

Chapitre 2:

Diagramme comportementaux:

Diagramme de cas d'utilisation

Les activités du projet

- ▶ Etude préalable : Analyse du contexte
- ▶ Analyse et spécification des besoins
- ▶ Analyse du problème ou métier ou système
- ▶ Conception de la solution

UML 2

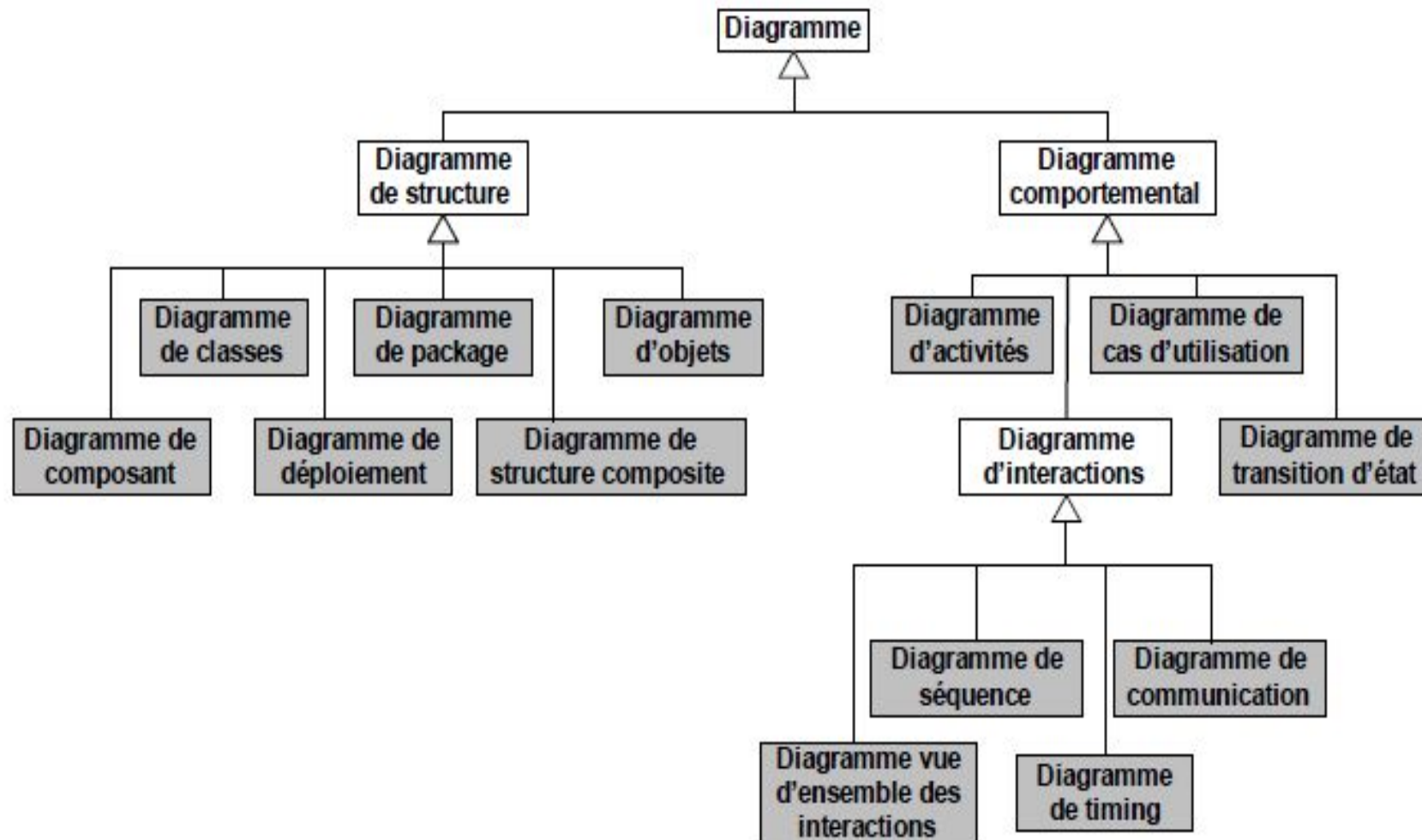


Diagramme des cas d'utilisation

- ▶ Permet de définir **les limites du système et ses relations** avec l'environnement.
- ▶ **Utilisé pour modéliser les exigences (besoins) du client.**

