Modélisation orientée objet - UML

Bibliographie   Les best-of :

- **Conception orientée objet et applications**,
G Booch (Addison-Wesley, 1992).

- **Le génie logiciel orienté objet**,I. Jacobson,
M. Christerson, P. Jonsson, G. Overgaard (Addison-Wesley, 1993).

Les outsiders :

- **Méthodes orientées objet**, 2nd édition,
I Graham (thompson Publisher, 1997).

- **Analyse des systèmes : de l'approche fonctionnelle à l'approche objet**, Ph. Larvet (InterEditions, 1994).

Bibliographie

Spécial UML :

- **UML La notation unifiée de modélisation objet,**
M. Lai (Dunod, 2000).

- **Modélisation objet avec UML,**
P.A. Muller (Eyrolles, 1998).

Les sites internet :

- **Index to Object-Oriented Information Sources**
http://iamwww.unibe.ch/ scg/OOinfo/index.html

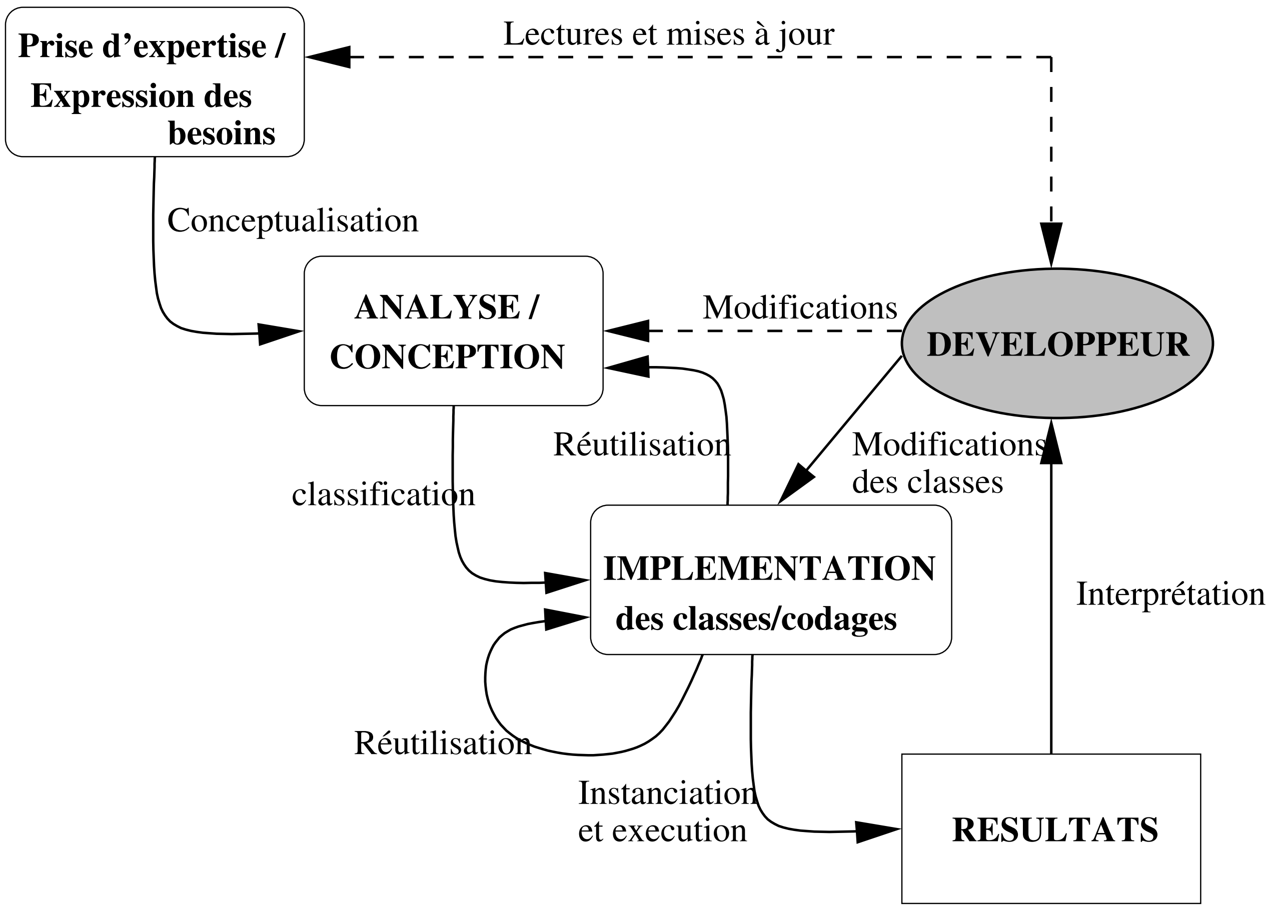
- **UML Resources**
http://www.omg.org/uml/ ou http://www.rational.com/uml/adobe.html

- **Aspect-Oriented Programming Home Page**
http://www.parc.xerox.com/csl/groups/sda/ ou http://aspectj.org http://aosd.net

Analyse des systèmes

1. Analyser un problème.
2. Proposer un modèle.
3. Concevoir un solution.
4. Réaliser (programmer) la solution.

Développement Orienté Objet



Développement Orienté Objet

- Conséquences de l'application des méthodes OO :
  - les phases d'analyse, de conception et de programmation sont très liées.

- Historique des méthodes orientées objet :
  1. langages de programmation,
  2. méthodes de conception,
  3. méthodes d'analyse.

Quelques repères

- Age de l'invention :
  - 1967 - le langage de programmation SIMULA.
    1970 - SMALLTALK (Palo Alto).
- Age de la confusion :
  - 1980 - les langages ++.
    Les méthodes de conception se multiplient
- Age de la maturité:
  - 1990 - Object Management Group : standardisation.
    Unification des méthodes OMT (Booch) OOSE (Jacobson) et Rumbaugh : Unified Modeling Language (version 1.0 1997, version actuelle 1.3).

Les concepts de base

- **Objets** : unités de base organisées en classes et partageant des traits communs (attributs ou procédures). Peuvent être des entités du monde réel, des concepts de l'application ou du domaine traité.
- **Encapsulation** :
  - les structures de données et les détails de l'implémentation sont cachés aux autres objets du système.
  - La seule façon d'accéder à l'état d'un objet est de lui envoyer un message qui déclenche l'exécution de l'une de ses méthodes.
  - Les types d'objets peuvent être assimilés aux types de données abstraites en programmation.
  - Abstraction et encapsulation sont complémentaires, l'encapsulation dessant des barrières entre les différentes abstractions.

Les concepts de base

- **Héritage** : chaque instance d'une classe d'objet hérite des caractéristiques (attributs et méthodes) de sa classe mais aussi d'une éventuelle super-classe. L'héritage est un des moyens d'organiser le monde c.-à-d. de décrire les liens qui unissent les différents objets.
- **Polymorphisme** : possibilité de recourir à la même expression pour dénoter différentes opérations. L'héritage est une forme particulière du polymorphisme caractéristique des systèmes orientés objet.
- **Modularité** : partition du programme qui crée des frontières bien définies (et documentées) à l'intérieur du programme dans l'objectif d'en réduire la complexité (Meyers). Le choix d'un bon ensemble de modules pour un problème donné, est presque aussi dificile que le choix d'un bon ensemble d'abstractions.

Trouver les bons objets

- **Méthode de désagrégation / agrégation** :
  - désagréger un module $\Rightarrow$ une suite de modules,
  - agréger une suite de modules $\Rightarrow$ un module.

- **Désagrégation**
  - On part d'un tout que l'on éclate en plusieurs parties. Chaque partie, formant à son tour un tout, est susceptible d'être à nouveau éclatée en parties plus petites.
  - Il est difficile d'exprimer en décomposition logicielle ce qu'est une partie.
  - La conception fait l'hypothèse que le système est un tout. Pour détailler et exprimer la solution, on postule que ce tout est composé de parties cohérentes séparables.

Trouver les bons objets

- Dans un premier temps, la décomposition est basée sur les entités du domaine du problème.
- La désagrégation est très différente de la décomposition fonctionnelle puisqu'une fonctionnalité n'est pas une entité du monde concret.
- La granularité de la taille des entités à utiliser est un facteur important de l'effort d'abstraction à réaliser.

Comment faire trouver les bons Objets ?

c.-à-d. :

1) comment trouver un objet ?
2) comment distinguer un bon objet d'un mauvais ?

Quelques règles d´écriture d'un module

- Un module représente un concept et tout le concept.
- Pour représenter une idée, il faut que cette idée existe.
- Ne pas regrouper dans un module des opérations qui n'ont pas de raisons particulières d'être ensemble (écriture de modules fourre-tout).
- Pour concrétiser une idée le choix du nom du module est un élément puissant d'expression (exemple les design patterns).
- Dans une première phase *"simpliste"* le choix des méthodes correspond aux verbes.

