

Méthodologie de conception des systèmes d'information



Ahmed MAALEL
maalel.ahmed@gmail.com

Objectifs du cours

- Rendre les étudiants en mesure de lire et comprendre les modèles d'un système d'information développés avec la notation UML.
- Rendre les étudiants capables d'implémenter ces modèles dans une Base de données ou avec un langage de programmation OO (Java, C++, ...).
- Familiariser les étudiants avec Rational Rose.

Plan du Cours

- **0- Préliminaires**
- **I- Introduction Générale**
 - I-1- Pourquoi Utiliser UML
 - I-2- Définition
- **II- Le Concept Objet**
 - II-1- Notion
 - II-2- Formalisme
- **III- La Modélisation avec UML**
 - III-1- Notions de Base
 - III-1-a- Les Modèles UML
 - III-1-b- Rédaction d'un Diagramme
 - III-1-c- Les différents types de diagrammes UML

Plan du Cours

suite

III-2- Modélisation Statique

III-2-a- Diagramme des Cas d'Utilisation

III-2-b- Diagramme d'Objet

III-2-c- Diagramme des Classes

III-2-d- Diagramme des Composants

III-2-e- Diagramme de Déploiement

III-3- Modélisation Dynamique

III-3-a- Diagramme de Séquences

III-3-b- Diagramme de Collaboration

III-3-c- Diagramme d'Etat Transition

III-3-d- Diagramme d'Activité

PLAN du COURS

suite

- IV- Le mécanisme d'extensions
- V- Avantages et Inconvénients d'UML
- VI- UML et les AGL

Vision objet d'un système d'information (1)



Un **SI** = un ensemble d'objets qui collaborent entre eux

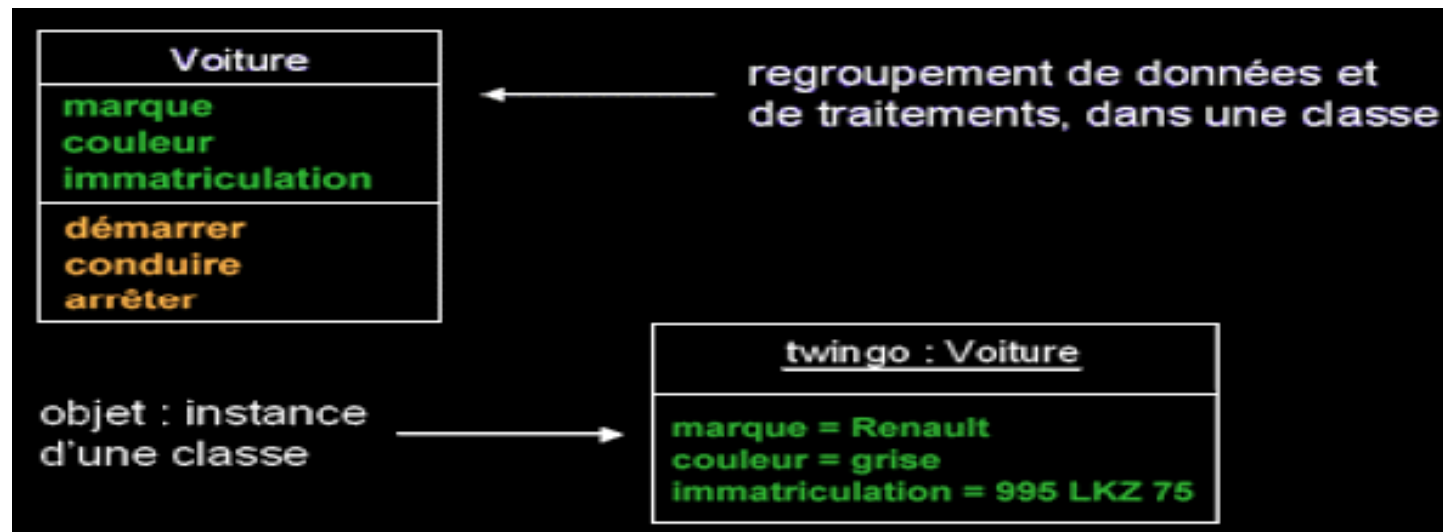
Un objet représente une entité du système qui est caractérisée par:

- Des frontières précises
- Une identité (ou référence)
- Un ensemble d'attributs (propriétés) décrivant son état
- Un ensemble de méthodes (opérations) définissant son comportement

Vision objet d'un système d'information (2a)



- Un objet est une instance de classe (une occurrence d'un type abstrait)
- Une **classe** est un type de données abstrait(modèle), caractérisé par des propriétés (attributs et méthodes) communes à des objets et permettant de créer des objets possédant ces propriétés.



Vision objet d'un système d'information (2b)

Présenter les propriétés suivantes :

Héritage

Polymorphisme

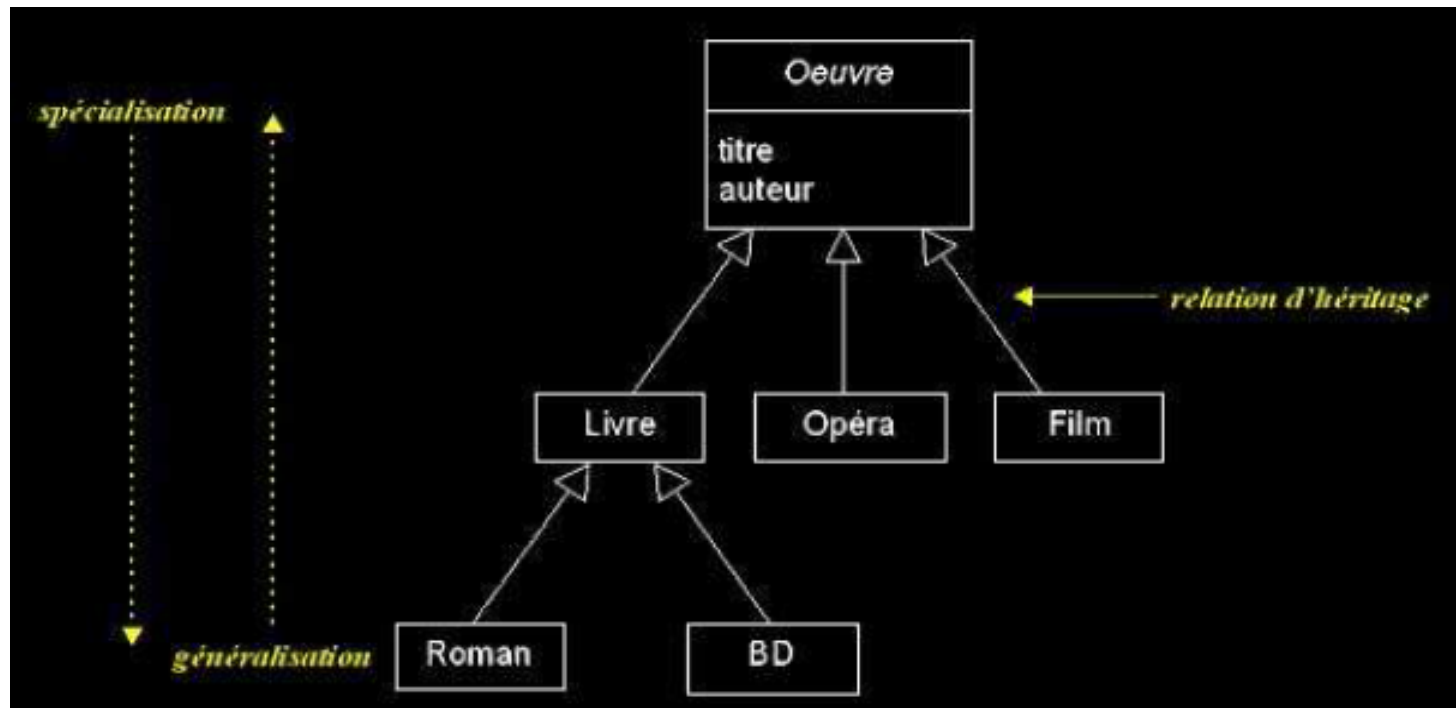
L'agrégation



Vision objet d'un système d'information (3)

■ Héritage

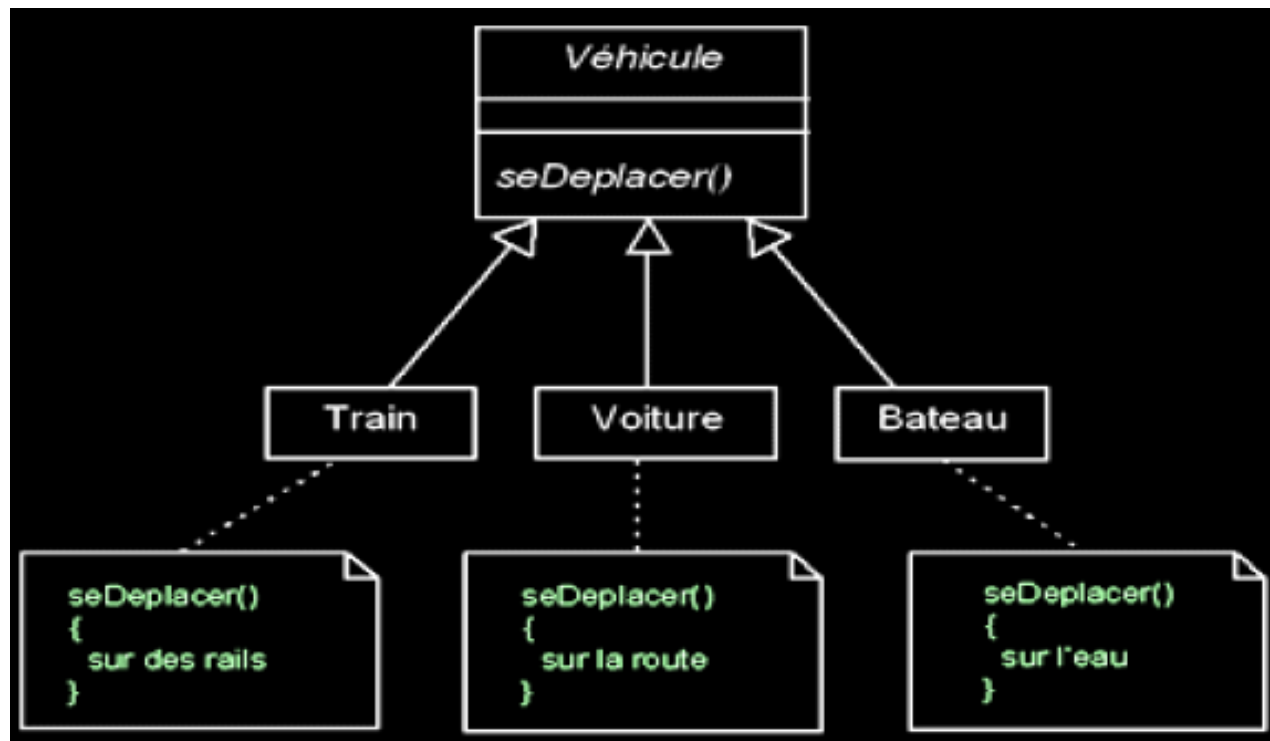
Transmission de propriétés (attributs et méthodes) d'une classe à une sous classe d'objets



Vision objet d'un système d'information (4)

■ Polymorphisme

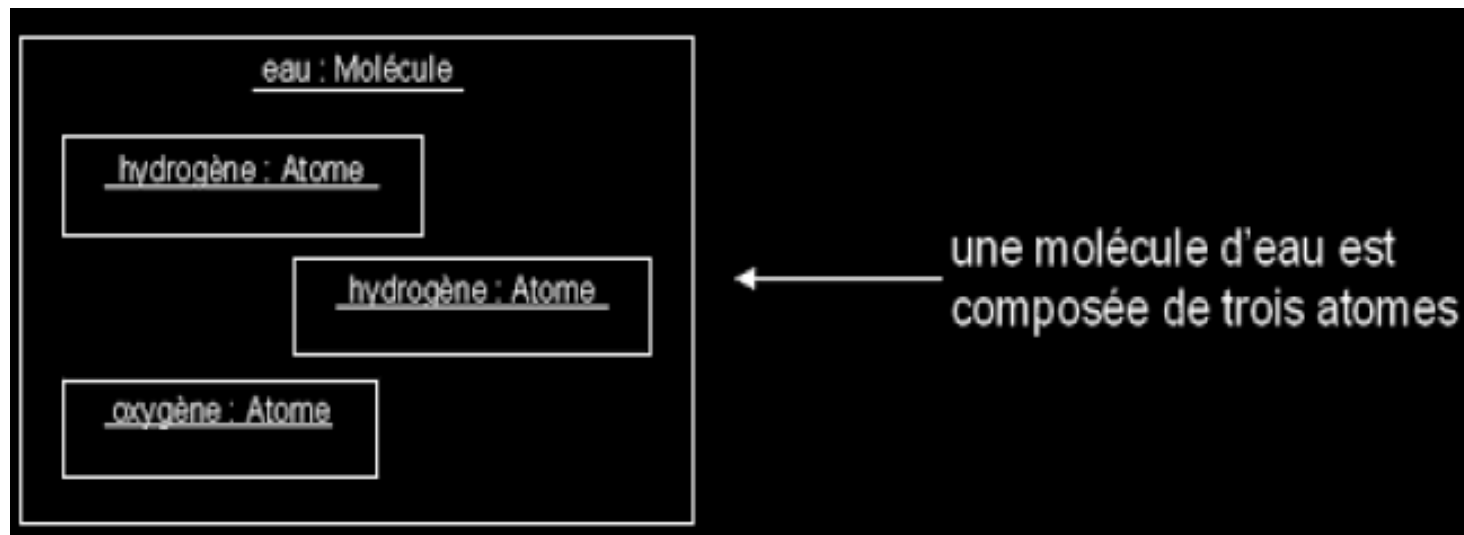
Factorisation de comportement (méthodes) commun d'objets



Vision objet d'un système d'information (5)

■ L'agrégation

Une relation d'agrégation permet de définir des objets composés d'autres objets. L'agrégation permet d'assembler des objets de base, afin de construire des objets plus complexes.



Résumé des concepts fondateurs de l'approche objet (1)

- L'héritage est un mécanisme de transmission des propriétés d'une classe (ses attributs et méthodes) vers une sous-classe.
- Une classe peut être spécialisée en d'autres classes, afin d'y ajouter des caractéristiques spécifiques ou d'en adapter certaines.
- Plusieurs classes peuvent être généralisées en une classe qui les factorise afin de regrouper les caractéristiques communes d'un ensemble de classes.

Résumé des concepts fondateurs de l'approche objet (2)



- **La spécialisation et la généralisation** permettent de construire des hiérarchies de classes.
- **L'héritage** peut être simple ou multiple. L'héritage évite la duplication et encourage la réutilisation.
- **Le polymorphisme** représente la faculté d'une méthode à pouvoir s'appliquer à des objets de classes différentes. Le polymorphisme augmente la généricité du code.

L'approche objet : une solution parfaite?



- L'approche objet est moins intuitive que l'approche fonctionnelle
 - Quel moyen utiliser pour faciliter l'analyse objet?
 - Quels critères identifient une conception objet pertinente?

- L'application des concepts objets nécessite une grande rigueur
 - Le vocabulaire présente des d'ambiguïtés et de d'incompréhension
 - Comment décrire la structure objet d'un système de manière pertinente?
 - Comment décrire l'interaction entre ces objets de manière précise?

Remèdes aux inconvénients de l'approche objet



- Un langage (ou modèle) pour exprimer les concepts objet qu'on utilise, afin de pouvoir :
 1. Représenter des concepts abstraits (graphiquement par exemple)
 2. Limiter les ambiguïtés (parler un langage commun)
 3. Faciliter l'analyse
- Une démarche d'analyse et de conception objet pour :
 1. Ne pas effectuer une analyse fonctionnelle et se contenter d'une implémentation objet, mais penser objet dès le départ
 2. Définir les vues qui permettent de couvrir tous les aspects d'un système, avec des concepts objets et l'évaluation de solutions)

Et c'est quoi un modèle?

- Un modèle est une abstraction de la réalité
- Il s'agit d'un processus qui consiste à identifier les caractéristiques intéressantes d'une entité, en vue d'une utilisation précise.
- L'abstraction désigne aussi le résultat de ce processus, c'est-à dire l'ensemble des caractéristiques essentielles d'une entité, retenues par un observateur.
- Un modèle est 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 très subjective de la réalité.
- Bien qu'un modèle ne représente pas une réalité absolue, un modèle reflète des aspects importants de la réalité, il en donne donc une vue juste et pertinente.

Et UML dans tout ça?

- **UML: *langage standard de modélisation* des systèmes d'information**
 - **UML: langage visuel pour**
 - comprendre le système
 - communiquer et travailler à plusieurs
 - aider à spécifier, concevoir et développer un système d'information
- **avec différents modèles et différentes vue**

Et dans quel cas l'utiliser?

■ Systèmes à forte composante logicielle

- systèmes d'information pour les entreprises
- les services bancaires et financiers
- les télécommunications
- les transports
- la défense / l'aérospatiale
- le commerce de détail
- l'électronique médicale
- les services distribués et les applications WEB

■ Pas limité à un domaine précis

Unified Modeling Language



Introduction Générale (Rappel)

■ Historique

Le développement d'UML a commencé 1994 quand G. Booch et J. Rumbaugh ont débuté leur travail sur **l'unification des méthodes** Booch et OMT (Object Modeling Technique).

Ces deux méthodes se développaient déjà indépendamment l'une de l'autre et étaient mondialement reconnues comme les principales méthodes orientées objet, Booch et Rumbaugh ont joint leurs forces pour réaliser une unification complète de leurs méthodes.

Introduction Générale (Rappel)

■ Pourquoi utiliser UML ?

Unified Modeling Language n'est pas un éloignement radical des méthodes Booch, OMT ou OOSE c'est une succession légitime de celles-ci

- Si vous utilisez Booch, OMT, ou OOSE, votre formation, votre expérience et vos outils seront préservés
- UML est plus expressif, plus propre et plus uniforme
- UML permet aux projets de modéliser des choses qui n'auraient pas pu l'être avant
- UML supprime toutes les différences non nécessaires de notation et de terminologie qui obscurcissent les similarités de bases de ces différentes approches.

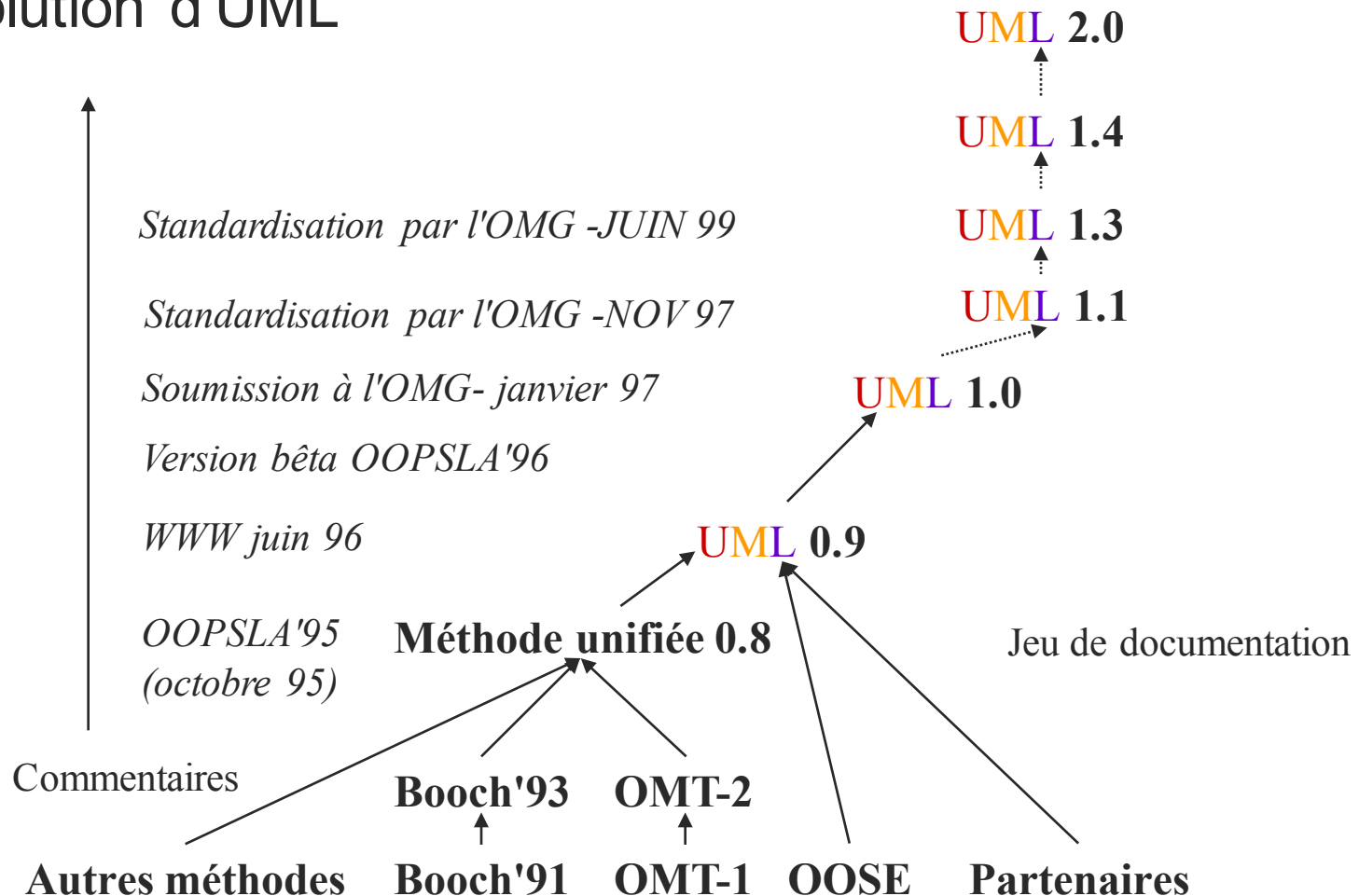
Introduction Générale (Rappel)

■ I-2- Définition

- UML est un ensemble de notations orientées objet dont la syntaxe et la sémantique sont précisément définis
- Il permet de modéliser tout type de système et peut être utilisé à toute étape d'un cycle de vie
- UML est un langage de modélisation objet et non une méthode
- C'est un langage formel (ou pseudo formel) basé sur un métamodèle qui permet de définir :
 - les concepts et éléments de modélisation
 - la sémantique de ces éléments

Introduction Générale (Rappel)

■ L'évolution d'UML



Les diagrammes UML (Rappel)

Axe de Modélisation



Statique

- Diagramme de Classes
- Diagramme d'Objets
- Diagramme de Composants
- Diagramme de Déploiement

Fonctionnel

- Diagramme de Use Case*

Dynamique

- Diagramme d'Etats-Transitions
- Diagramme d'Activité
- Diagramme de Séquence
- Diagramme de Collaboration

II- Le Concept Objet

■ II-1- La Notion d'Objet

- La POO consiste à modéliser conceptuellement un ensemble d'éléments d'une partie du monde réel en un ensemble d'entités informatiques
- Ces entités sont appelées *objets*. Il s'agit de données informatiques regroupant les principales caractéristiques des éléments du monde réel
- La difficulté de cette modélisation consiste à créer une représentation abstraite, sous forme d'objets, d'entités ayant une existence matérielle (chien, voiture, ampoule, ...) ou bien virtuelle (sécurité sociale, temps,...).

II- Le Concept Objet

■ Définition d'un objet

Un objet est défini par :

1. Un état : Ensemble de valeurs associées à des propriétés qui permettent de décrire un objet à un instant t .
2. Un comportement : Ensemble de services ou d'opérations que peut rendre un objet ou qui modifient son état.
3. Une identité : Propriété qui permet de distinguer un objet des autres objet.

II- Le Concept Objet

■ II-2- Formalisme

Un objet est caractérisé par plusieurs notions:

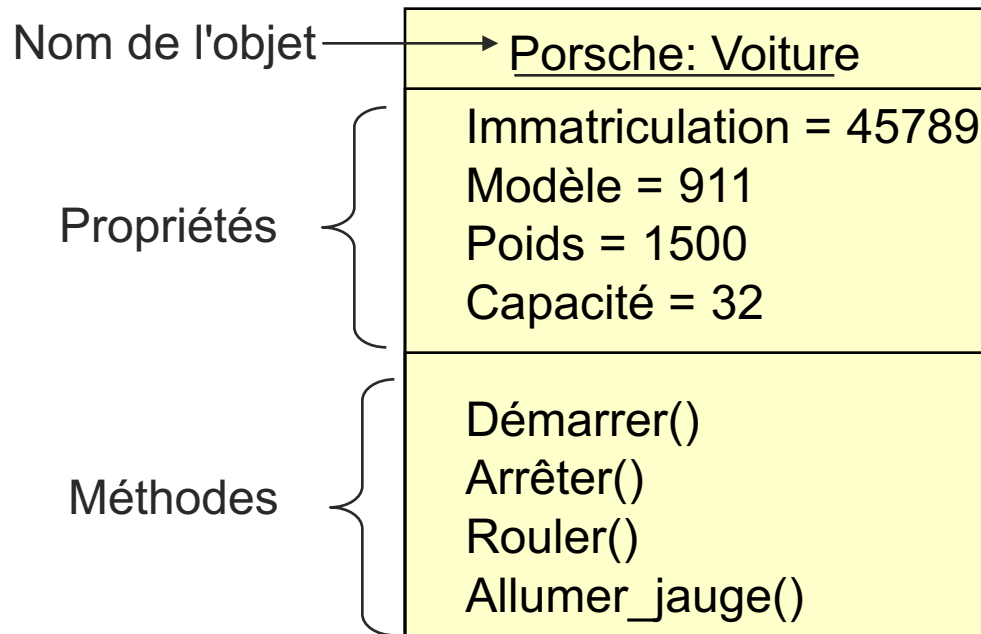
- ➔ Les attributs ou propriétés : Il s'agit des données caractérisant l'objet. Ce sont des variables stockant des informations d'état de l'objet
- ➔ Les méthodes ou fonctions membres : Les méthodes d'un objet caractérisent son comportement, c'est-à-dire l'ensemble des actions ou opérations que l'objet est à même de réaliser. elles permettent de faire réagir l'objet aux sollicitations extérieures (ou d'agir sur les autres objets)
- ➔ Les opérations sont étroitement liées aux attributs, car leurs actions peuvent dépendre des valeurs des attributs, ou bien les modifier

II- Le Concept Objet

- L'identité: L'objet possède une identité, qui permet de le distinguer des autres objets, indépendamment de son état
- On construit généralement cette identité grâce à un identifiant découlant naturellement du problème (un produit pourra être repéré par un code, une voiture par un numéro d'immatriculation, ...)

II- Le Concept Objet

Représentation d'un objet avec UML :



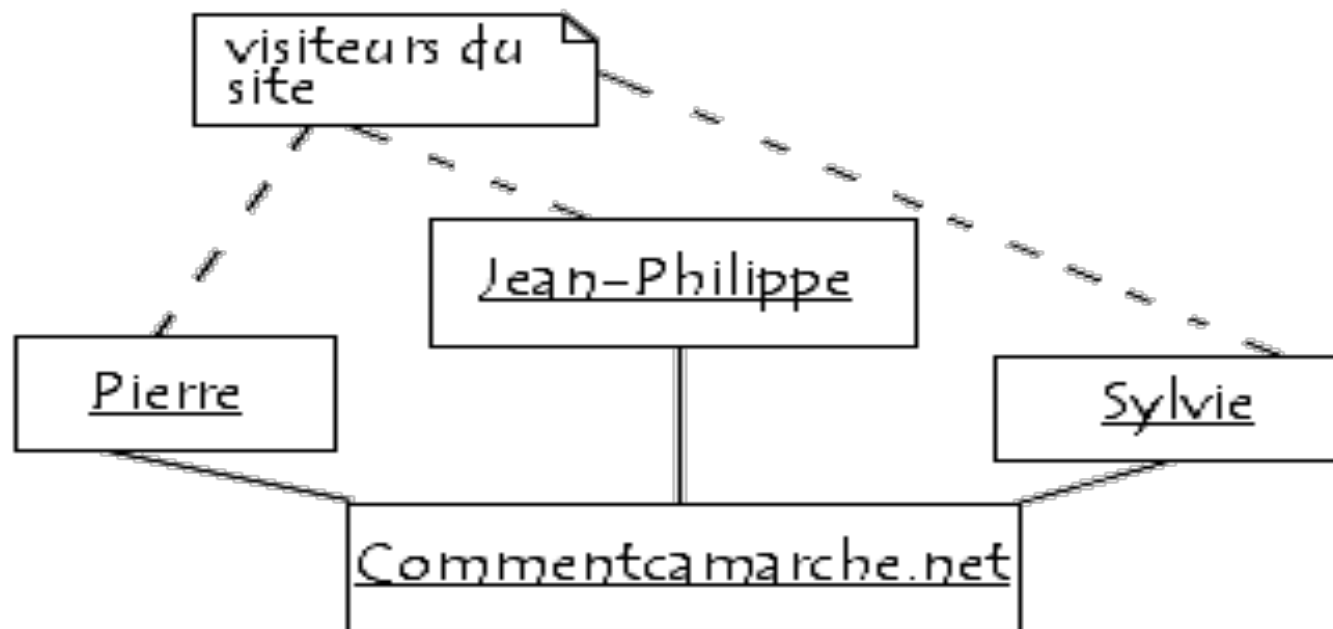
II- Le Concept Objet

■ Les liens entre objets

- Les objets ne sont pas des corps inertes isolés, même s'ils possèdent leurs caractéristiques propres par l'intermédiaire des valeurs de leurs attributs, ils ont la possibilité d'interagir entre eux grâce à leurs méthodes.
- UML propose de représenter les liens qui existent entre les objets grâce à un trait continu.
- Il est possible d'ajouter des commentaires sur le modèle sous forme de notes
- Une note est représentée par un rectangle dont le coin supérieur droit est replié. Pour relier une note à un objet il faut utiliser un trait discontinu (en pointillés).

II- Le Concept Objet

Exemple d'objets ayant des liens entre eux :



II- Le Concept Objet

■ Modélisation d'une classe

- Une classe est la structure d'un objet, c'est-à-dire la déclaration de l'ensemble des entités qui composeront un objet.
- Un objet est donc "issu" d'une classe, c'est le produit qui sort d'un moule.
- En réalité on dit qu'un objet est une instantiation d'une classe
- On peut parler indifféremment d'objet ou d'instance (éventuellement d'occurrence)

II- Le Concept Objet

■ Modélisation d'une classe

Une classe est composée:

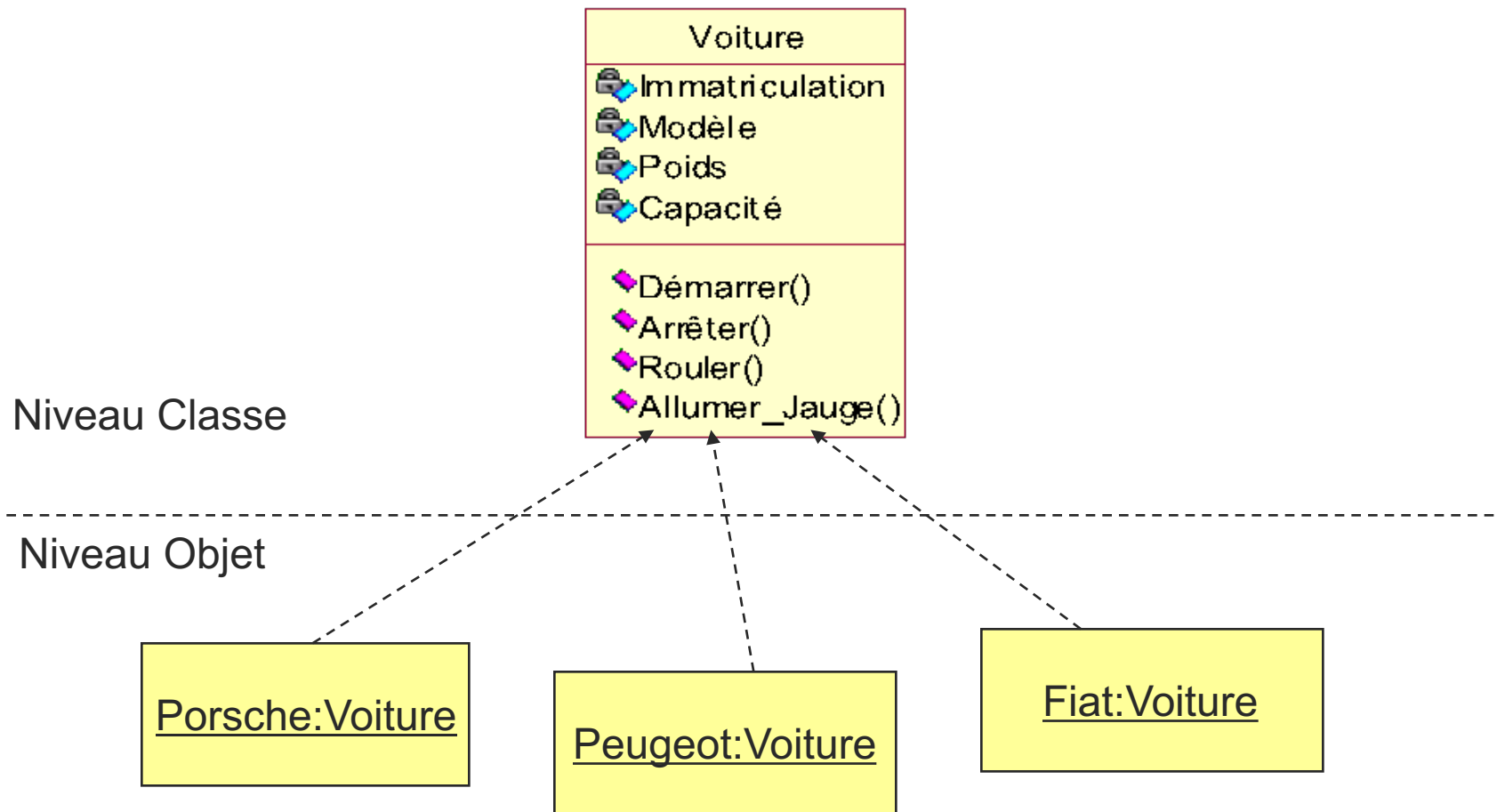
- D'un nom qui caractérise une classe d'une manière unique dans un modèle.
- D'un ou plusieurs attributs: il s'agit des données ainsi que leur visibilité
- De méthodes : il s'agit des opérations applicables aux objets de la classe

II- Le Concept Objet

Une classe se représente avec UML sous forme d'un rectangle divisé en trois sections:

- ➔ Le premier contient le nom donné à la classe (non souligné)
- ➔ Les attributs d'une classe sont définis par un nom, un type (éventuellement une valeur par défaut, c'est-à-dire une valeur affectée à la propriété lors de l'instanciation) dans le second compartiment
- ➔ Les opérations sont répertoriées dans le troisième volet du rectangle.

II- Le Concept Objet



Représentation d'une classe

- 1 à 4 champs :
 - nom de la classe : obligatoire
 - 0 ou n attributs
 - 0 ou n opérations
 - 0 ou n invariants de classe, exceptions, ...

Classe A

Classe B

Classe C

 name : type = initval

 opname()

Les attributs

- Sémantique : constitue un élément de l'état de l'objet
- Visualisé ou pas, selon le niveau de détail souhaité
- ... : liste d'attributs incomplète
- Visibilité = type d'accessibilité
 - + : public - visible et modifiable par tout objet du même paquetage
 - - : private - seulement visible et modifiable par les opérations de l'objet auquel i appartient. Le principe de masquage impose de rendre chaque attribut *private*
 - # : protected - seulement accessible et modifiable par les opérations des classes descendantes
- Multiplicité : intervalle ou nombre

Les opérations

- Représente un service spécifique offert par un objet
- Visibilité : +, -, #
- Direction : in, out, inout (in est la valeur par défaut)

Identification des classes, attributs et opérations

- C'est lors de la construction des diagrammes d'interaction (séquences et/ou collaboration) que les objets (ceux qui par abstraction donneront lieu à des classes) sont identifiés
- Ils sont transformés :
 - en classes s'il y a des messages échangés
 - en attribut
- Les opérations des classes correspondent aux messages échangés entre objets réels

Identification des classes, attributs et opérations

■ Exercice 1:

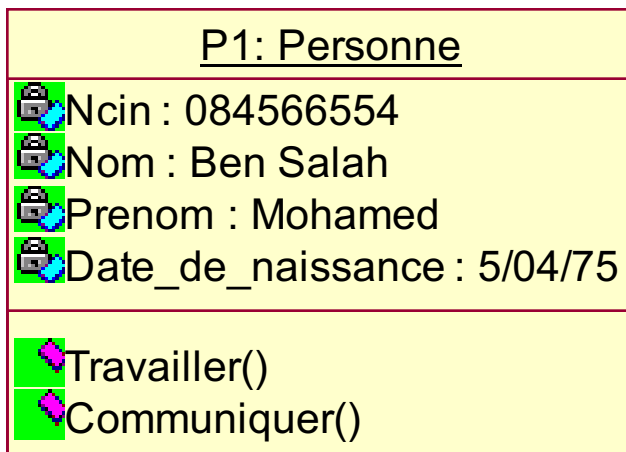
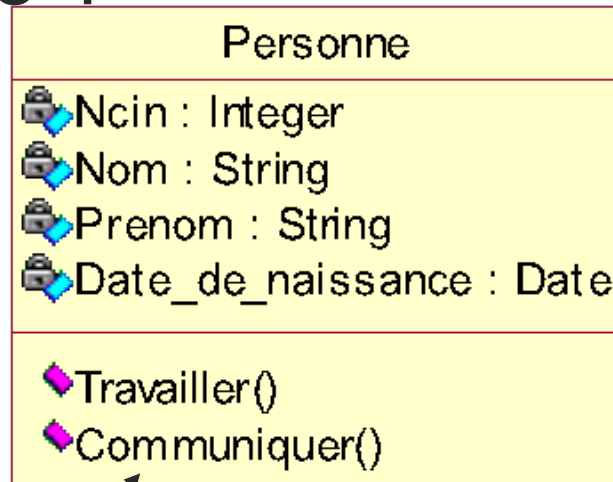
Une personne possède un ncin, un nom, un prénom, une date de naissance. La personne peut travailler et communiquer

Déduire la classe personne

Donner une instance possible de cette classe

Identification des classes, attributs et opérations

■ Solution exercice 1



Les relations entre les classes

- Elles sont de 3 types et traduisent :
 - l'association entre une (réflexivité) ou plusieurs classes
 - la notion d'assemblage/composition
 - le concept de spécialisation/généralisation
- Elles représentent des liens statiques / structurels entre objets et à longue durée de vie
- Précaution : les associations ne sont en général pas directement supportées par les langages de programmation orienté-objet.