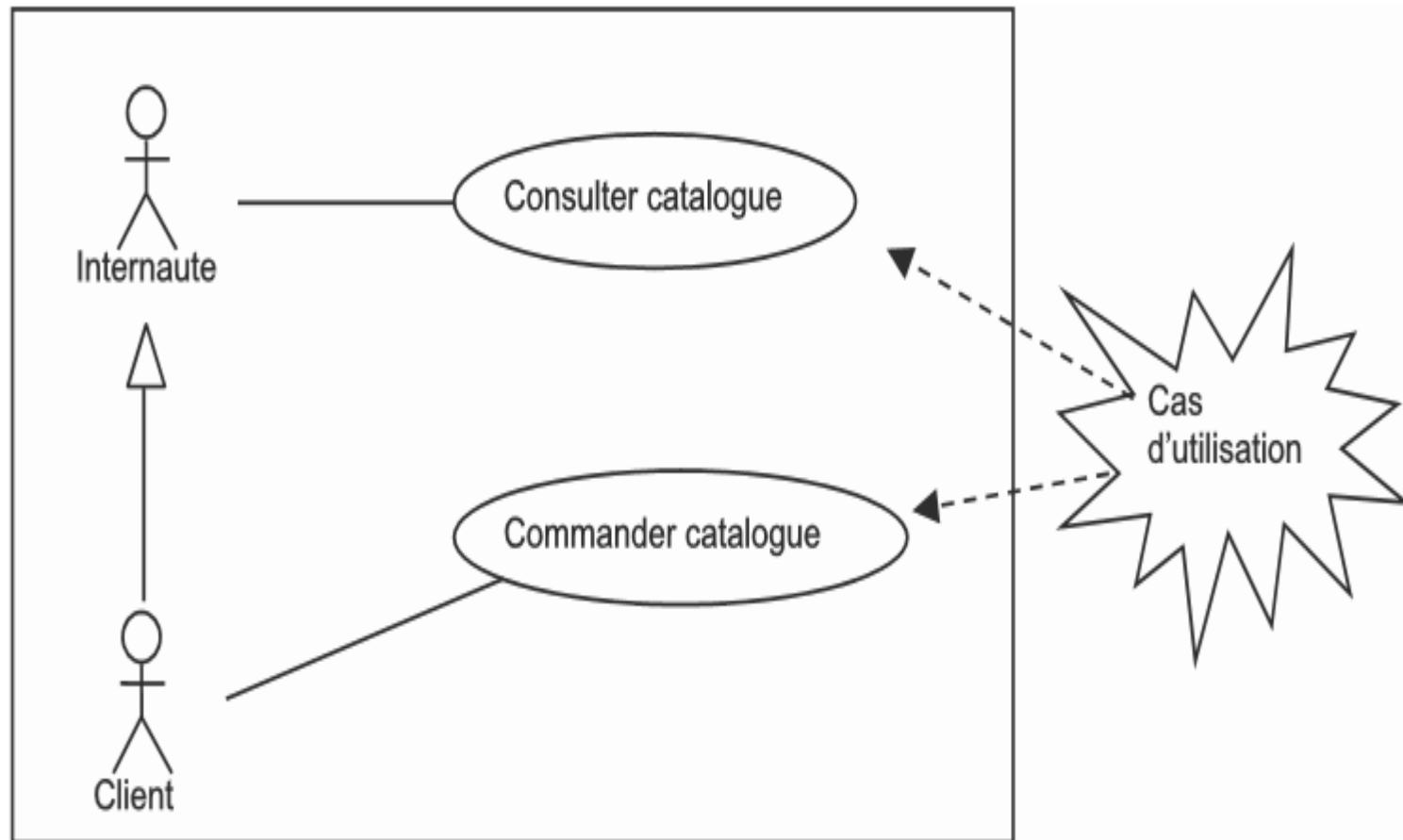
Concepts de base: Cas d'utilisation

- ▶ Les cas d'utilisation constituent un moyen **de recueillir et de décrire** les besoins **des acteurs du système**.
- ▶ Ils peuvent être aussi utilisés ensuite comme **moyen d'organisation du développement du logiciel**:
 - ▶ structuration et le déroulement de **la conception et des tests logiciels**.

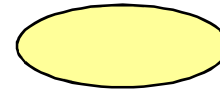
Objectifs: Cas d'utilisation

- Capturer **le comportement** désiré du système,
- **Servir d'entente** entre les différents intervenants (développeurs, utilisateurs et experts) sur **les fonctions disponibles** et sur la façon d'interagir avec le système.
- Spécifier **ce que le système fait** (fonctions), **mais pas comment il le fait.**

Exemple: Cas d'utilisation



Cas d'utilisation



- ▶ Un cas d'utilisation (*use case*) décrit:
 - ▶ Une **fonctionnalité** du système suivant le point de vue de l'utilisateur.
 - ▶ **Les interactions** entre les acteurs et le système,
 - ▶ Un **comportement** attendu **du système** du point de vue d'un ou de plusieurs acteurs,
 - ▶ Un **service** rendu par le système.

Cas d'utilisation: Scénario

- ▶ **Un cas d'utilisation** = Ensemble de « chemins d'exécution » possibles
- ▶ **Un scénario** = Un chemin particulier d'exécution
- ▶ **Un scénario** = Instance de cas d'utilisation
- ▶ Exemple:
- ▶ Cas d'utilisation : *Acheter un objet sur internet*
- ▶ Mais il peut y avoir des scénarios tels que:
 - *échec lors du paiement*
 - *Article non disponible*

Cas d'utilisation

- ▶ La représentation d'un cas d'utilisation met en jeu trois concepts :
 - ▶ l'acteur,
 - ▶ le cas d'utilisation,
 - ▶ l'interaction entre l'acteur et le cas d'utilisation.

Diagramme de cas d'utilisation

- ▶ Un diagramme de cas d'utilisation :

- ▶ décrit

- ▶ les **acteurs**



- ▶ les **cas d'utilisation**

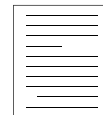
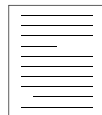
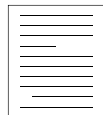
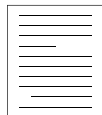