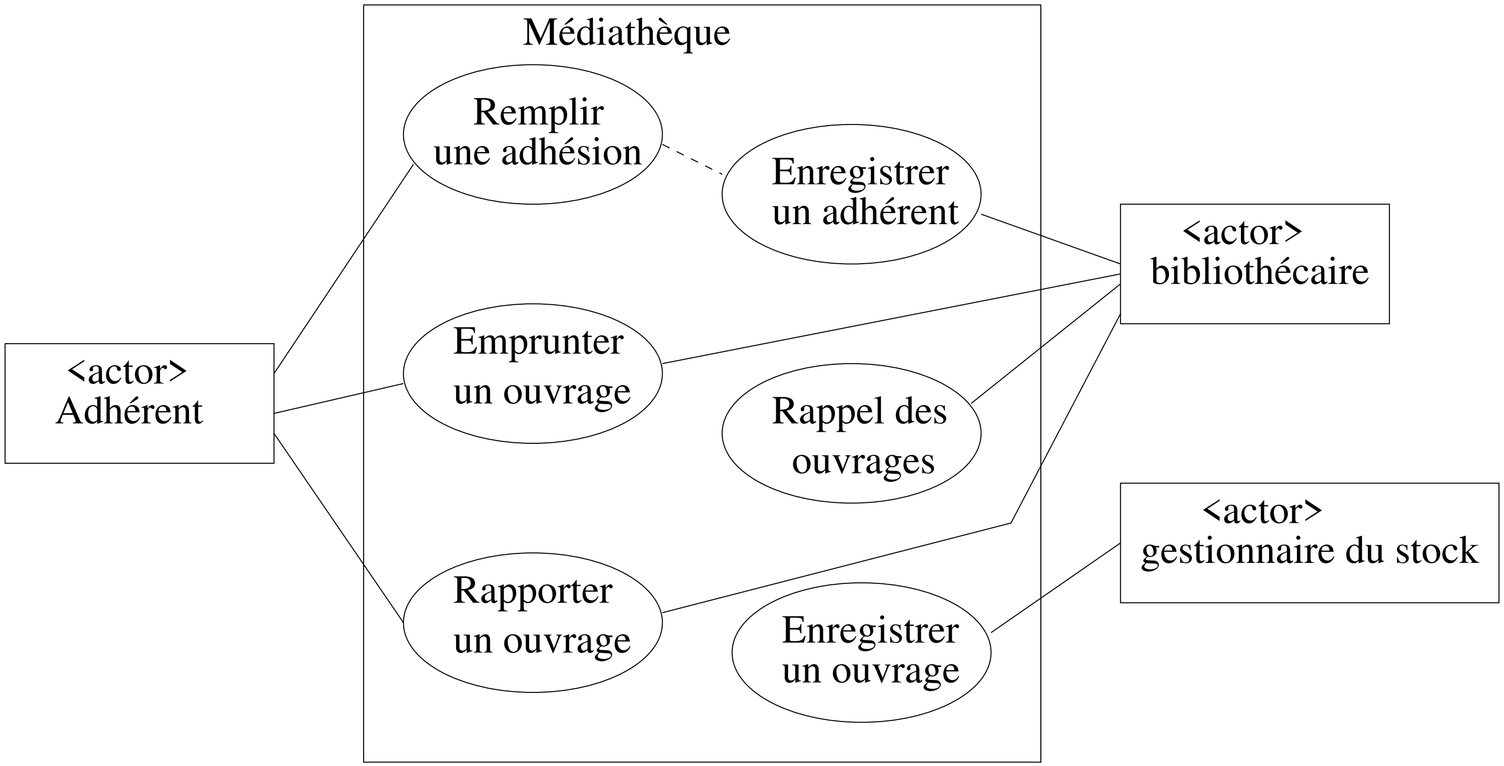
Démarche de Modélisation

- **Le modèle conceptuel** : obtenu à partir du cahier des charges ou de l'expression des besoins.
- **Le modèle statique (structurel)** : diagrammes composés de classes, catégories, paquetages et des relations qui les unissent. Peut être rapproché du modèle *entité/relation*.
- **Le modèle dynamique** : décrit le comportement des objets et leurs relations lors de l'exécution des messages(méthodes).
- **Le modèle des composants** : permet de spécifier l'architecture logicielle dans un environnement de développement donné.

Le modèle conceptuel

- **Diagramme de cas d'utilisation** : diagramme de communication des flux d'évènements ou de données entre les entités **externes** et le système à concevoir.
- 1 cas d'utilisation représente une fonction de haut niveau du système (entre un acteur et le système).
- L'identification d'un cas d'utilisation apporte une connaissance sur l'interface externe du système vis-à vis des utilisateurs de ce système.
- En général un cas d'utilisation nécessite, pour être compris, une description plus détaillée par exemple sous forme de texte ou de **diagramme de séquence**. Dans ce cas les diagrammes de séquences ne comportent que deux catégories d'objets : les acteurs externs et les composants du système qui interagissent directement avec les acteurs.

Diagramme des cas d'utilisation de la médiathèque



Le modèle statique

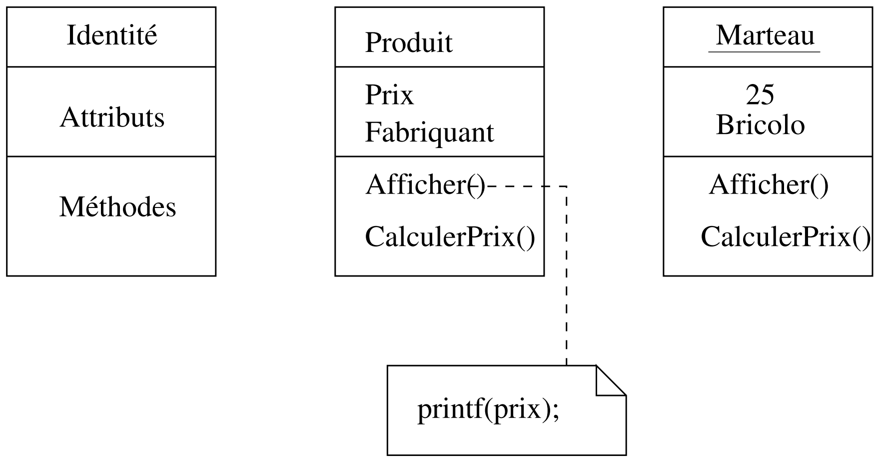
Le modèle structurel ou statique montre les relations entre les classes ou les paquetages. Le niveau de détails dépend de la phase dans laquelle on se trouve : analyse ou conception.

- **Diagramme de classes** : décrire les classes **clés**, ne donner que les attributs et les méthodes essentielles. On peut réaliser deux modèles des classes : modèle d’analyse et modèle de conception (classes techniques). La construction d’un diagramme de classes nécessite une partition logique effectuée à partir des paquetages.
- Le **diagramme d’objet** fournit des informations identiques mais au niveau des instances,

Objets et Classe d'objets

Objet = Etat + Comportement + Identité

- **Conventions graphiques de UML**

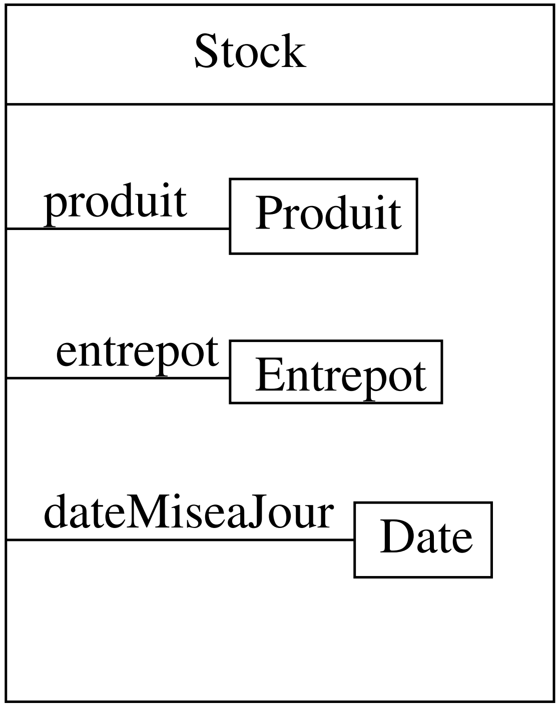


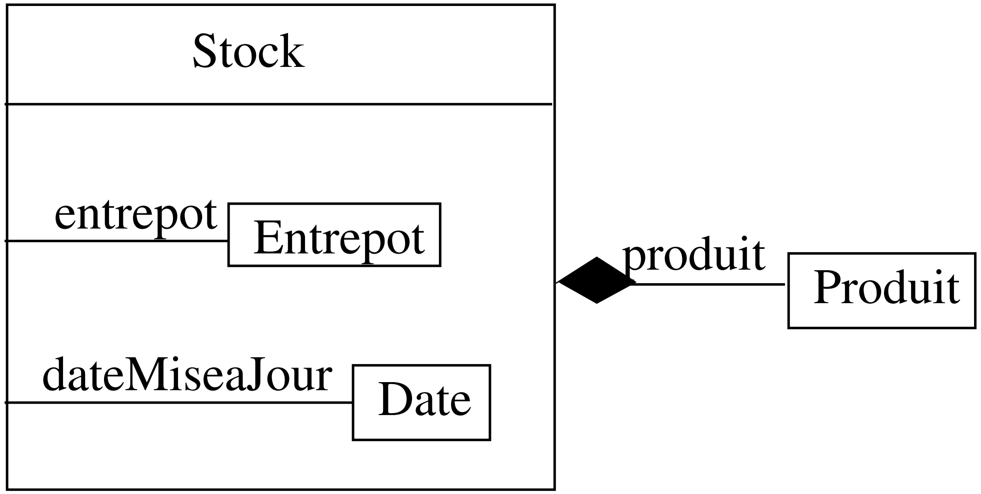
Les classes abstraites sont représentées uniquement par leur nom dans un rectangle. Il est possible de faire apparaitre le mot `abstract` sous ce nom.

La composition d'objets - 1

Les attributs d'un objet peuvent être eux-même des objets.