Les relations (Associations)

- Représente une propriété statique
- Lie une ou plusieurs classes : arité 2 ou plus
- Toute association doit être nommée par un verbe d'état
- Toute association peut être navigable
- Présence de rôle aux extrémités (rôle de chaque classe dans l'association)
- Multiplicité :
 - 0
 - 0..1
 - N
 - M..N
 - *, 0..*
 - 1..*

Multiplicités et contraintes

■ Multiplicité : Contraintes valables durant toute l'existence des objets

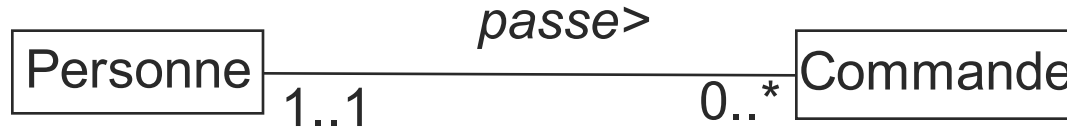
- une et une seule
- plusieurs (0 ou plus)
- optionnelle (0 ou 1)
- 1 ou plus
- de m à n



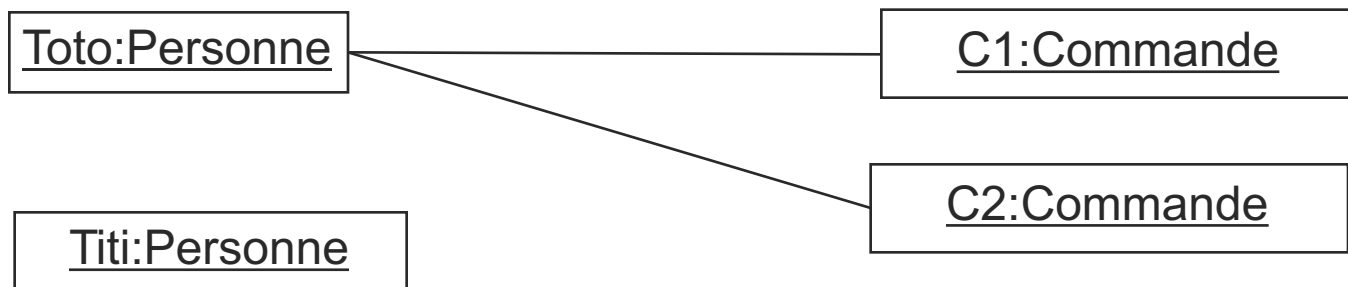
■ Autres contraintes sur association

- {ordonné},{sous-ensemble},{ou-exclusif},

Les relations (Association)

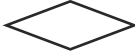



- Une personne peut passer plusieurs commandes
- Une commande est toujours passée par une et une seule personne



Les relations

Agrégation/composition

- Relation non symétrique
- Une extrémité supporte la notion de responsabilité :
 - toute opération / attribut du responsable peut se propager aux éléments contraints
- Agrégat / agrégé : durée de vie des agrégés indépendante de l'agrégat 
- Composite / composant : durée de vie des composants dépendante du composite / conteneur - on parle d'embarquement 

Les relations (Association)

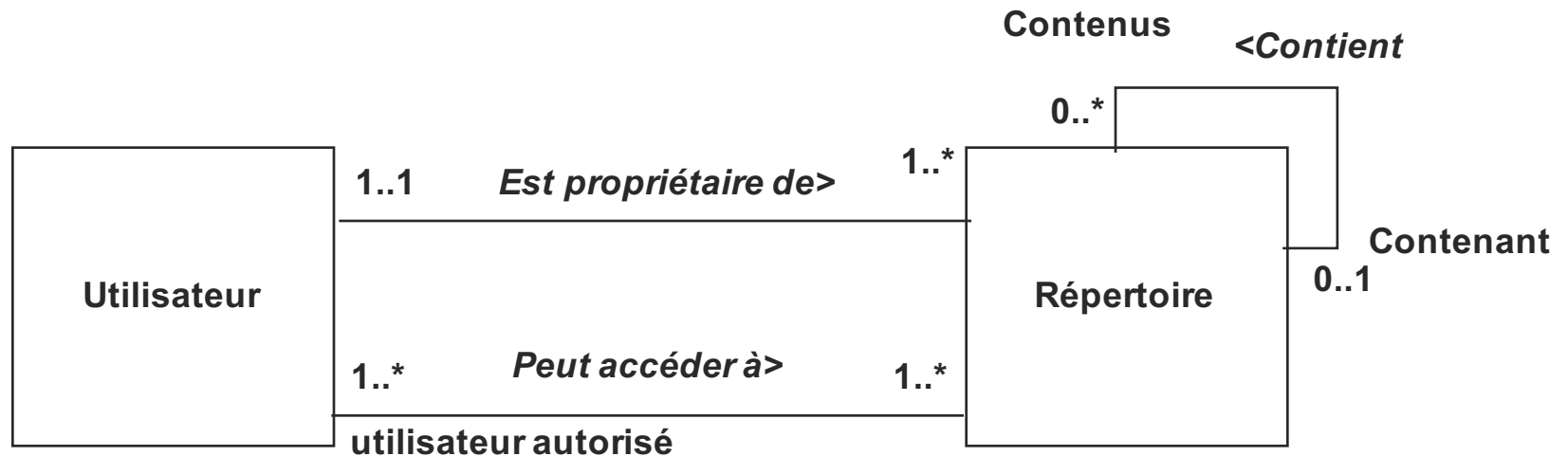
■ Exercice 2:

Description d'un système de fichiers

- Un utilisateur possède au moins un répertoire
- Un répertoire appartient à un et un seul utilisateur
- Un répertoire peut contenir d'autres répertoires
- Un utilisateur peut accéder à au moins un répertoire
- Un répertoire peut être accédé par au moins un utilisateur

Les relations (Association)

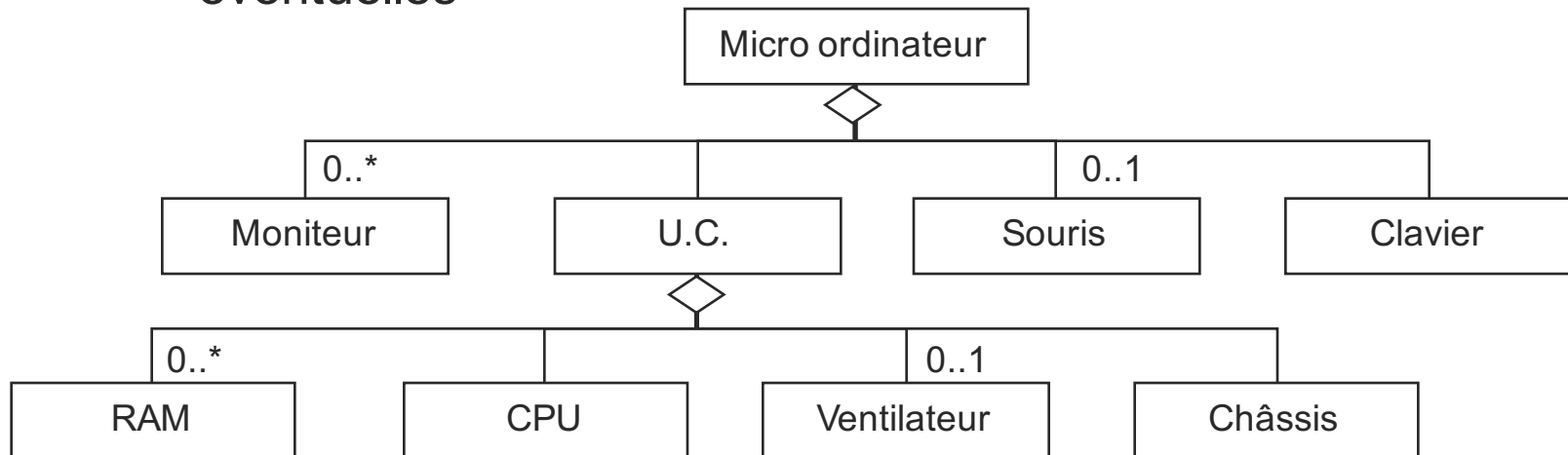
Solution Exercice 2 :



Les relations (Agrégation)

Association avec une sémantique spécifique

- Relation "made of" "part of"
- Transitive
- Antisymétrique
- Propagation des propriétés avec modifications locales éventuelles



- Agrégations récursives
- Propagation des opérations

Agrégation

■ Agrégation

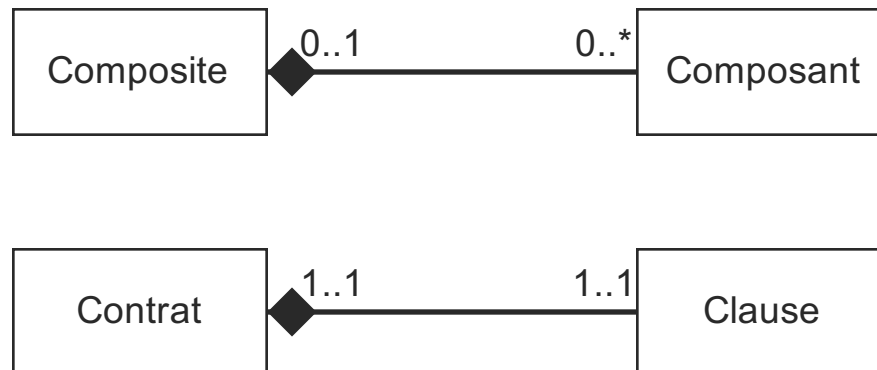
- Relation binaire de type tout-parties
- Relation d'appartenance faible : le composant peut être partagé entre plusieurs composites : multiplicité



Composition

■ Composition

- Agrégation avec relation d'appartenance forte et coïncidence des durées de vie
- Les composants peuvent être créés après le composite, mais après création ils ont la même durée de vie
- Les composants peuvent être enlevés avant la mort du composite
- Chaque composant appartient à, au plus, un composite
- Propagation composite vers composant



Généralisation/spécialisation

- Relation de classification entre un élément plus général et des éléments spécialisés
- L'élément spécialisé hérite des propriétés (attributs + opérations + relations) et des contraintes de l'élément dont il est la spécialisation
- L'élément spécialisé peut avoir ses propres attributs, opérations, relations
- Spécialisation simple ou multiple
- La spécialisation peut être contrainte à l'aide d'un stéréotype

Généralisation

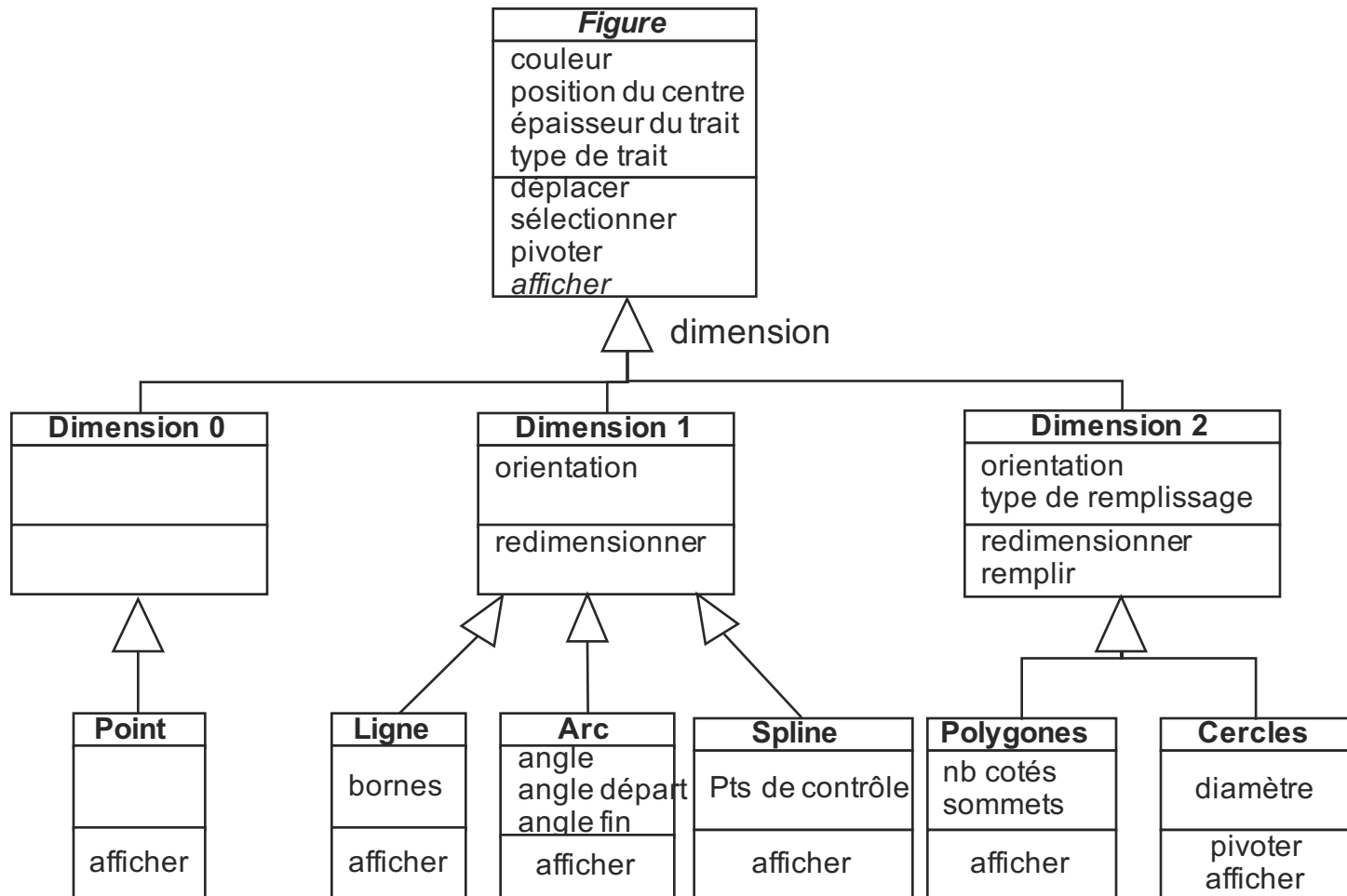
■ Généralisation

- Au niveaux des concepts
- Relation "is a" ou "a kind of" (Classification)
- Généralisation/Spécialisation
- Relation entre une classe (super-classe) et une ou plusieurs spécialisations (sous-classes)
- Une instance d'une sous-classe est également une instance de la super-classe
- Classe concrète et classe abstraite

Héritage

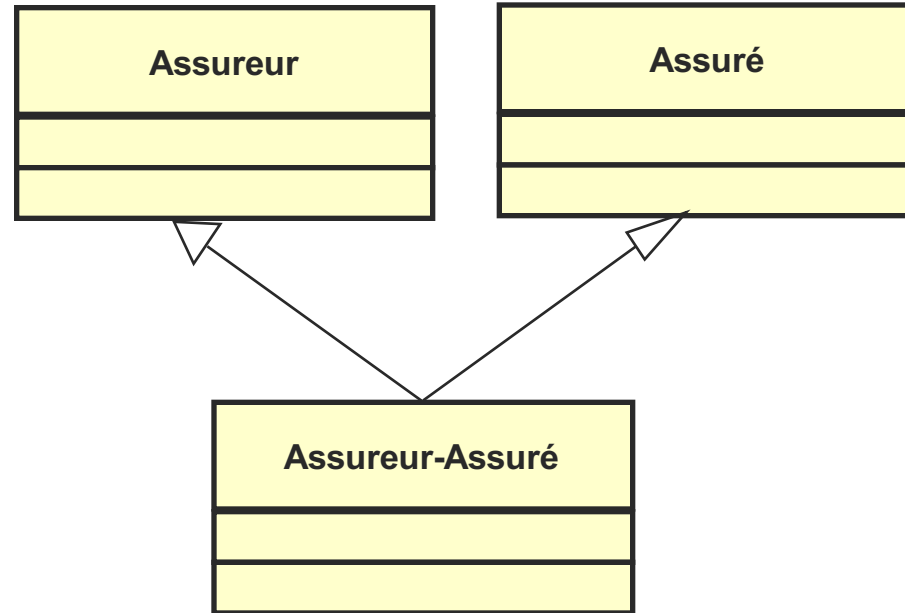
- Au niveau de le mise en œuvre (programmation OO)
- N'est pas la seule technique d'implantation d'une généralisation
- Héritage : Relation transitive sur un nombre arbitraire de niveaux
- Relation statique à couplage fort
- Réduire la complexité du code
- Maximiser la réutilisation

Généralisation et Héritage



Généralisation et Héritage

Héritage multiple :

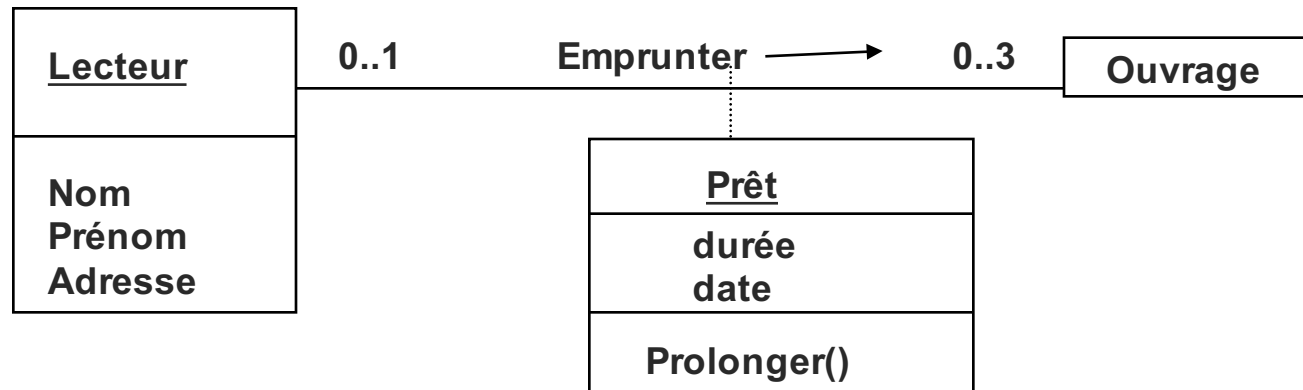


Remarques sur les Associations

- Ne prendre en compte que des associations binaires ! Si ce n'est pas possible, il faut privilégier les classes associations
- Le nommage est indispensable quand il y a plusieurs associations entre 2 classes
- Importance des contraintes !
 - Sous-ensemble, et/ou, etc.
 - {ordonné}, ...

Remarques sur les Associations

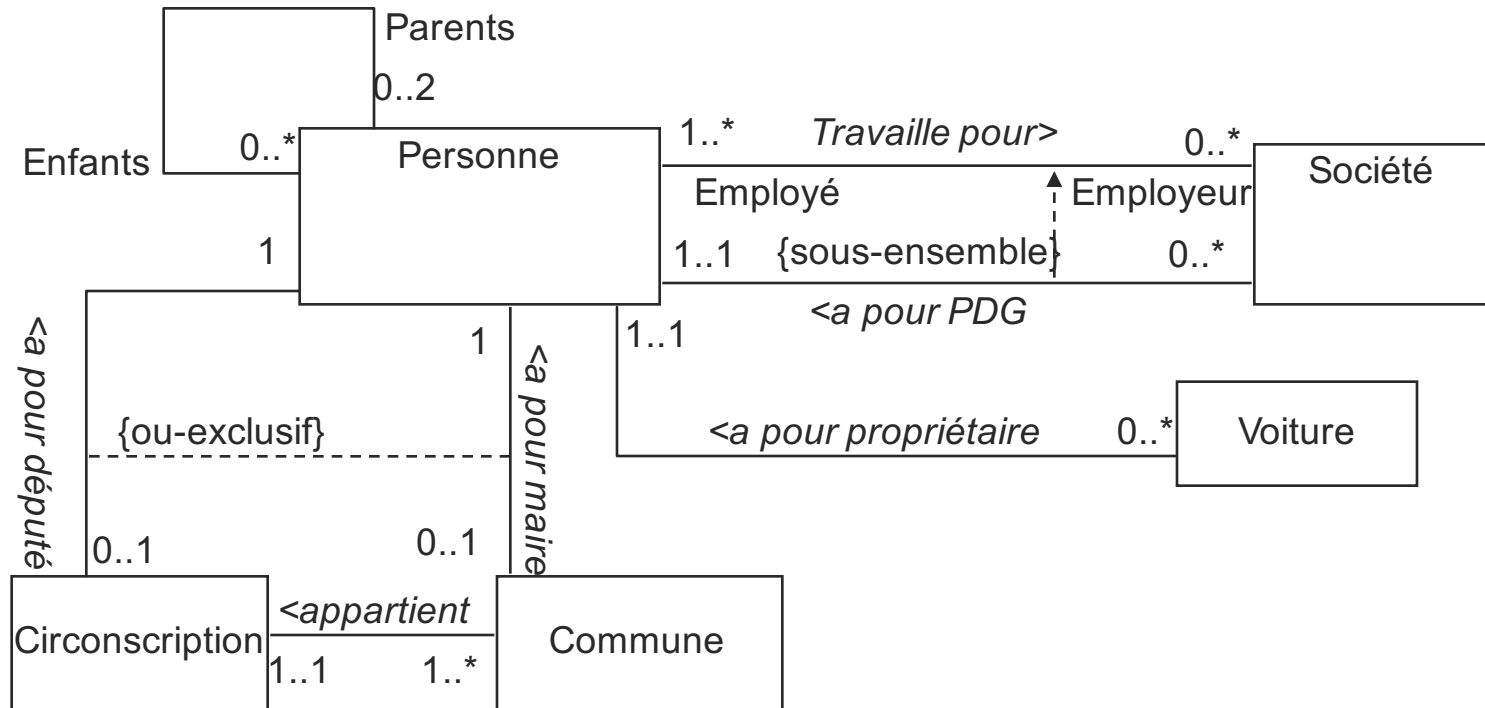
Exemple de Classe association



Exercice

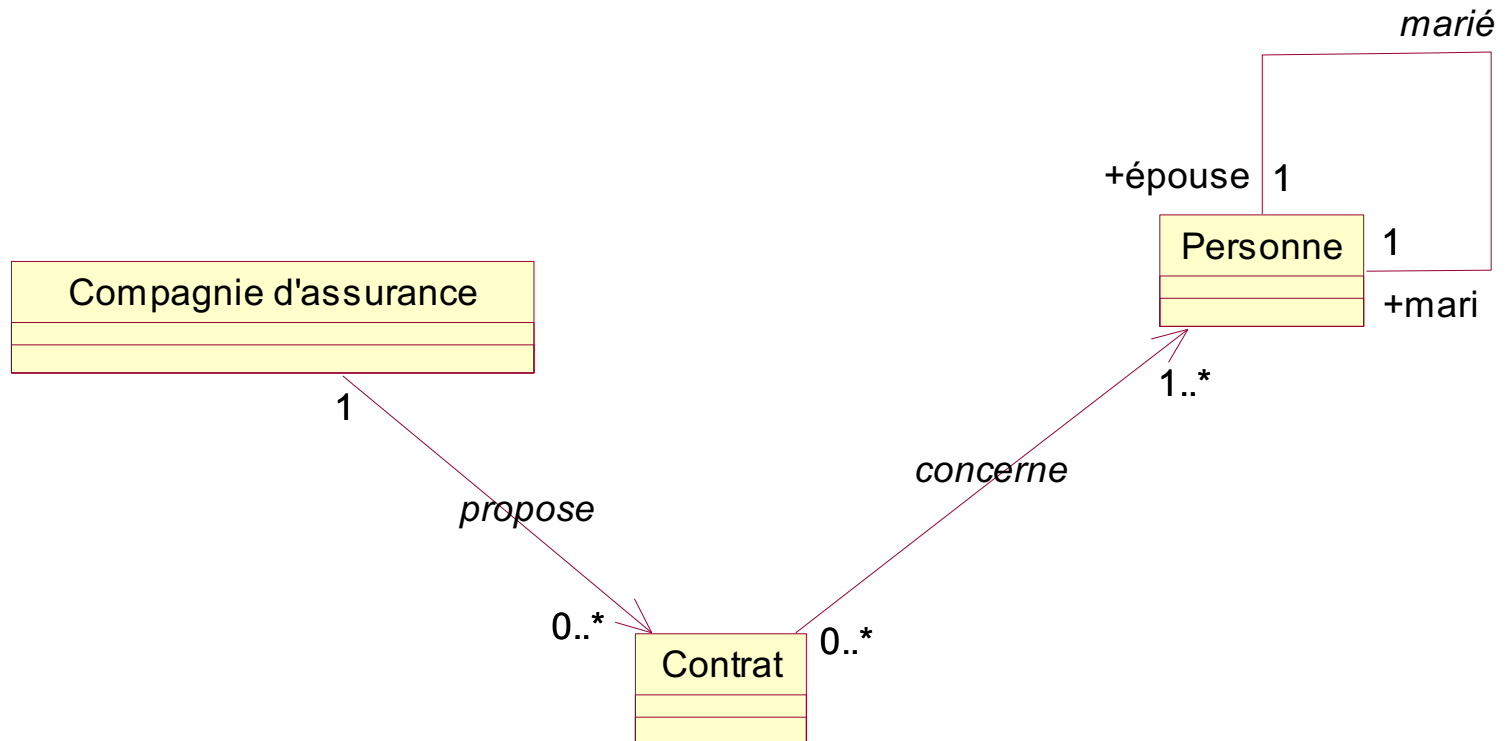
- Une personne peut travailler pour plusieurs sociétés
- Une société a un et un seul PDG
- Une personne peut être PDG de plusieurs sociétés
- Une personne peut posséder plusieurs voitures
- Une voiture a un seul propriétaire
- Une personne peut avoir des enfants
- Une personne peut être le maire d'une commune
- Une personne peut être le député d'une circonscription
- Une commune appartient à une circonscription

Exercice 3



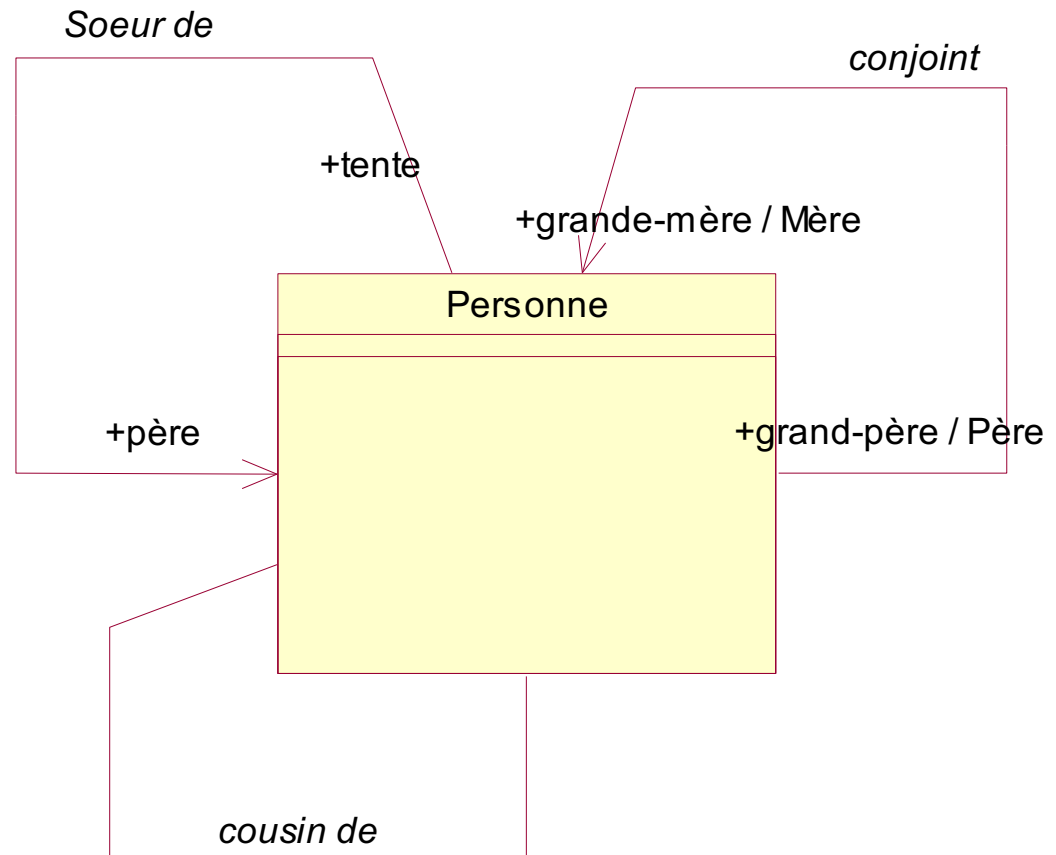
Exercice 1

Réalisez le diagramme de classes d'une compagnie d'assurance proposant des contrats d'assurance à des époux (mari et femme).



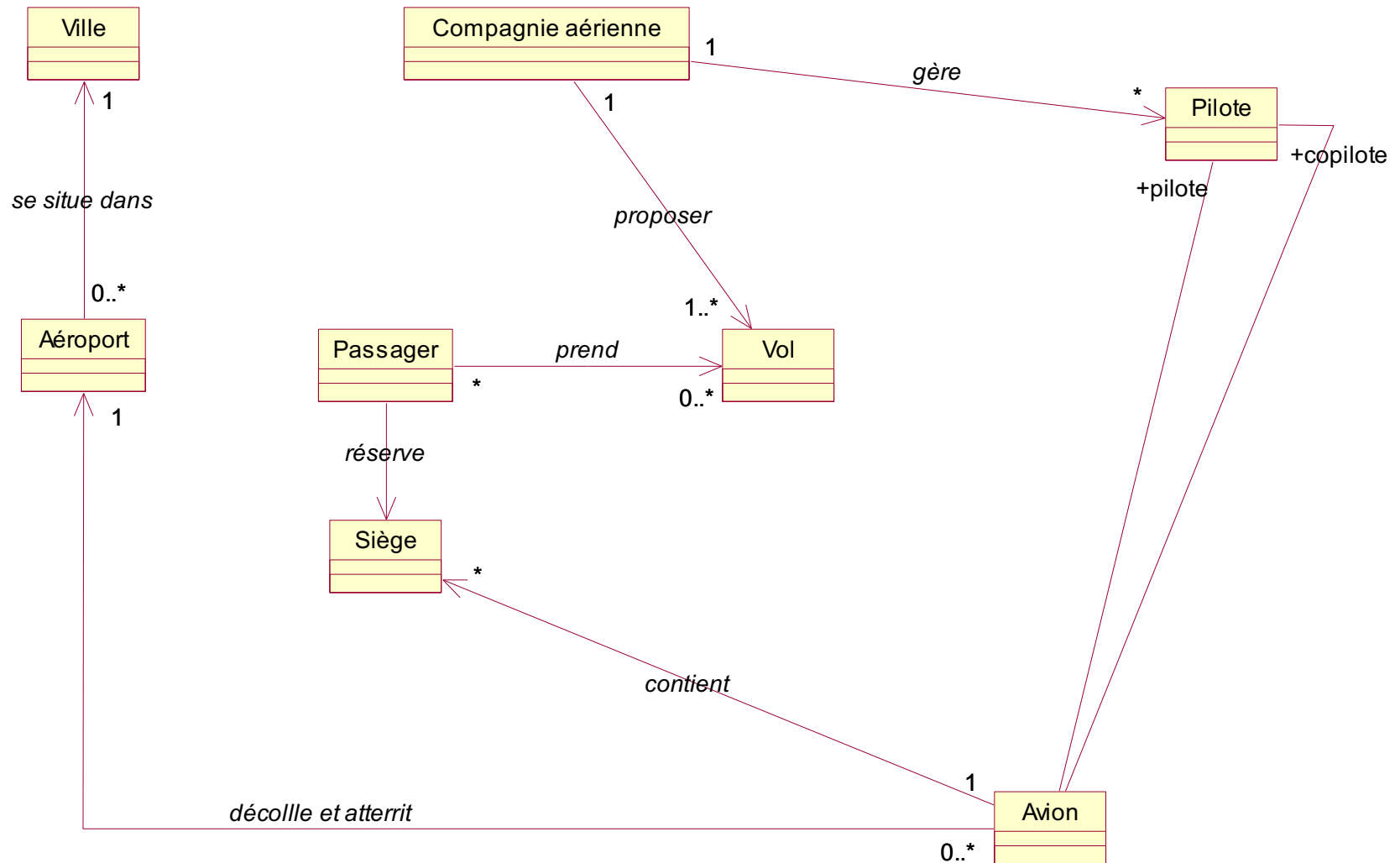
Exercice 2

Réalisez le diagramme de classes qui correspond à l'arbre généalogique de votre famille comprenant les membres suivants : votre Grand-père conjoint de votre Grand-mère, votre tante sœur de votre père, votre père conjoint de votre mère, votre cousin et vous même.



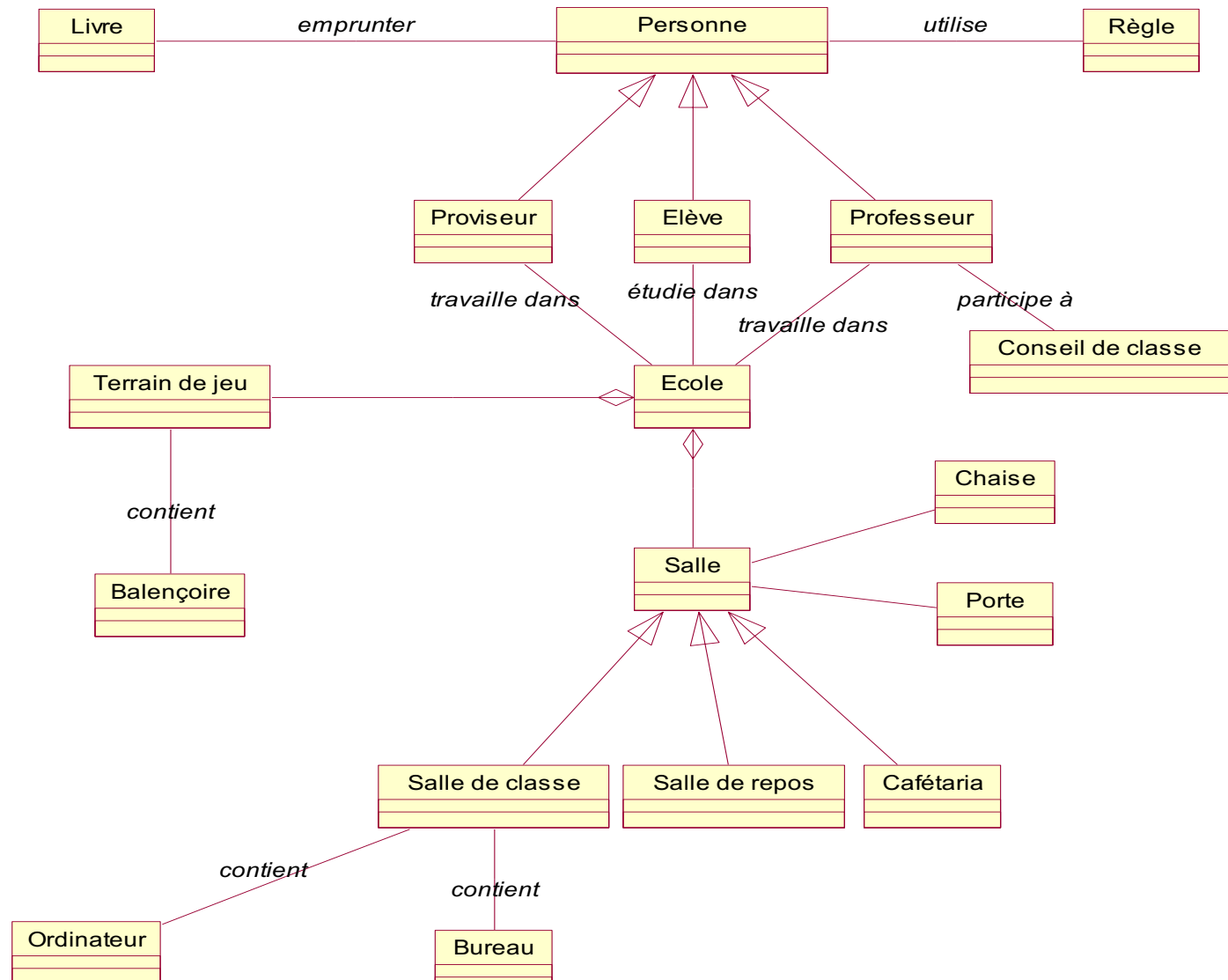
Exercice 4

Réalisez le diagramme de classes comprenant les objets suivants : ville, compagnie aérienne, pilote, copilote, avion, vol, aéroport, passager, siège.



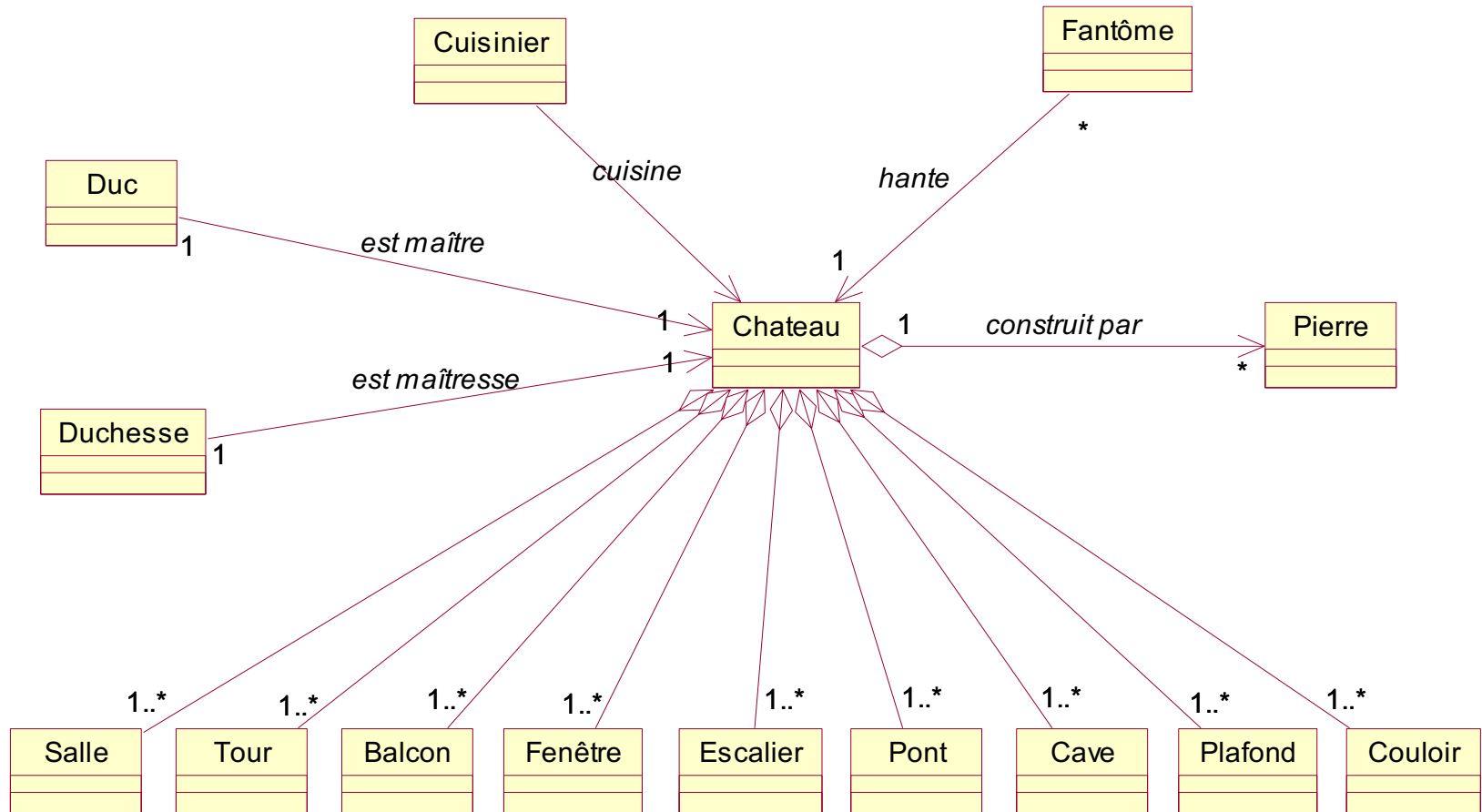
Exercice 5

Réalisez le diagramme de classes comprenant les objets suivants : école, terrain de jeu, proviseur, conseil de classe, salle de classe, livre, élève, professeur, cafétéria, salle de repos, ordinateur, bureau, chaise, règle, balançoire.