- ▶ le **système**

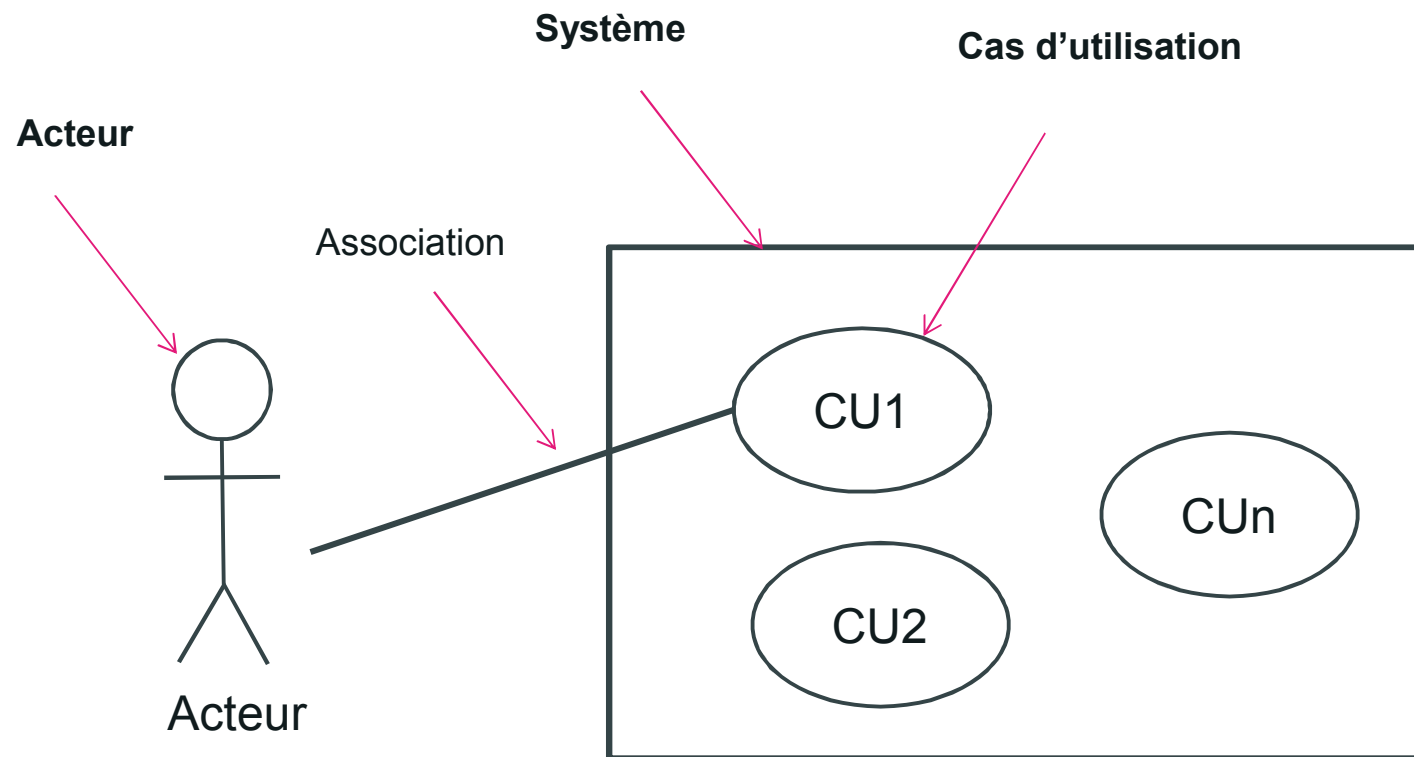


- ▶ contient

- ▶ des **descriptions** textuelles

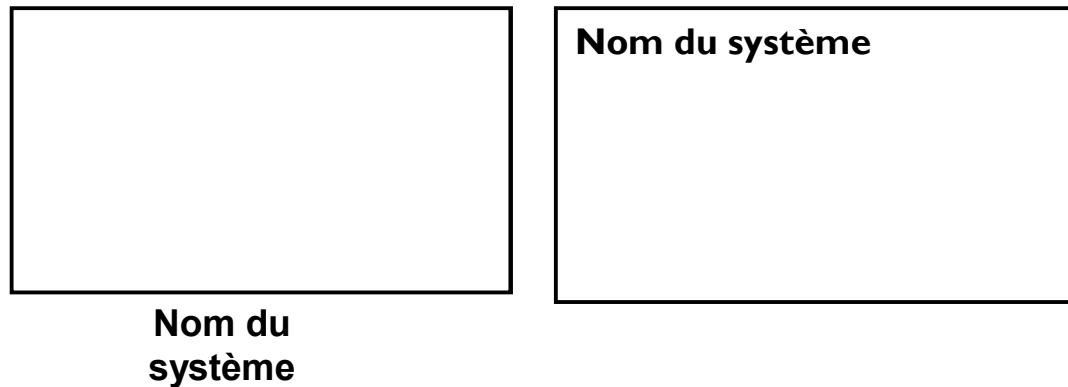


UML : Diagramme De Cas D'utilisation



Le système

- ▶ Le système est **un ensemble de cas d'utilisation**
- ▶ Le système ne comprend pas les acteurs.



Cas d'utilisation

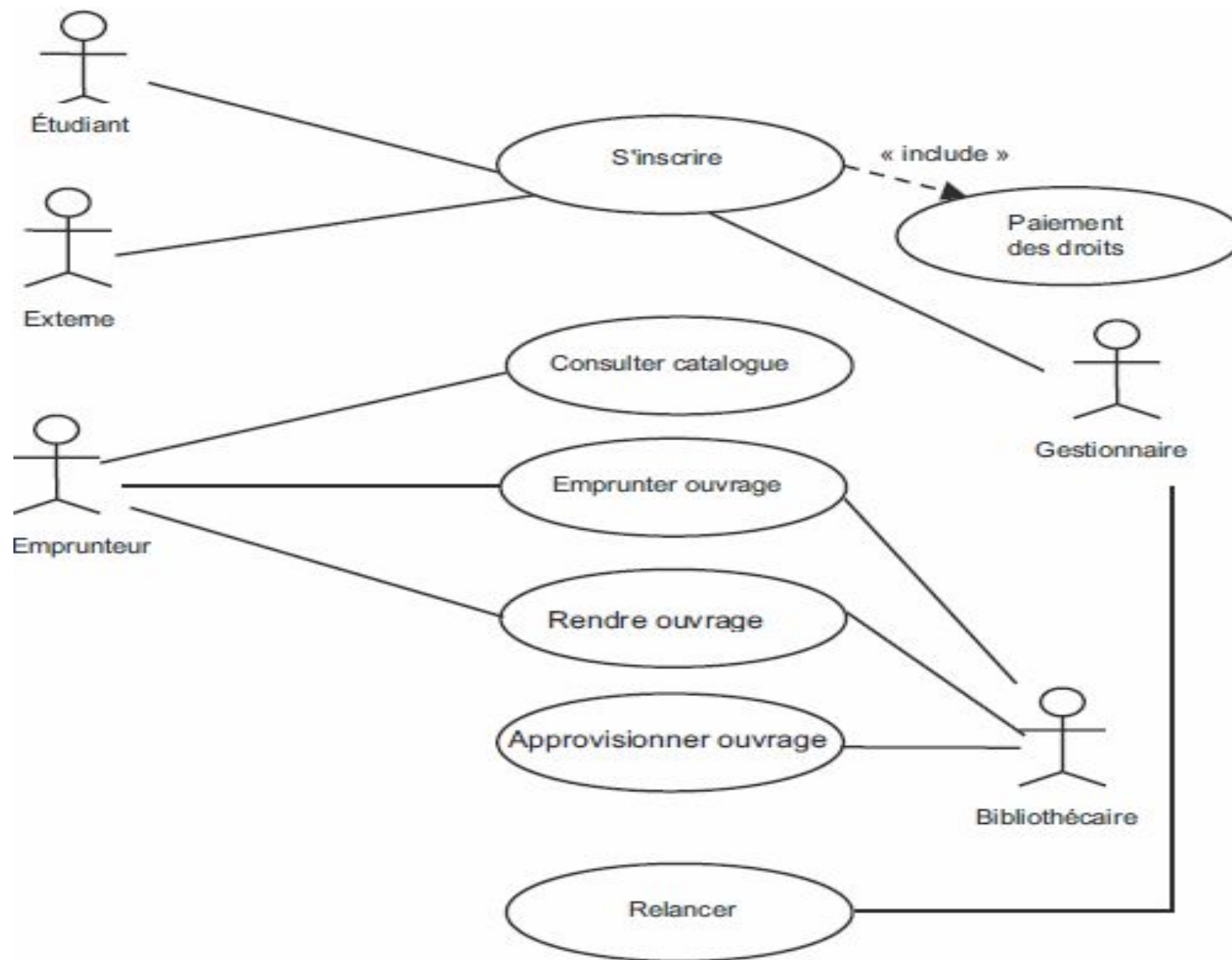
- ▶ Use Case :
 - ▶ Ensemble de séquences d'actions réalisées par le système et qui produisent un résultat **intéressant** pour un acteur particulier.