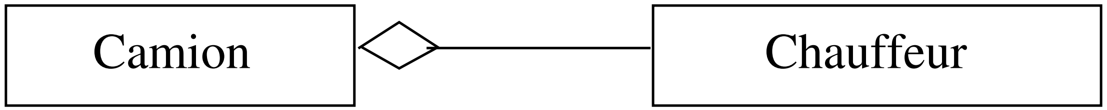
- **La composition par valeur** : la construction d'un objet physique implique la construction de ses attributs par valeur.





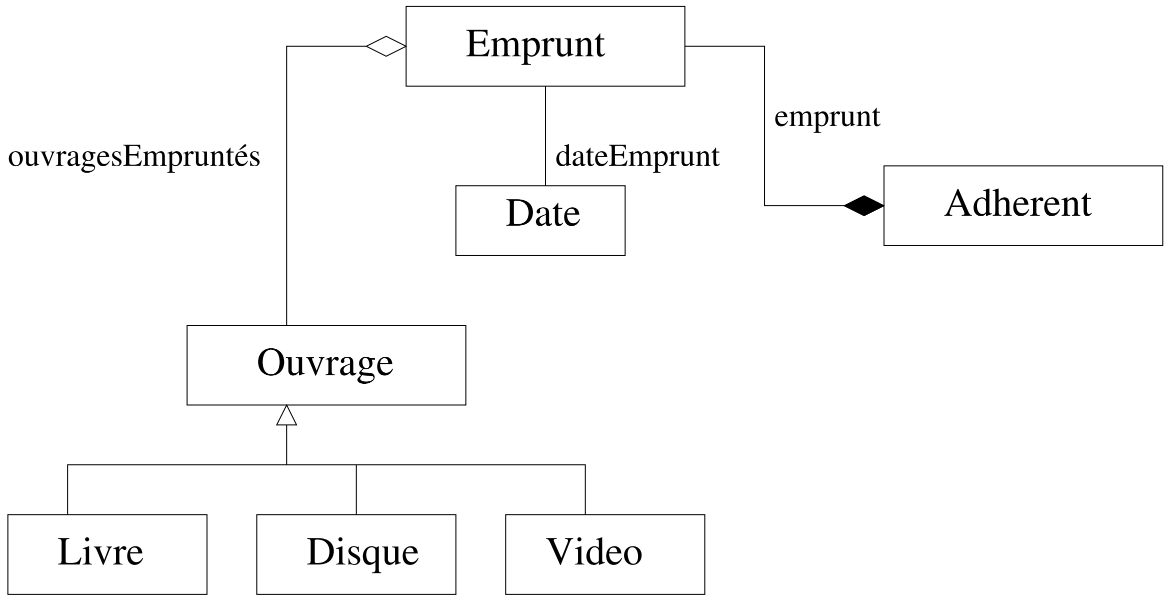
La composition d'objets - 2

- **La composition par référence** : c'est un lien de référence qui peut être partagé par plusieurs objets.



La construction du conteneur n'implique pas la construction de l'objet référencé.
NB : le losange se place du coté de l'objet référençant.

Diagramme de classe de la médiathèque



La visibilité des attributs et des méthodes

- **Publique** : un attribut ou une méthode publique est spécifée avec le signe +.
- **Privée** : un attribut ou une méthode privée est spécifée avec le signe −.
- **Protected** : un attribut ou une méthode protégée est spécifée avec le signe #.

Signature

La signature d’une méthode se compose de :

- son nom,
- le nombre et le type de ses paramètres en entrée,

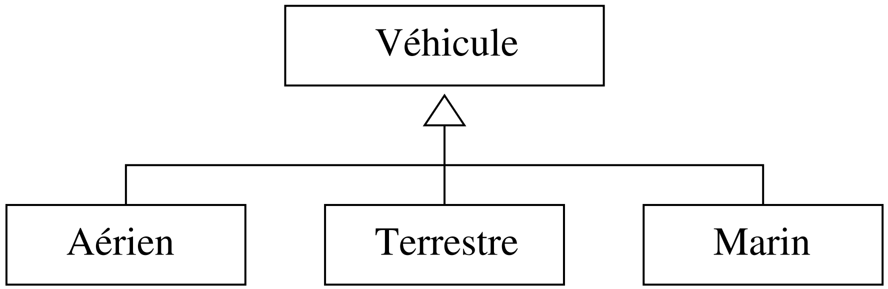
Exemple :

```
public void afficher(String , Integer);
```

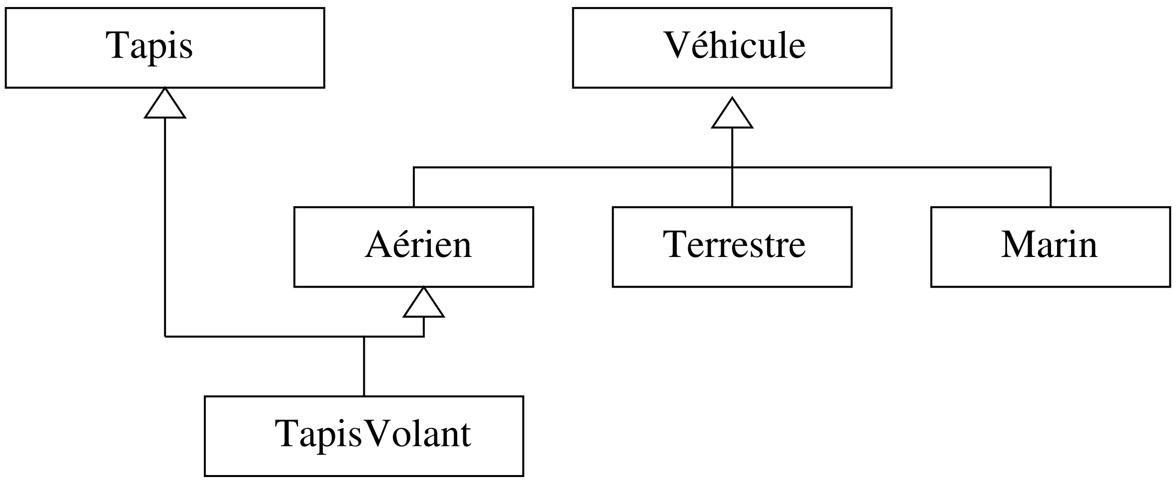
Dans un espace défini (le même espace de noms), deux méthodes peuvent avoir le même nom si elles n’ont pas la même signature.