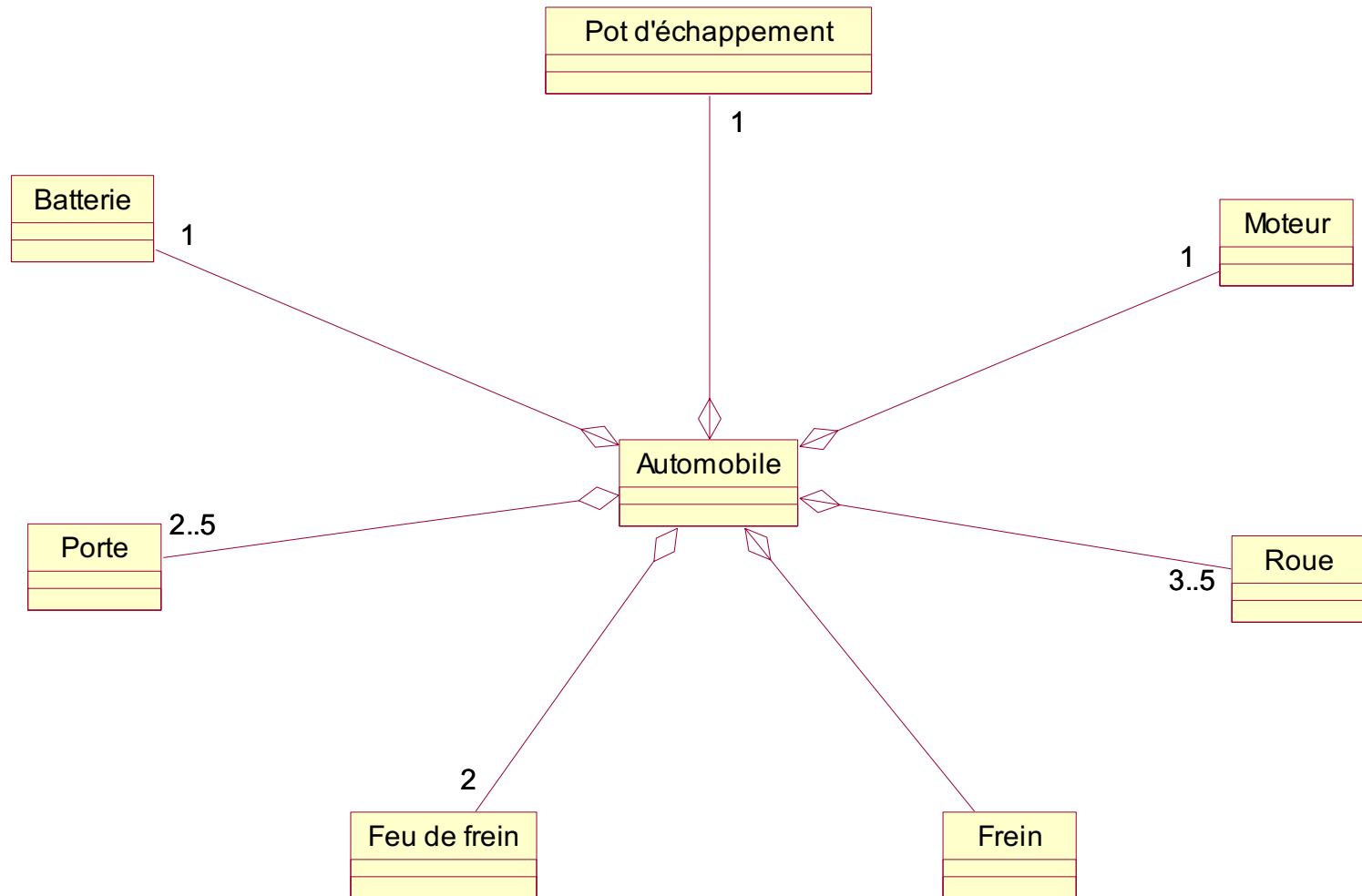
Exercice 6

Réalisez le diagramme de classes comprenant les objets suivants :
château, balcon, pont, tour, fantôme, escaliers, cave, plafond, couloir, salle
fenêtre, pierre, duc, duchesse, cuisinier.



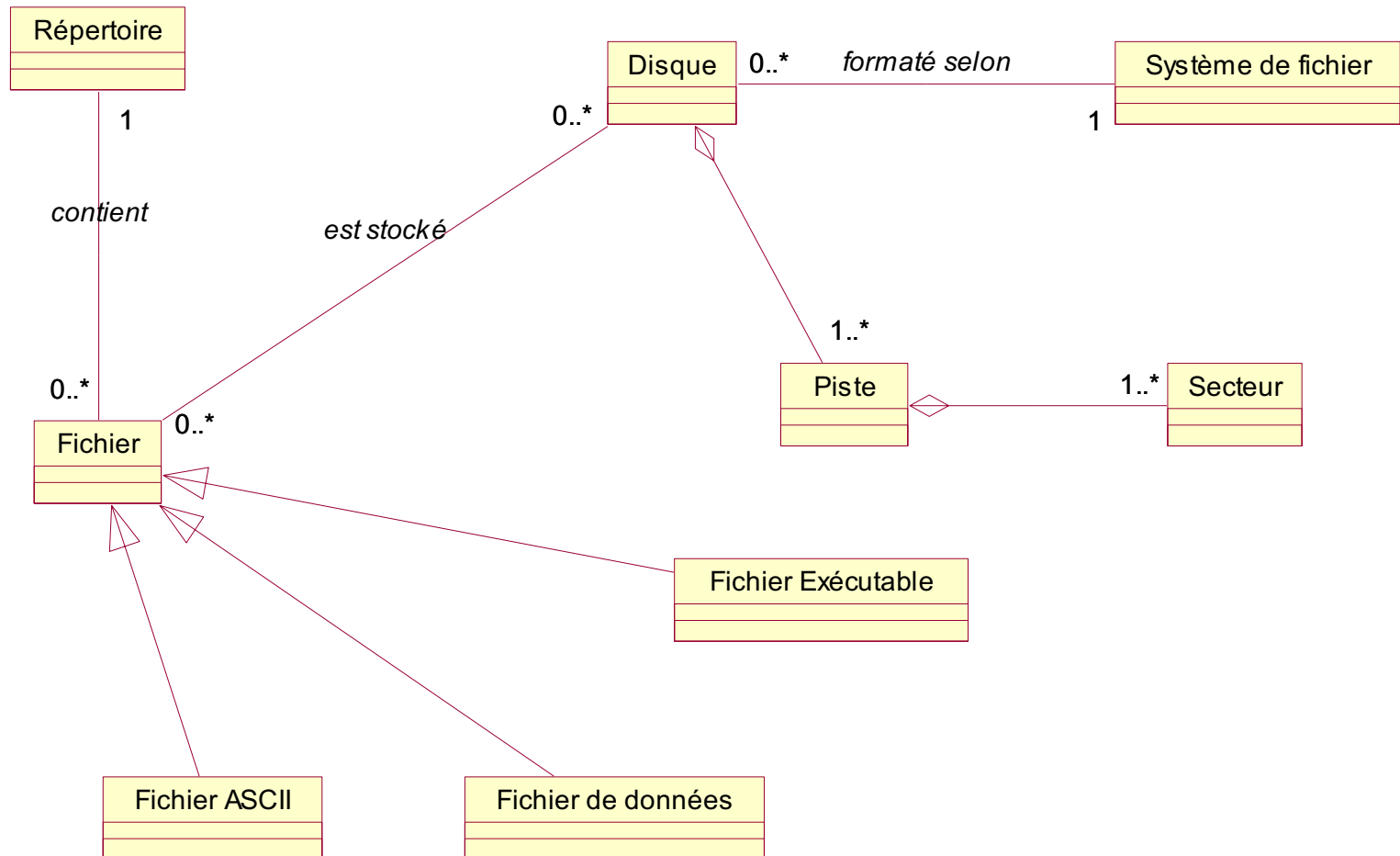
Exercice 7

Réalisez le diagramme de classes comprenant les objets suivants : automobile, moteur, roue, frein, feu de frein, porte, batterie, pot d'échappement.



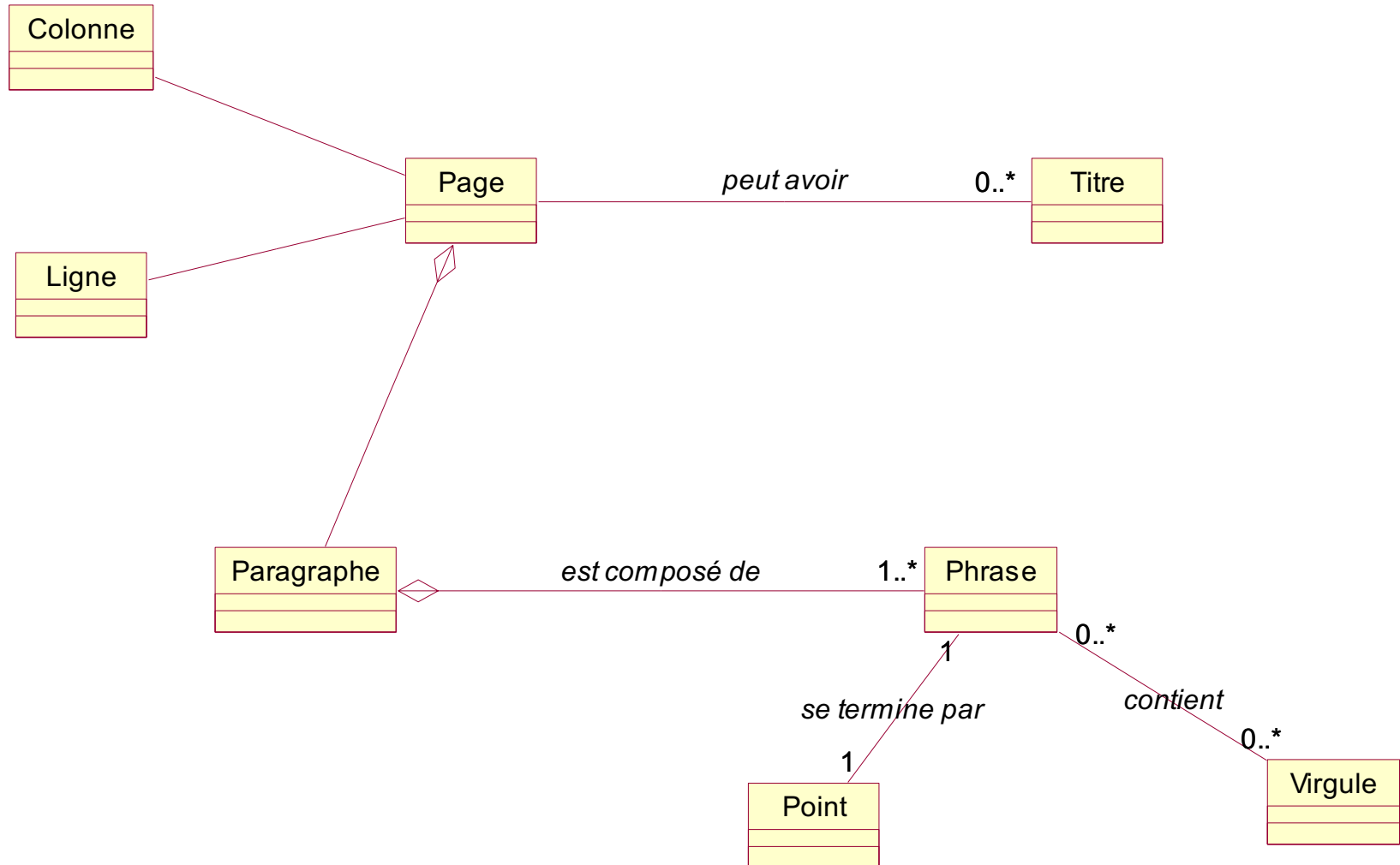
Exercice 8

Réalisez le diagramme de classes comprenant les objets suivants : système de fichier, fichier, nom de fichier, fichier ASCII, fichier exécutable, fichier répertoire, disque, piste, secteur.



Exercice 9

Réalisez le diagramme de classes comprenant les objets suivants : page, paragraphe, phrase, ligne, colonne, titre, sous-titre, point, virgule.








Exercice 10

Un écolier utilise un stylo à encre. Mohamed utilise un stylo à encre bleu. Un écolier est âgé de 6 à 12 ans. Mohamed est âgé de 7 ans. Représenter le diagramme de classe.



Récapitulons !

- Un objet est une représentation valuée et abstraite du monde réel
- Les objets de même type sont représentés par une seule classe
- 3 types de relations entre les classes :
 - Association 
 - Agrégation/Composition  
 - Généralisation/Héritage  

III- La Modélisation avec UML

■ III-1- Notions de Base

III-1-a- Les Modèles UML

Un modèle est une abstraction de la réalité :

- ➔ L'abstraction est un des piliers de l'approche objet
 - ➔ un processus qui consiste à identifier les caractéristiques intéressantes d'une entité, en vue d'une utilisation précise
 - ➔ L'abstraction désigne aussi le résultat de ce processus, c'est-à-dire l'ensemble des caractéristiques essentielles d'une entité, retenues par un observateur.

III- La Modélisation avec UML

- Un modèle est 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 très subjective de la réalité

III-1-b- Rédaction d'un Diagramme

UML est un langage qui permet de définir et de visualiser un modèle, à l'aide de diagrammes.

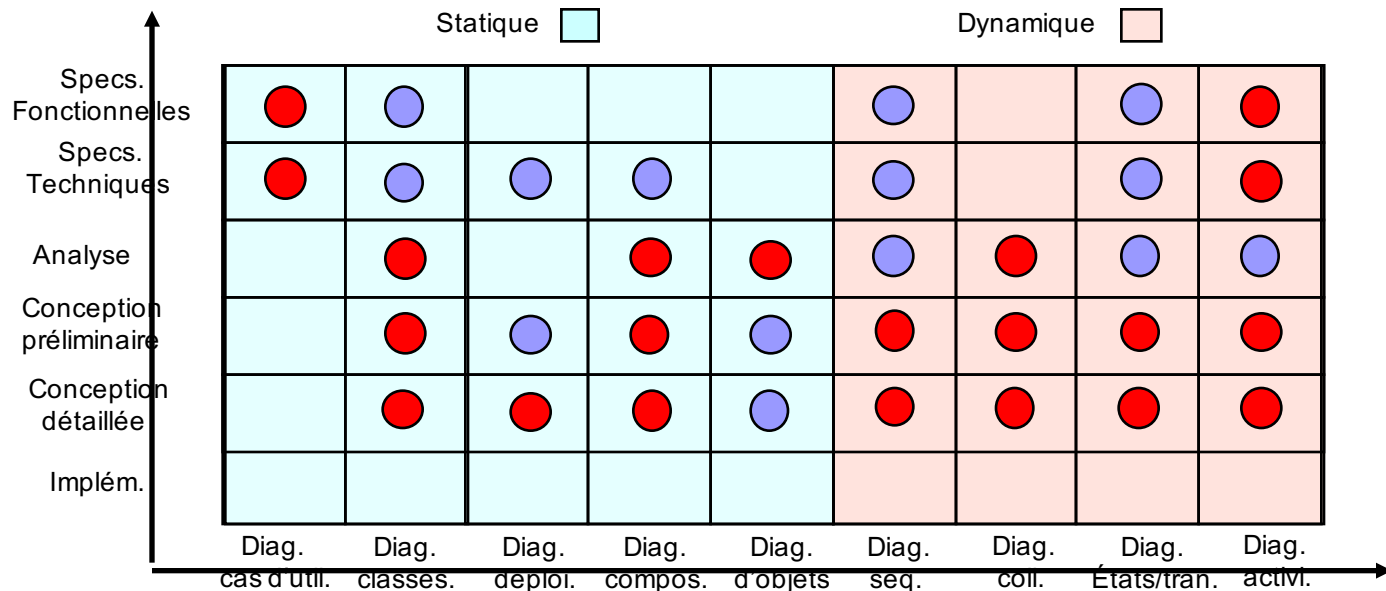
Un diagramme UML est une représentation graphique, qui s'intéresse à un aspect précis du modèle ; c'est une perspective du modèle, pas "le modèle".

III- La Modélisation avec UML

- Chaque type de diagramme UML possède une structure (les types des éléments de modélisation qui le composent sont prédéfinis).
- Un type de diagramme UML véhicule une sémantique précise (un type de diagramme offre toujours la même vue d'un système).
- Combinés, les différents types de diagrammes UML offrent une vue complète des aspects statiques et dynamiques d'un système.
- Par extension et abus de langage, un diagramme UML est aussi un modèle (un diagramme modélise un aspect du modèle global).

La Modélisation avec UML

- UML se base sur une notation graphique et propose neuf types de diagrammes. Ces derniers représentent d'une part les aspects **statiques** et **dynamiques** et couvrent d'autre part l'ensemble des phases de développement.



III- La Modélisation avec UML

■ *III-1-c- Les différents types de diagrammes UML :*

Vues statiques du système :

- diagrammes de cas d'utilisation
- diagrammes d'objets
- diagrammes de classes
- diagrammes de composants
- diagrammes de déploiement

III- La Modélisation avec UML

■ *III-1-c- Les différents types de diagrammes UML :*

Vues Dynamiques du système :

- diagrammes de collaboration
- diagrammes de séquence
- diagrammes d'états-transitions
- diagrammes d'activités

Diagramme de Cas d'utilisation

III- La Modélisation Statique

1- Diagramme de Cas d'utilisation

- Représentation du modèle conceptuel
- Structuration des besoins des utilisateurs
- Structuration des objectifs correspondants d'un système
- Expression des exigences du système sur ses utilisateurs
- Limitation des préoccupations "réelles" des utilisateurs
- Pas de présentation de solutions d'implémentation

III- La Modélisation Statique

1- Diagramme de Cas d'utilisation

- Ils identifient les utilisateurs du système (acteurs) et leur interaction avec le système
- Ils permettent de classer les acteurs et structurer les objectifs du système
- Les Use Case servent de base à la traçabilité des exigences d'un système dans un processus de développement intégrant UML

III- La Modélisation Statique

1- Diagramme de Cas d'utilisation

■ *Éléments de base des cas d'utilisation :*

➔ **Acteur** : entité **externe** qui agit sur le système (opérateur, autre système...)

- L'acteur peut consulter ou modifier l'état du système
- En réponse à l'action d'un acteur, le système fournit un service qui correspond à son besoin
- Les acteurs peuvent être classés (hiérarchisés)

■ On distingue :

- les *acteurs primaires* : ceux sont les utilisateurs du système
- les *acteurs secondaires* : ceux qui administrent le système

III- La Modélisation Statique

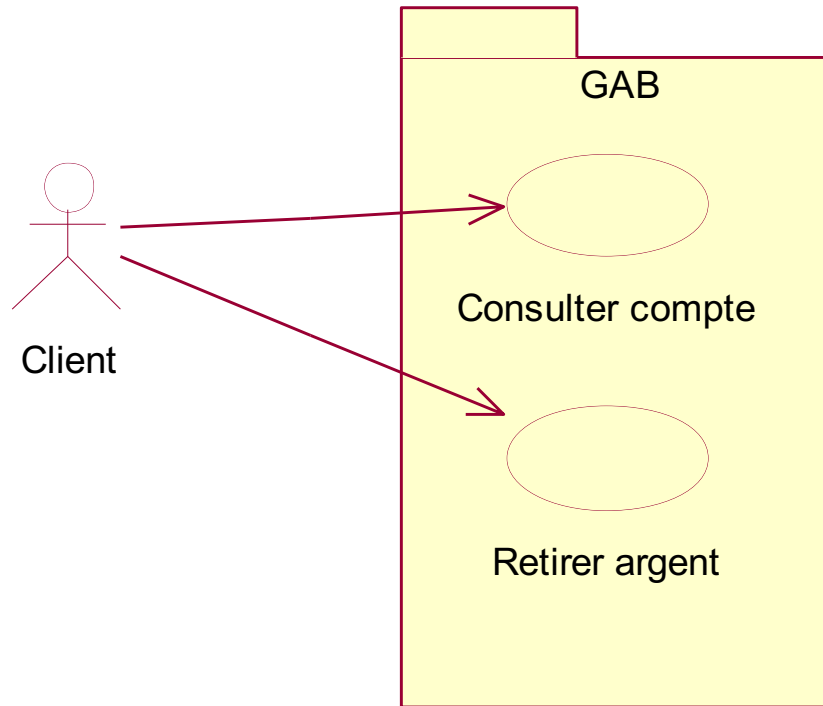
1- Diagramme de Cas d'utilisation

- **Use case** : ensemble d'actions réalisées par le système, en réponse à une action d'un acteur.
 - Les uses cases peuvent être structurés
 - Les uses cases peuvent être organisés en paquetages (packages)
 - L'ensemble des use cases décrit les objectifs (le but) du système

III- La Modélisation Statique

1- Diagramme de Cas d'utilisation

- ***Formalisme de représentation des cas d'utilisation avec UML :***



III- La Modélisation Statique

1- Diagramme de Cas d'utilisation

« Un cas d'utilisation spécifie une séquence d'interactions, avec ses variantes, entre les acteurs et le système, produisant un résultat satisfaisant pour un acteur particulier »

➔ un cas d'utilisation est une abstraction de plusieurs chemins d'exécution d'une utilisation du système

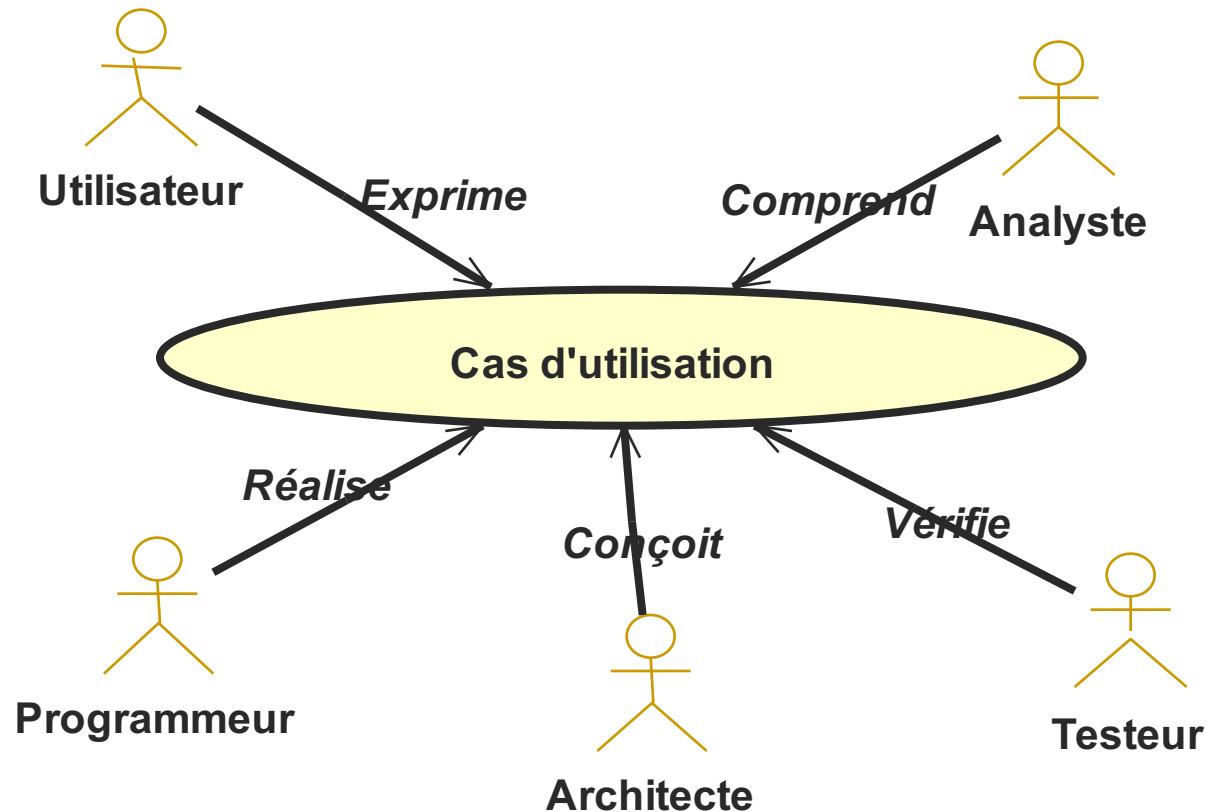
➔ Pour décrire un cas d'utilisation, il faut décrire un maximum de chemins d'exécution possibles pour la séquence d'actions correspondant à ce cas. On étudiera donc un certain nombre de scénarios d'un cas d'utilisation

➔ Un scénario est donc une instance de cas d'utilisation

Cas d'utilisation

- Ce que le système doit faire (comportement souhaité)
- Mais pas comment réaliser ce comportement
 - Pas de détails de programmation, mise en oeuvre, etc.
 - Indépendant de la réalisation
- Un outil pour communiquer
 - utilisateur final / expert du domaine <---> concepteur / développeur

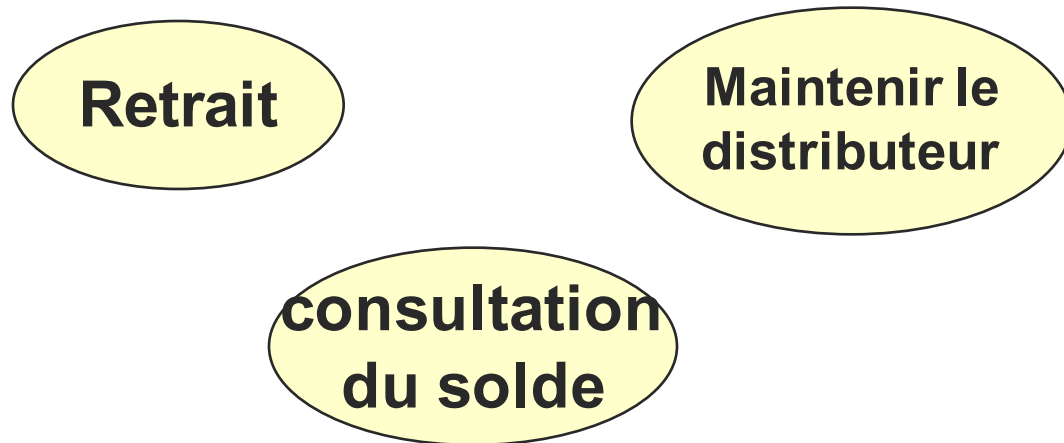
Cas d'utilisation



Les cas d'utilisation servent de Fil conducteur du projet

Cas d'utilisation

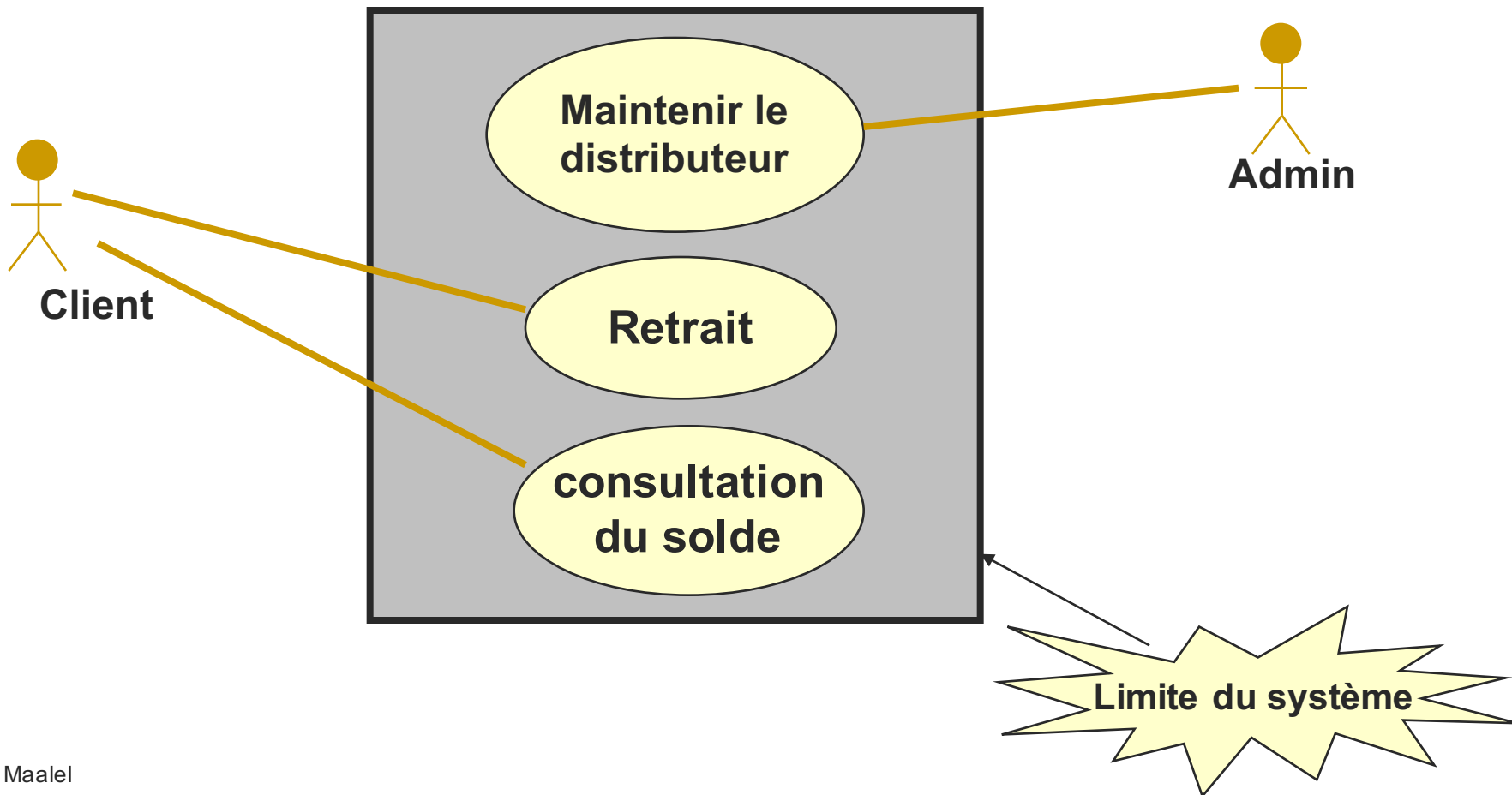
■ Exemple :



On considère le système comme un ensemble de besoins et non comme une solution.

Cas d'utilisation

- Relation entre acteur et cas d'utilisation :

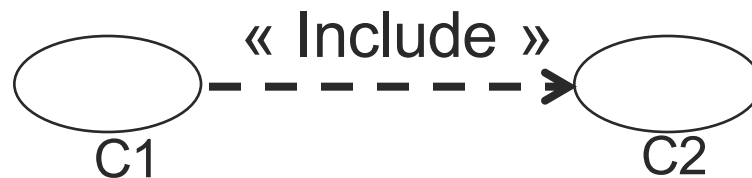


Organisation de cas d'utilisation

- Organisation des cas d'utilisation
 - Inclusion : formaliser par le terme « include ».
 - Extension : formaliser par le terme « extend ».
 - Généralisation/spécialisation.

Organisation de cas d'utilisation

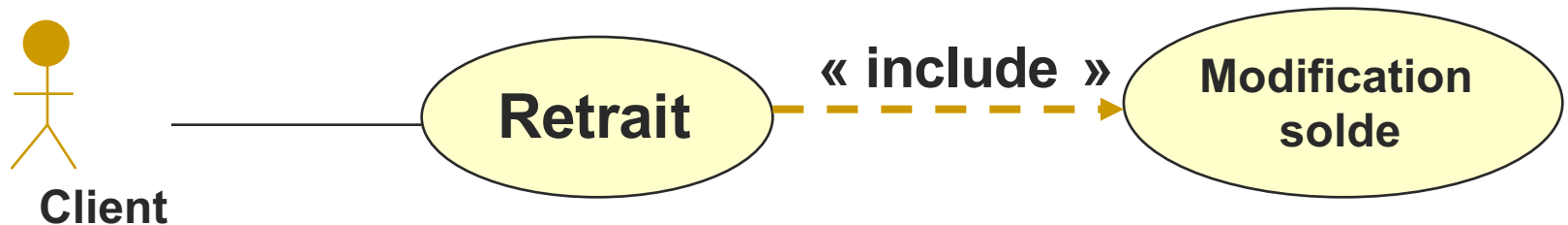
- Inclusion :
 - Parfois un cas d'utilisation **peut** avoir besoin de l'aide d'un autre cas c'est l'inclusion.
 - Permet d'identifier un sous ensemble commun à plusieurs cas



- C1 utilise C2  toute activation de C1 entraîne une activation de C2

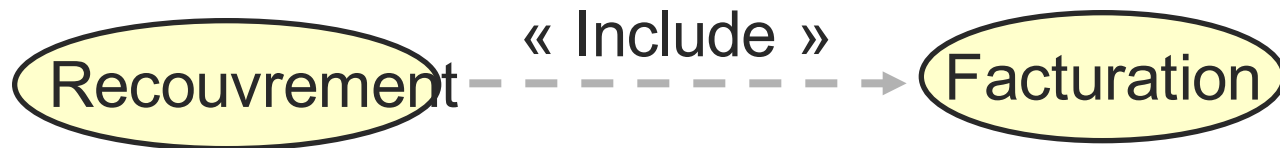
Organisation de cas d'utilisation

- Inclusion :
 - Ex : le cas d'utilisation retrait peuvent ne pas modifier le compte bancaire mais délégué cette opération a un autre cas d'utilisation mise a jour compte.



Organisation de cas d'utilisation

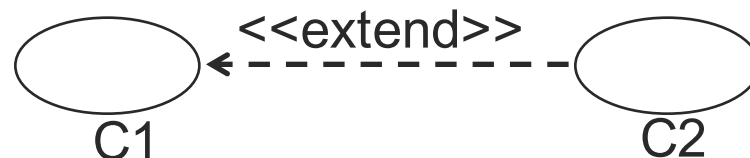
- Inclusion :
 - Ex : le service recouvrement a besoin de savoir les facturations qui ont été faites par le service de facturation a fin de traiter les chèques reçus et les impayés...



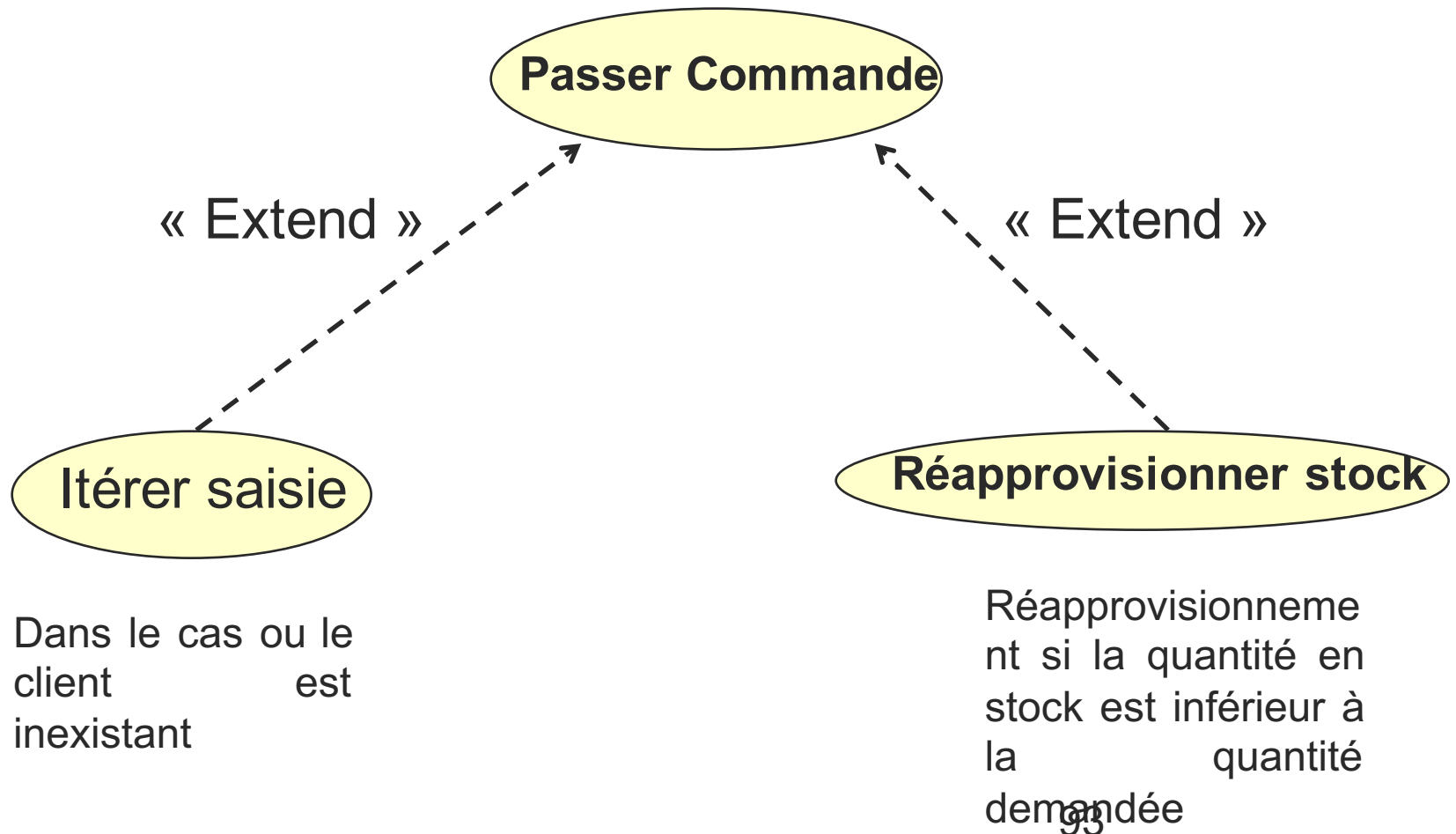
Organisation de cas d'utilisation

■ Extension :

- Parfois un cas d'utilisation *pourrait* avoir besoin d'un autre cas.
- Permet d'identifier les comportements alternatifs ou exceptionnels (erreurs,...).
- C2 étend C1 🦋 C2 est une façon particulière de réaliser C1.
- C2 n'est pas activable directement.