Acteurs

- ▶ UML n'emploie pas le terme d'utilisateur mais d'acteur.
- ▶ Le terme acteur ne désigne pas seulement des utilisateurs humains mais également les autres systèmes (machines, programmes, ...)

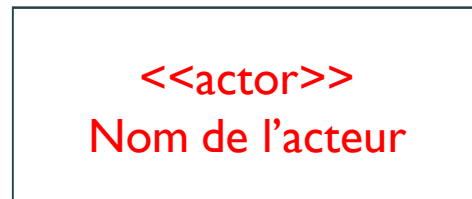
Acteurs

- ▶ Un acteur est un rôle joué par une entité externe qui est en interaction avec le système (échange de l'information en entrée et/ou en sortie):
 - ▶ des utilisateurs humains,
 - ▶ matériels,
 - ▶ logiciels.



Acteur

- ▶ Est représenté par:
 - ▶ Un petit bonhomme (*stick man*) avec son nom dessous **ou**
 - ▶ Par un rectangle contenant le mot-clé **<< actor >>** avec son nom dessous **ou**
 - ▶ Par un mélange de ces 2 représentations.



UML : diagramme de cas d'utilisation

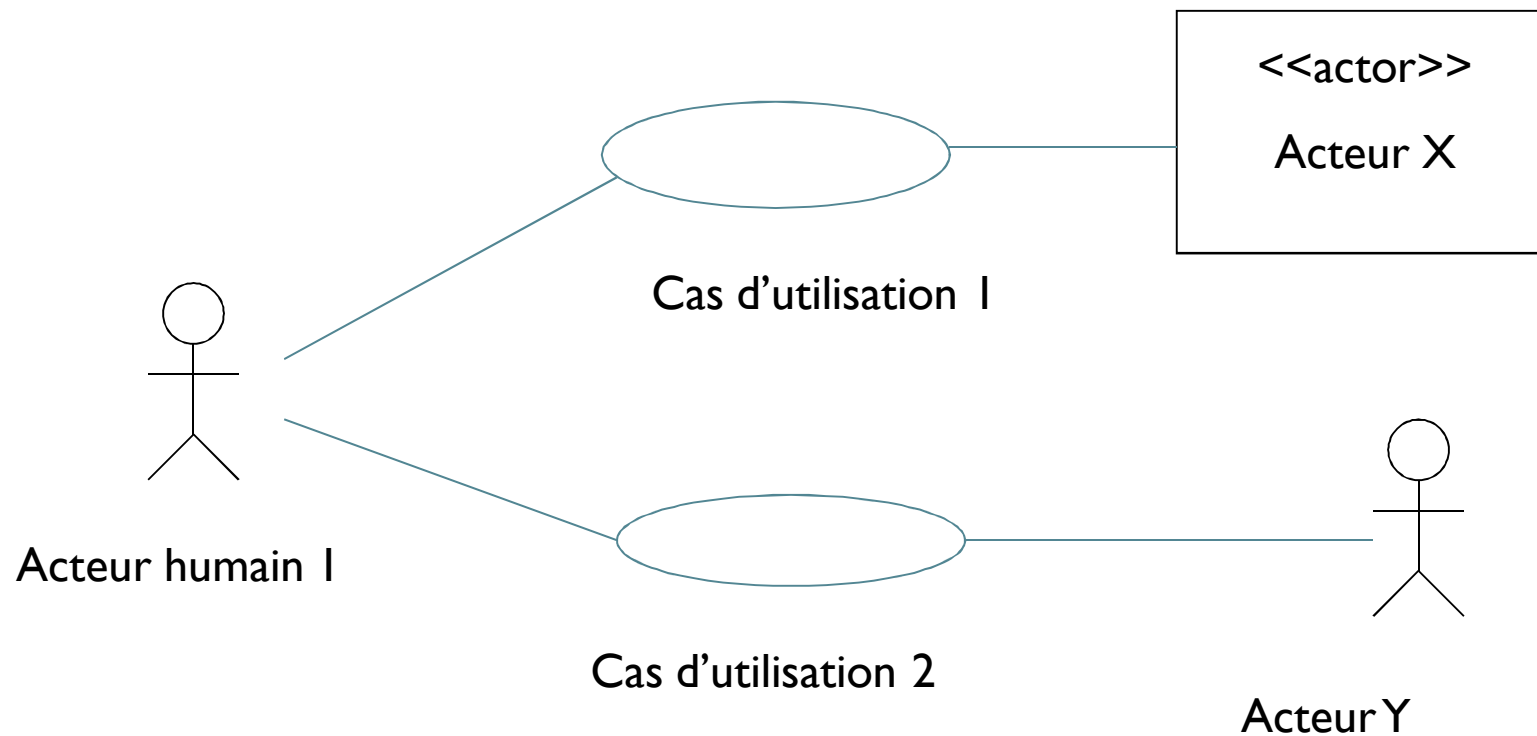
Acteurs vs Personnes :

Ne pas confondre **acteur** et **personne** utilisant le système :

- Une même personne peut jouer plusieurs rôles,
- Plusieurs personnes peuvent jouer un même rôle,
- Un acteur n'est pas forcément une personne physique.