Représentation de l'héritage

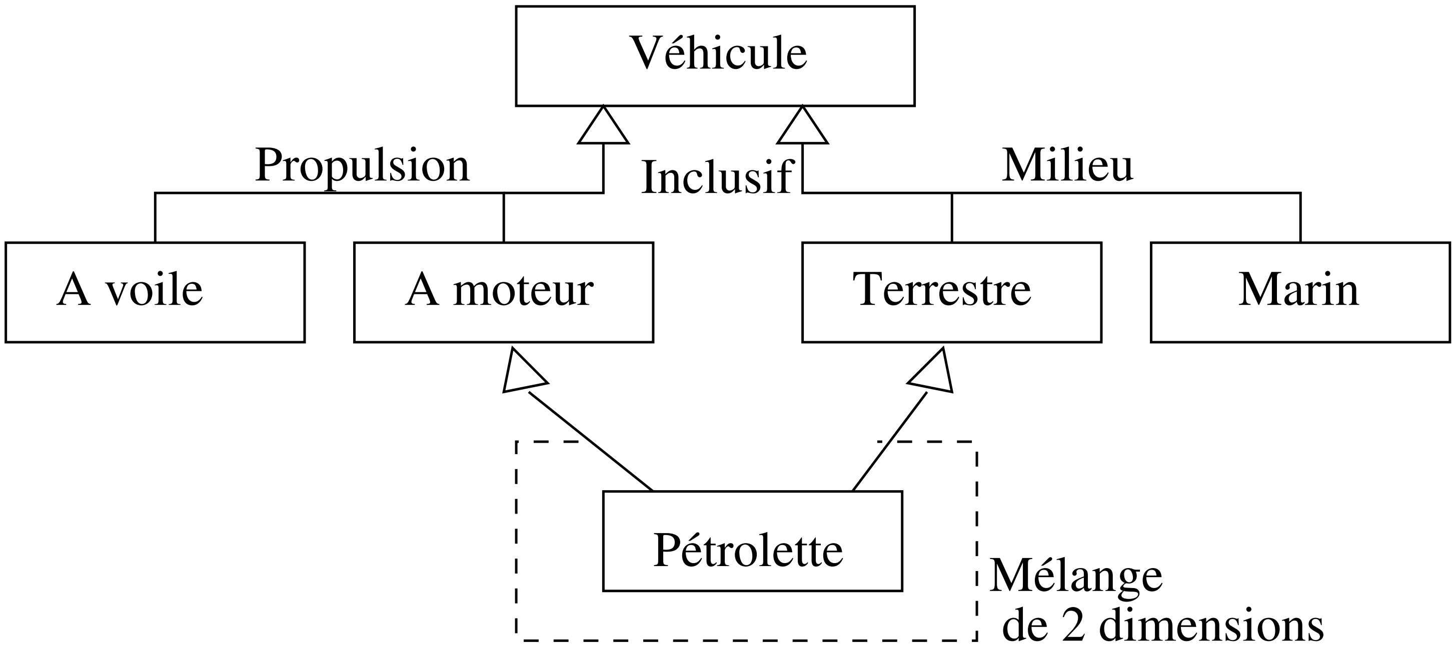
Héritage simple



Héritage multiple



Héritage multiple inclusif

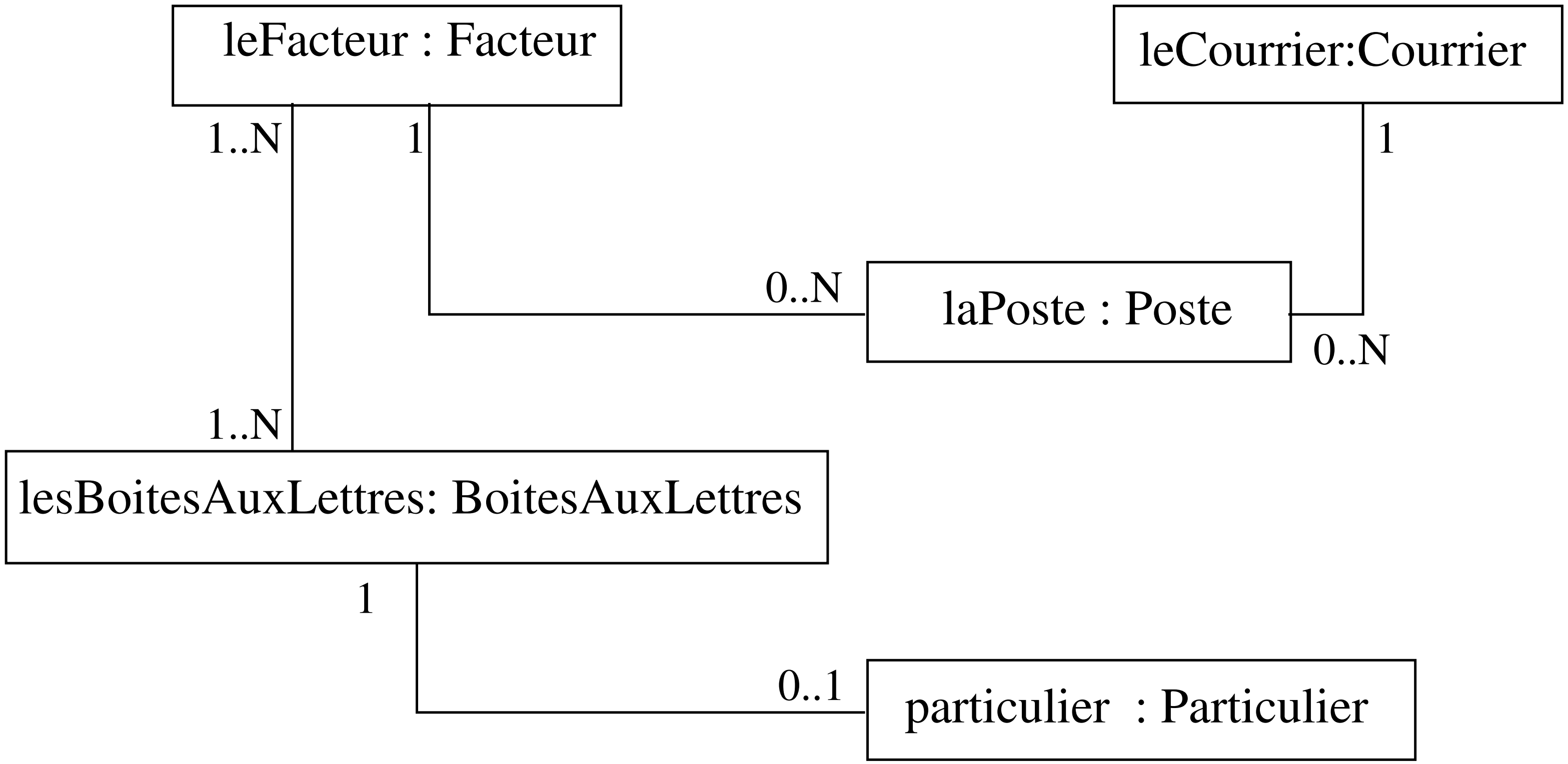


Association de classes

- Les liens d'association doivent être portés sur une classe pas sur les champs (instances).
- Arité d'une association

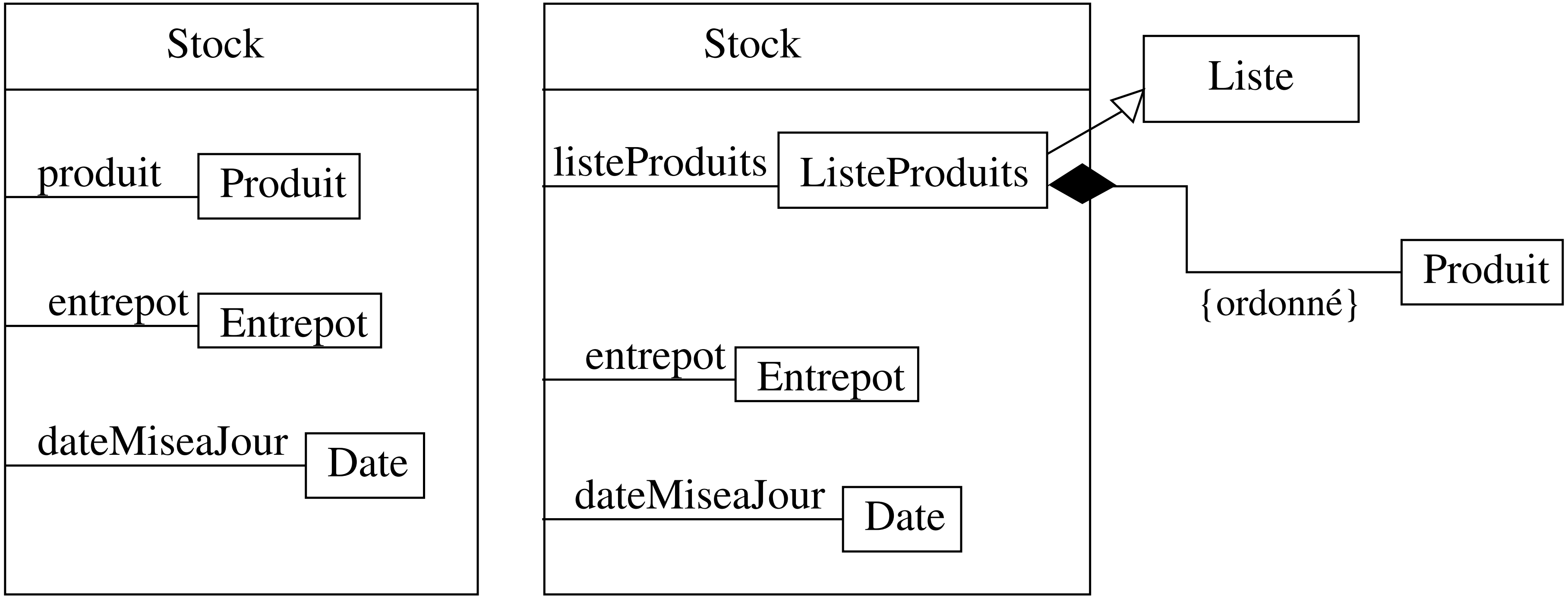
1	1 est une seule
$0..1$	0 ou 1 association
M..N	de M à N (M et N entiers naturels)
$\star$	de 0 à plusieurs
$0..\star$	de 0 à plusieurs
$1..\star$	de 1 à plusieurs

Marquer les arités dans le diagramme de classe



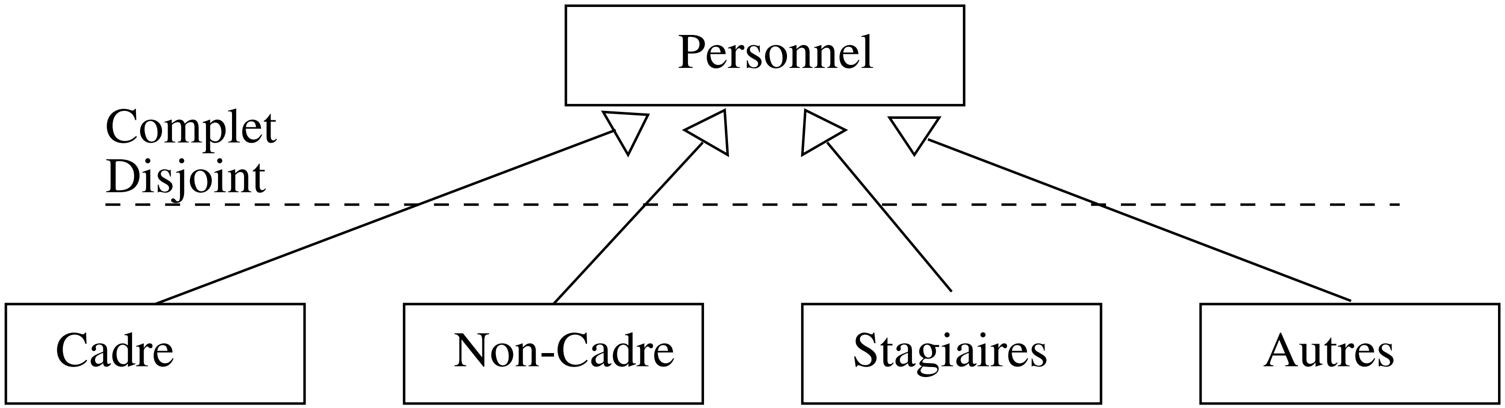
Les contraintes

Il est possible de préciser les contraintes d'une association directement sur le diagramme de classe.



Les contraintes - 2

Les liens d’héritage peuvent aussi être étiquetés par des contraintes.