Organisation de cas d'utilisation



Organisation de cas d'utilisation

■ Passer commande

1. Entrer le nom et le numéro de compte
2. Vérifier qu'ils sont valides

Extension: itérer la saisie si client inexistant

3. Saisir l'article commandé et la quantité
4. Vérifier la quantité en stock

Extension : réapprovisionner si la quantité en stock n'est pas suffisante

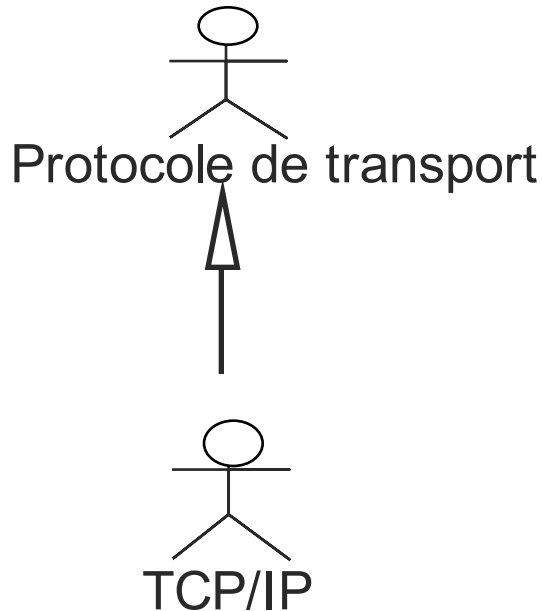
5. Enregistrer la commande

Organisation de cas d'utilisation

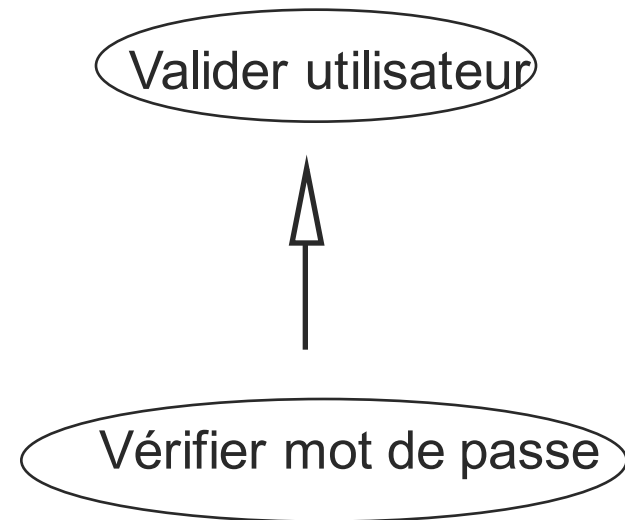
- Généralisation :
 - L'héritage est un concept fondamentale en programmation, en analyse et en conception Orienté Objet.
 - Cette idée appliquée aux acteurs et aux cas d'utilisation et appelée généralisation et est souvent surnommée relation « **est un** ».

Organisation de cas d'utilisation

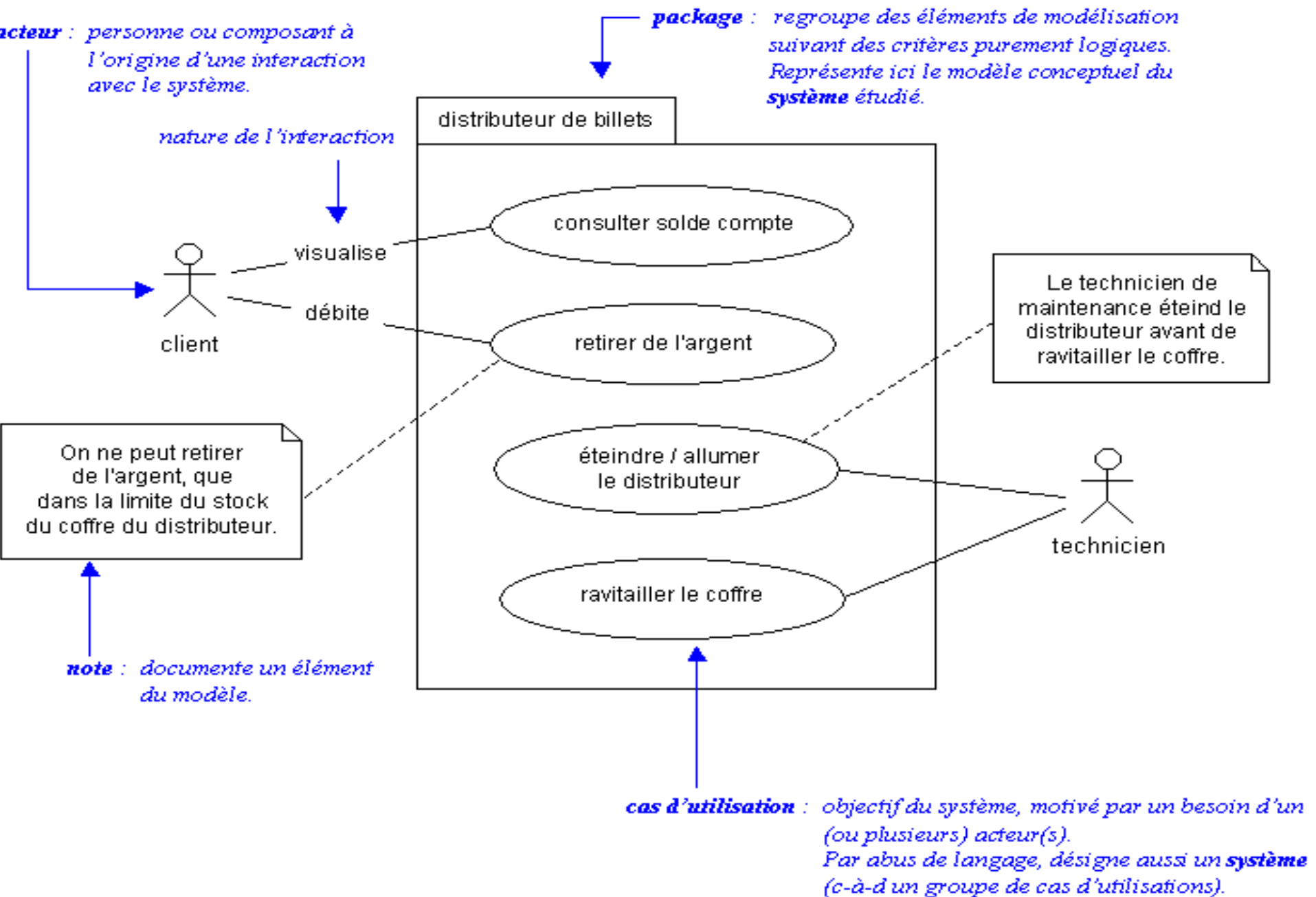
- Généralisation



L'acteur spécialisé *hérite* des interactions de l'acteur père.



Le cas d'utilisation spécialisé est *substituable* au cas d'utilisation⁹⁶ père.



Construction du diagramme des cas d'utilisation

1. Définir le contexte du système.
2. Identifier les acteurs.
3. Identifier les messages.
4. Dresser le diagramme de contexte dynamique.
5. Identifications des cas d'utilisation.
6. Définition des associations entre acteur et cas d'utilisation.
7. Amélioration (évaluations des acteurs et des cas d'utilisations).
8. Voir les dépendances (include, extend, généralisation).
9. Représenter le diagramme final.

Exemple

1. Définir le contexte du système.

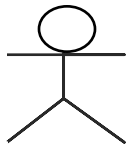
- **Réception:** les employés de la réception reçoivent les livraisons et vérifient que la marchandise livrée correspond au bon de commande. Il informe la comptabilité fournisseur de la réception.
- **Stockage:** les produits reçus peuvent provenir de commande annulée, retour ou commande fournisseur. Les employés du stock mettent à jour le stock en entrant la quantité reçue.
- **Traitement commandes:** d'autres employés préparent la commande à traiter et informant le service de commande que cette dernière a été traitée.
- **Expédition:** les employés du service expédition emballent et les préparent pour l'expédition, mettent à jour le stock, contactent le transporteur et notifient la livraison au service commande.

Construction du diagramme des cas d'utilisation

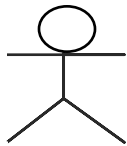
2. Identifier les acteurs.

« Acteur »
Syst service
commande

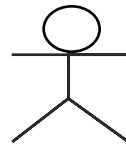
« Acteur »
Syst comptabilité
fournisseur



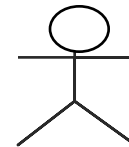
Réception



Expédition



Traitement
commande



stockage

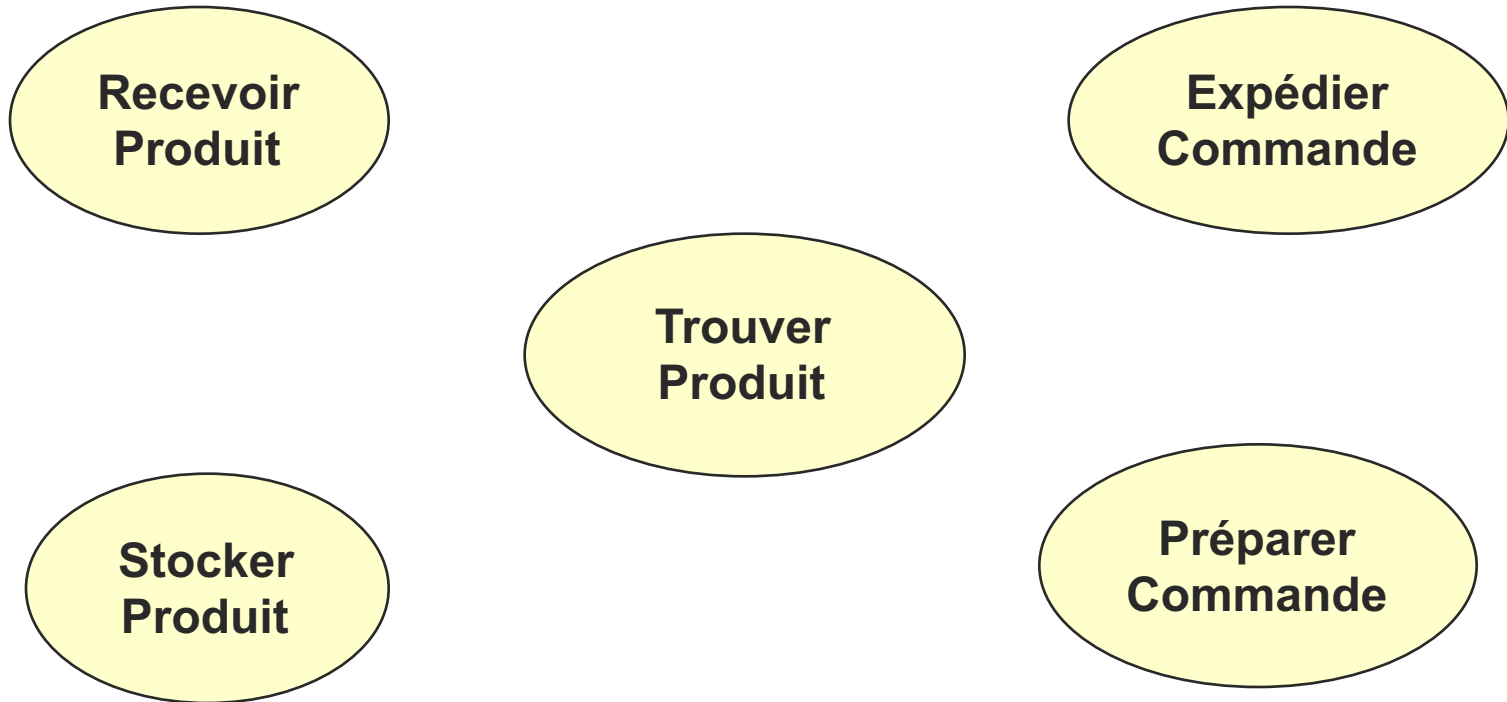
Construction du diagramme des cas d'utilisation

3. Identifier les cas d'utilisation.

- Que doit produire le système pour chaque acteurs. (identifie le travail du système)
- Comment chaque acteur doit il aider le système. (identifie le travail du système)
- Quel sont les tache pour les quel le système aide les acteurs. (identifie les règles)

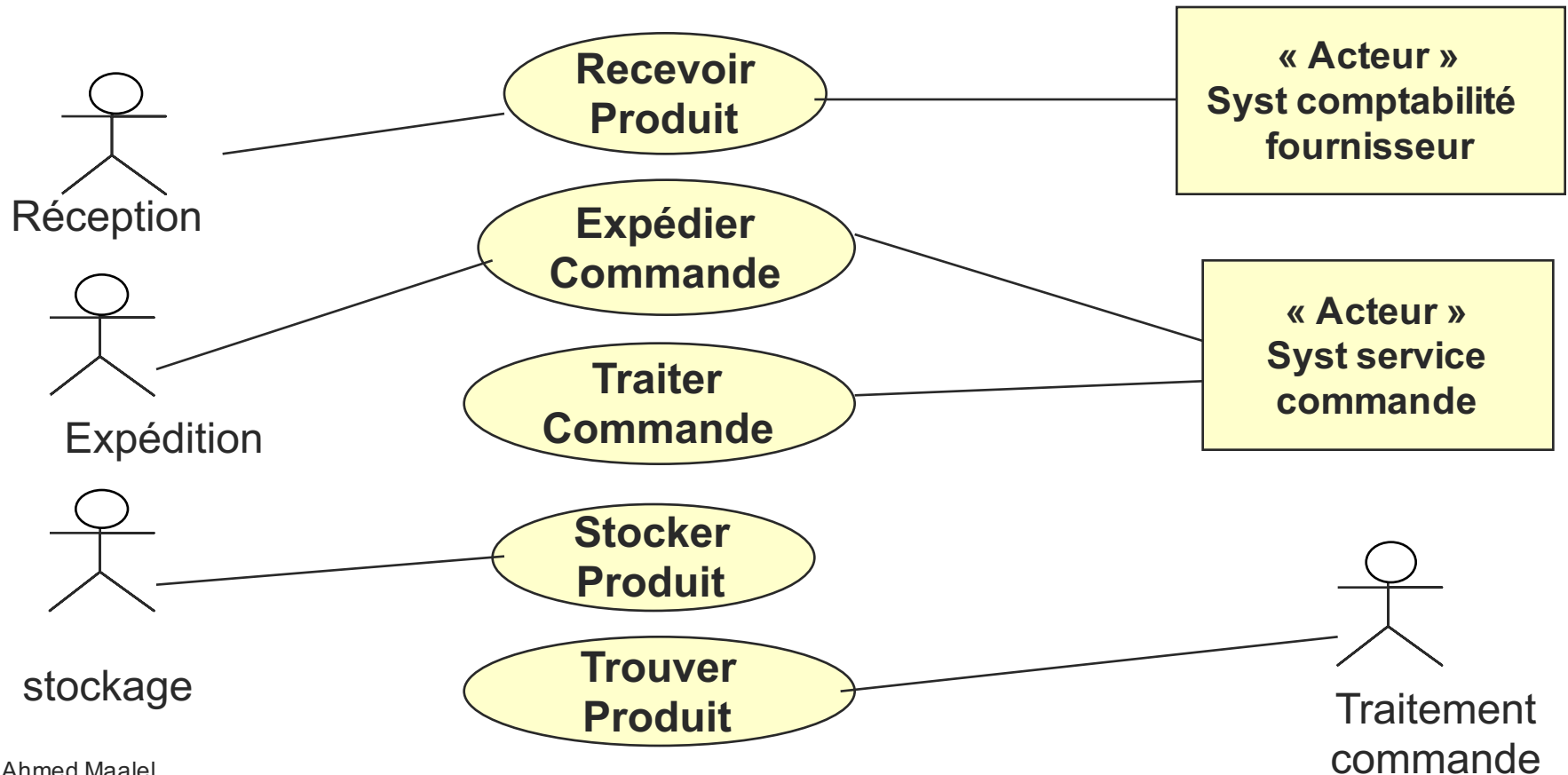
Construction du diagramme des cas d'utilisation

4. Identifier les cas d'utilisation.



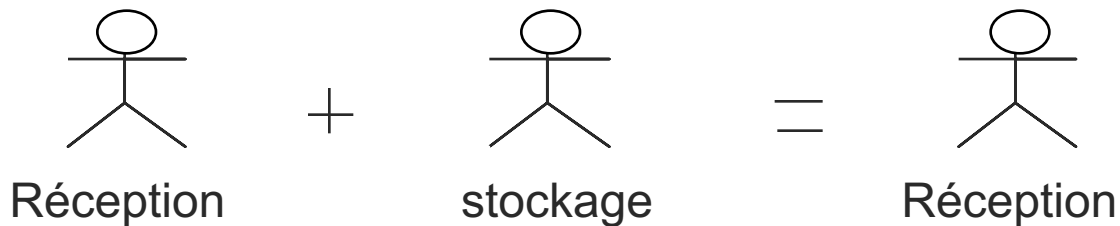
Construction du diagramme des cas d'utilisation

5. Définition des associations entre acteur et cas d'utilisation.



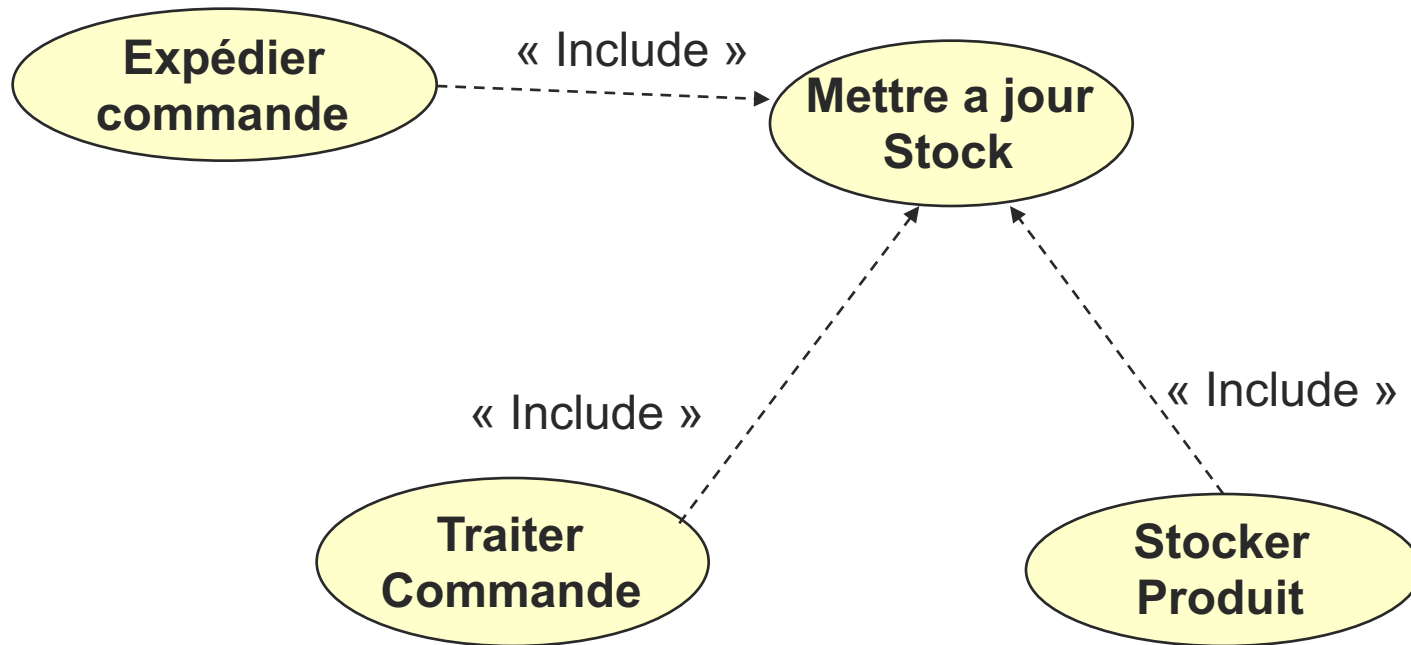
Construction du diagramme des cas d'utilisation

6. Amélioration (évaluations des acteurs et des cas d'utilisations).



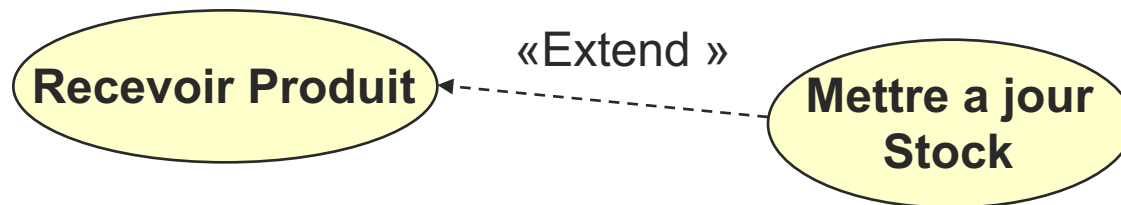
Construction du diagramme des cas d'utilisation

7. Voir les dépendances (include).



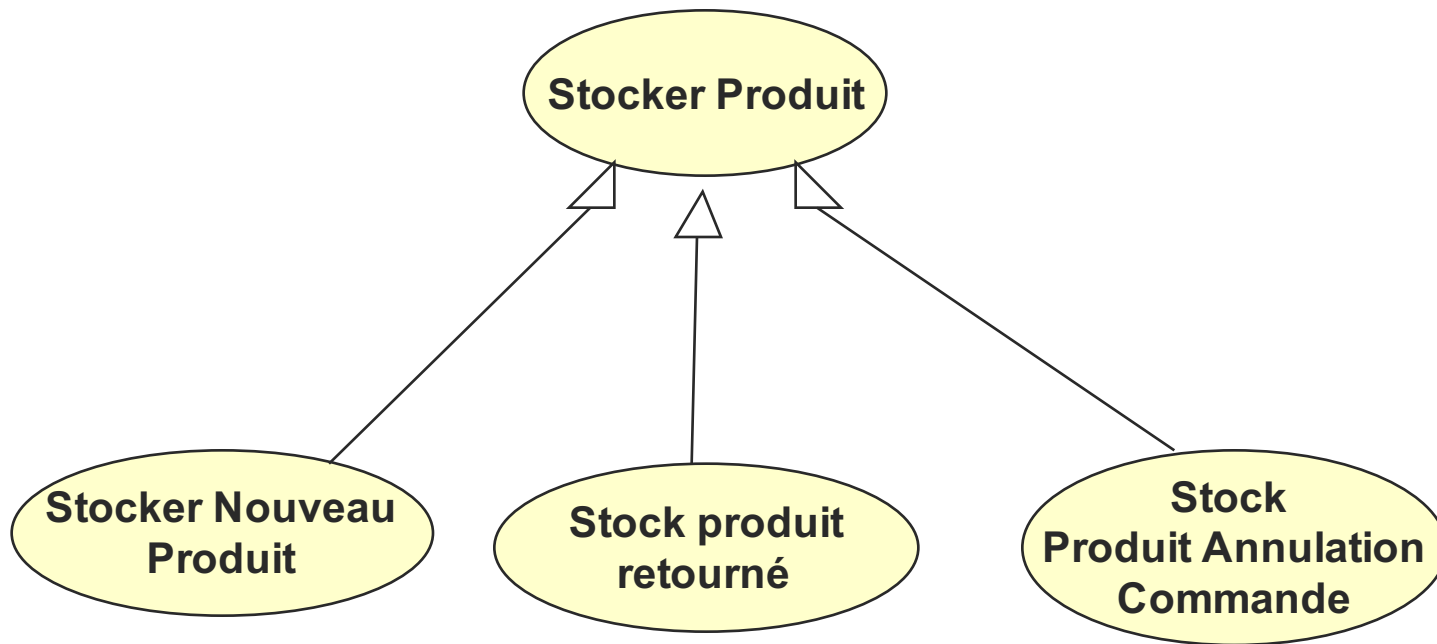
Construction du diagramme des cas d'utilisation

8. Voir les dépendances (extend).



Construction du diagramme des cas d'utilisation

8. Voir les dépendances (généralisation).



Insuffisance des cas d'utilisations

- Diagramme de cas d'utilisation à lui tout seul est insuffisant.
- Il faut :
 1. établir la traçabilité entre le cahier des charges et le modèle des cas d'utilisations.
 2. Expliciter le comportement du système pour chacun des cas d'utilisations par :
 - Description textuelle
 - Prototype des interfaces
 - Un ou plusieurs diagrammes de séquences/scénarios
 - Un ou plusieurs diagrammes Etats/transitions

Description des cas d'utilisation

■ Exemple de description :

Sommaire d'identification

Titre :

But :

Résumé :

Acteurs :

Date de création :

Date de mise à jour :

Version :

Responsable :

Description des cas d'utilisation

■ Exemple de description (2) :

Description des enchaînements

- ***Préconditions*** : Ce qui doit être vérifié avant que le Cas d'utilisation commence.
- ***Enchaînements*** :
 - Evènements de déclenchement
 - Séquence nominale/C.U. référencés
 - Séquences Exceptionnelles/Exceptions
- ***Exceptions*** :
- ***Postconditions*** : ce qui est vrai après que le Cas d'utilisation se soit déroulé.

Description des cas d'utilisation

■ Exemple de description (3) :

Description des enchaînements

Besoin d'IHM

Expression de contraintes liées à l'interface

Contrainte non fonctionnelle :

Fréquences

Volumétries

QoS: Disponibilité, fiabilité, Performances

Concurrence

Description des cas d'utilisation

- Un cas d'utilisation doit avoir un début et une fin clairement identifié.
- Il faut également préciser les variantes possibles (cas alternatif, exception, erreur,...).
- Chaque unité de description de séquence d'actions est appelé *enchaînement*.
- Un *scénario* représente une succession d'enchaînement qui s'exécute du début a la fin du cas d'utilisation.

Description des cas d'utilisation

■ Exemple :

Sommaire d'identification

Titre : traiter commande

But : traiter les commandes des clients et maj stock.

Résumé : c'est la principale activité de l'entreprise, le personnel autorisé prend les produits commandés et met à jour la commande et le stock.

Acteurs : Syst gestion commande

Date de création : 01/08/12

Date de mise à jour : 10/09/12

Version : 1.3

Responsable : Ahmed Maalel

Description des cas d'utilisation

■ Exemple :

Description des enchaînements

- ***Préconditions*** : l'utilisateur est reconnu et autorisé à accéder a cette ressource. Fournir un num de commande valide.
- ***Enchainements*** :
 - Le système demande a l'utilisateur un um de commande.
 - L'utilisateur fournis un num de commande.
 - Le système recherche la commande.
 - Si la commande n'est pas trouver [Exception1].
 - Sinon le système fournit la commande a l'utilisateur.

Description des cas d'utilisation

■ Exemple :

Description des enchaînements

- Faire
 - Trouver l'article rechercher (CU Trouver produit).
 - Affecter la quantité a délivrer si disponible
 - Sinon [Exception2].
- Jusqu'à avoir terminer tous les articles de la cmd.
- Fin normale de la transaction.
- ***Exceptions :***
 - ***[Exception 1]*** : Commande non trouvée → sortir.
 - ***[Exception 2]*** : Article non disponible → création d'une fiche complémentaire (passer a un 2 cas d'utilisation).

Description des cas d'utilisation

■ Exemple :

Description des enchaînements

- ***Postconditions :***
 - Fin normale : les modification de la commande doivent etre enregistrées.
 - Annulation : si l'utilisateur annule la commande a n'importe quel moment aucune modification ne sera enregistrée

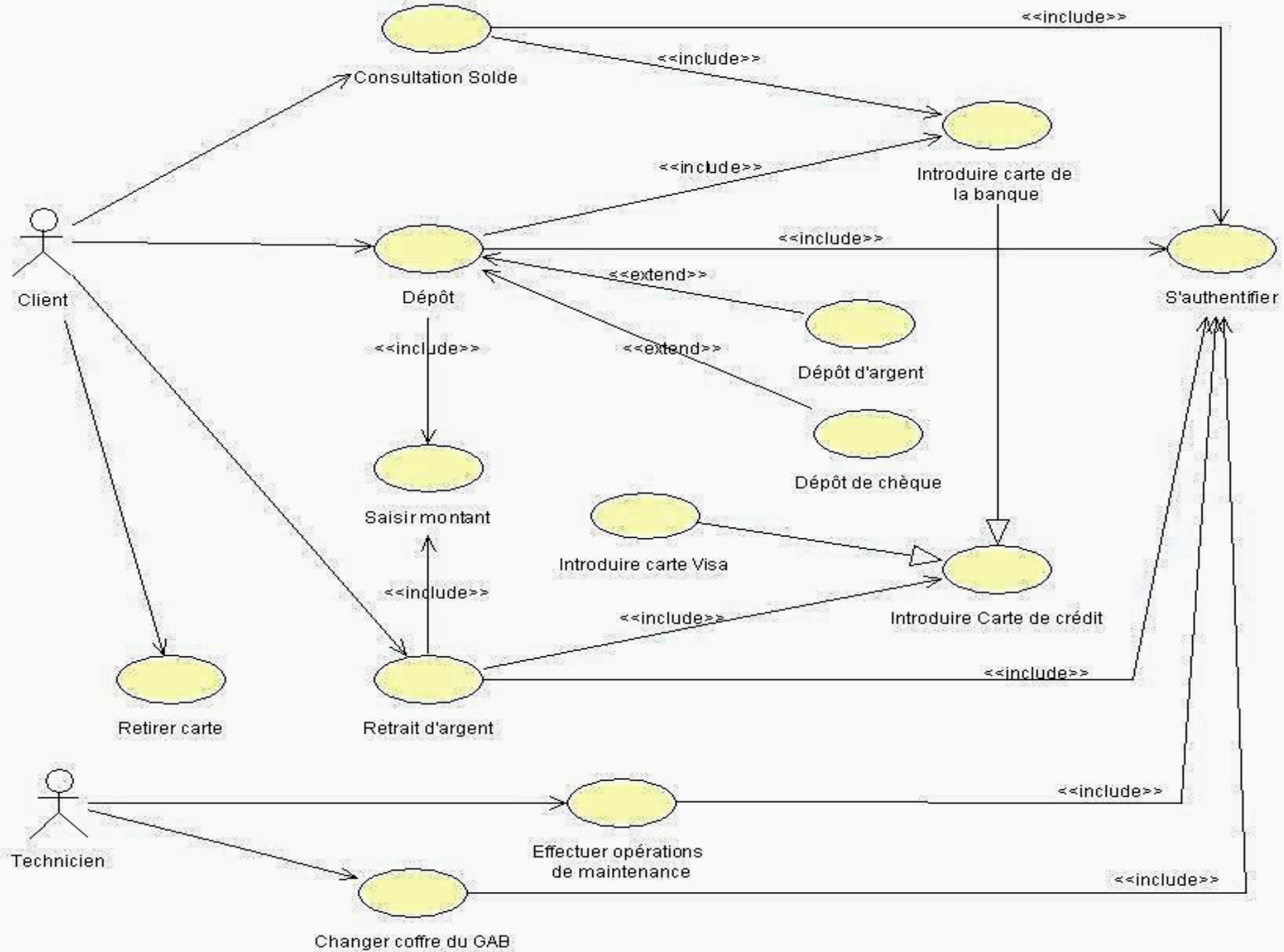
Exercice 2 (TD2)

Étude d'un système de Guichet Automatique de Banque (GAB) simplifié. Le GAB offre les services suivants :

- distribution d'argent à tout porteur de carte de crédit (carte visa, carte de la banque) via un lecteur de carte et un distributeur de billets,
- consultation de solde de compte, dépôt en numéraire et dépôt de chèques pour les clients de la banque porteurs d'une carte de crédit de la banque,
- opérations de rechargement et de maintenance diverses

Ne pas oublier que toute transaction doit être sécurisée.

- Identifier les acteurs du GAB
- Donner une description textuelle des cas d'utilisation



En vue de la mise en place d'un logiciel dédié à l'industrie textile, nous étudions tout d'abord quelques fonctionnalités requises.

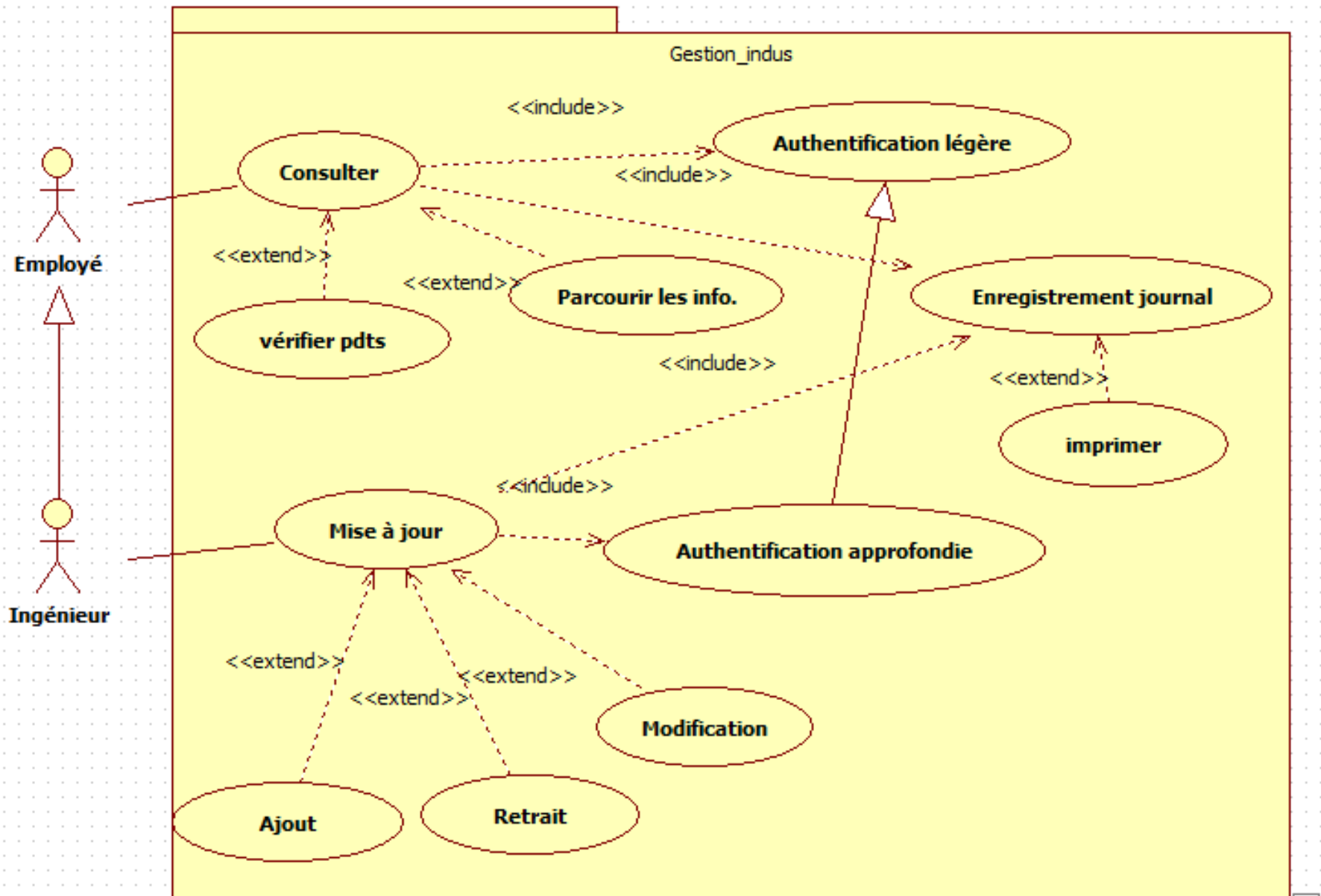
Ce logiciel s'adresse à différentes catégories d'utilisateurs, et doit servir principalement à recueillir l'information sur les produits développés dans l'entreprise.

Tous les personnels de l'entreprise peuvent consulter le système, soit pour vérifier qu'un produit particulier existe, soit pour un parcours libre des informations. Toute consultation doit être précédée d'une authentification légère dans laquelle la personne précise son nom et son service à des fins de statistiques ultérieures.