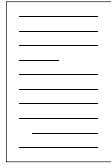
Cas d'utilisation

► Représentation :



Acteurs principaux et secondaires

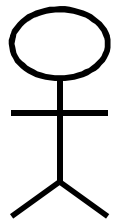
- ▶ Acteur principal d'un CU
 - ▶ Celui pour qui le CU produit un **résultat observable**
 - ▶ **A gauche des CU (généralement)**
- ▶ Acteur secondaire d'un CU
 - ▶ Celui pour qui le CU ne produit pas un résultat observable par l'utilisateur
 - ▶ Souvent sollicités pour **des informations complémentaires**
 - ▶ **A droite des CU (généralement)**
 - ▶ *Ex. : système d'autorisation appelé par le distributeur de billets*



Description des acteurs



- ▶ Pour chaque acteur :
 - ▶ choisir un **identificateur** représentatif de son rôle
 - ▶ donner une brève **description textuelle (facultatif)**



Guichetier

Un guichetier est un employé de la banque.
Interface entre le système informatique et les clients.
Peut effectuer une série d'opérations: création d'un compte, dépôt et retrait d'argent, etc.

Utilité des acteurs

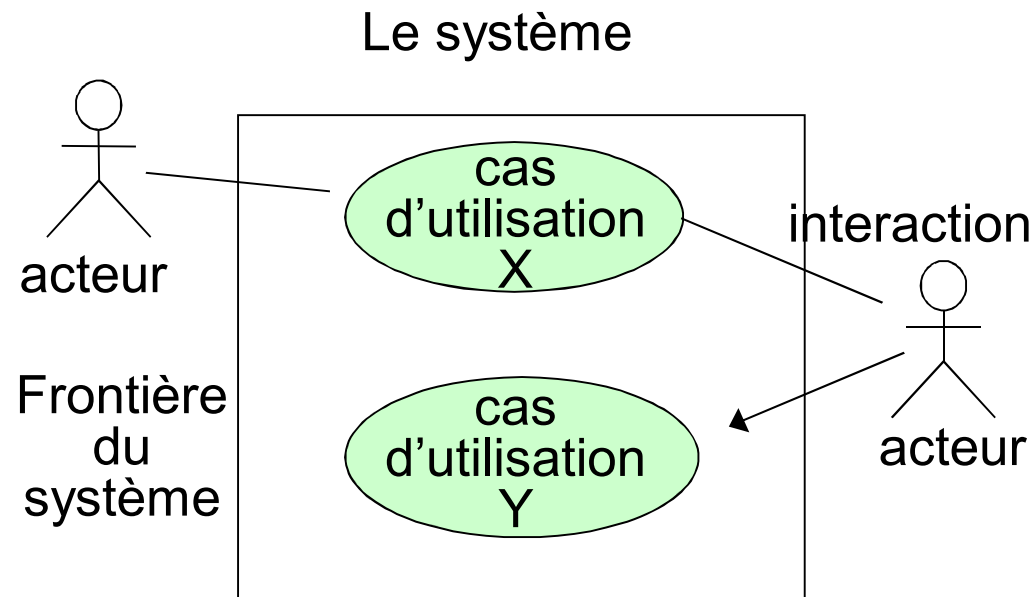


- ▶ La définition d'acteurs permet d'identifier les cas d'utilisation.

Exemple: que peut faire un guichetier ? un client ?

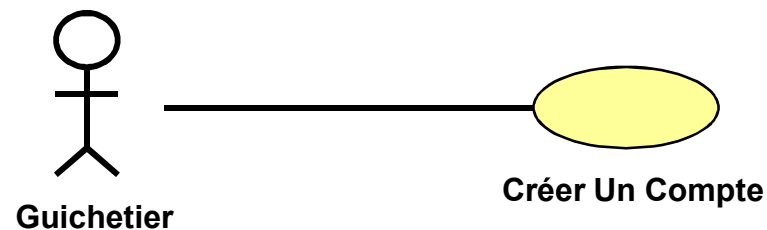
Diagramme de cas d'utilisation

- Un diagramme de cas d'utilisation regroupe plusieurs cas d'utilisation qui ont un fort lien fonctionnel.



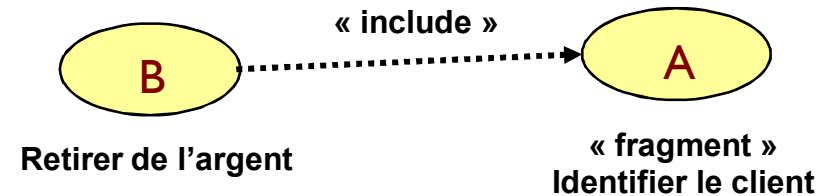
L'acteur est source et/ou destination d'une interaction

Relations entre acteurs et cas d'utilisation



- ▶ **Relation d'association** (« participe à ») :
Lien entre un acteur et un cas d'utilisation qu'il peut exécuter
- ▶ Un cas d'utilisation doit être relié au moins à un acteur

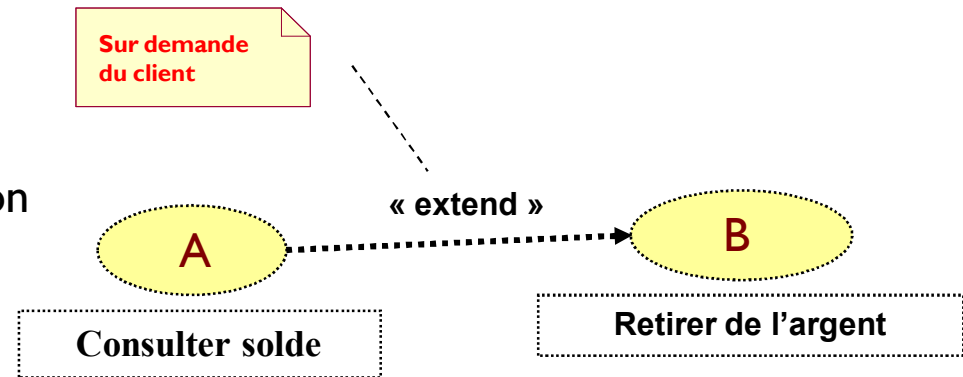
Relations entre cas d'utilisation: Inclusion



