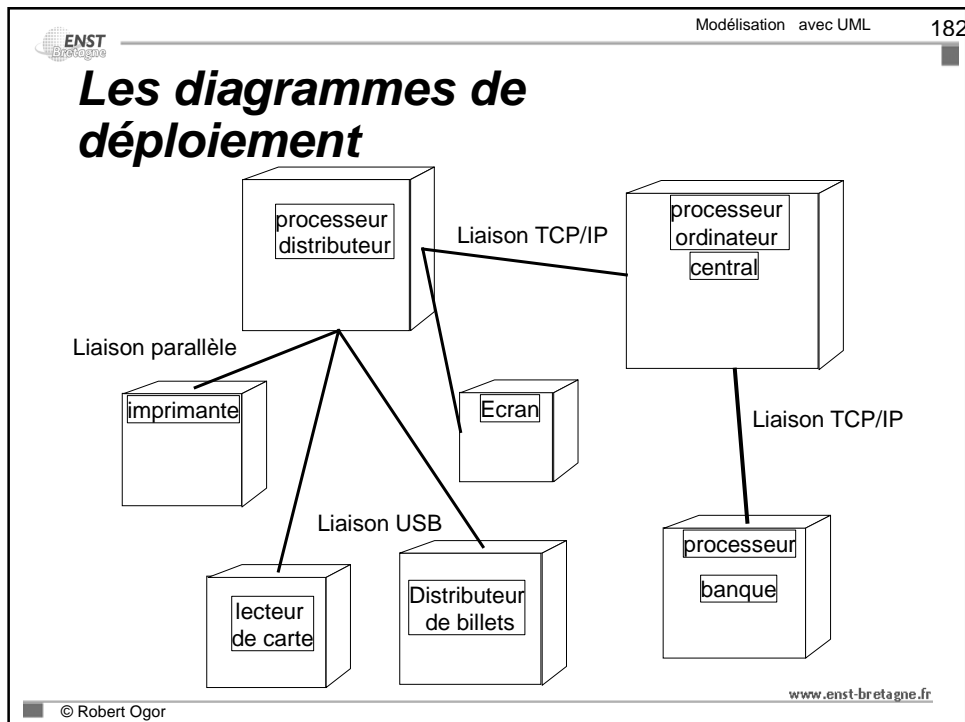
Modélisation avec UML181

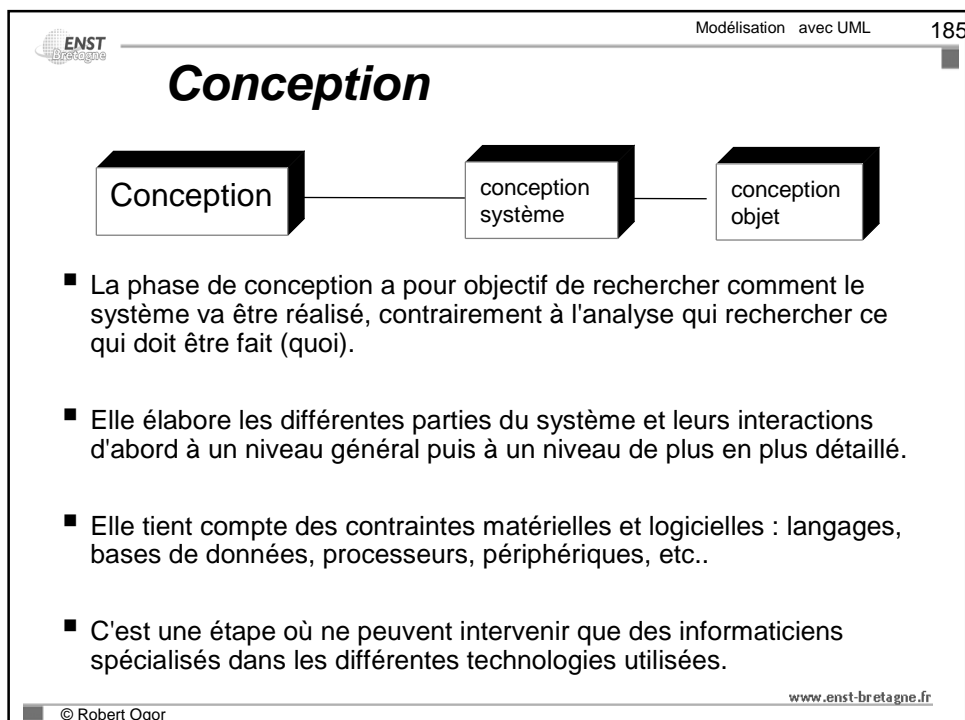
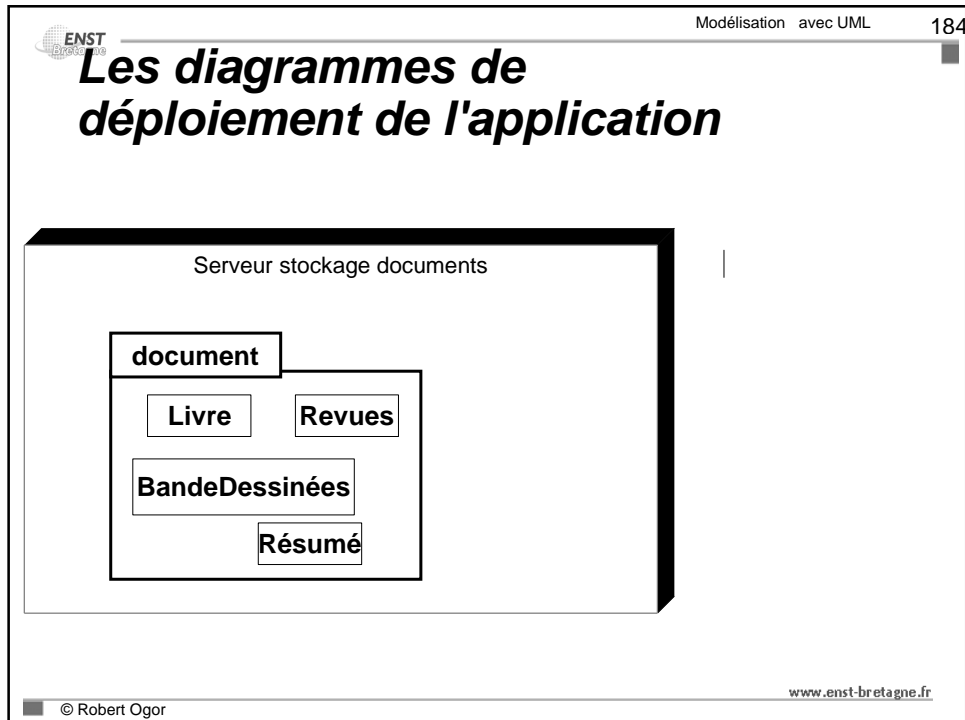
## ***Les diagrammes de déploiement***

- Ce modèle définit le diagramme de l'architecture matérielle du système ou de l'application.
- Il représente les différents processeurs, périphériques et la répartition du système sur ces différents éléments.
- Il montre les liens de communication entre ces différentes entités.