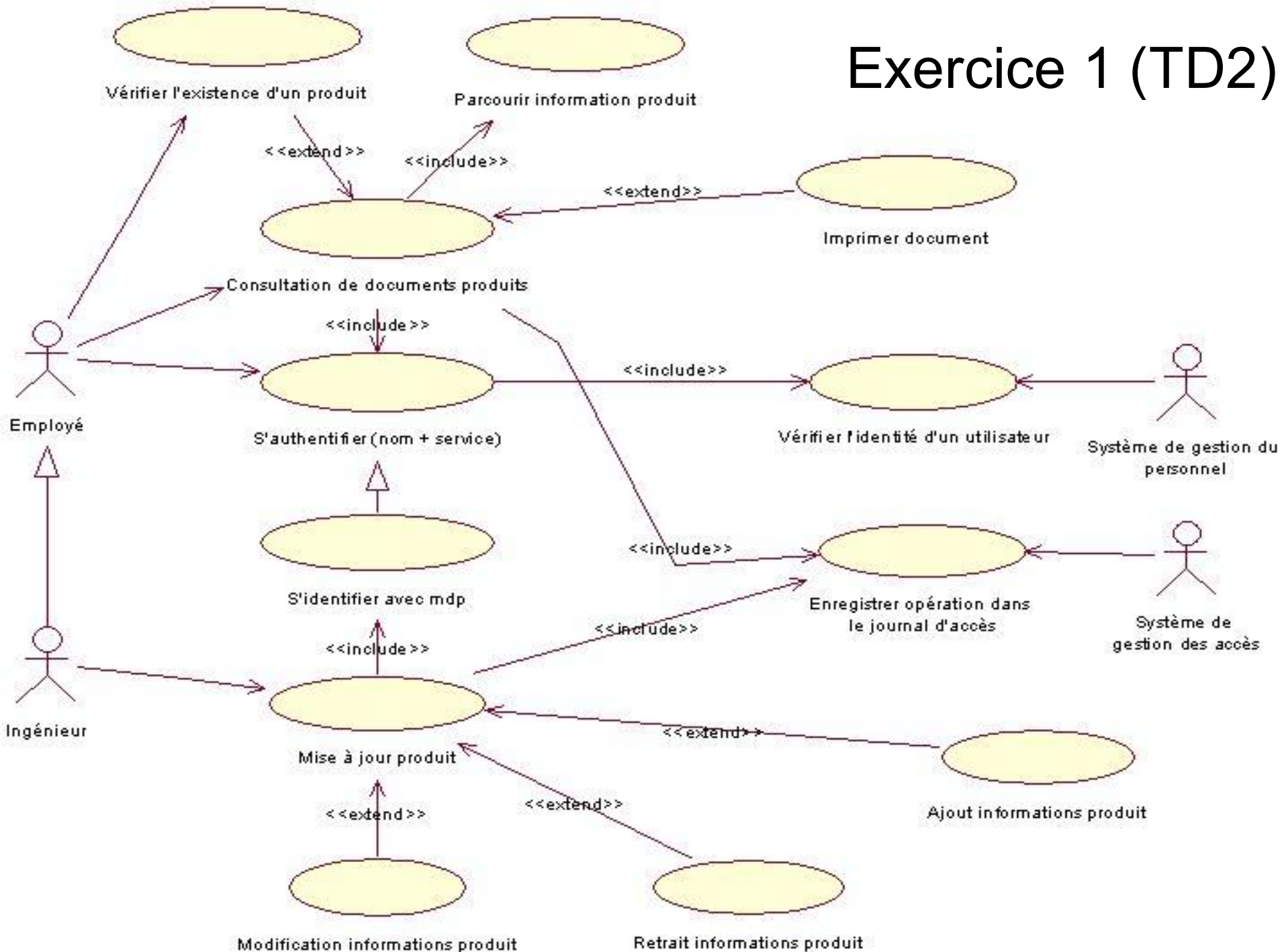
Les ingénieurs peuvent effectuer différentes opérations de mise à jour pour les produits dont ils sont responsables : ajout, retrait, modification des informations sur les produits. Ces opérations doivent être précédées d'une authentification plus approfondie lors de laquelle l'ingénieur précise son nom, son service et donne un mot de passe qui est vérifié en contactant le système de gestion des personnels.

Toutes les opérations (consultation et mise à jour) donnent lieu à un enregistrement dans le journal des accès et peuvent optionnellement s'accompagner d'une impression des documents accédés.

- Identifier les acteurs du système
- Identifier et réaliser les cas d'utilisation



Exercice 1 (TD2)



Étude d'un système simplifié de caisse enregistreuse de supermarché :

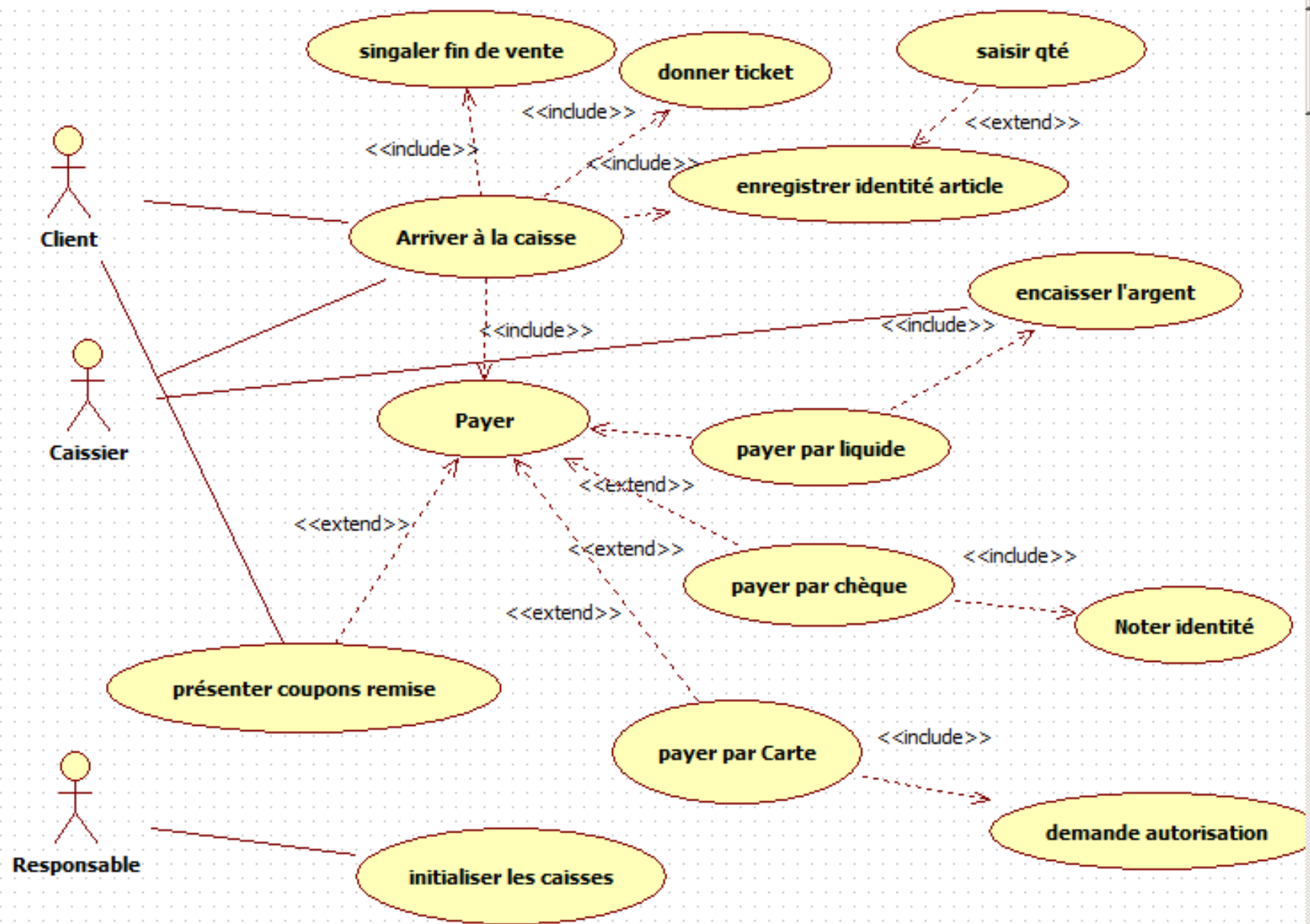
Le déroulement normal d'utilisation de la caisse est le suivant :

- un client arrive à la caisse avec des articles
- le caissier enregistre le numéro d'identification de chaque article, ainsi que la quantité si celle-ci est supérieure à 1
- la caisse affiche le prix de chaque article et son libellé
- lorsque tous les articles ont été enregistrés, le caissier signale la fin de la vente
- la caisse affiche le total des achats
- le client choisit son mode de paiement :
 - Liquide : le caissier encaisse l'argent et la caisse indique le montant éventuel à rendre au client,
 - Chèque : le caissier note l'identité du client et la caisse enregistre le montant sur le chèque
 - Carte de crédit : un terminal bancaire fait partie de la caisse, il transmet la demande à un centre d'autorisation multi banques
- la caisse enregistre la vente et imprime un ticket
- le caissier transmet le ticket imprimé au client

Un client peut présenter des coupons de réduction avant paiement.

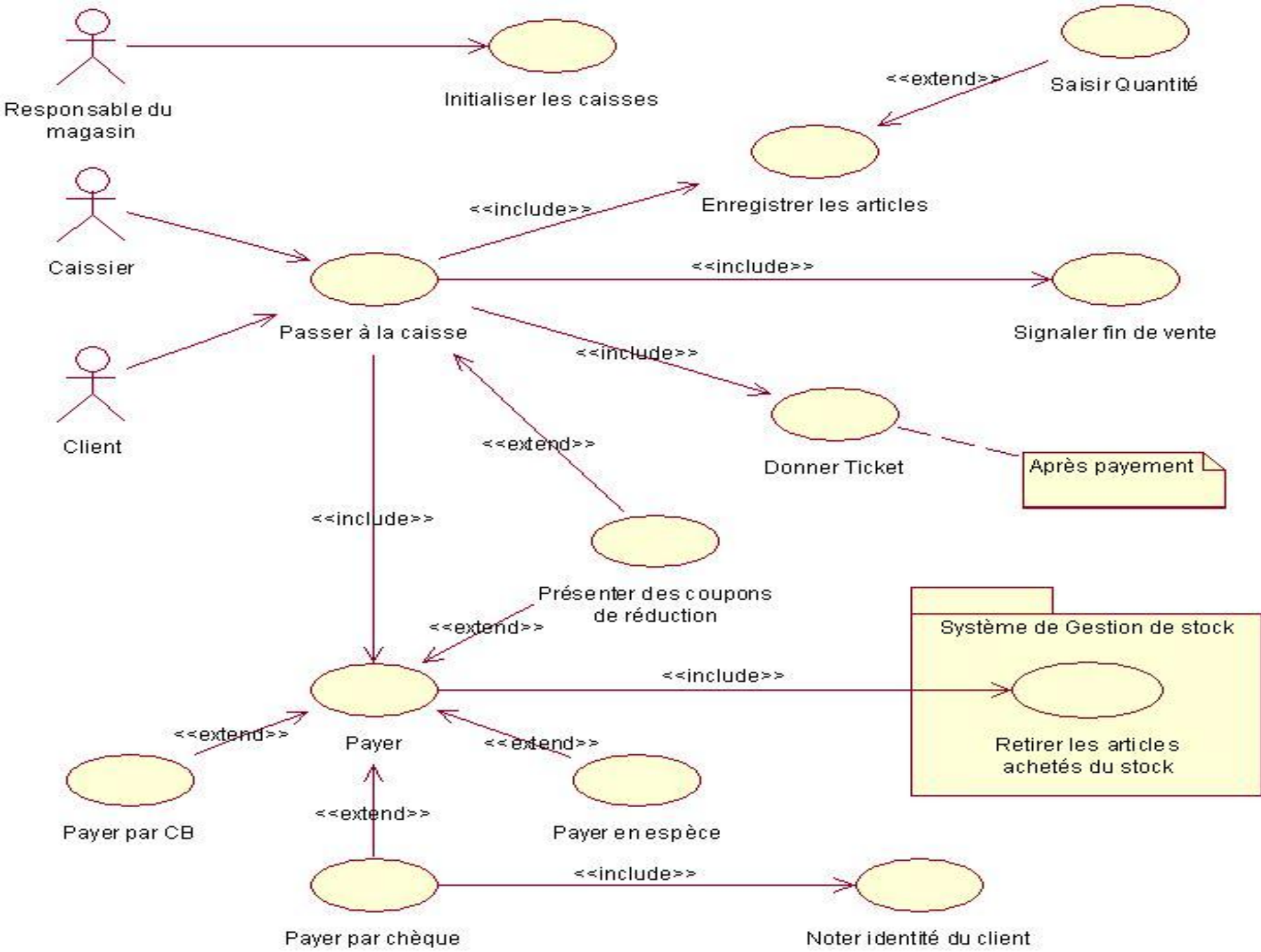
Lorsque le paiement est terminé, la caisse transmet les informations relatives aux articles vendus au système de gestion des stocks.

Tous les matins, le responsable du magasin initialise les caisses pour la journée.



Travail à faire

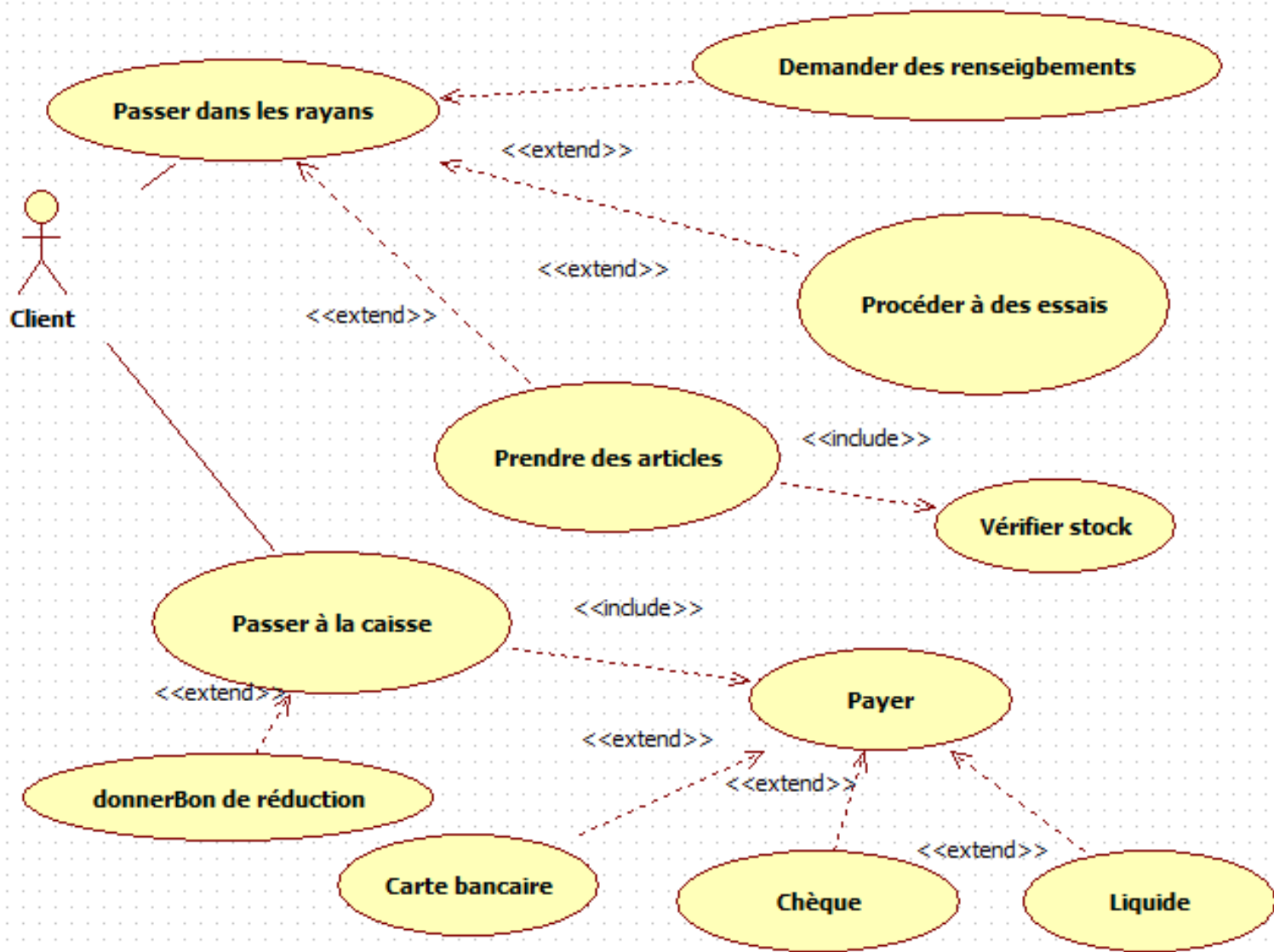
1. Identifier les acteurs du système
2. Identifier et réaliser les cas d'utilisation
3. Lister les différentes sections présentes dans un cas d'utilisation.
4. Rédiger le cas d'utilisation «Passer à la caisse»
5. Rédiger de nouveau le cas d'utilisation « Passer à la caisse » en considérant que le caissier fait partie du système.



Exercice d'application

Dans un magasin, le processus de vente est le suivant : le client entre, passe dans les rayons, demande éventuellement des renseignements ou procède à des essais, prend des articles (si le stock est suffisant), passe à la caisse où il règle ses achats (avec tout moyen de paiement accepté). Il peut éventuellement bénéficier d'une réduction.

Modéliser cette situation par un diagramme de cas d'utilisation



Description des cas d'utilisation

■ Exemple :

Sommaire d'identification

Titre : Achat produits

But : Prospection et/ou Achat d'un produit

Résumé : Un client peut faire la prospection et notamment l'achat d'un produit

Acteurs : Client & Vendeur & caissier

Date de création : 05/03/13

Date de mise à jour : 05/03/13

Version : 1.0

Responsable :

Description des cas d'utilisation

■ Exemple :

Description des enchaînements

- ***Pré conditions : ...***
- ***Enchainements :***
 - Le client entre et peut passer dans les rayons
 - Le client peut se renseigner et/ou essayer un produit puis il procède à l'achat ou il quitte.
 - Le client peut faire l'achat d'un produit et précise la qté demandée
 - Si la qté demandée n'est pas présente dans le stock alors [Exception1].
- le client choisi son mode de paiement (chèque, CB, liquide) valide son achat auprès de caissier

Description des cas d'utilisation

■ Exemple :

Description des enchaînements

- ***Exceptions :***
 - ***[Exception 1]*** : Regret → sortir.

Exercice d'application 2

Dans un magasin, un commerçant dispose d'un système de gestion de son stock d'articles, dont les fonctionnalités sont les suivantes :

- **Edition de la fiche d'un fournisseur**
- **Possibilité d'ajouter un nouvel article (dans ce cas, la fiche fournisseur est automatiquement éditée. Si le fournisseur n'existe pas, on peut alors le créer)**
- **Edition de l'inventaire. Depuis cet écran, on a le choix d'imprimer l'inventaire, d'effacer un article ou d'éditer la fiche d'un article).**

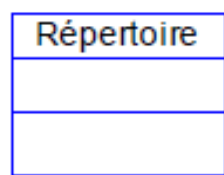
Modéliser cette situation par un diagramme de cas d'utilisation

Exercice d'application 3

Un répertoire contient des fichiers

- **Une pièce contient des murs**
- **Les modems et claviers sont des périphériques d'entrée / sortie**
- **Une transaction boursière est un achat ou une vente**
- **Un compte bancaire peut appartenir à une personne physique ou morale**

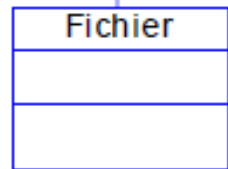
Elaborez les diagrammes de classe correspondants en choisissant le type de relation approprié



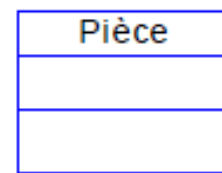
1..1

Contenir

0..*



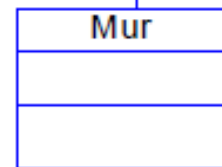
Fichier



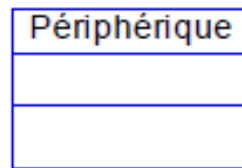
1..*

composer

1..*

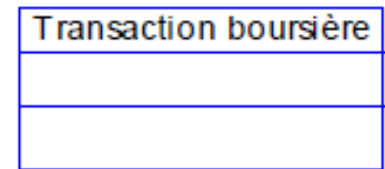
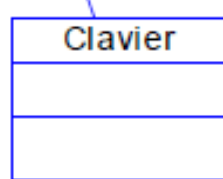
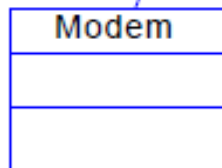


Mur



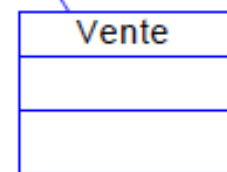
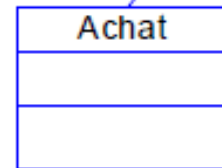
Modem

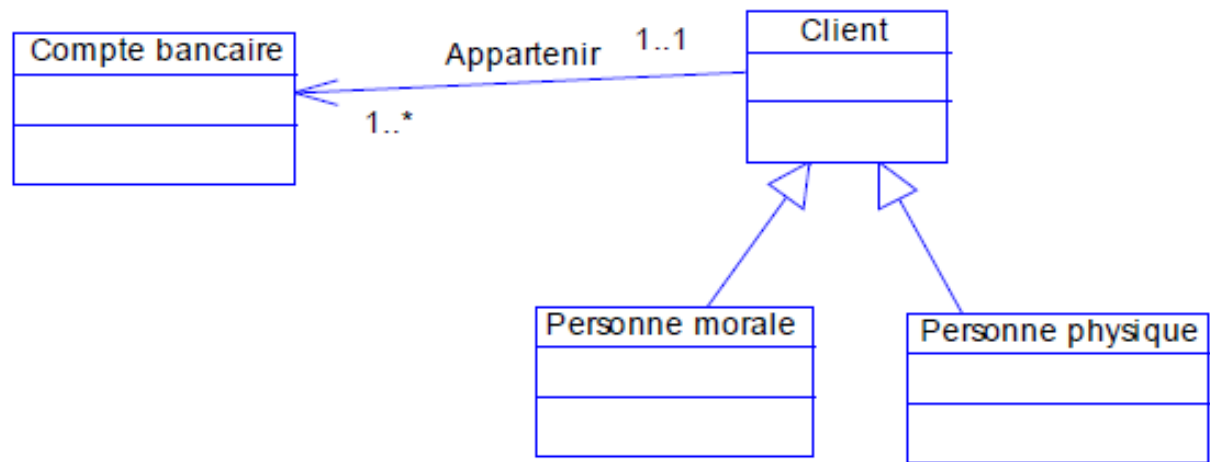
Clavier



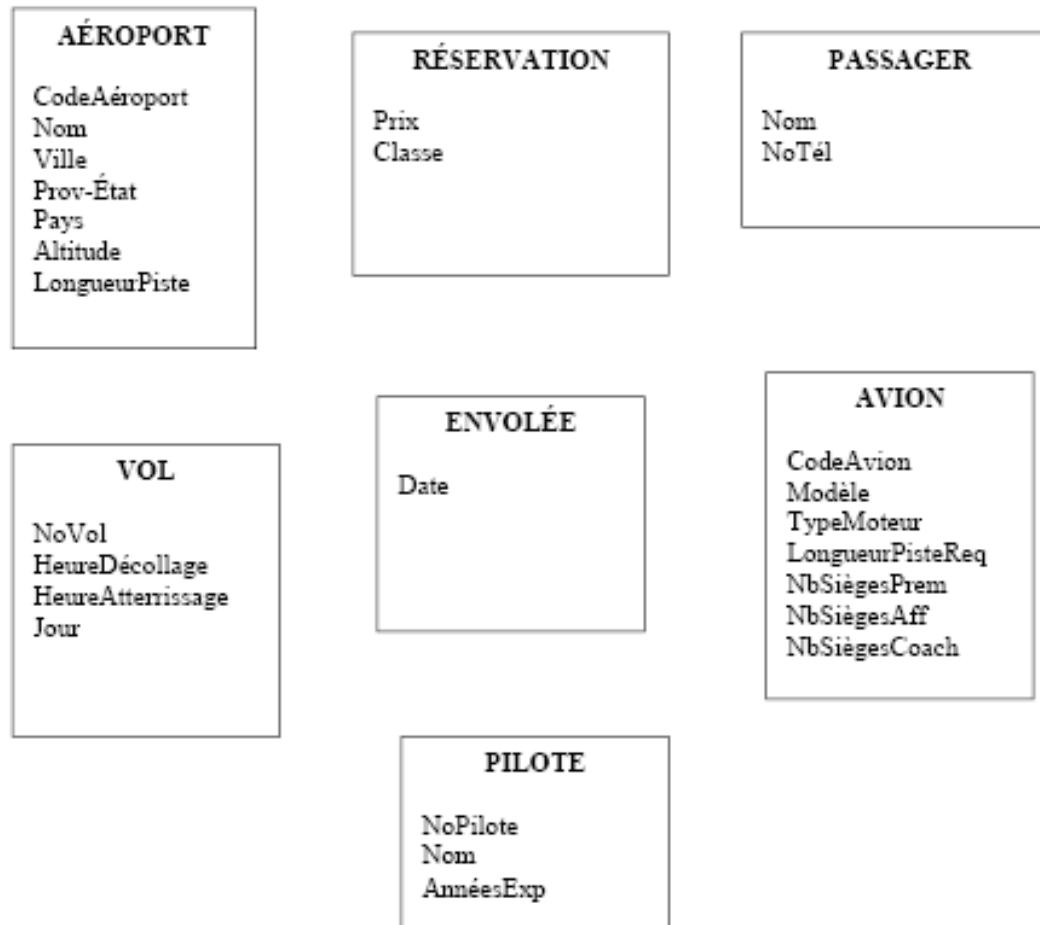
Achat

Vente





Exercice d'application 4



On souhaite gérer les réservations de vols effectuées dans une agence. D'après les interviews réalisées avec les membres de l'agence, on sait que :

Les compagnies aériennes proposent différents vols

Un vol est ouvert à la réservation et refermé sur ordre de la compagnie

Un client peut réserver un ou plusieurs vols, pour des passagers différents

Une réservation concerne un seul vol et un seul passager

Une réservation peut être confirmée ou annulée

Un vol a un aéroport de départ et un aéroport d'arrivée

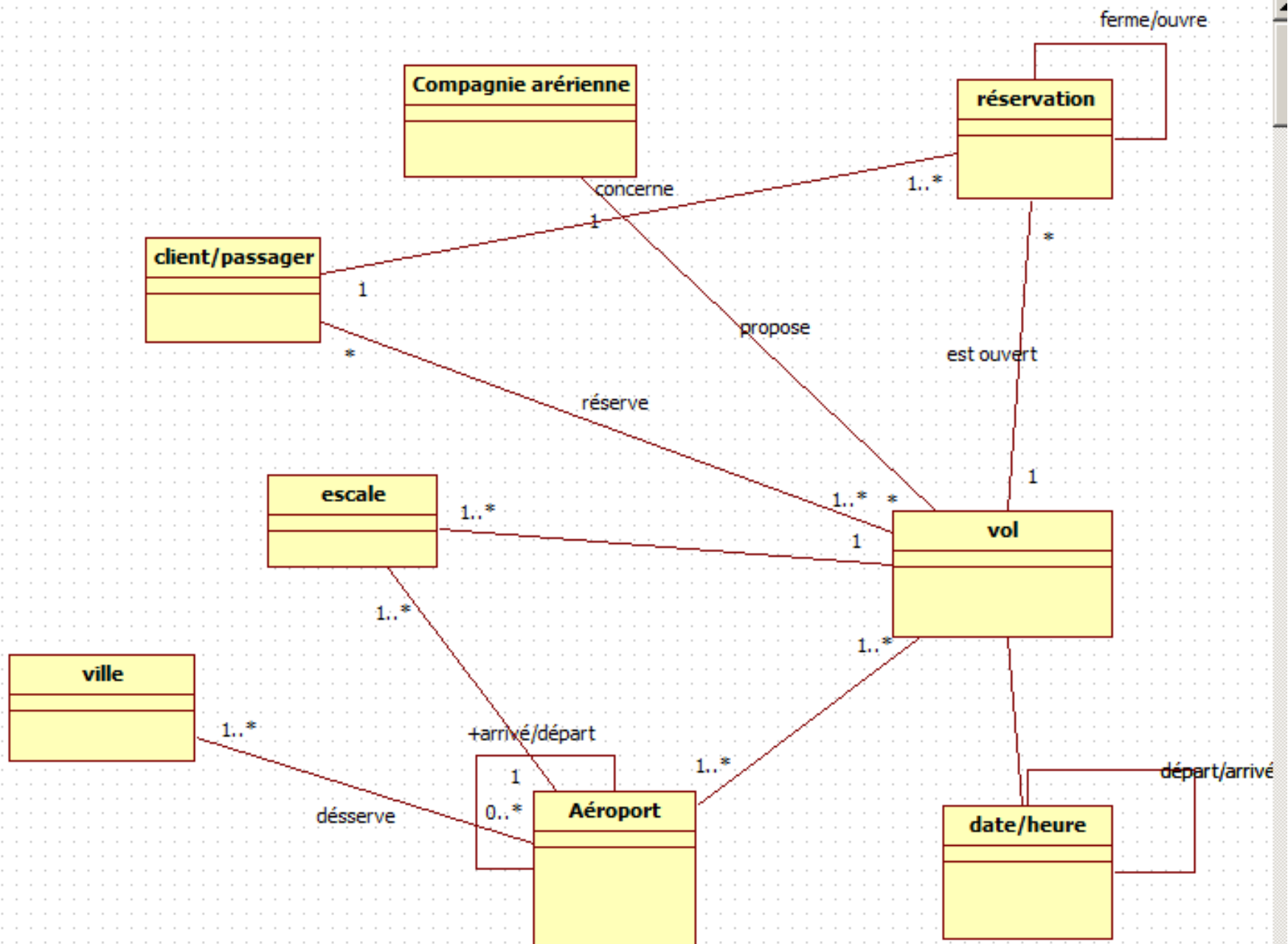
Un vol a un jour et une heure de départ, et un jour et une heure d'arrivée

Un vol peut comporter des escales dans un ou plusieurs aéroport(s)

Une escale à une heure de départ et une heure d'arrivée

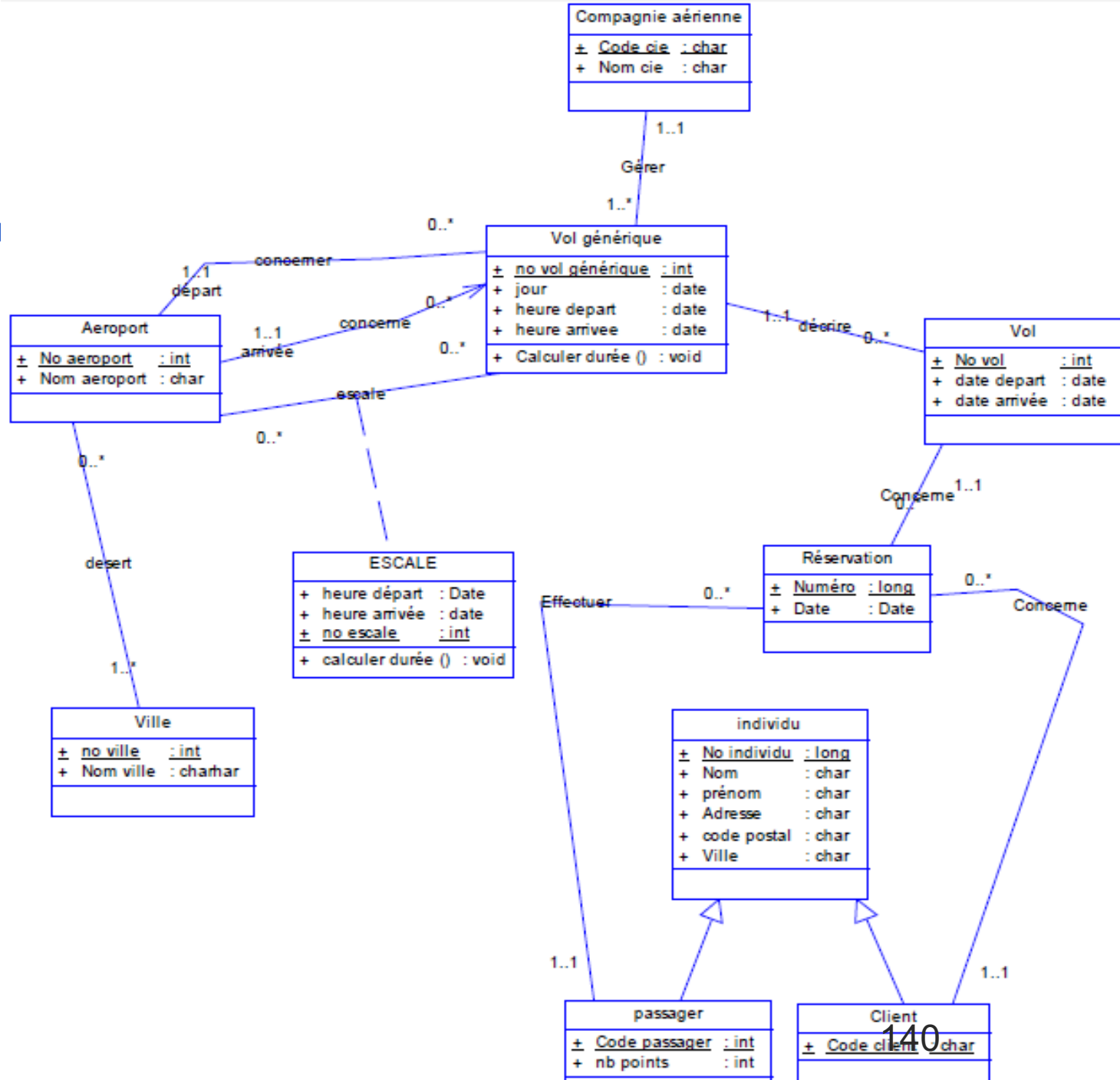
Chaque aéroport dessert une ou plusieurs villes

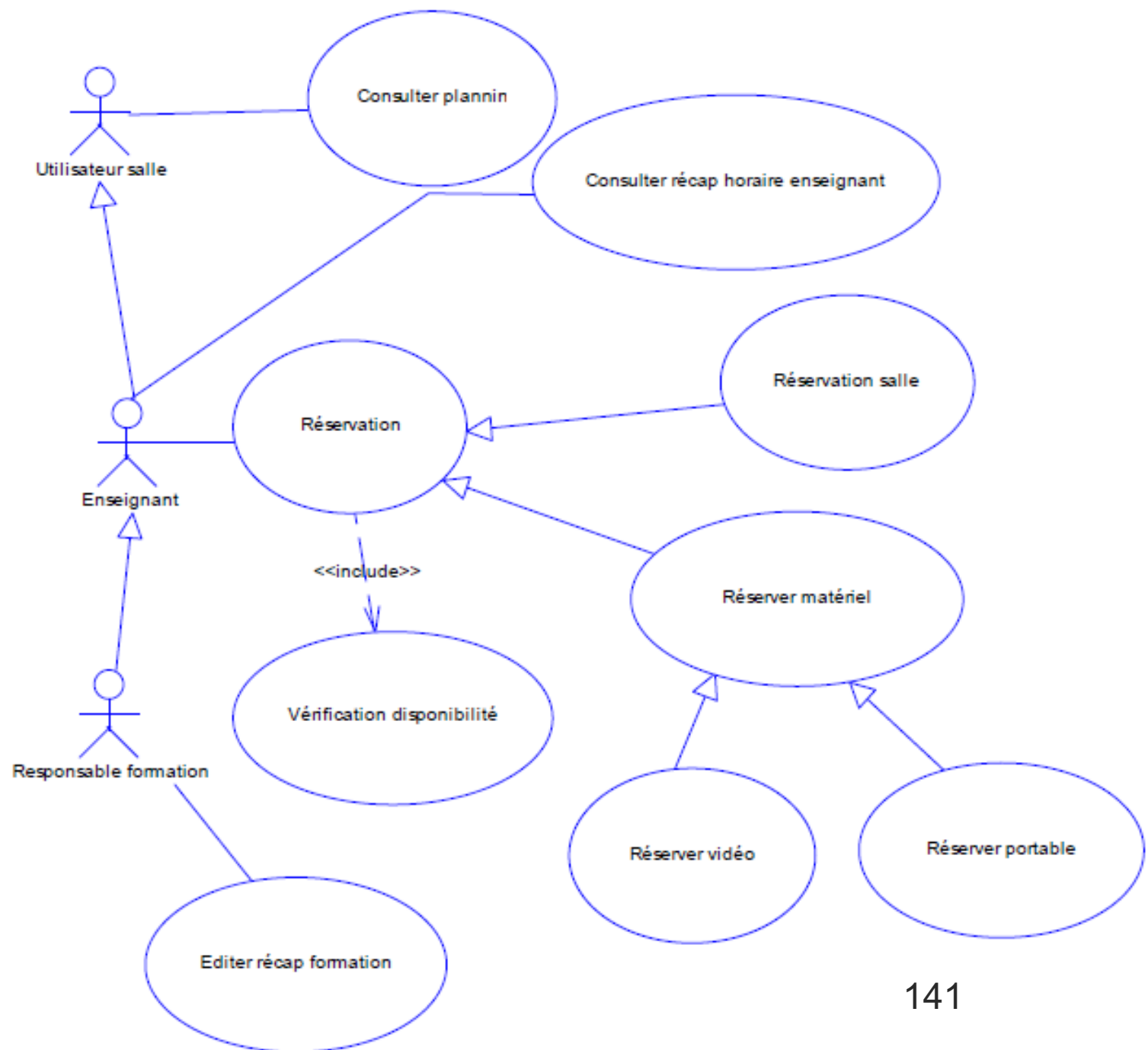
→ A partir des éléments qui vous sont fournis ci-dessus, élaborer le diagramme de classes (en y ajoutant tout attribut que vous jugez pertinent et qui n'a pas été décrit ci-dessus).



Dans un établissement scolaire, on désire gérer la réservation des salles de cours ainsi que du matériel pédagogique (ordinateur portable ou/et Vidéo projecteur). Seuls **les enseignants** sont habilités à **effectuer des réservations (sous réserve de disponibilité de la salle ou du matériel)**. Le planning des salles peut quant à lui être **consulté par tout le monde** (enseignants et étudiants). Par contre, le récapitulatif horaire par enseignant (calculé à partir du planning des salles) ne peut être consulté que par les enseignants. Enfin, il existe pour **chaque formation** un enseignant responsable qui seul peut éditer **le récapitulatif horaire** pour l'ensemble de la formation.

→ Modéliser cette situation par un diagramme de cas





Diagrammes De classe

III- La Modélisation Statique

■ *III-2-c- Diagrammes de Classe*

- Une classe est un type abstrait caractérisé par des propriétés (attributs et méthodes) et permettant de créer des objets ayant ces propriétés.

Classe = attributs + méthodes + instantiation

- Un diagramme de classes est une collection d'éléments de modélisation statiques (classes, paquetages...), qui montre la structure d'un modèle
- Il fait abstraction des aspects dynamiques et temporels
- Pour un modèle complexe, plusieurs diagrammes de classes complémentaires doivent être construits.

III- La Modélisation Statique

- On peut par exemple se focaliser sur :
 - les classes qui participent à un cas d'utilisation
 - les classes associées dans la réalisation d'un scénario précis,
 - les classes qui composent un paquetage,
 - la structure hiérarchique d'un ensemble de classes.
- Pour représenter un contexte précis, un diagramme de classes peut être instancié en plusieurs diagrammes d'objets.
- UML permet de définir trois types de stéréotypes pour les classes :
 - a) **les classes « frontière »**(interface): classes qui servent à modéliser les interactions entre le système et ses acteurs.
 - b) **les classes « contrôle »** : classes qui servent à représenter la coordination, le séquençement, les transactions et le contrôle d'autres objets.
 - c) **les classes « entité »** : classes qui servent à modéliser les informations durables et persistantes.