- ▶ **Relation d'inclusion**
- ▶ Utilisation du stéréotype « **include** »
- ▶ Un cas A (identifier le client) **est inclus** dans un cas B (retirer de l'argent) si **le comportement** décrit par le cas A est inclus dans le comportement de B
- ▶ Le CU de base B incorpore explicitement le CU A de façon **obligatoire** à un endroit spécifié dans ses enchaînements.
- ▶ Le CU inclus peut porter le stéréotype « **fragment** »

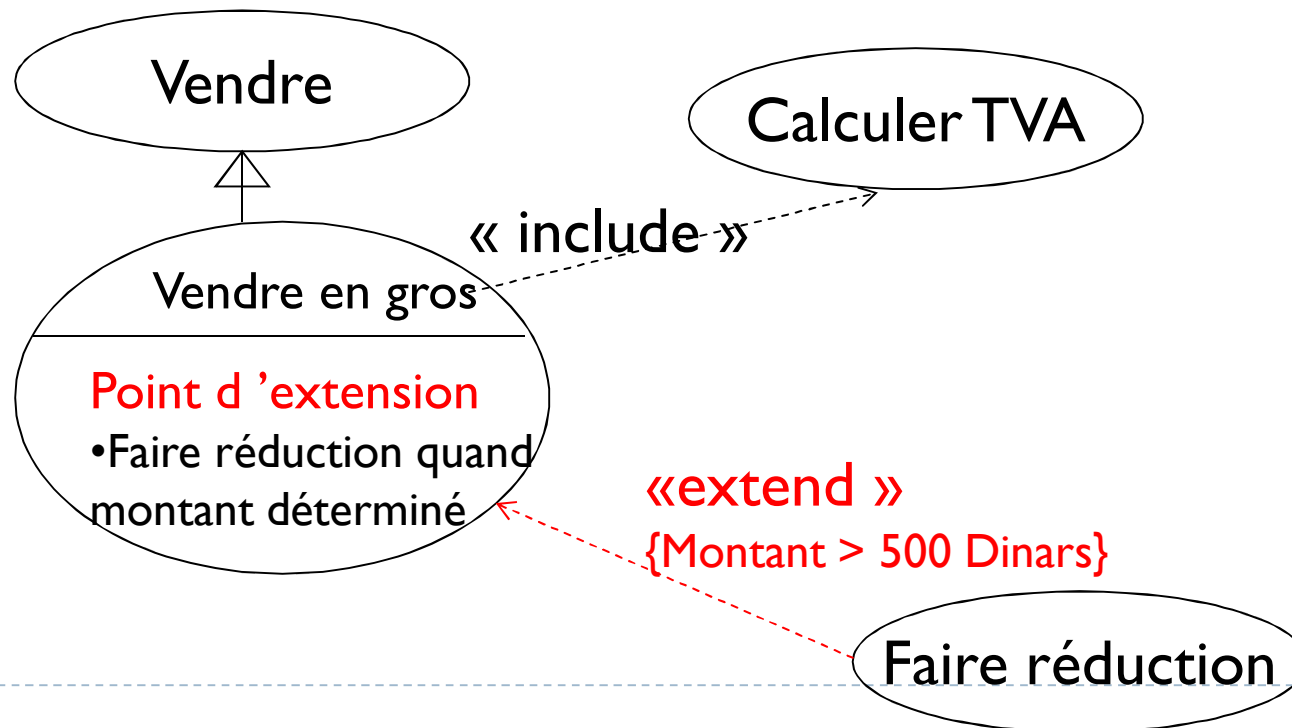
Relations entre cas d'utilisation: extension

- ▶ Relation d'**extension optionnelle** :
- ▶ le CU de base B incorpore un CU A de façon optionnelle :
- ▶ Ex. Le CU 'Consulter solde' se fait uniquement sur demande du client de la banque; il étend le CU 'Retirer de l'argent'
- ▶ Utilisée pour décrire une variation du comportement normal
- ▶ Souvent soumise à condition exprimée sous la forme d'une note
- ▶ Utilisation du stéréotype « **extend** »



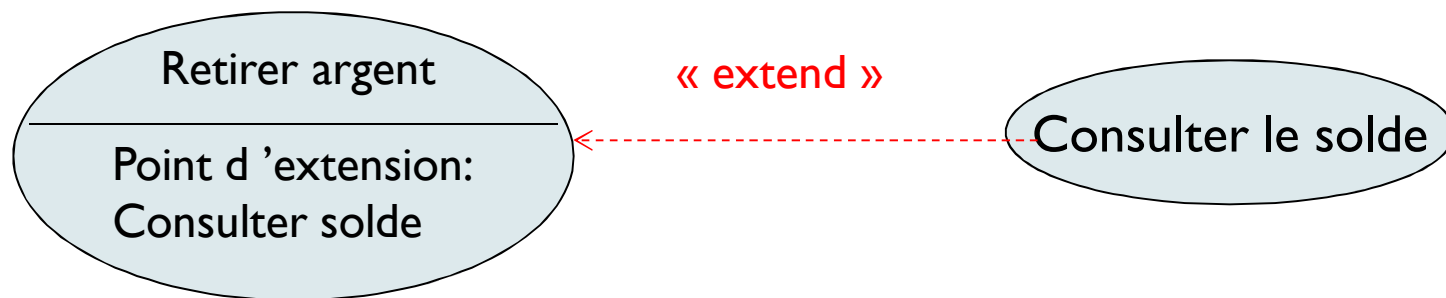
Relations entre cas d'utilisation: extension

- ▶ **Option:** La relation « **extend** » peut spécifier à la fois :
 - ▶ la condition de l'extension,
 - ▶ le point d'extension, c'est-à-dire l'endroit où l'extension doit être faite dans le Use Case.



Relations entre cas d'utilisation: extension

- Dans le cas « Retirer argent », le client peut vouloir vérifier son solde.
- Attention au sens des flèches dans les relations inclut/étend



Relations entre cas d'utilisation

Ce point d'extension doit être déclaré dans la description textuelle, par exemple en modifiant comme ceci l'enchaînement nominal :

...