Les Méta-Classes

- Définition :
  - Une méta-classe définit les caractéristiques d'une classe, c'est un modèle générique de classsse (attribut ou comportement).
- Exemple :
  - classe abstraite, interface, par extension (abus de langage !) toute classe d'une classe.
- On notera qu'une méta-classe est également un objet dont la classe est la classe de base de référence à partir de laquelle tous les objets du système sont construits (classe `Object en Java`).
- A l'étape d'analyse et de conception, il n'existe pas de différence entre une classe et une méta-classe.

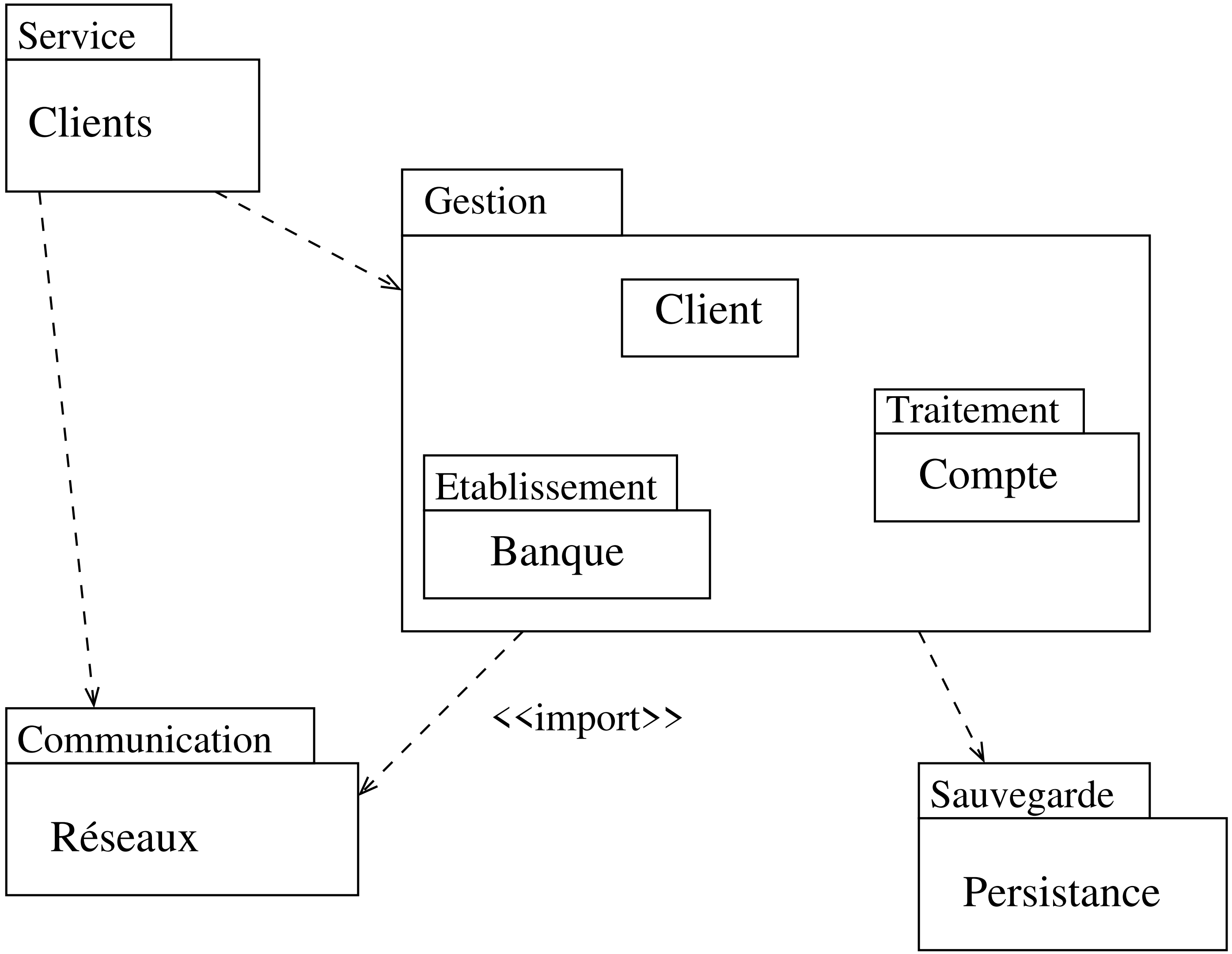
Le diagramme de classes

- Le diagramme de classe est une **vue statique** du modèle.
- Il décrit la structure interne des classes et leurs relations (dépendances, héritage, composition . . .) .
- Les termes : *static structural diagram* et *class diagram* sont équivalents dans la terminologie UML.
- Il est une collection des éléments du modèle déclaratif. Ces éléments sont classifiés grâce au mécanisme de typage.

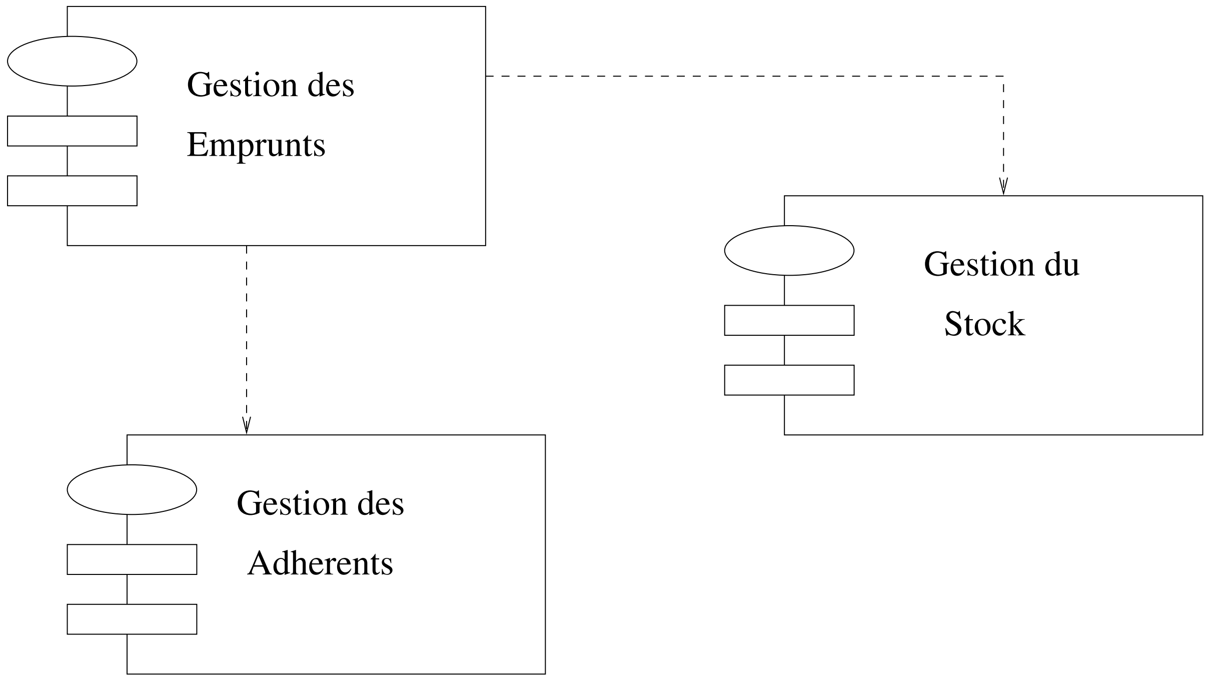
Le diagramme d'objets

- C'est le graphe des instances des différentes classes d'objets.
- Il est lui-même une instance du diagramme de classes.
- Il sert uniquement à illustrer des exemples.

Représentation des Paquetages



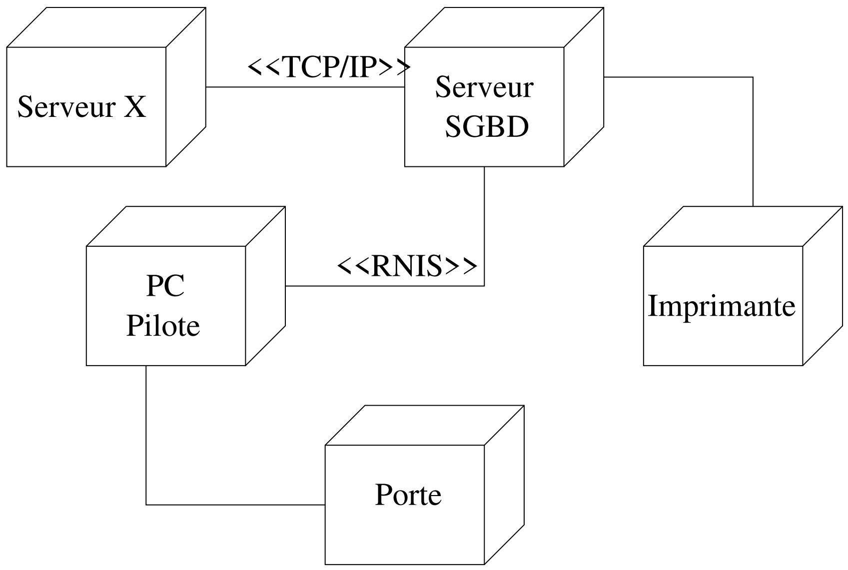
Modèle de composants



Modèle de déploiement

- Diagramme de l'organisation matériel du système ou de l'application.
- Les liens de dépendances entre unités représentent un lien de communication entre les unités.
- Permet de donner la structure d'une plate-forme technique.
- Permet de préciser où se trouve les processus et de montrer comment les objets se créent et se déplacent dans une architecture distribuée..