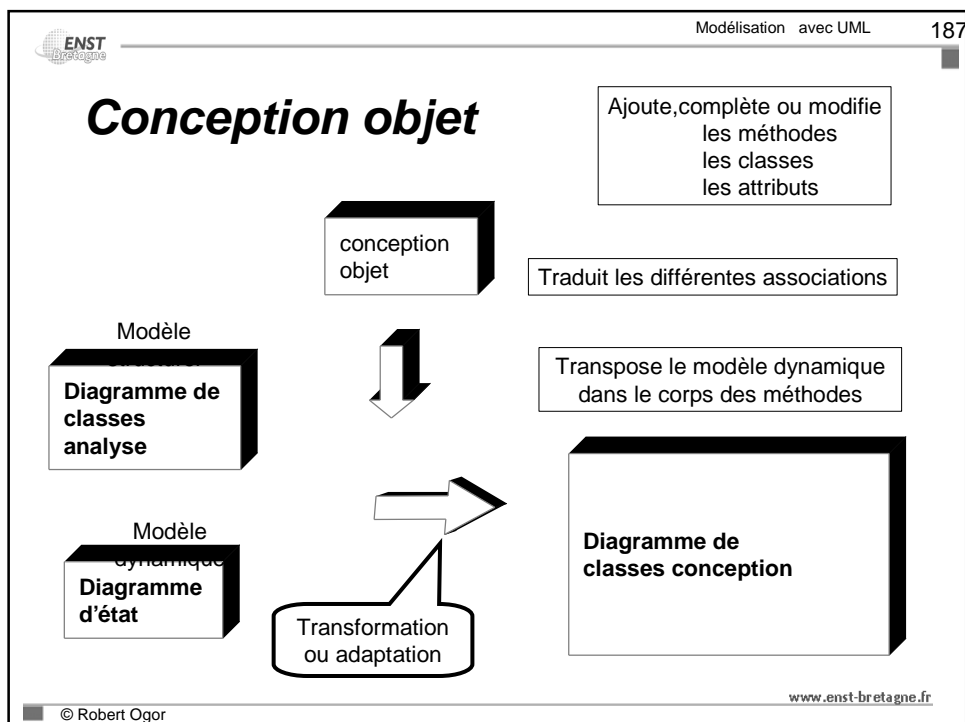
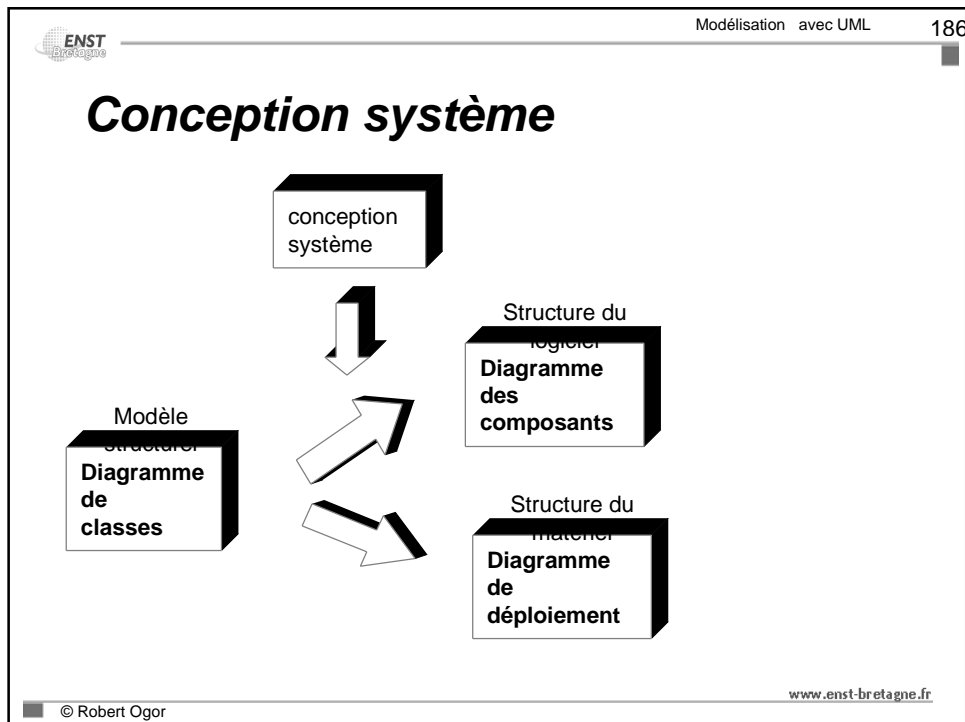
© Robert Ogor[www.enst-bretagne.fr](http://www.enst-bretagne.fr)