III- La Modélisation Statique

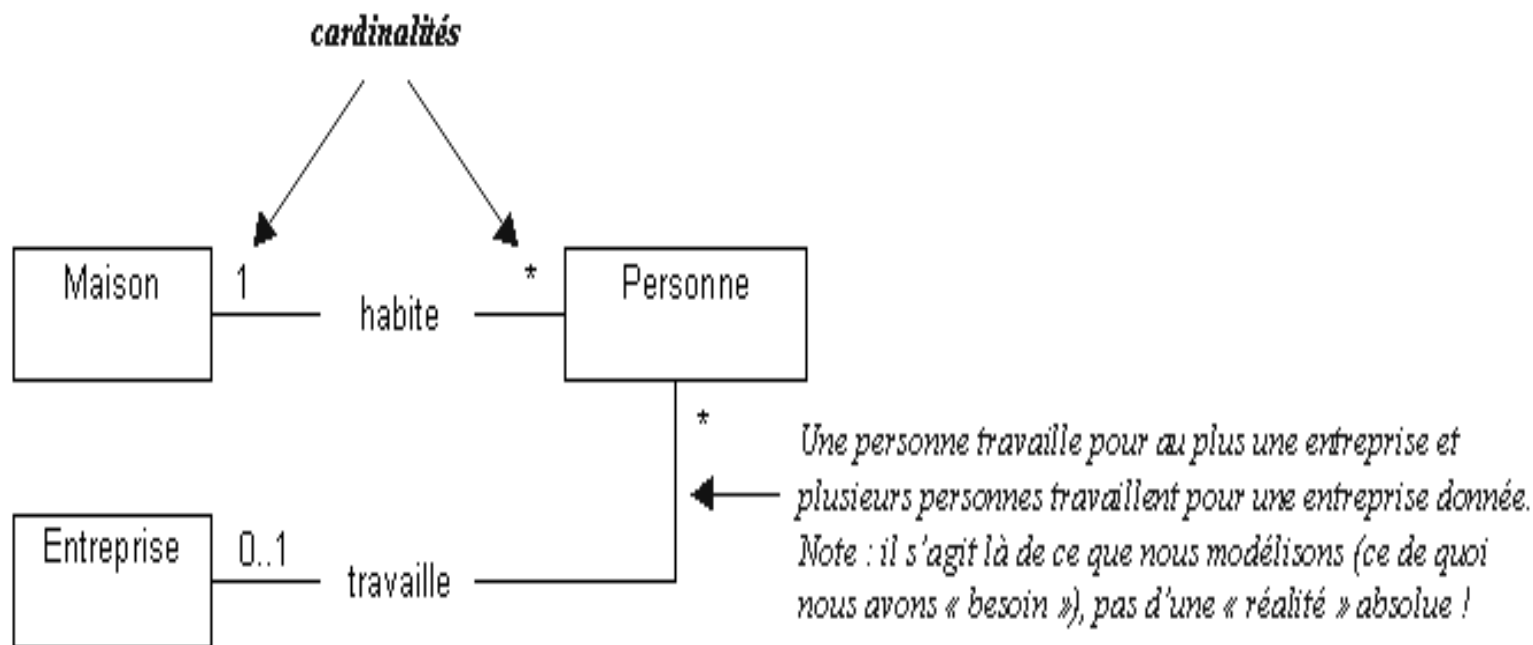
Buts du diagramme de classe:

1. Structurer l'application - Relations entre classes
2. Base pour générer le code
3. Base pour générer la structure de données

III- La Modélisation Statique

■ Relation entre Classes :

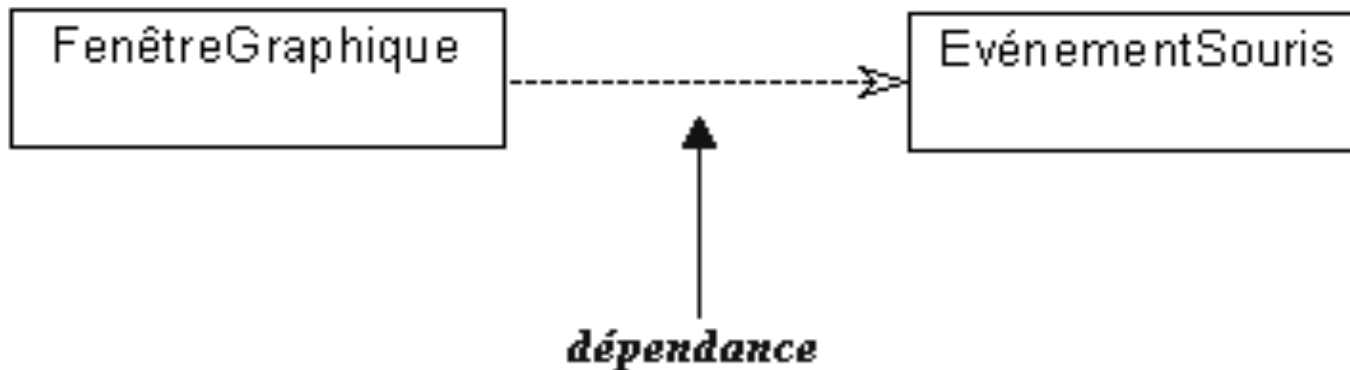
Cardinalités : précise le nombre d'instances qui participent à une relation.



III- La Modélisation Statique

■ Relation entre 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).

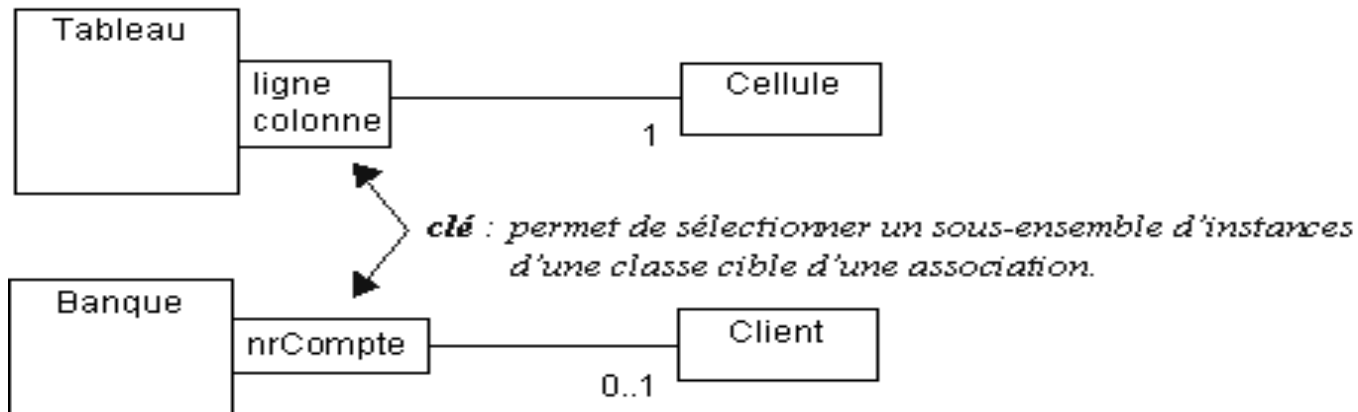


III- La Modélisation Statique

■ Relation entre Classes :

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.



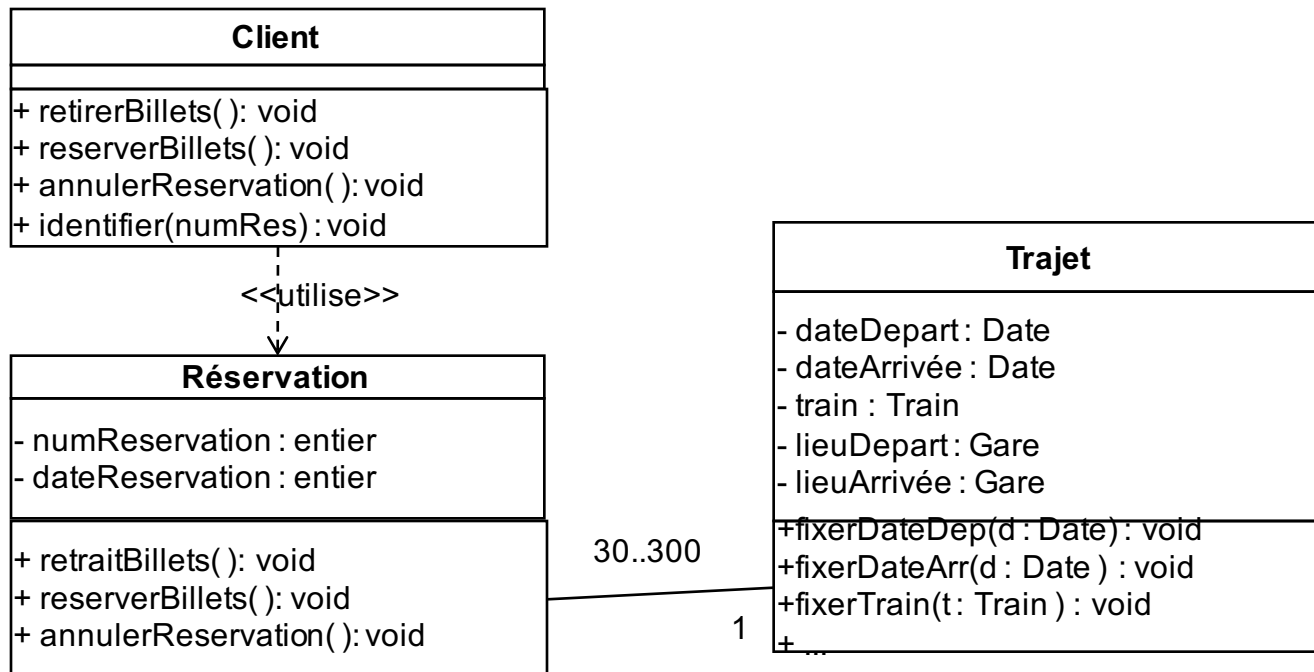
III- La Modélisation Statique

Les concepts supportés par les diagrammes de classe

- Les classes :
 - classes réelles
 - classes paramétrables/génériques
 - classes abstraites
 - classes interfaces
- Les relations :
 - généralisation/spécialisation
 - agrégation/composition
 - associations binaire

III- La Modélisation Statique

■ Exemple récapitulatif



Exercice d'application

- Construire le diagramme de classe de votre projet

Diagrammes d'objets

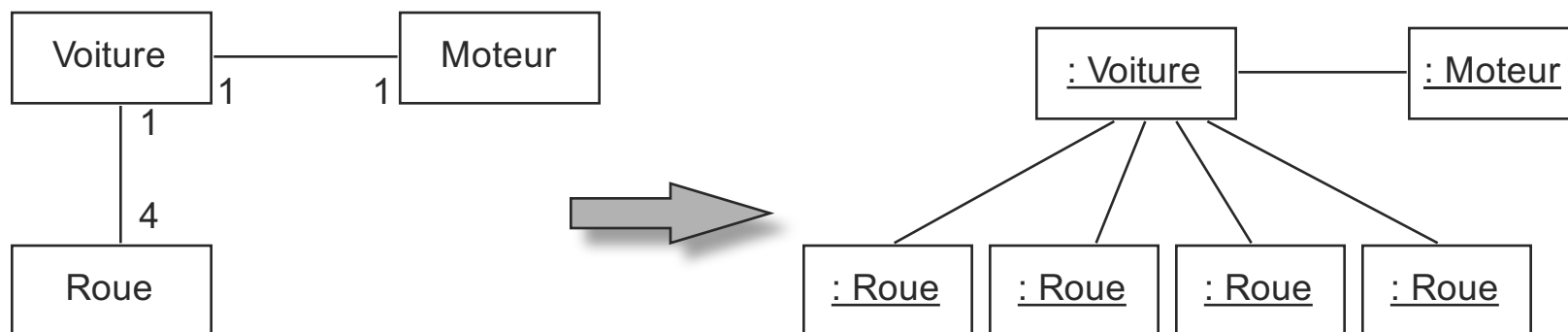
III- La Modélisation Statique

■ *III-2-b- Diagrammes d'objets*

- Illustration des objets (instances de classes dans un état particulier) et des liens entre ces objets
- Les diagrammes d'objets s'utilisent pour montrer un contexte (avant ou après une interaction entre objets par exemple)
- Sert essentiellement en phase **exploratoire**, car il possède un très haut niveau d'abstraction
- Il a pour but de décrire schématiquement les relations entre classes et de servir comme support pour les diagrammes de collaboration
- Faciliter la compréhension des structures complexes

III- La Modélisation Statique

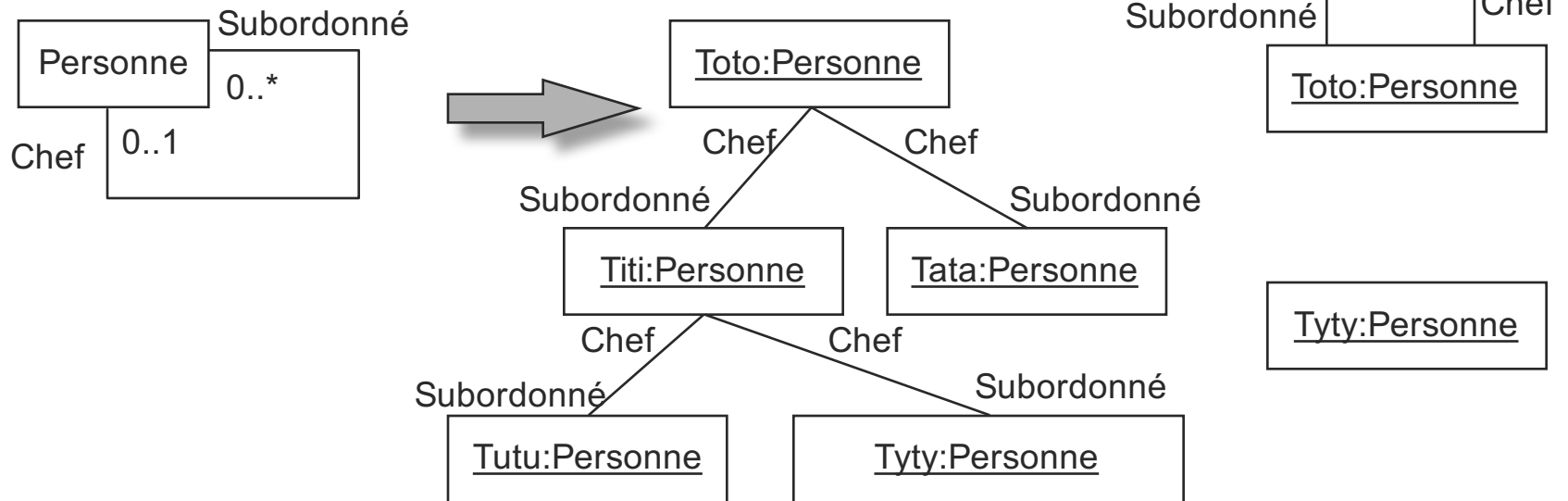
Exemple de diagramme d'Objet :



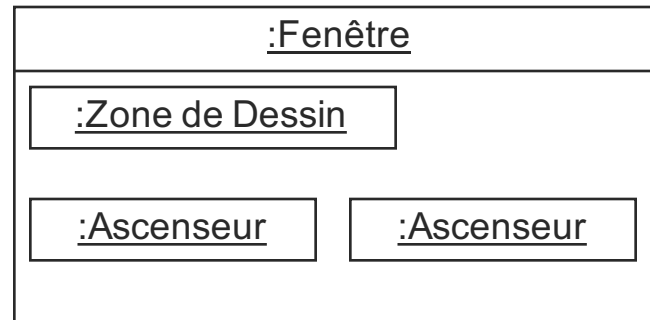
III- La Modélisation Statique

Exemple de diagramme d'Objet :

Relations réflexives



Objets composites



Diagrammes de composants

III- La Modélisation Statique

■ *III-2-d- Diagrammes de Composants*

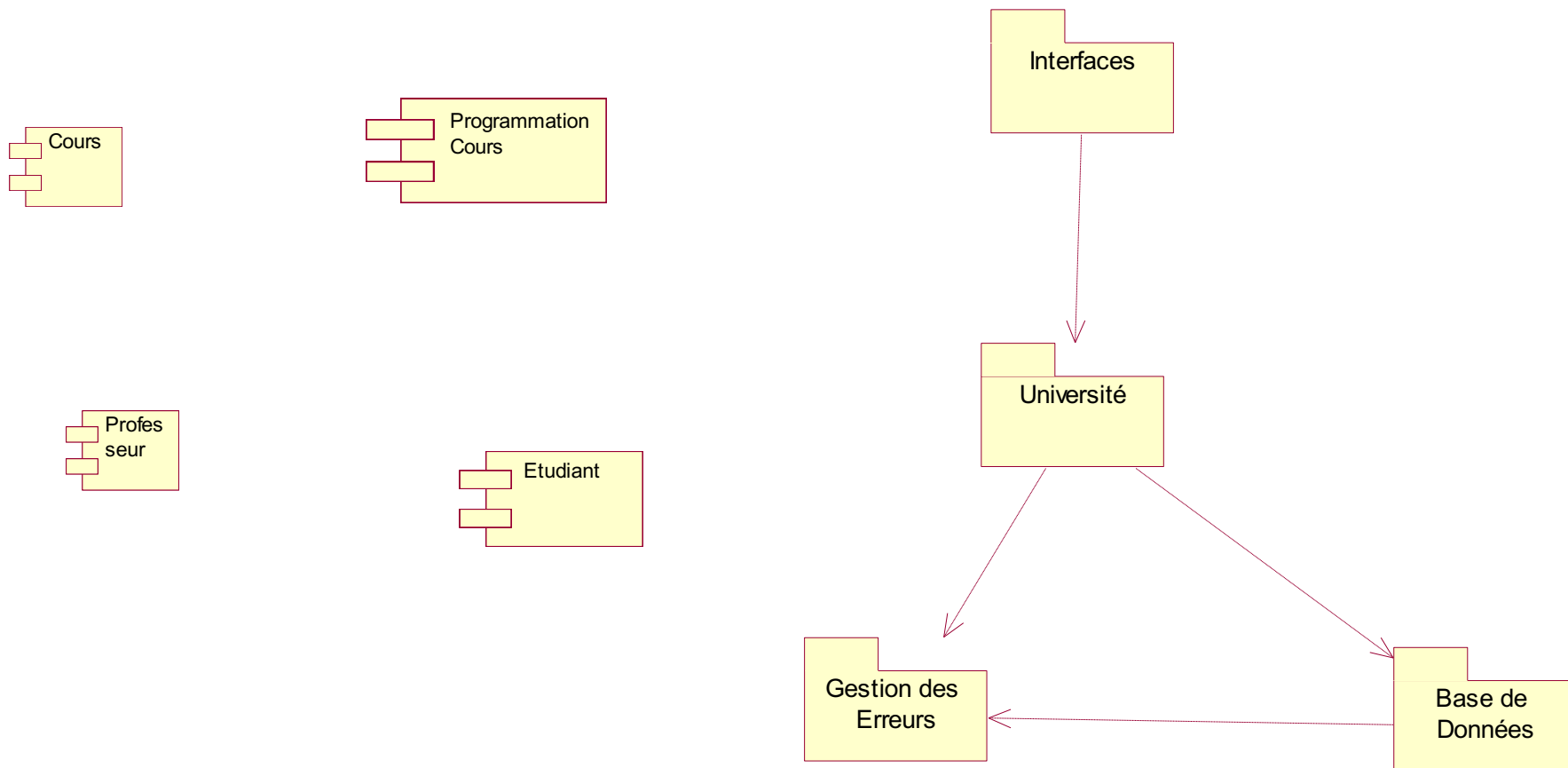
- Ils permettent de décrire l'architecture physique et statique d'une application en termes de modules : fichiers sources, librairies, exécutables, etc.
Ils montrent la mise en œuvre physique des modèles de la vue logique avec l'environnement de développement
- Les dépendances entre composants permettent notamment d'identifier les contraintes de compilation et de mettre en évidence la réutilisation de composants
- Les composants peuvent être organisés en paquetages, qui définissent des sous-systèmes. Les sous-systèmes organisent la vue des composants (de réalisation) d'un système. Ils permettent de gérer la complexité, par encapsulation des détails d'implémentation.

III- La Modélisation Statique

- Les composants représentent les éléments de conception réutilisables :
 - ils contiennent des classes
 - ils présentent des interfaces
 - et leurs interdépendances
- Propriétés importantes :
 - au sein d'un composant couplage fort
 - entre composants, couplage lâche
- Paquetage = ensemble de composants

III- La Modélisation Statique

Exemples de diagramme de Composants :



Différences entre composants et classes:

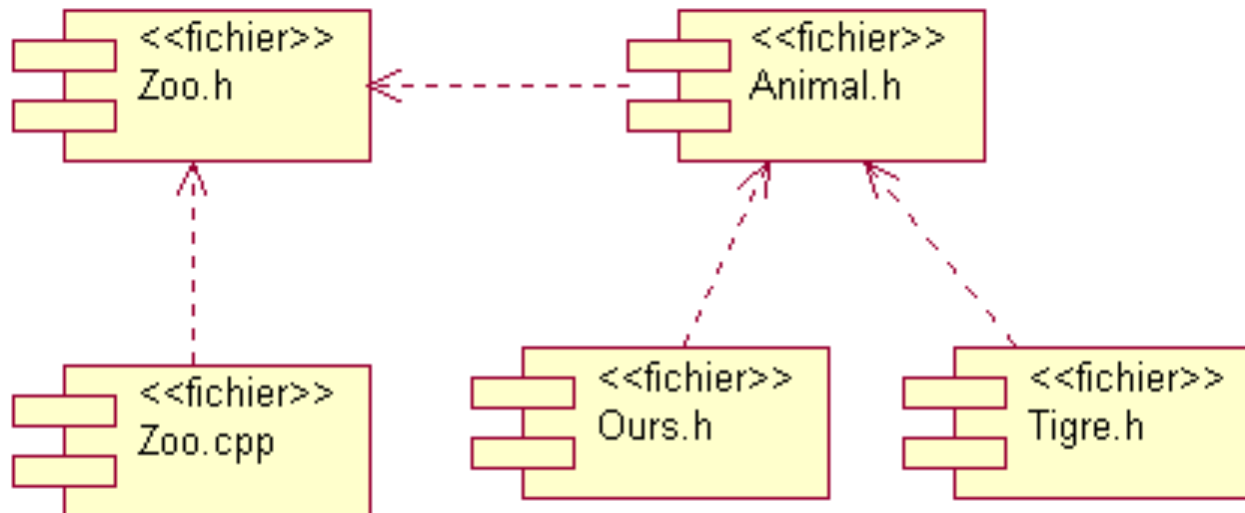
- Les classes représentent des **abstractions logiques** alors que les composants représentent des **abstractions physiques** qui existent dans le monde des bits.
- Les composants représentent le **regroupement physique** de ce qu'on pourrait appeler des composants logiques et se situent à un niveau d'abstraction différent.
- Les classes peuvent avoir directement des attributs et des opérations. En général, **les composants comportent seulement des opérations que l'on peut atteindre uniquement par leur interface.**

III- La Modélisation Statique

UML définit cinq stéréotypes standards qui s'appliquent aux composants:

- « document »
- « exécutable »
- « fichier »
- « bibliothèque »
- « table »

Les relations de dépendance sont utilisées dans les diagrammes de composants pour indiquer **qu'un élément d'implémentation d'un composant fait appel aux services offerts** par les éléments d'implémentation d'un autre composant.



III- La Modélisation Statique

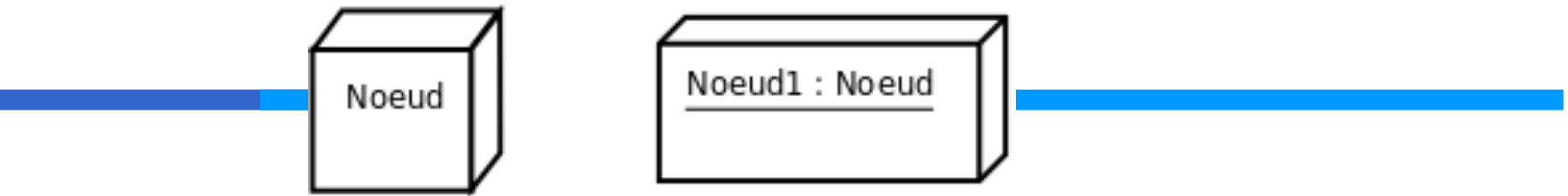
Les diagrammes de composants sont des vues statiques de l'implémentation des systèmes qui montrent les choix de réalisation. En général, ils ne sont utilisés que pour des systèmes complexes.

Diagrammes de déploiement

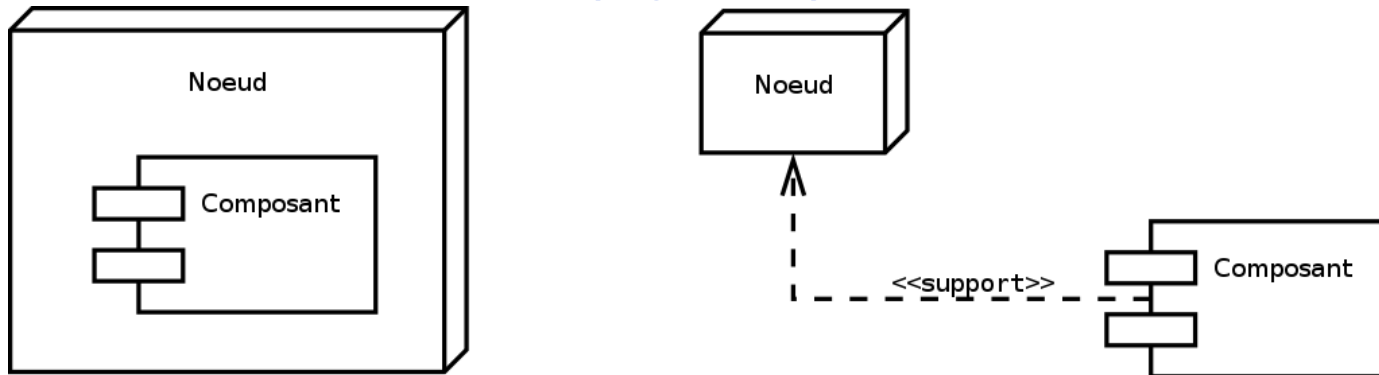
Diagrammes de Déploiement :

- Ils montrent la disposition physique des matériels qui composent le système et la répartition des composants sur ces matériels.
- Les ressources matérielles sont représentées sous forme de nœuds:
 - Les nœuds sont connectés entre eux, à l'aide d'un support de communication.
La nature des lignes de communication et leurs caractéristiques peuvent être précisées
- Les diagrammes de déploiement peuvent montrer des instances de nœuds (un matériel précis), ou des classes de nœuds
- Les diagrammes de déploiement correspondent à la vue de déploiement d'une architecture logicielle .

Diagrammes de Déploiement



Représentation d'un nœud (à gauche) et d'une instance de nœud (à droite).

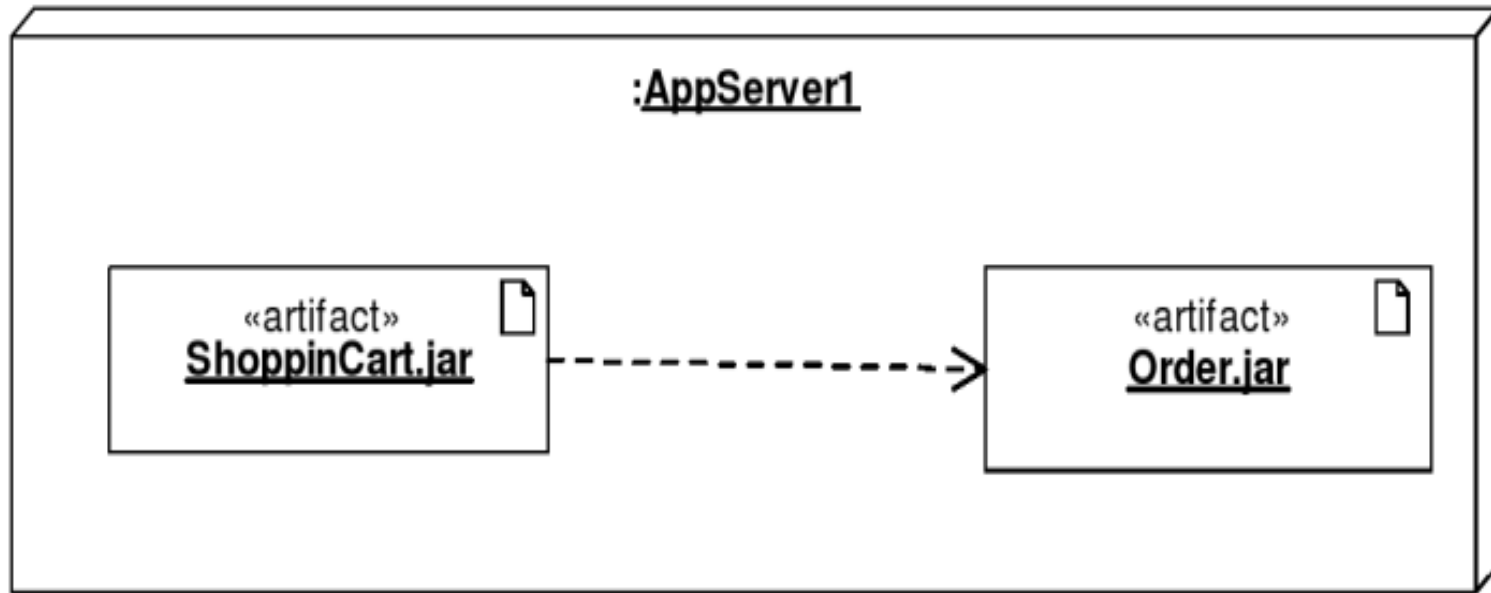


Deux possibilités pour représenter l'affectation d'un composant à un nœud.

- Chaque ressource est matérialisée par un nœud représenté par un cube comportant un nom
- Un nœud est un classeur et peut posséder des attributs (quantité de mémoire, vitesse du processeur, ...).

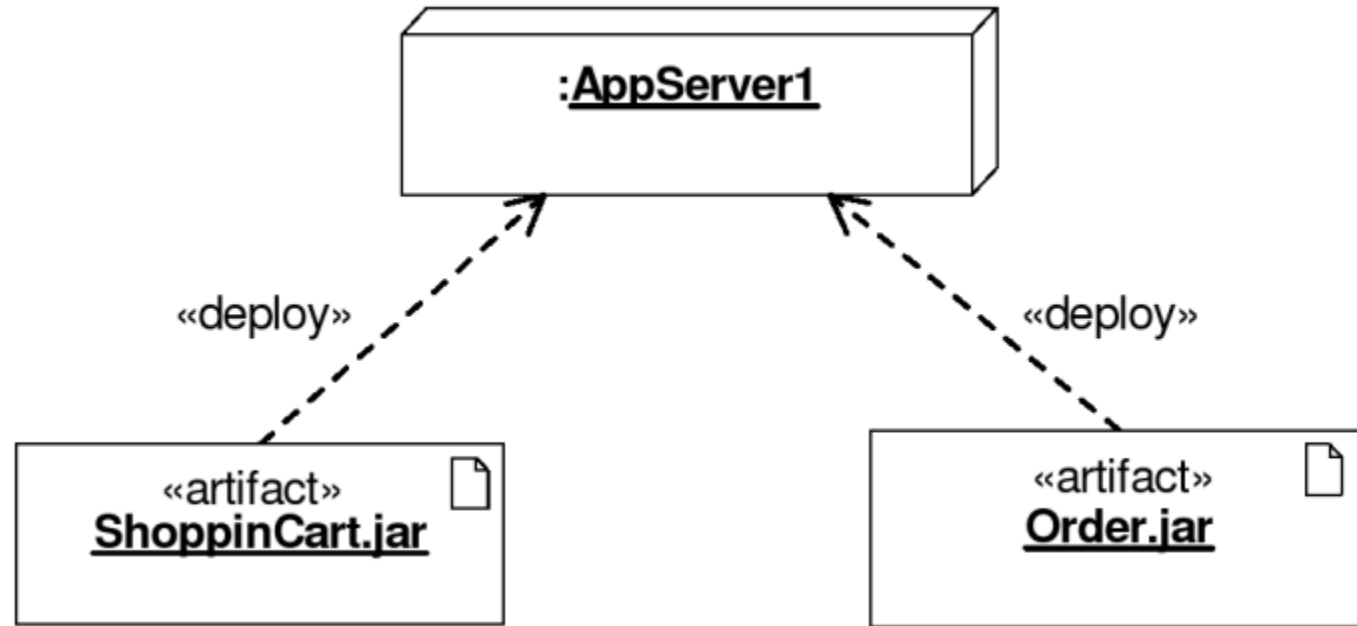
Diagrammes de Déploiement

Notion d'artefact (*artifact*)



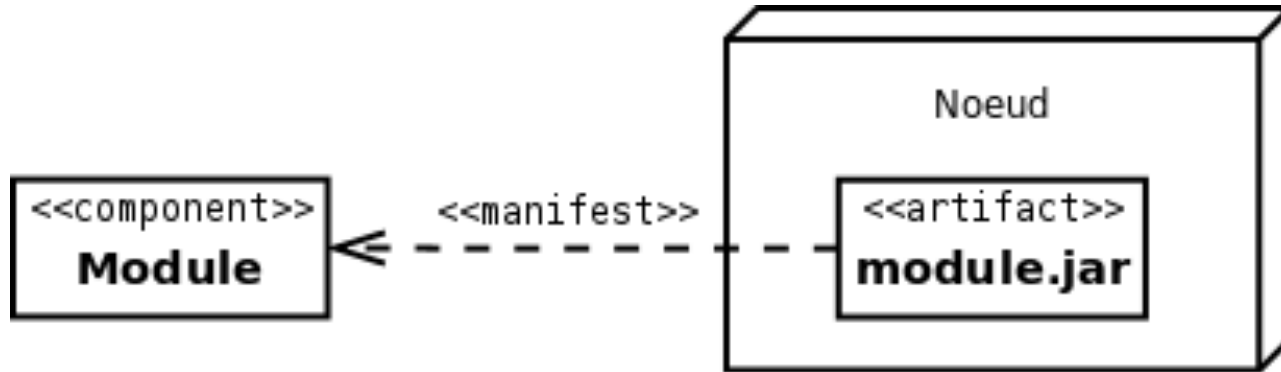
Représentation du déploiement de deux artefacts dans un nœud. La dépendance entre les deux artefacts est également représentée.

Diagrammes de Déploiement



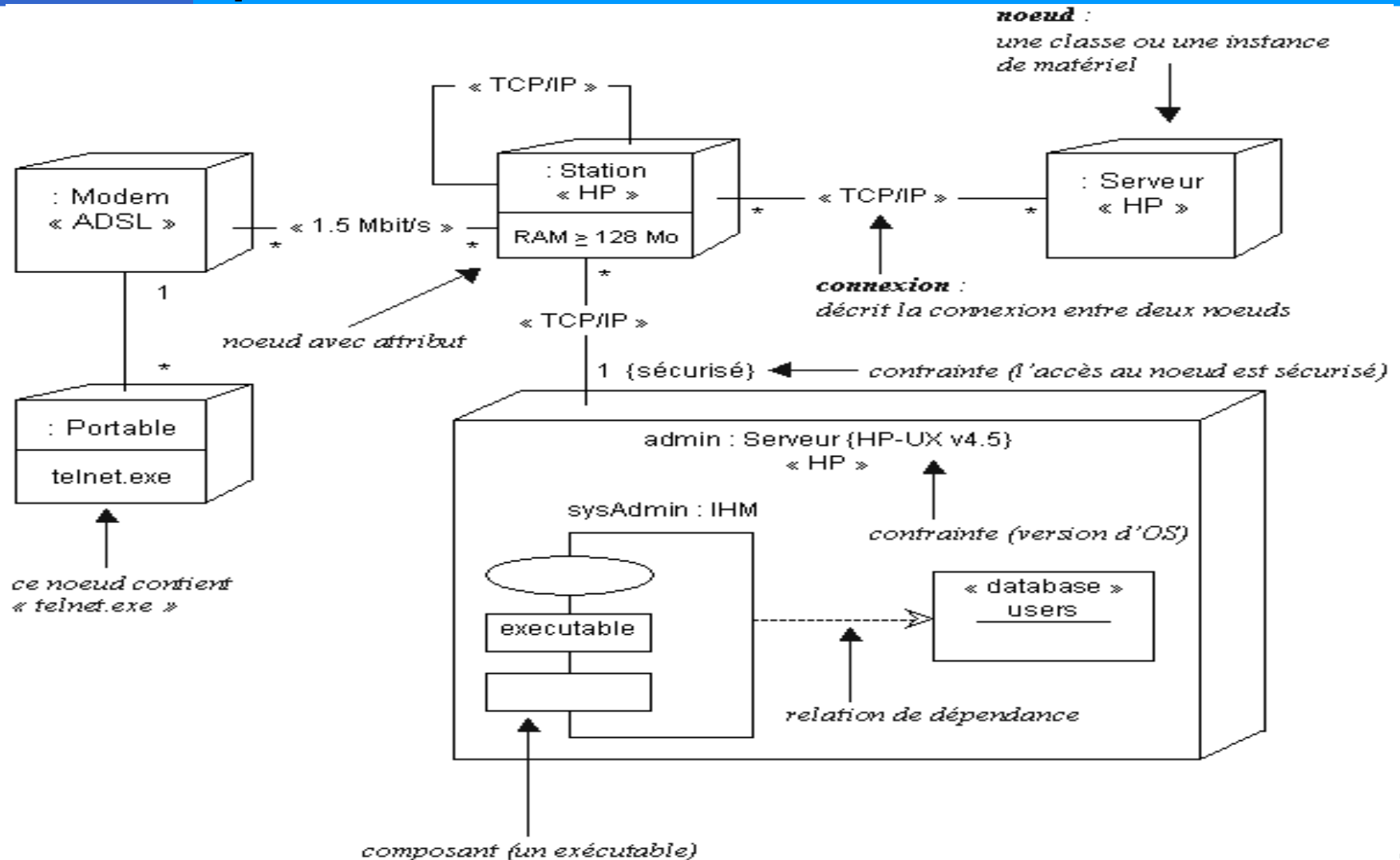
Représentation du déploiement de deux artefacts dans un nœud utilisant la relation de dépendance stéréotypée *«deploy»*.

Diagrammes de Déploiement



Représentation du déploiement dans un nœud d'un artefact manifestant un composant.

Exemple de diagramme de Déploiement :



Diagrammes de séquences

III- Modélisation des Vues Dynamiques du système

III-3-a- Diagrammes De Séquences :

Les diagrammes de séquences permettent de représenter des collaborations entre objets selon un point de vue temporel, on y met l'accent sur la chronologie des envois de messages.

Contrairement au diagramme de collaboration, on n'y décrit pas le contexte ou l'état des objets, la représentation se concentre sur l'expression des interactions.

Les diagrammes de séquences peuvent servir à illustrer un cas d'utilisation.