Modèle de déploiement



Le modèle dynamique

Deux diagrammes représentent presque la même information mais avec un formalisme différent.

- Diagramme de séquence : permettent de spécifier des séquences de messages échangés entre des objets ainsi que la création, la période d'activité et la destruction d'un objet.
- Diagramme de collaboration : aspect chronologique mais pas temporel !!

Le modèle dynamique - 1

- **Le diagramme de séquence** : permet de définir un algorithme en fonction du temps.
- Les flêches modèlisent l'échange de message entre deux objets.
- Concerne les instances des classes.
- Pas de représentation explicite du contexte des objets.
- Notation [a] :
  - Un objet est matérialisé par un rectangle est une barre verticale appelée ligne de vie

---

[a] Object Message Sequence Chart, Siemens Pattern Group. Wiley 1996 *Pattern-riented Softawre*

Diagramme de séquence

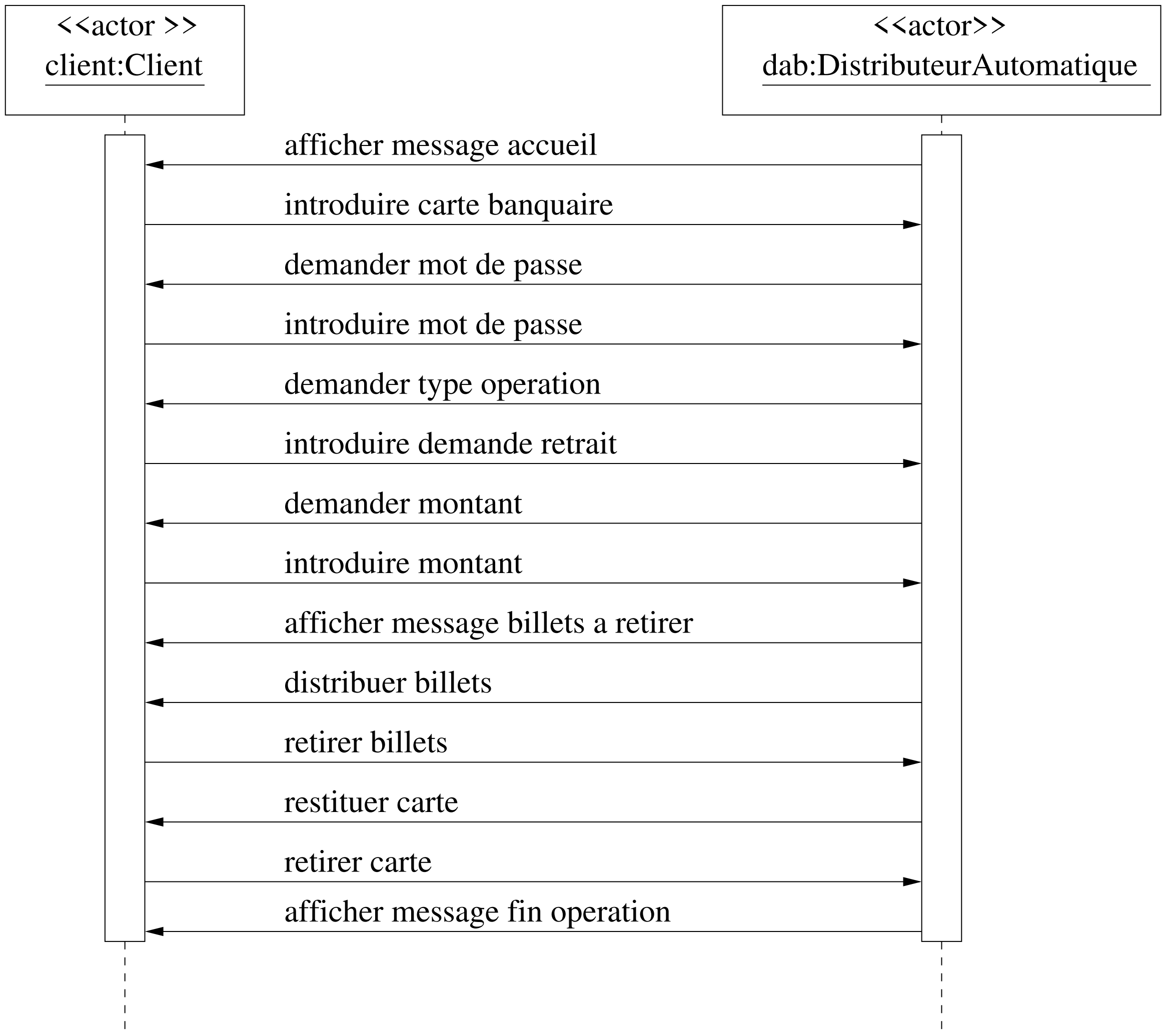
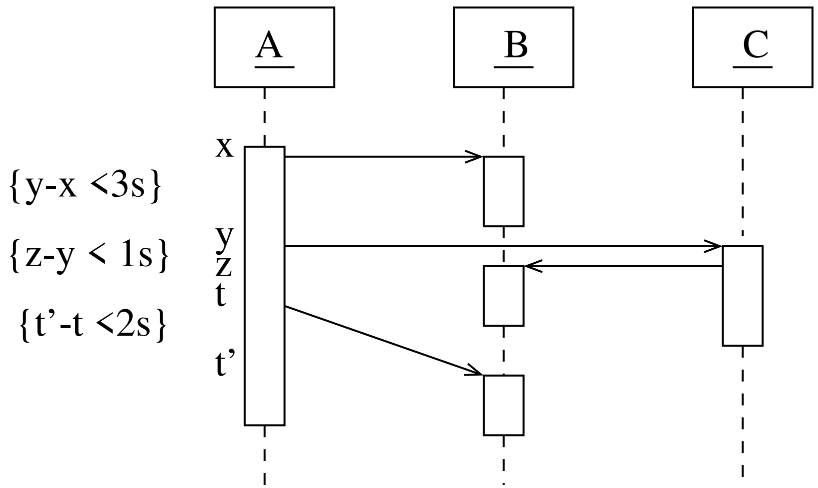


Diagramme de séquence

- L'ordre d'envoi d'un message est donné par la position sur l'axe vertical.



Mode centralisé - Mode décentralisé

- Les diagrammes de séquences reflètent le choix des structures de controle.

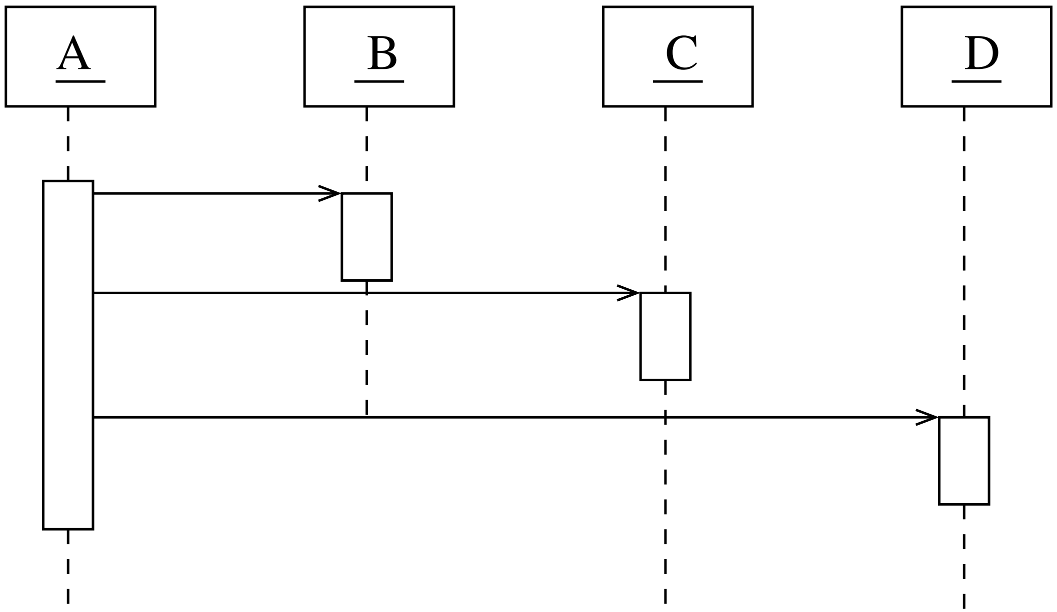


Diagramme de séquence - utilisation

Deux utilisations possibles :

- Documentation des cas d'utilisation : description des interactions entre objets sans détails de synchronisation. Les flèches correspondent à des **événements** qui surviennent dans le domaine de l'application. Pas de distinction entre **flots de contrôle** et **flots de données**.