# Modélisation avec UML

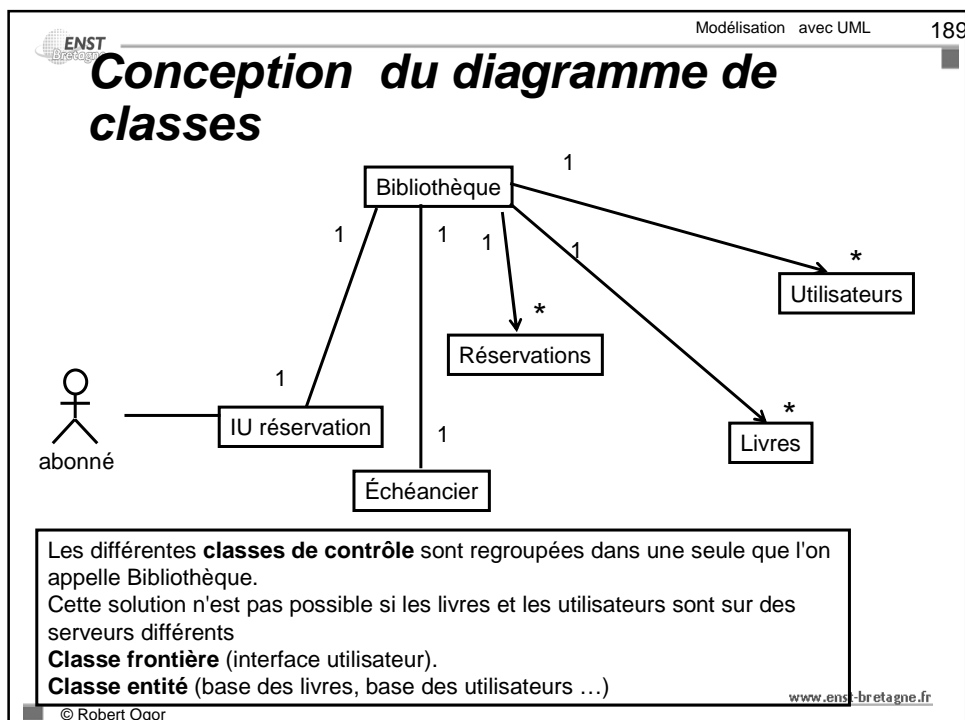
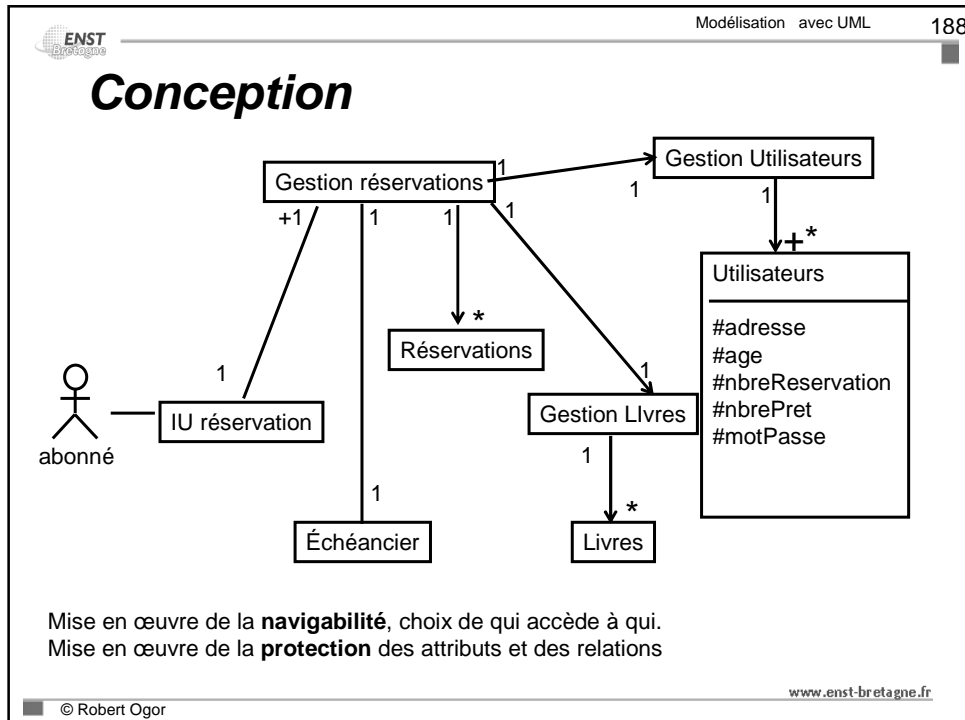