L'ordre d'envoi d'un message est déterminé par sa position sur l'axe vertical du diagramme ; le temps s'écoule "de haut en bas" de cet axe.

La disposition des objets sur l'axe horizontal n'a pas de conséquence pour la sémantique du diagramme.

Les diagrammes de séquences et les diagrammes d'état-transitions sont les vues dynamiques les plus importantes d'UML.


■ **Buts :**

1. décrire les cas d'utilisation (scénarios)
2. décrire les interactions entre objets
3. Identifier des objets, et donc des classes potentielles
4. identification des échanges entre objets, issus des scénarios

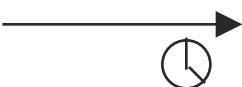
III- La Modélisation Dynamique

■ Types de messages :

message simple :


Message dont on ne spécifie aucune caractéristique d'envoi ou de réception particulière. 

message minuté (timeout) :

Bloque l'expéditeur pendant un temps donné (qui peut être spécifié dans une contrainte), en attendant la prise en compte du message par le récepteur. L'expéditeur est libéré si la prise en compte n'a pas eu lieu pendant le délai spécifié. 

III- La Modélisation Dynamique


message synchrone :

Bloque l'expéditeur jusqu'à prise en compte du message par le destinataire. Le flot de contrôle passe de l'émetteur au récepteur (l'émetteur devient passif et le récepteur actif) à la prise en compte du message. 

message asynchrone :

N'interrompt pas l'exécution de l'expéditeur. Le message envoyé peut être pris en compte par le récepteur à tout moment ou ignoré (jamais traité). 

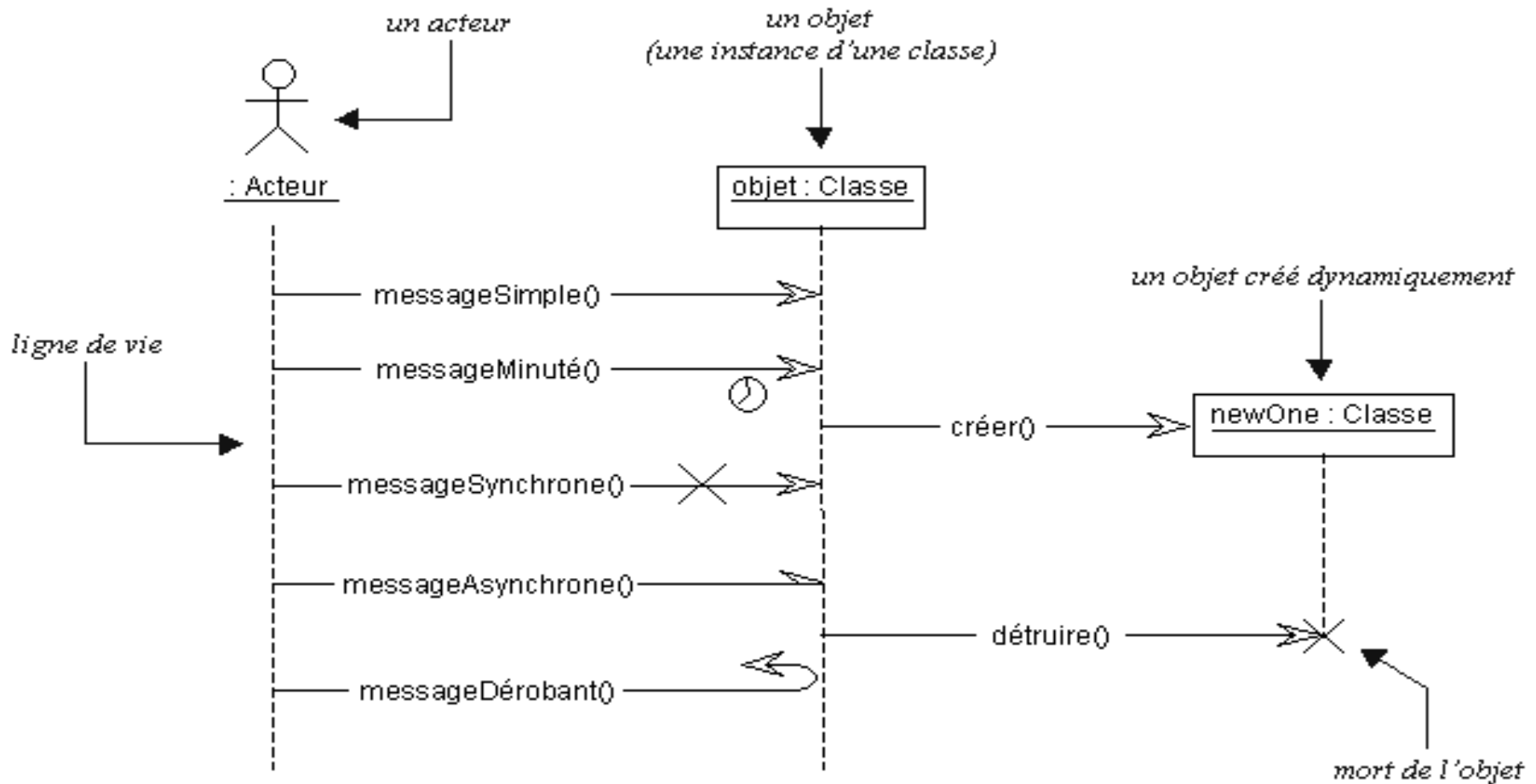
message dérobant :

N'interrompt pas l'exécution de l'expéditeur et ne déclenche une opération chez le récepteur que s'il s'est préalablement mis en attente de ce message. 

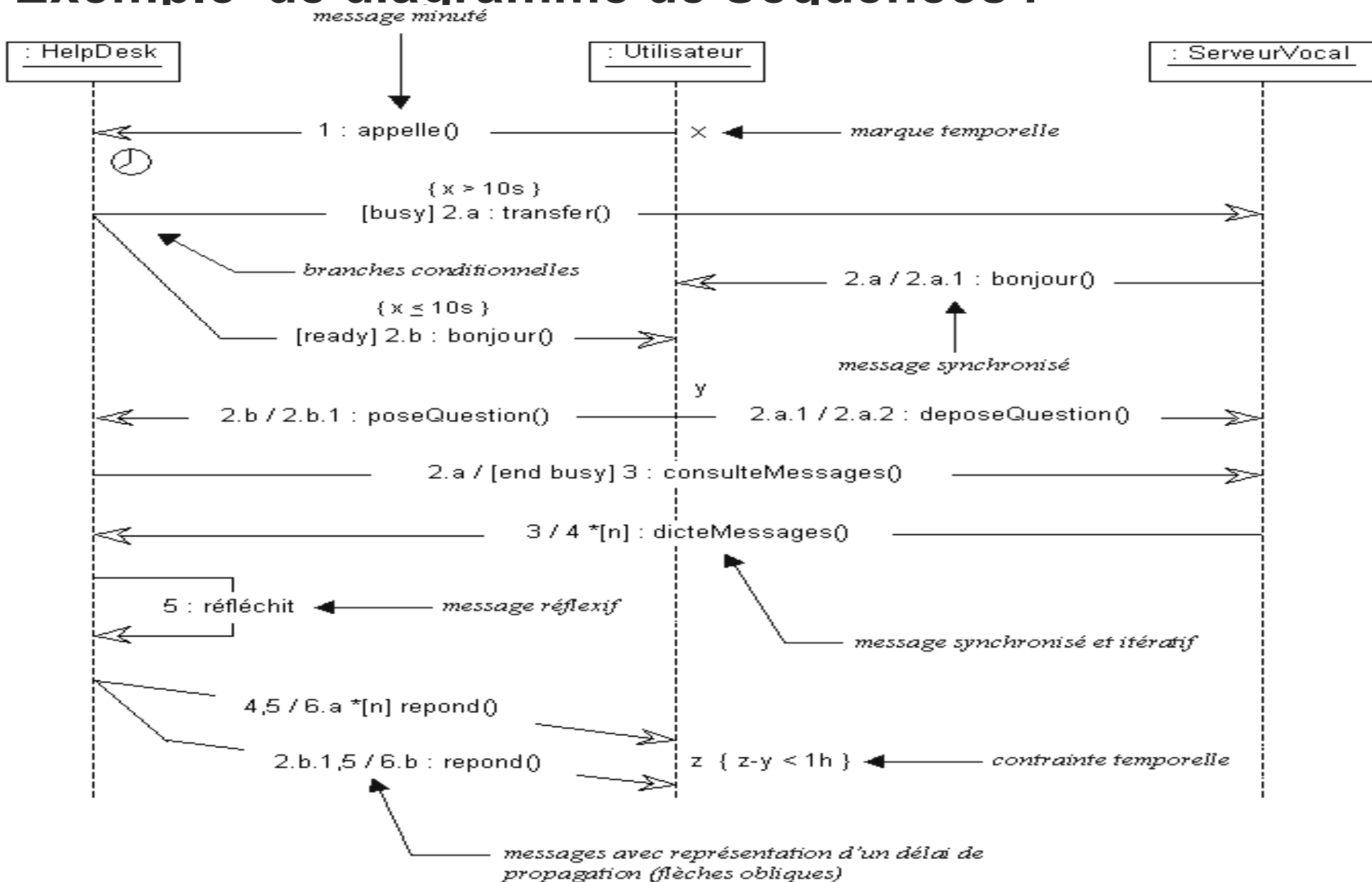
III- La Modélisation Dynamique

- Chaque réception de message donne lieu à une durée d'activation : le temps de traitement du message
- La durée d'activation de l'émetteur recouvre celle du récepteur
- Type de messages :
 - flot de contrôle à plat :
 - message synchrone
 - message asynchrone
 - flot de contrôle emboîté ou appel de procédure (avec attente implicite du retour)
 - retour d'un appel de procédure, avec ou sans paramètre de retour
- Autres structures de contrôle (mais à éviter)
 - alternative
 - répétition

Formalisme de diagramme de Séquences :



Exemple de diagramme de Séquences :



III- La Modélisation Dynamique

■ Exercice

A partir d'un cas d'utilisation de votre projet, construire un diagramme de séquence

—**X**→ **synchrone**

—→ **asynchrone**

—→ **avec délai**

← **reflexif**

—→**X** **détruire**

—→ **activer**

Exercice 1 :

Le déroulement normal d'utilisation d'un distributeur automatique de billets est le suivant :

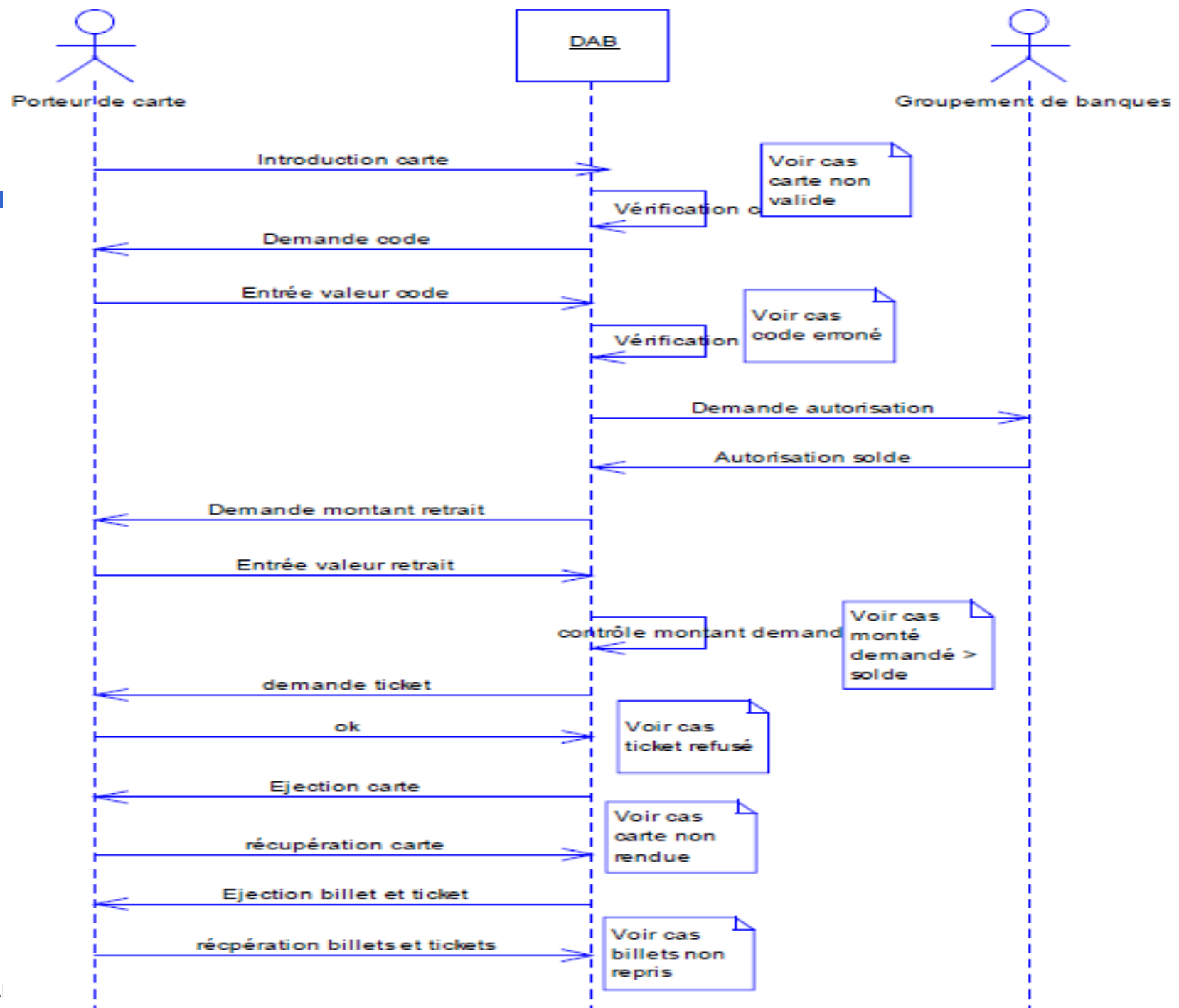
le client introduit sa carte bancaire

la machine vérifie alors la validité de la carte et demande le code au client si le code est correct, elle envoie une demande d'autorisation de prélèvement a groupement de banques. Ce dernier renvoie le solde autorisé à prélever. le distributeur propose alors plusieurs montants à prélever le client saisit le montant à retirer après contrôle du montant par rapport au solde autorisé, le distributeur demande a client s'il désire un ticket. Après la réponse du client, la carte est éjectée et récupérée par le client les billets sont alors délivrés (ainsi que le ticket)

le client récupère enfin les billets et son ticket

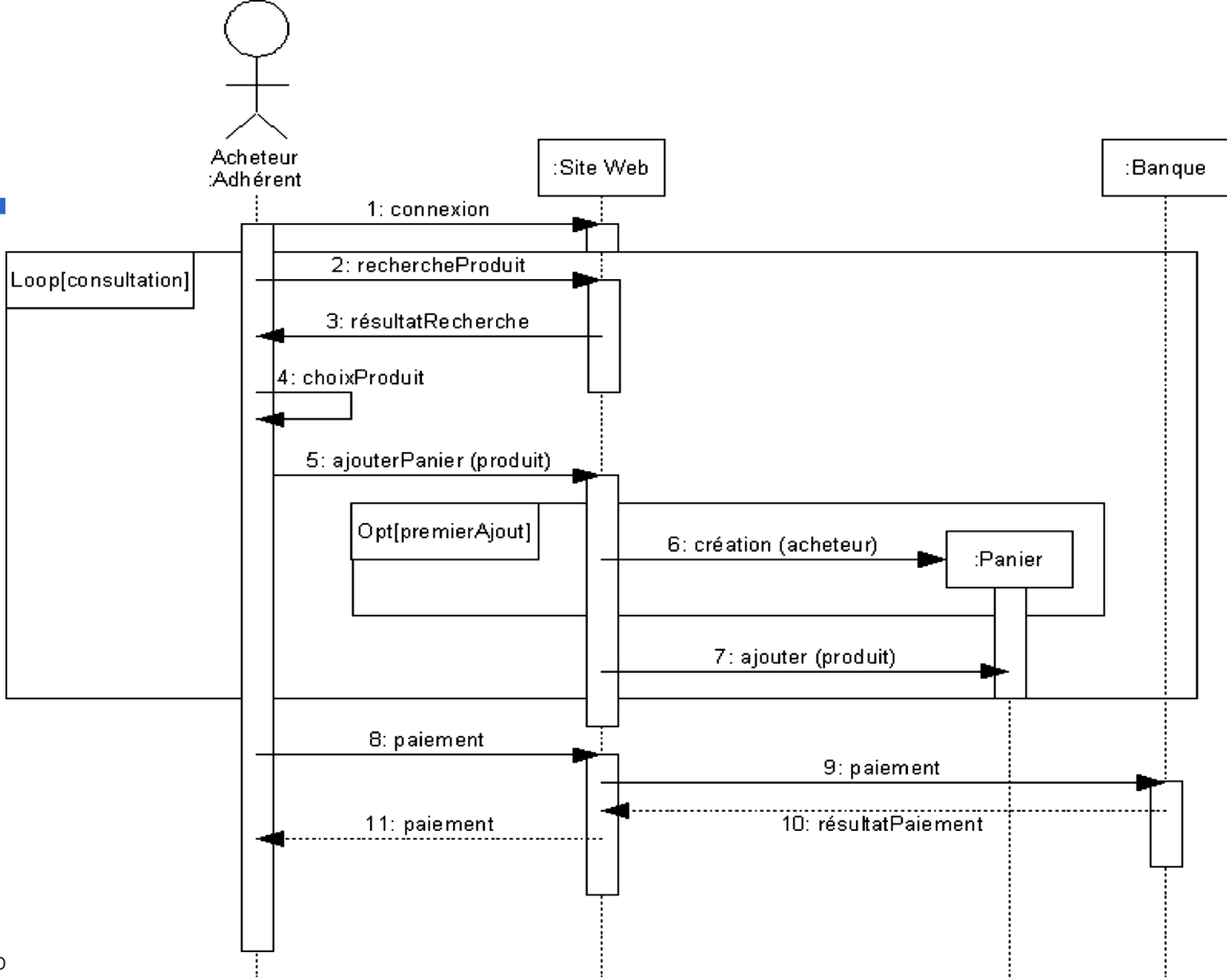
Modéliser cette situation à l'aide d'un diagramme de séquence en ne prenant en compte que le cas où tout se passe bien.

NB : on identifiera les scénarios qui peuvent poser problème en incluant des commentaires dans le diagramme



Ex 2

- *Construisez le diagramme de séquence d'une prise de commande sur un site Internet de vente de produits en ligne.



On souhaite gérer les différents objets qui concourent à l'activité d'un magasin de vente de fleurs.

Le client demande au vendeur des renseignements des renseignements sur les compositions florales

Le vendeur lui fournit toutes les informations nécessaires

Le client commande alors la composition de son choix et le vendeur émet le bon de fabrication qu'il transmet à son ouvrier fleuriste.

Le vendeur édite ensuite la facture correspondante.

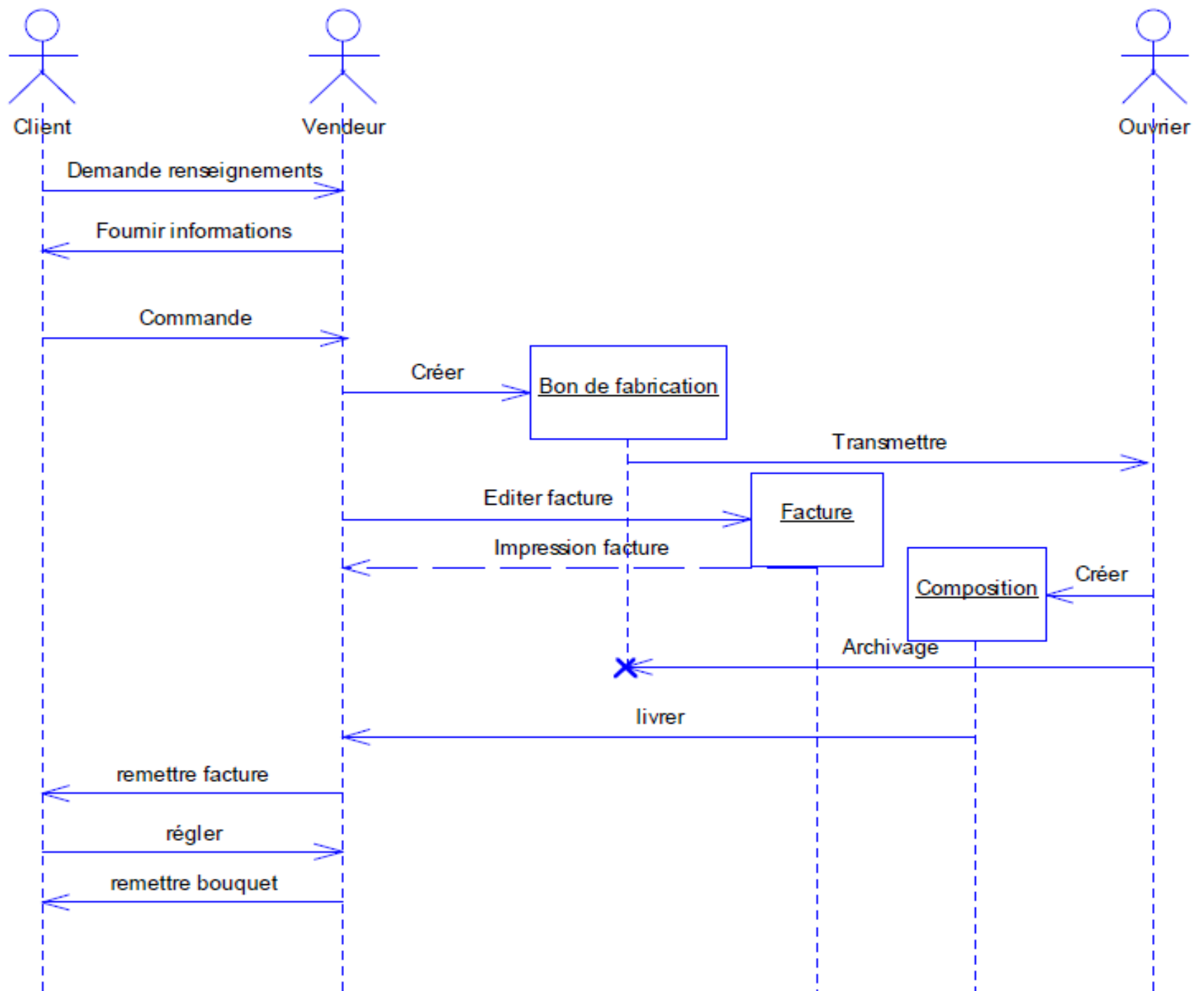
L'ouvrier fleuriste crée la composition puis archive le bon de fabrication. Il remet alors la composition au vendeur

La facture est remise au client pour règlement une fois le bouquet réalisé

Une fois la facture réglée, le client récupère sa composition et quitte le magasin.

Identifier les acteurs, les objets et les différents liens sémantiques de ce système

Modéliser cette situation à l'aide d'un diagramme de séquence et d'un diagramme de collaboration.



Diagrammes de collaboration

III- La Modélisation Dynamique

■ III-3-b- Diagrammes de Collaboration

Les collaborations sont des interactions entre objets, dont le but est de réaliser un objectif du système (c'est-à-dire aussi de répondre à un besoin d'un utilisateur).

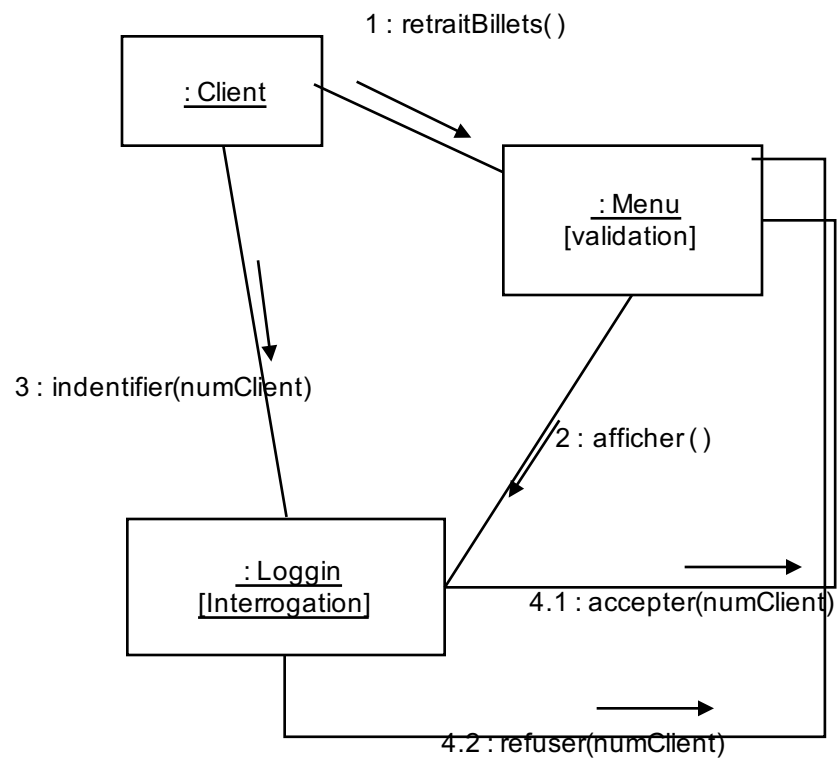
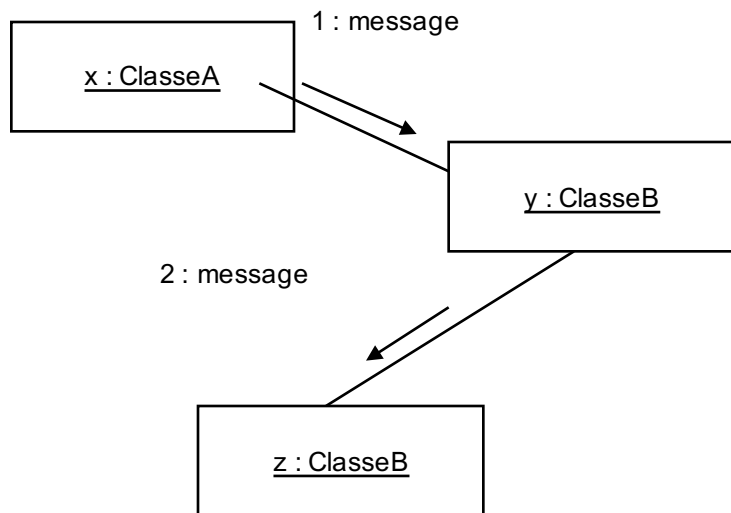
Attention : ne confondez pas l'élément de modélisation "collaboration" avec le diagramme de collaboration, qui représente des interactions entre instances de classes.

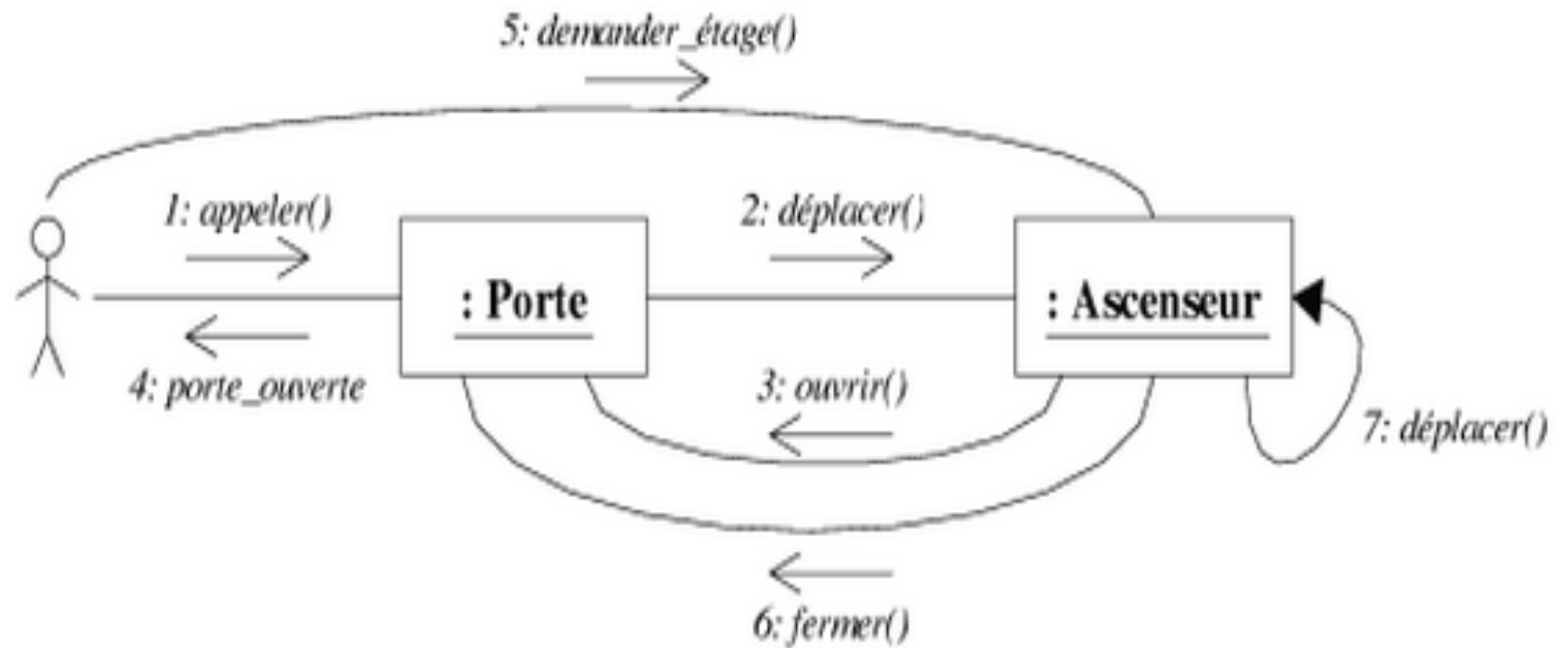
III- La Modélisation Dynamique

■ Buts :

1. Décrire l'interaction des objets entre eux
2. Illustrer les scénarios des use cases
3. Valider les choix d'analyse et de conception (prototypage)
4. Aider à élaborer des diagrammes de classes de conception

Exemple de Diagramme de Collaboration





Diagrammes d'états transitions

State machine diagram

Diagrammes d'Etats Transitions

Ce diagramme sert à représenter des automates d'états finis, sous forme de graphes d'états, reliés par des arcs orientés qui décrivent les transitions.

Les diagrammes d'états-transitions permettent de décrire les changements d'états d'un objet ou d'un composant, en réponse aux interactions avec d'autres objets/composants ou avec des acteurs.

Un état se caractérise par sa durée et sa stabilité, il représente une conjonction instantanée des valeurs des attributs d'un objet.

Une transition représente le passage instantané d'un état vers un autre

III- La Modélisation Dynamique

Une transition est déclenchée par un événement. En d'autres termes : c'est l'arrivée d'un événement qui conditionne la transition.

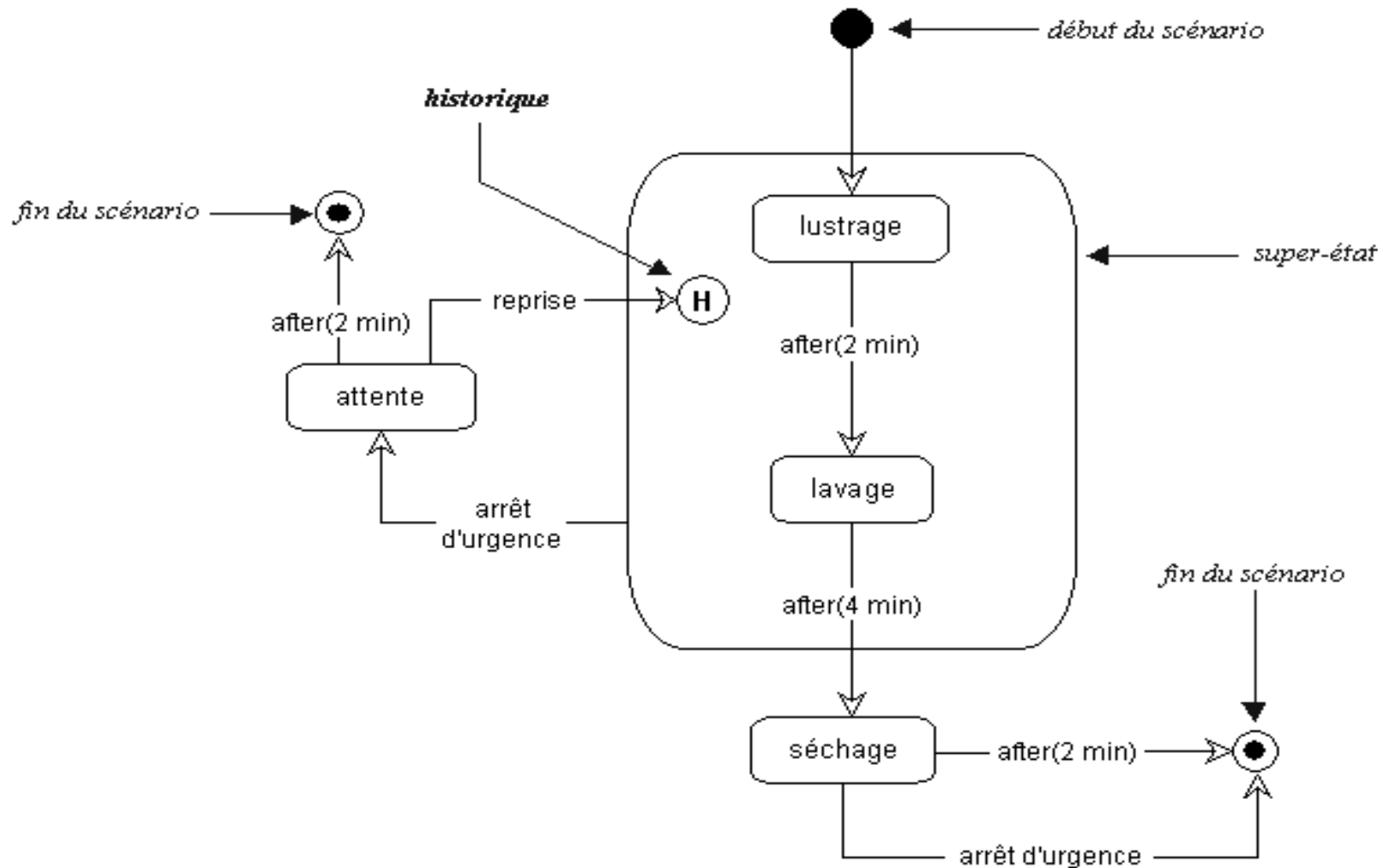
Les transitions peuvent aussi être automatiques, lorsqu'on ne spécifie pas l'événement qui la déclenche.


En plus de spécifier un événement précis, il est aussi possible de conditionner une transition, à l'aide de "gardes" : il s'agit d'expressions booléennes, exprimées en langage naturel (et encadrées de crochets).

■ Buts :

1. Illustrer les cas d'utilisation
2. Décrire en détail le comportement des classes

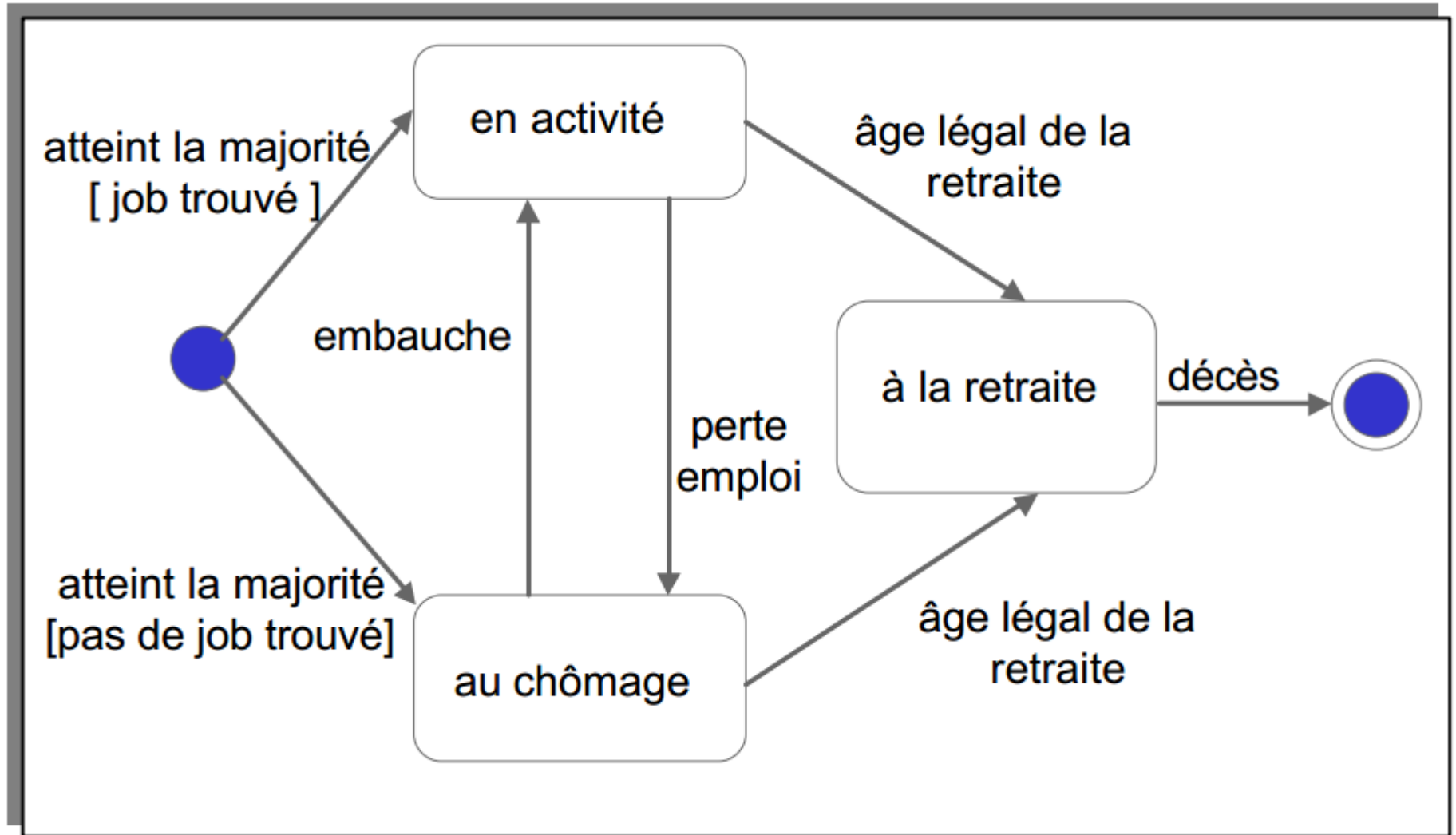
Exemple de Diagramme d'Etats Transition :





Le diagramme d'états-transitions ci-dessous, montre les différents états par lesquels passe une machine à laver les voitures. En phase de lustrage ou de lavage, le client peut appuyer sur le bouton d'arrêt d'urgence. S'il appuie sur ce bouton, la machine se met en attente. Il a alors deux minutes pour reprendre le lavage ou le lustrage (la machine continue en phase de lavage ou de lustrage, suivant l'état dans lequel elle a été interrompue), sans quoi la machine s'arrête. En phase de séchage, le client peut aussi interrompre la machine. Mais dans ce cas, la machine s'arrêtera définitivement (avant de reprendre un autre cycle entier).