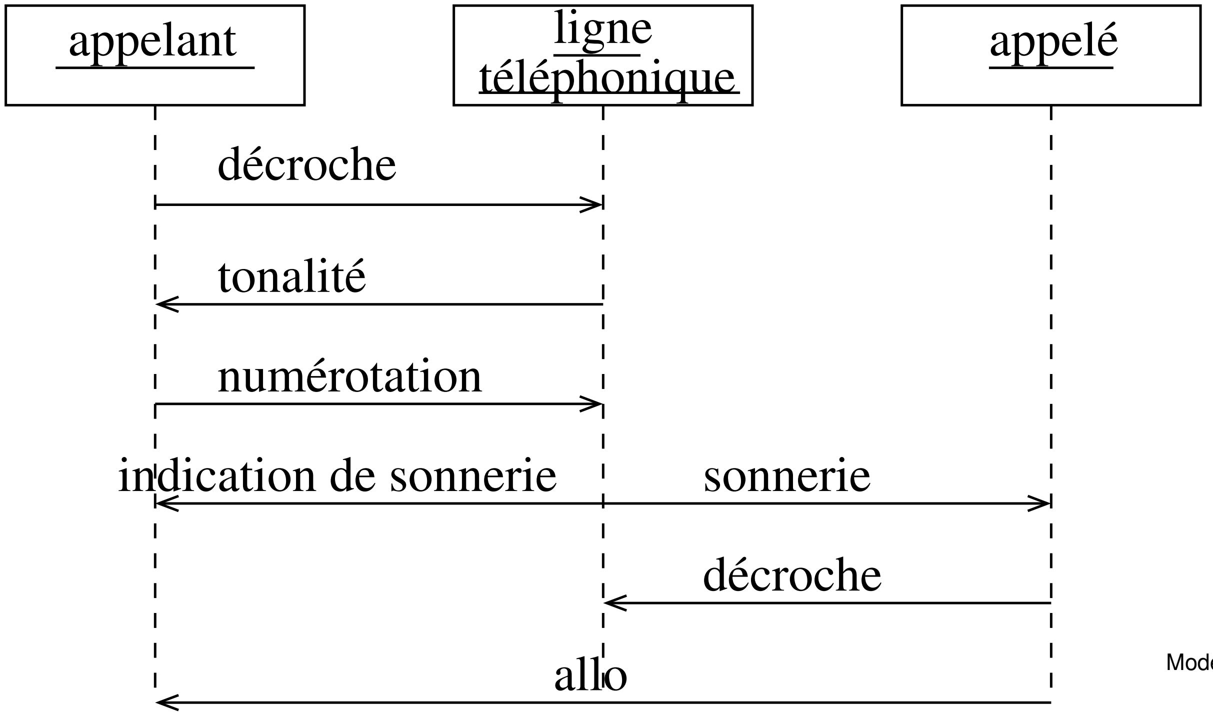
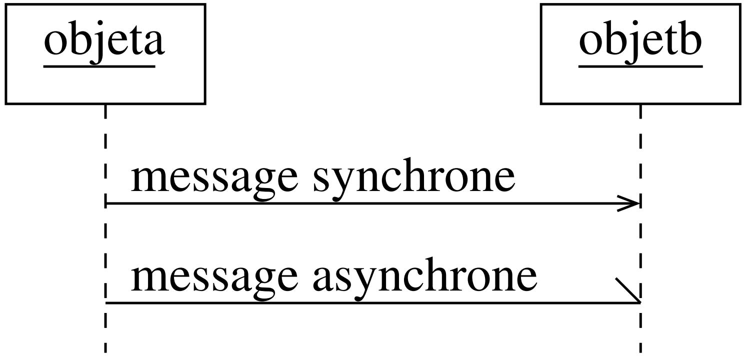
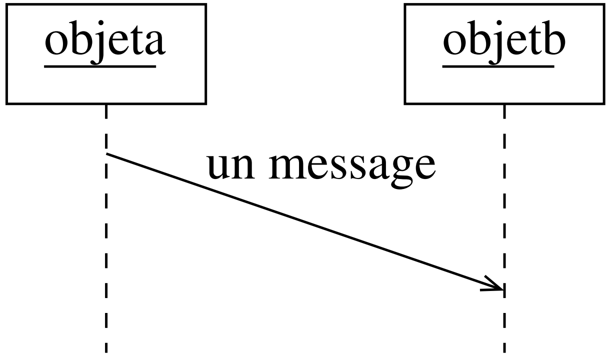
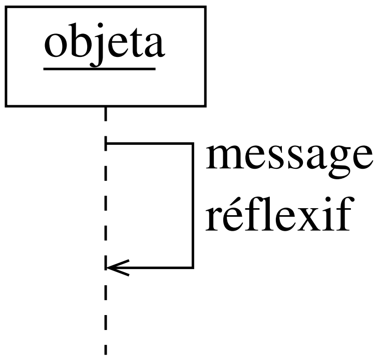


Diagramme de séquence - utilisation - 2

- Représentation précise des interactions entre objets.
  - le concept de message unifie toutes les formes de communication : appel de procédures, événemnt discret, signal de flots, interruption matérielle . . .



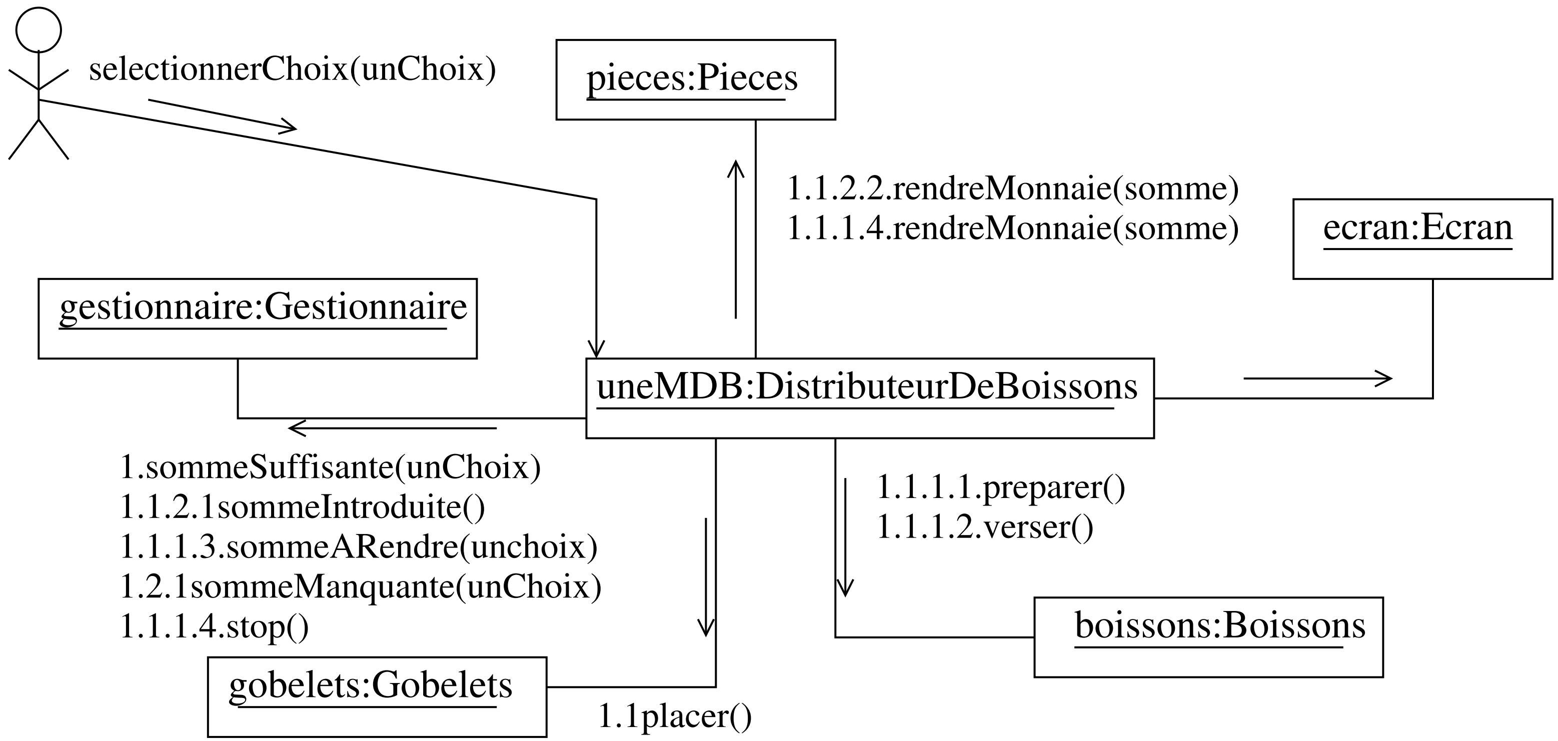




Le modèle dynamique - 2

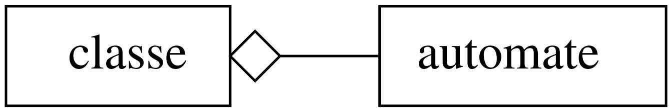
- **Le diagramme de collaboration** : stipule les echanges de message sans information temporelle.
- Une autre manière de modéliser un algorithme.
- Une interaction est réalisée par un groupe d’objets qui collaborent en échangeant des messages.
- Ces messages sont représentés le long des liens qui relient les objets avec des flêches orientées vers le destinataire du message.

Diagramme de collaboration



Le modèle dynamique-3

- Diagrammes d'état : modélisation du comportement d'un objet.
- Ils visualisent des automates déterministes [a].
- On relie l'automate à la classe considérée. On ne représente pas les automates des objets qui ne changent pas (ou peu) d'état.

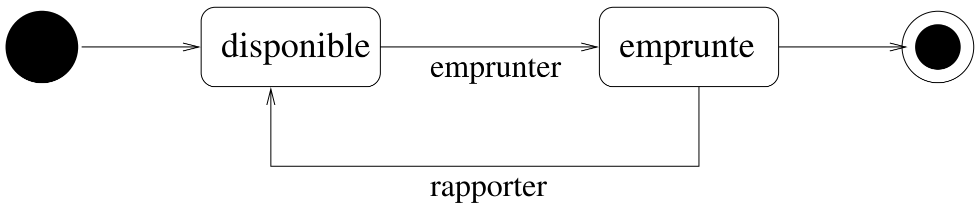


[a]formalisme de Harel, D. 1987. *Statecharts : a Visual Formalism for Complex Systems*. Science of Computer Programming vol 8.