7. Le SI banque donne son accord et indique le solde hebdomadaire.

8. Le GAB demande au Client banque de saisir le montant désiré du retrait.

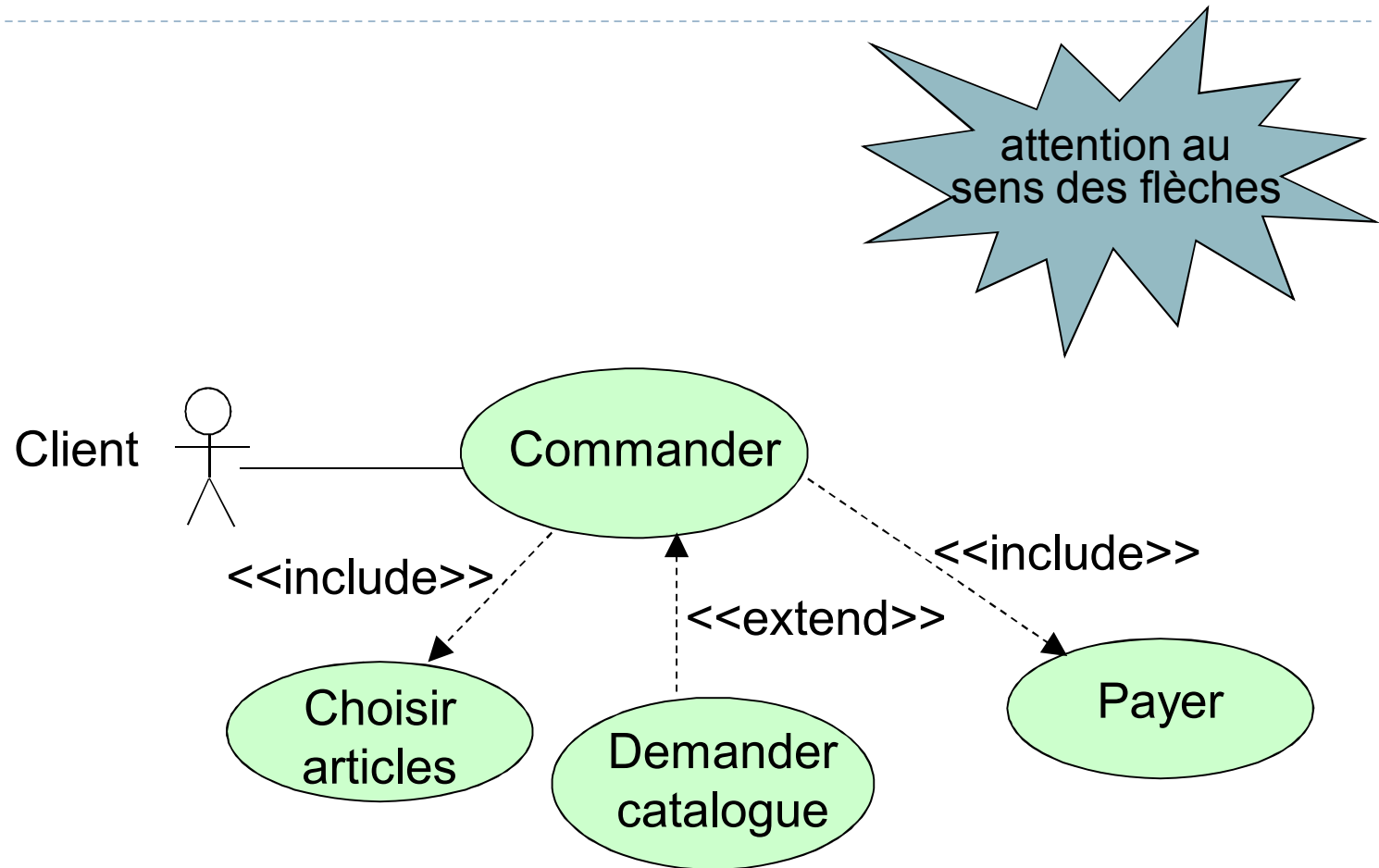
Point d'extension : Consulter solde

9. Le Client banque saisit le montant désiré du retrait.

10. Le GAB contrôle le montant demandé par rapport au solde hebdomadaire.

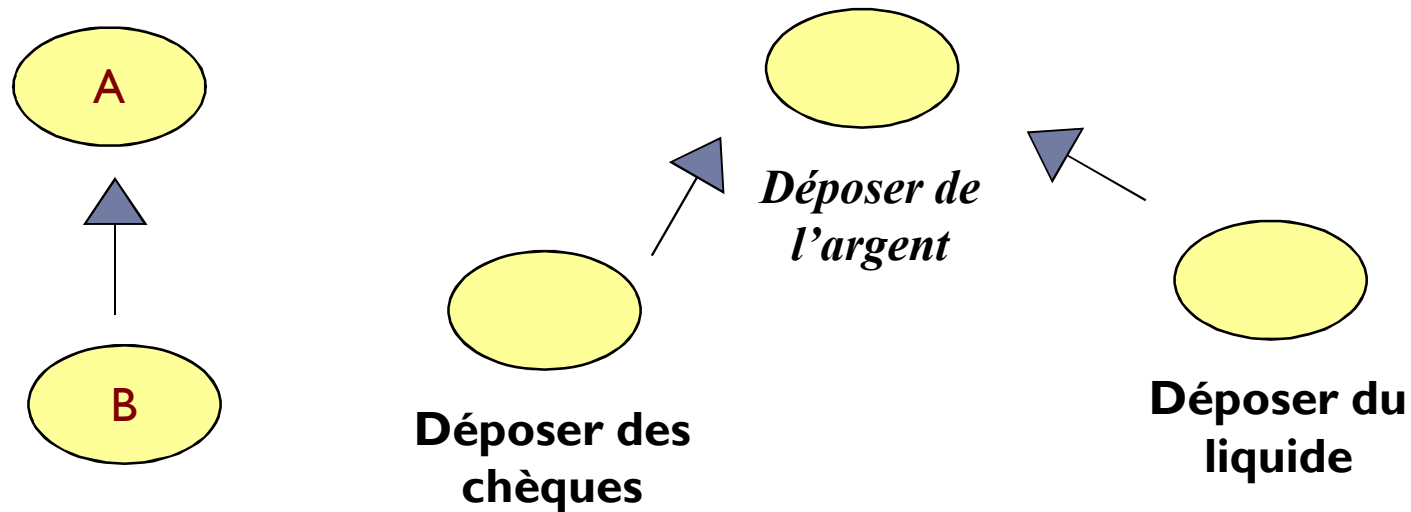
...