Diagramme d'Etats Transitions



III- La Modélisation Dynamique

■ *Les états :*

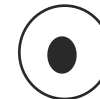
- Un objet est à tout moment dans un état donné
- L'état d'un objet est constitué des valeurs instantanées de ses attributs.
- L'objet passe d'un état à un autre par les transitions.
- Ces états sont représentés par des rectangles au coin arrondi
- L'état initial se représente par un point noir et l'état final par un point noir encerclé



état



état initial



état Final

Les transitions

- Déclenché par un événement, les transitions permettent le passage d'un état à un autre instantanément. La syntaxe d'un événement dans un diagramme est la suivante :

Nom_événement (Nom_paramètre : type, ...)[condition]

- " condition " est la garde qui valide ou non le déclenchement d'une transition quand l'événement s'est produit.
- Il est possible de préciser les actions à exécuter lorsque l'on est dans un état donné, en entrant ou en sortant. Pour cela UML donne plusieurs mots-clés :

entry : action à exécuter dès l'entrée dans l'état

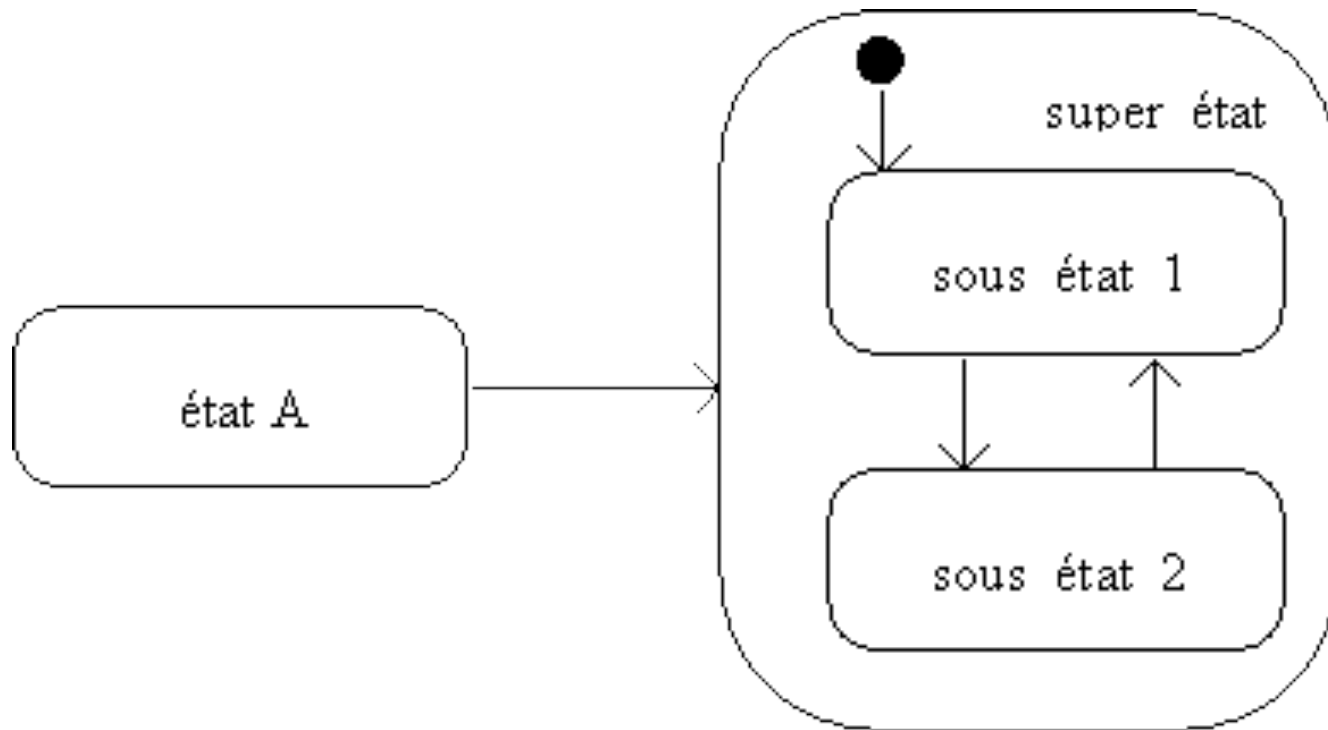
exit : action à exécuter lors de la sortie de l'état

on : action interne provoquée par un événement qui ne provoque pas le passage à un nouvel état.

Do : activité à exécuter (une activité est une action dont le temps d'exécution est non négligeable)

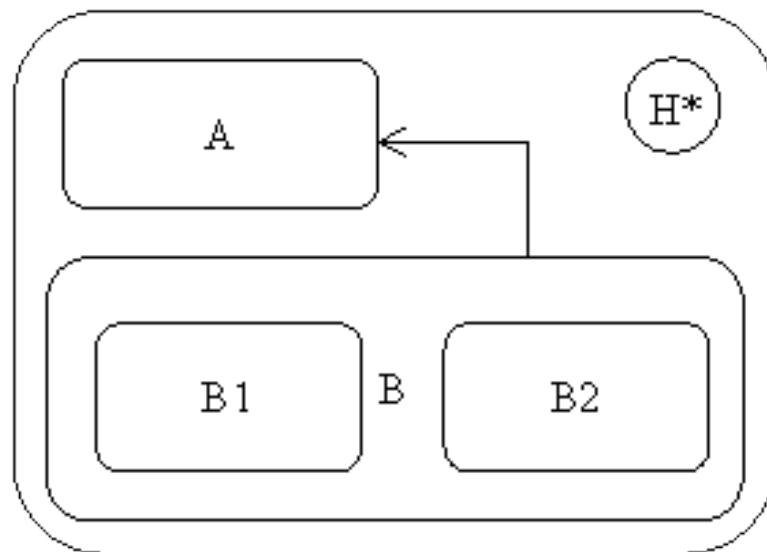
Généralisation d'état

Pour plus de clarté, la notation UML permet la généralisation d'état. Les états les plus généraux sont appelés les super états et les états les plus spécifiques sont appelés les sous états. Un état peut être décomposé en sous état.



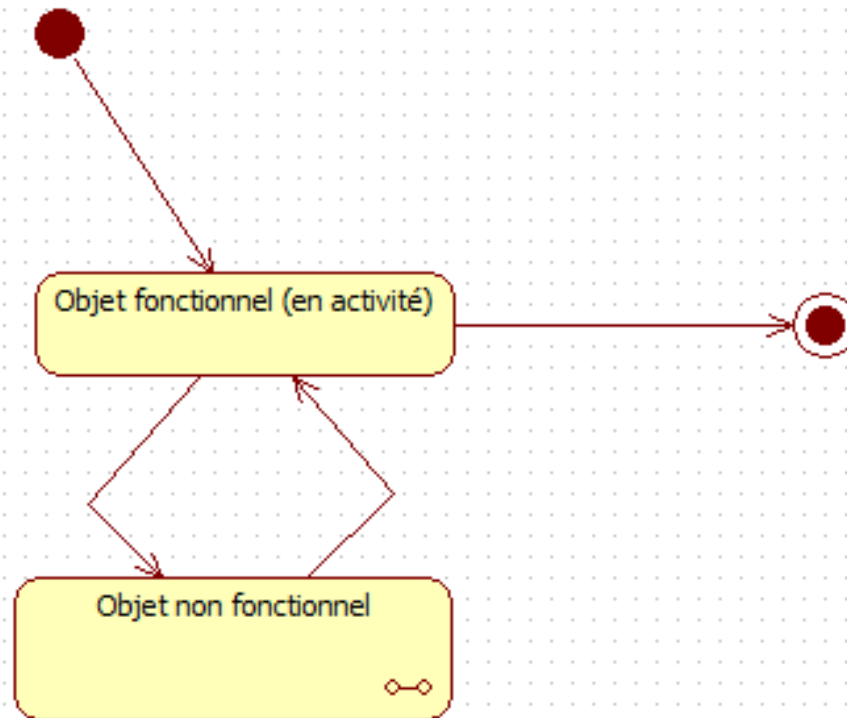
L'historique

Il peut sembler nécessaire de mémoriser les sous-états qui ont été actifs dans une agrégations d'états. On parle alors d'historie. En mettant le symbole " H " dans un super état, on indique que l'on voudrait mémoriser les sous états qui ont été actifs. " H* " indique que l'on veut mémoriser les sous états actifs quel que soit la profondeur de l'imbrications des sous états.



Application

Construire le diagramme d'état de l'objet garantie



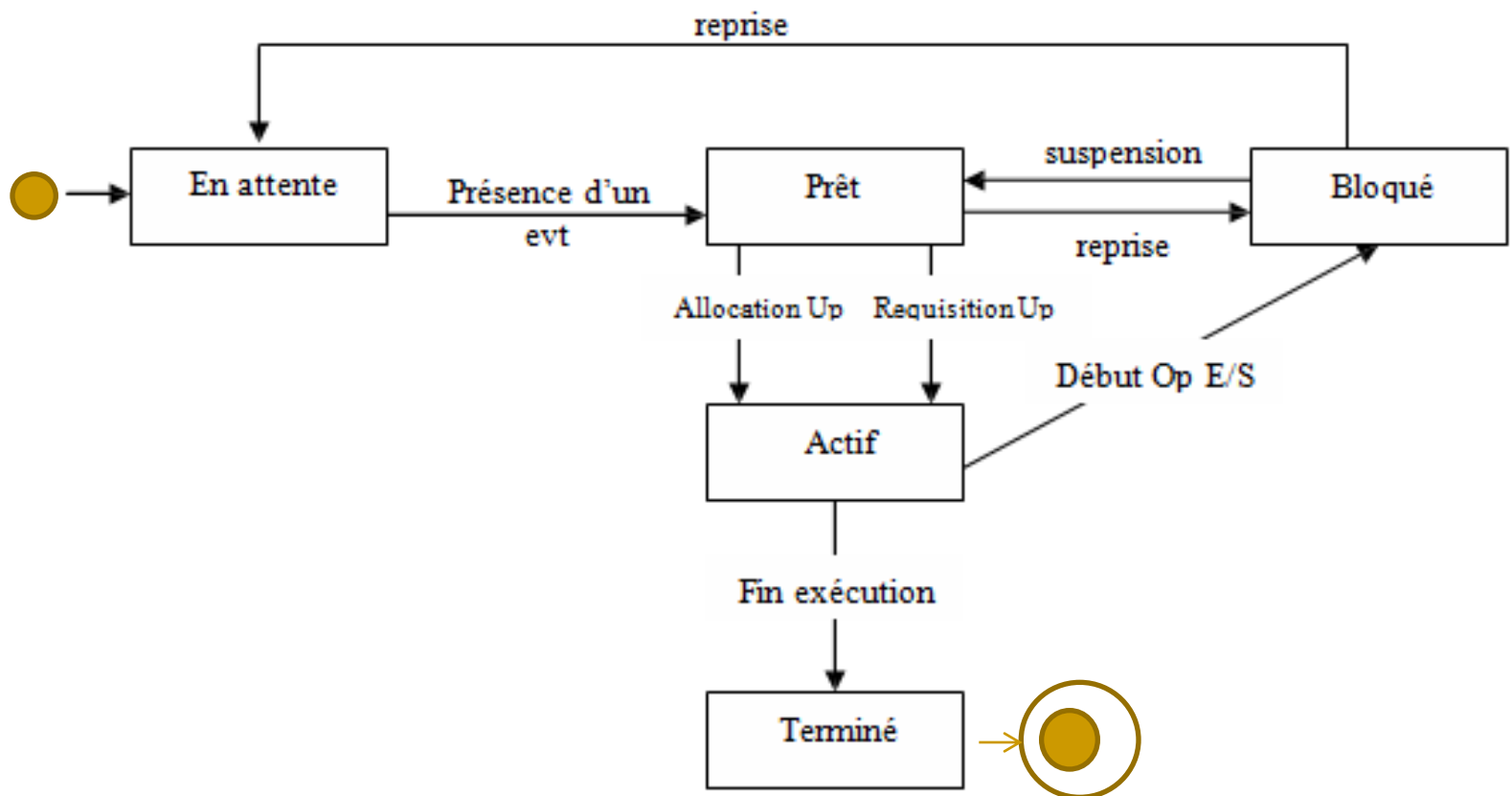
Les opérations sur les processus provoquent le changement de l'état d'un processus, On parle alors de états suivants :

Etat actif (en exécution) : le processus est entrain d'utiliser le processeur c'est à dire quand le processus lui est alloué.

Etat en attente ou bloqué : le processus est entrain de faire une opération E/S, l'attente vient du fait qu'il est entrain d'attendre la fin d'une opération E/S.

Etat prêt : quand le processus est prêt à avoir le contrôle de l'UC, cependant l'UC est occupé par l'exécution d'un autre processus et ceci dans le cas de multiprogrammation. On associe une file d'attente aux processus à l'état prêt.

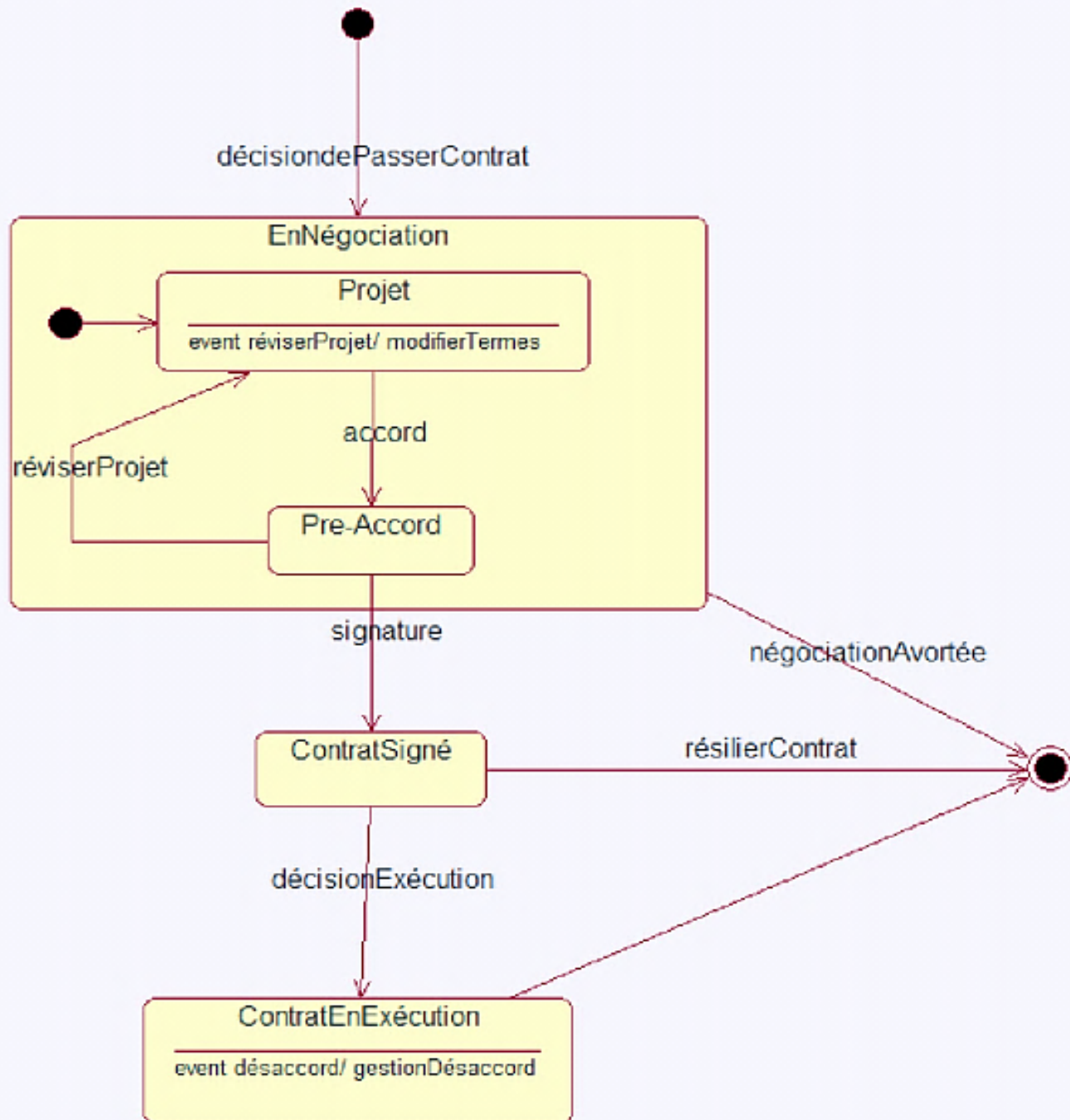
Etablir le diagramme d'état transition correspondant

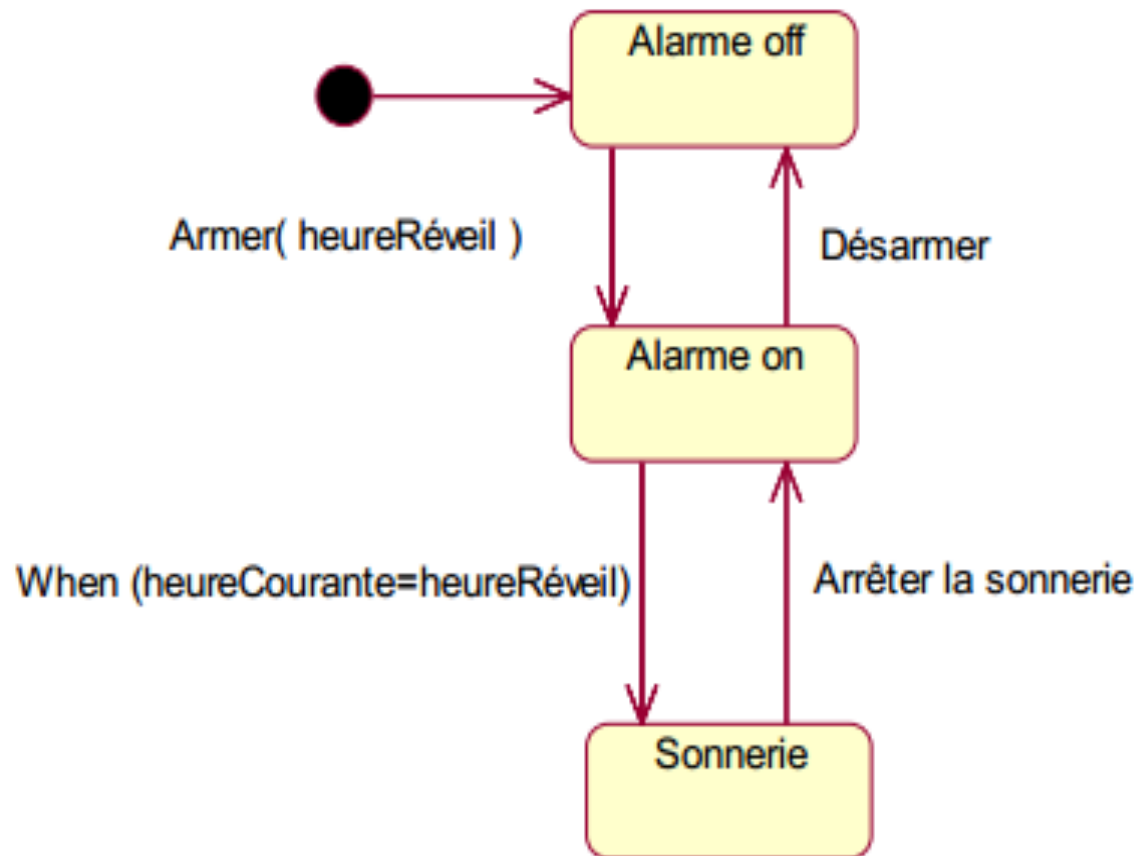


Les états d'un processus

Dessinez un diagramme d'état/transition résumant les états possibles d'un objet "contrat" tel que décrit dans l'énoncé suivant.

Un ensemble de personnes décident d'établir un contrat. Pour ce faire elles rédigent un projet par itération successive. Le contrat est ensuite informellement accepté par les parties, et devient ce que l'on appelle un préaccord. A ce stade il peut toujours être l'objet de modification et revenir à l'état de projet. Une fois le préaccord définitivement établi, le contrat est signé par les parties. Dès ce moment les partenaires sont liés. Une fois signé, le contrat peut être rendu exécutoire par une décision d'une des parties. Un contrat en exécution peut faire l'objet de discussions qui sont réglées par un arbitre désigné à cet effet. Le contrat une fois exécuté prend fin.





Diagrammes d'activités

Diagrammes d' Activités

UML permet de représenter graphiquement le comportement d'une méthode ou le déroulement d'un cas d'utilisation, à l'aide de diagrammes d'activités (une variante des diagrammes d'états-transitions).

Une activité représente une exécution d'un mécanisme, un déroulement d'étapes séquentielles.

Le passage d'une activité vers une autre est matérialisé par une transition.

III- La Modélisation Dynamique

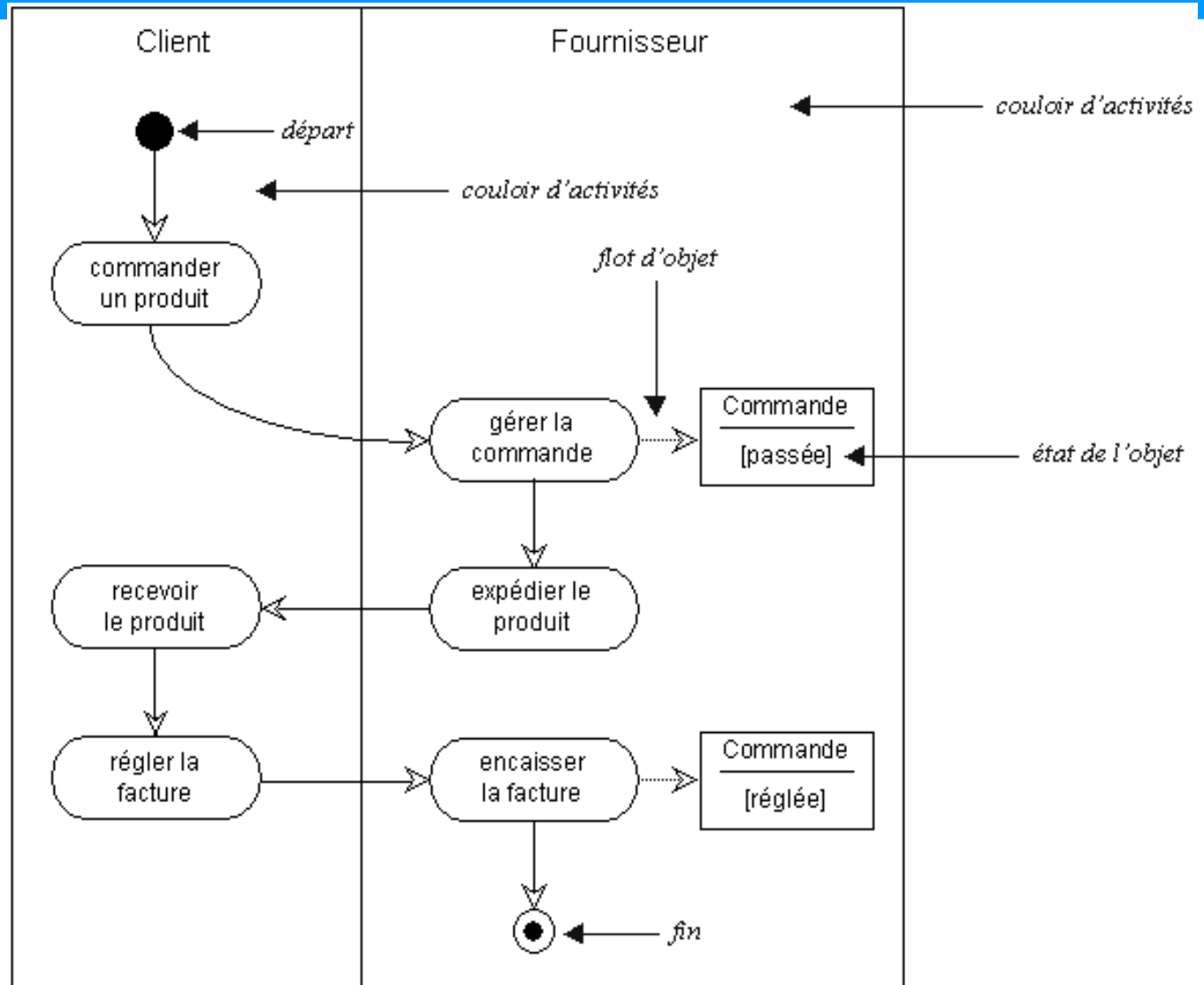
Les transitions sont déclenchées par la fin d'une activité et provoquent le début immédiat d'une autre (elles sont automatiques).

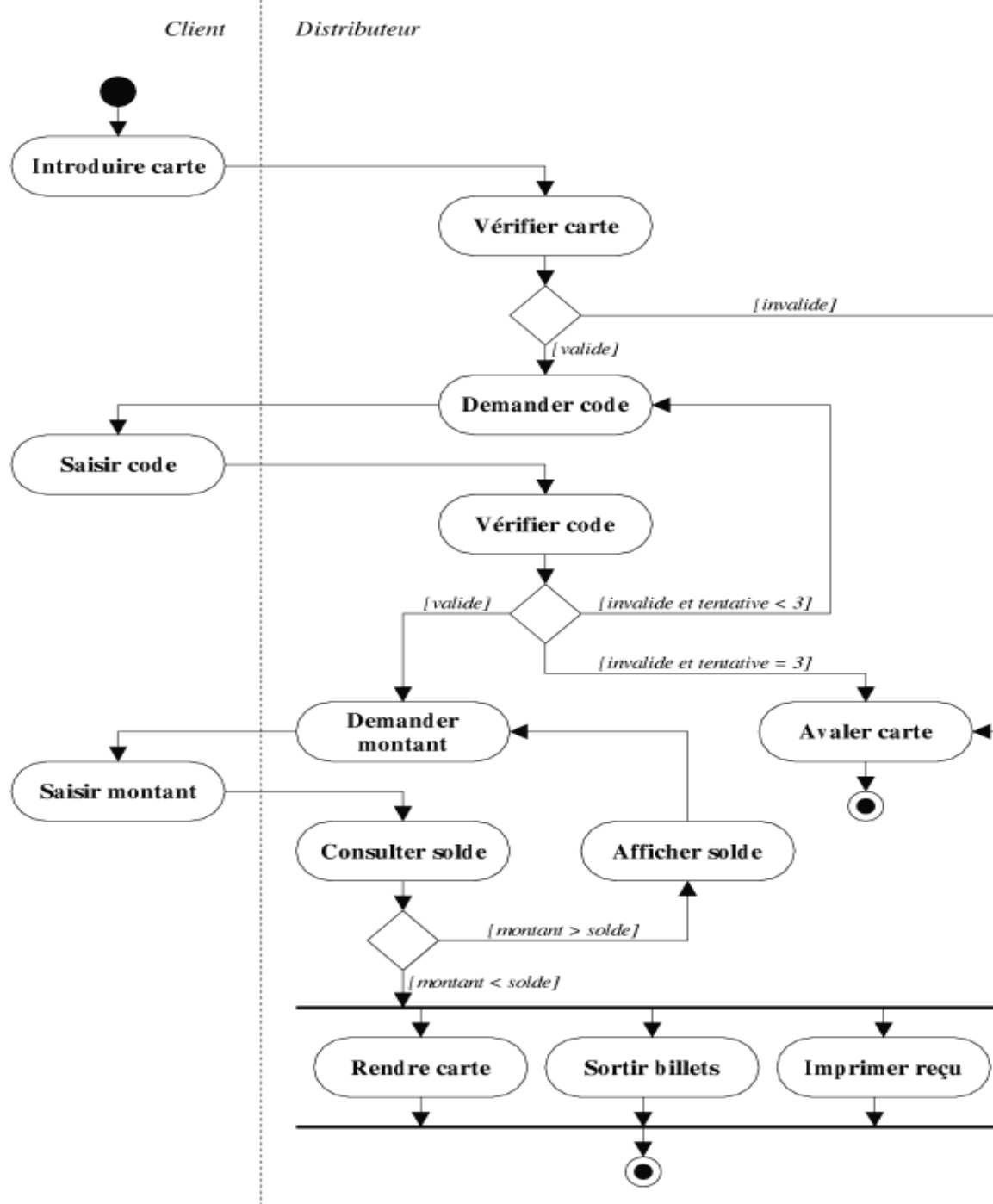
En théorie, tous les mécanismes dynamiques pourraient être décrits par un diagramme d'activités, mais seuls les mécanismes complexes ou intéressants méritent d'être représentés.

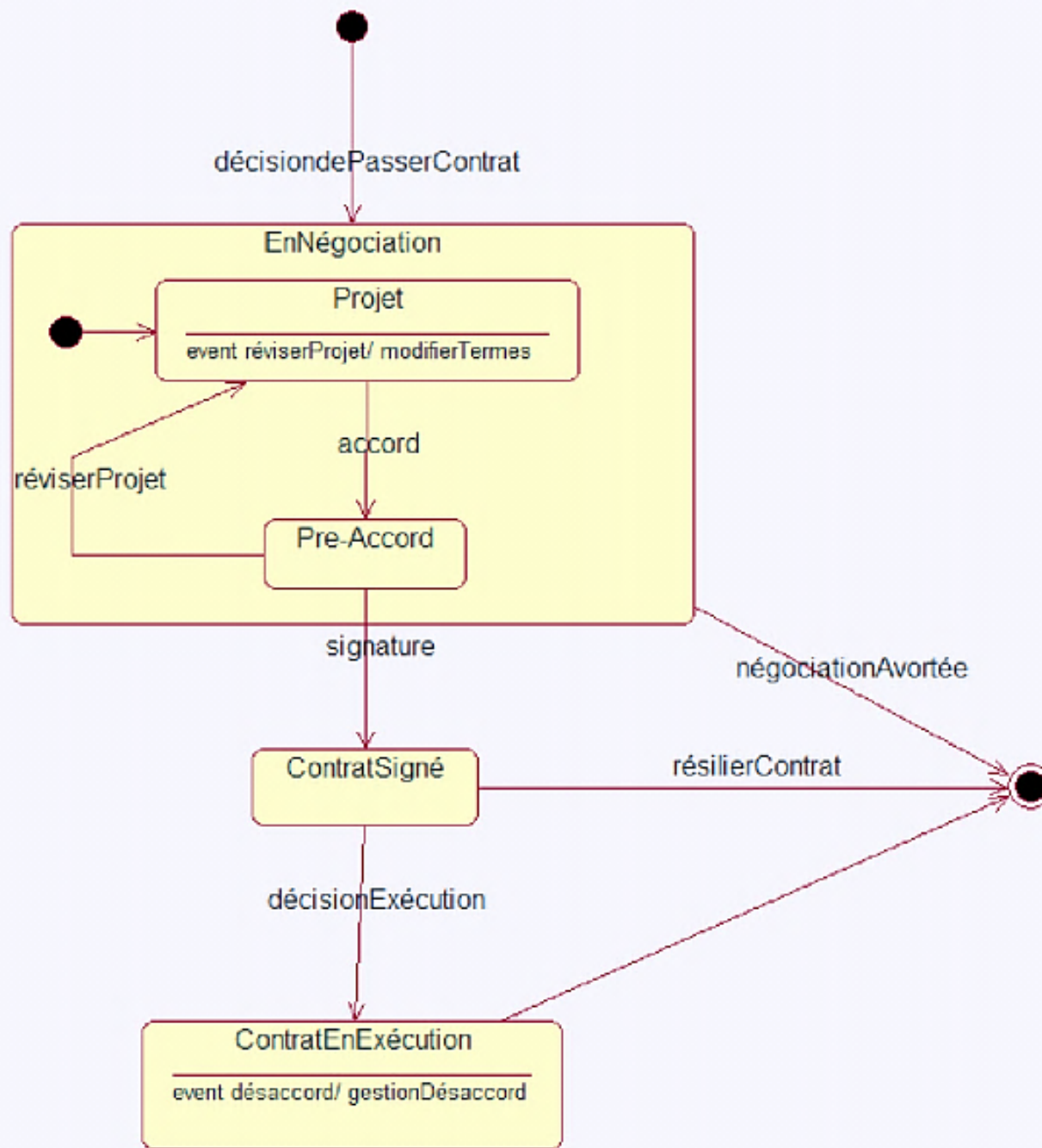
■ Buts :

1. Décrire le comportement générique d'un use case
2. Décrire en détail le comportement d'une opération
3. Modéliser les processus métiers

Exemple de Diagramme d'Activité :







IV- Le mécanisme d'extensions

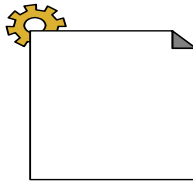
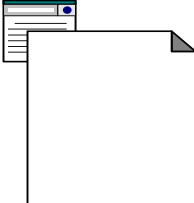
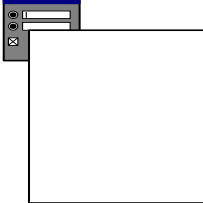
■ Le mécanisme d'extensions :

Beaucoup de concepteurs utilisant UML reconnaissent que ce langage n'est pas toujours adapté à toutes les solutions. Il existe des cas où le processus de conception nécessite des informations supplémentaires et une sémantique différente appliquée aux éléments de bases. UML propose à travers le mécanisme d'extensions d'ajouter ces sémantiques et ces contraintes additionnelles au moyen de la définition de stéréotypes.

Les stéréotypes peuvent être des icônes représentant les classes particulières (pages contenant des scripts cotés serveur par exemple) ou simplement des étiquettes (tels que link pour représenter un lien hypertexte entre deux pages).

IV- Le mécanisme d'extensions

- Les stéréotypes les plus utilisés sont :

| <i>Stéréotypes</i> | <i>Description</i> |
|---|---|
|  | <i>Page contenant des scripts s'exécutant du coté serveur</i> |
|  | <i>Page contenant des scripts s'exécutant du coté client</i> |
|  | <i>Les formulaires</i> |

Avantages et inconvénients d'UML

V- Avantages et Inconvénients d'UML

■ IV-1- Diagrammes d 'UML

■ *Avantages :*

- Représentation en 2D
- Représentation sémantiquement riche
- Chaque notation est simple , facilité d'apprentissage et par des non informaticiens, validation possible par des utilisateurs finaux

■ *Inconvénients :*

- Sémantique pas totalement définie , risque d'interprétation variées
- Beaucoup de diagrammes différents , difficulté a maîtriser toute les notations
- Besoin d'une méthode sous-jacente : Fusion ,RUP...

V- Avantages et Inconvénients d'UML

■ V-2- UML

■ *Avantages :*

- Langage formel
- De nombreux AGL
- Formalismes graphiques
- Plusieurs types de diagrammes - différents niveaux et vues
- Forte cohérence entre diagrammes
- Les diagrammes sont issus de formalismes existants
- Extensibilité

V- Avantages et Inconvénients d'UML

- UML est un langage standardisé et normalisé → très répandu
 - gain de précision
 - gage de stabilité
 - encourage l'utilisation d'outils
- UML est un support de communication performant
 - Il cadre l'analyse.
 - Il facilite la compréhension de représentations abstraites complexes.
- Il Son caractère polyvalent et sa souplesse en font un langage universel.

V- Avantages et Inconvénients d'UML

■ *Inconvénients :*

- Long à maîtriser
- Pas de méthode
- La mise en pratique d'UML nécessite un apprentissage et passe par une période d'adaptation.
- Le processus (non couvert par UML) est une autre clé de la réussite d'un projet.

Processus Unifié (Unified Process)

- Cycle de vie itératif et incrémental Méthode générique
- Générique signifie qu'il est nécessaire d'adapter UP au contexte du projet, de l'équipe, du domaine et/ou de l'organisation (*exemple: X.UP*). C'est, entre parenthèses, plus ou moins vrai pour toute méthode, qu'elle se définisse elle-même comme générique ou pas.
- Le processus unifié est piloté par les cas d'utilisation

| | |
|---|---|
| Modèle des cas d'utilisation | Expose les cas d'utilisation et leurs relations avec les utilisateurs |
| Modèle d'analyse | Détaille les cas d'utilisation et procède à une première répartition du comportement du système entre divers objets |
| Modèle de conception | Définit la structure statique du système sous forme de sous système, classes et interfaces ; Définit les cas d'utilisation réalisés sous forme de collaborations entre les sous systèmes les classes et les interfaces |
| Modèle d'implémentation | Intègre les composants (code source) et la correspondance entre les classes et les composants |
| Modèle de déploiement | Définit les nœuds physiques des ordinateurs et l'affectation de ces composants sur ces nœuds. |
| Modèle de test | Décrit les cas de test vérifiant les cas d'utilisation |
| Représentation de l'architecture | Description de l'architecture |

VI- UML et les AGL

■ Critères de sélection :

- Nombre de diagrammes supportés
- Génération automatique de diagrammes
- Génération de code – langages supportés
- Rétro ingénierie (reverse engineering)
- Prototypage
- Synchronisation du code avec modification extérieures
- API
- Echange avec d'autres AGL UML
- Utilisation pratique (schéma de qualité, convivialité, générer des images...)
- Pattern de conception – et création de patterns

VI- UML et les AGL

■ Quelques AGL :

- Rational ROSE (Rational corp.) <http://www.rational.com>
- Together (Togethersoft) <http://www.togethersoft.com>
- Paradigm (Platinum) <http://www.platinum.com>
- Magic Draw UML (NoMagic) <http://www.nomagic.com>
- DOM (ObjetDirect) <http://www.objetdirect.com>
- Objecteering (SoftTeam) <http://www.objecteering.com>
- Aonix Life Cycle Desktop (Aonix) <http://www.aonix.com>, .fr
- UML Studio (Pragsoft) <http://www.pragsoft.com>

Bibliographies

- ❖ Cours « Atelier UML » Tarak Chaari Maître assistant à l'institut supérieur d'électronique et de communication
- ❖ Cours « GL ET ULM » Bouhlel Hamdi, ISSAT Sousse, 2004/2005
- ❖ Cours Méthodologie de conception U.M.L, Jamel Dimassi, FSM.
Jamil.Dimassi@malix.univ-paris1.fr