Les diagrammes d'états-transitions - 2

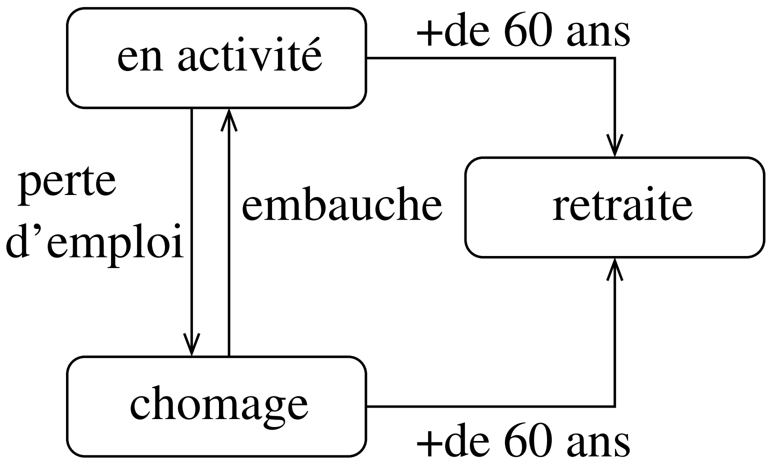
- Un objet est à tout moment dans un état donné.
- L'état d'un objet est constitué des valeurs instantanées de ses attributs.

Diagramme d'état d'un ouvrage

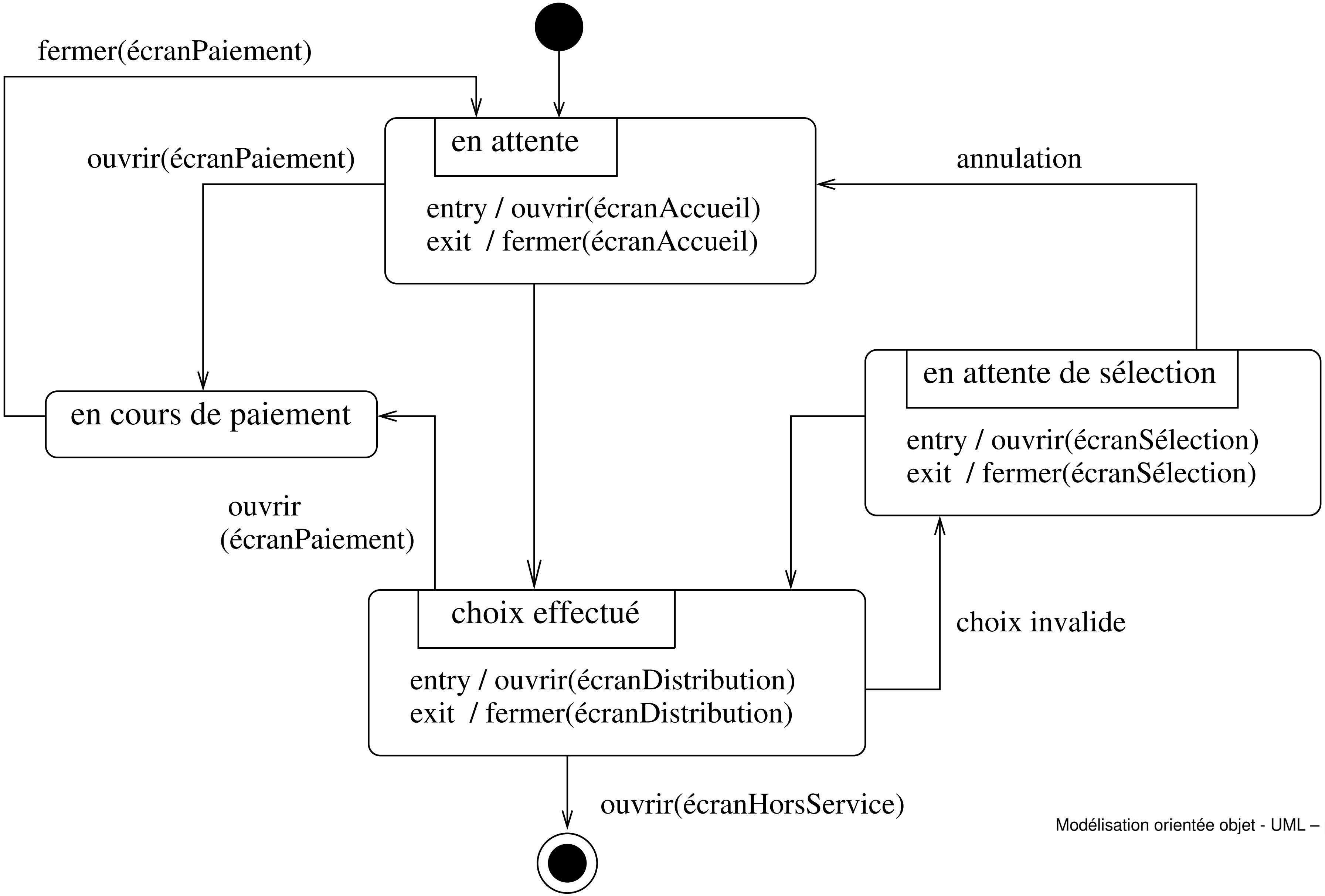


Les diagrammes d’états-transitions

- L’objet passe d’un état à un autre par les transitions.
- Déclenché par un événement, les transitions permettent le passage d’un état à un autre instantanément.



Exemple d’un diagramme d´états



Diagrammes d’activités

- Décomposition d’opérations en sous-opérations représentées par des états d’activités.
- Possibilité de modélisation des barres de *synchronisation* ou des activités menées en parallèle.

Diagramme d'activités

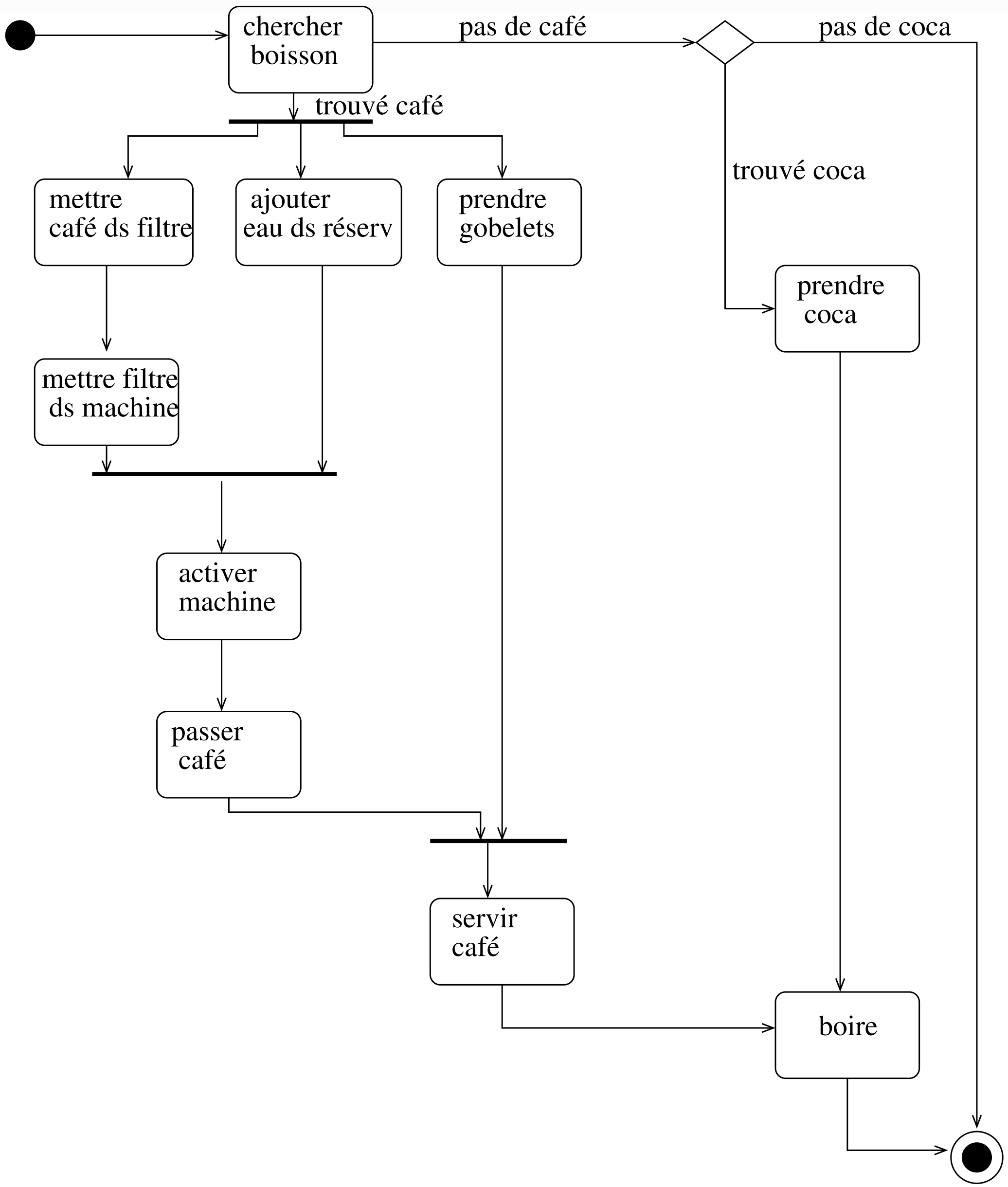


Diagramme d'activités