# Modélisation avec UML



# Modélisation avec UML





ENST  
Bretagne

Modélisation avec UML

190

## Conception : choix de persistance

```
classDiagram
    class GestionUtilisateurs
    class Utilisateurs
    GestionUtilisateurs "1" -- "*" Utilisateurs
```

Choix de conception :

Utilisateurs est une table dans une base de données  
Ce choix est guidé par la persistance.

```
classDiagram
    class GestionUtilisateurs
    class Utilisateurs
    GestionUtilisateurs "1" -- "*" Utilisateurs
```

Ou bien, il est défini, par exemple en Java, par une liste d'utilisateurs, au programmeur d'assurer la persistance par des fichiers.

```
classDiagram
    class GestionUtilisateurs
    class Bibliothèque
    class Utilisateurs
    GestionUtilisateurs "1" -- "1" Bibliothèque
    Bibliothèque "1" -- "*" Utilisateurs
```

© Robert Ogor

[www.enst-bretagne.fr](http://www.enst-bretagne.fr)

ENST  
Bretagne

Modélisation avec UML

223

## Conclusions

- Méthode pour appréhender la réalisation d'un système informatique
- Diagrammes qui permettent d'aider à la réflexion, de permettre la discussion (clients, développeurs)
- Pas de solution unique mais un ensemble de solutions plus ou moins acceptables suivant les contraintes du client et les logiciels et matériels disponibles
- Une solution acceptable sera obtenue avec des itérations successives.

© Robert Ogor

[www.enst-bretagne.fr](http://www.enst-bretagne.fr)

ENST  
Bretagne

Modélisation avec UML 224

## Bibliographie

### Livres

- Martin Fowler, Kendall Scott: **UML Distilled**, Addison-Wesley 2000
- Grady Booch, et al: **The Unified Modeling Language User Guide**, Addison-Wesley
- James Rumbaugh, et al: **The Unified Modeling Language Reference Manual**, Addison-Wesley
- Ivar Jacobson, et al: **Unified Software Development Process**, Addison-Wesley
- Jos B. Warmer, Anneke G. Kleppe: **The Object Constraint Language : Precise Modeling With UML**, Addison-Wesley
- Pierre-Alain Muller, Nathalie Gaertner: **Modélisation avec UML**, Eyrolles

© Robert Ogor

www.enst-bretagne.fr

ENST  
Bretagne

Modélisation avec UML 225

## Adresses utiles sur le WEB

- **OMG UML info:** <http://uml.systemhouse.mci.com>
- **survey of analysis and design methods**  
<http://www.awl.com/cp/awweb.htm>
- <http://st-www.cs.uiuc.edu/users/patterns/patterns.html>,
- **Patterns info:** <http://c2.com/ppr/index.html>
- **Rational doc:** <http://www.rational.com/uml/documentation.html>
- **Rational CASE tools:** <http://www.rational.com/products/rose/seed>
- **UML 1.4 RTF:** [www.celigent.com/omg/umlrtf](http://www.celigent.com/omg/umlrtf)
- **OMG UML Tutorials:** [www.celigent.com/omg/umlrtf/tutorials.htm](http://www.celigent.com/omg/umlrtf/tutorials.htm)
- **UML 2.0 Working Group:**  
[www.celigent.com/omg/adptf/wgs/uml2wg.htm](http://www.celigent.com/omg/adptf/wgs/uml2wg.htm)
- **OMG UML Resources:** [www.omg.org/uml/](http://www.omg.org/uml/)

© Robert Ogor

www.enst-bretagne.fr