Relation entre cas d'utilisation



Relations entre cas d'utilisation: héritage

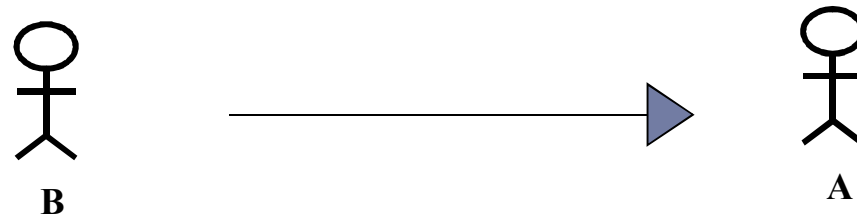
- ▶ Relation de **généralisation/spécialisation**
- ▶ Un cas A est une généralisation d'un cas B si B est un cas particulier de A



- ▶ Déposer de l'argent est cas d'utilisation généralisé. Il devient abstrait (italique) car il ne s'instancie pas directement, mais uniquement par le biais de l'un des 2 cas spécialisés

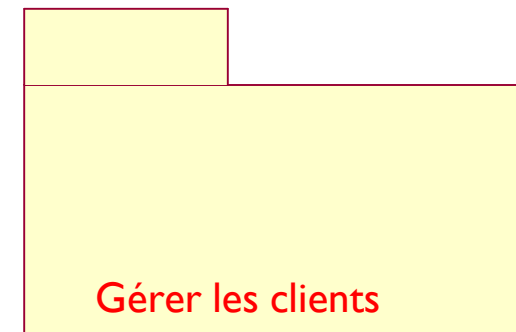
Relations entre acteurs: héritage

- ▶ Uniquement **relation de généralisation/spécialisation**
- ▶ A est une généralisation de B si tous les CU accessibles à A sont accessibles à B, mais pas l'inverse.



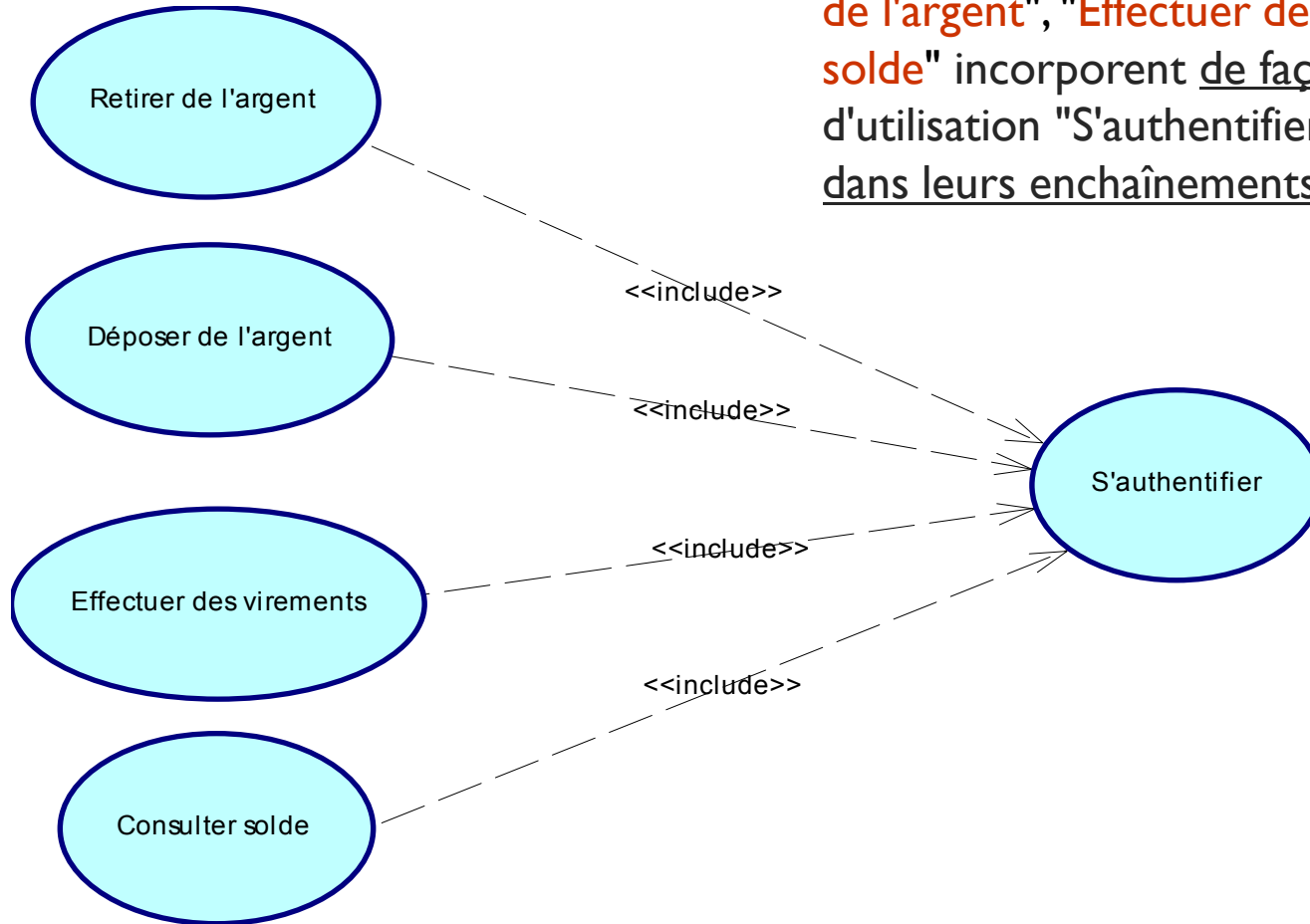
Structuration en package

- ▶ Si les diagrammes de cas d'utilisation sont nombreux
 - ▶ les regrouper **par acteur**
 - ▶ Opérations client
 - ▶ Opérations administrateur
 - ▶ etc.
 - ▶ les regrouper **par domaine fonctionnel**
 - ▶ Gérer les contrats
 - ▶ Gérer les clients
 - ▶ Etc.

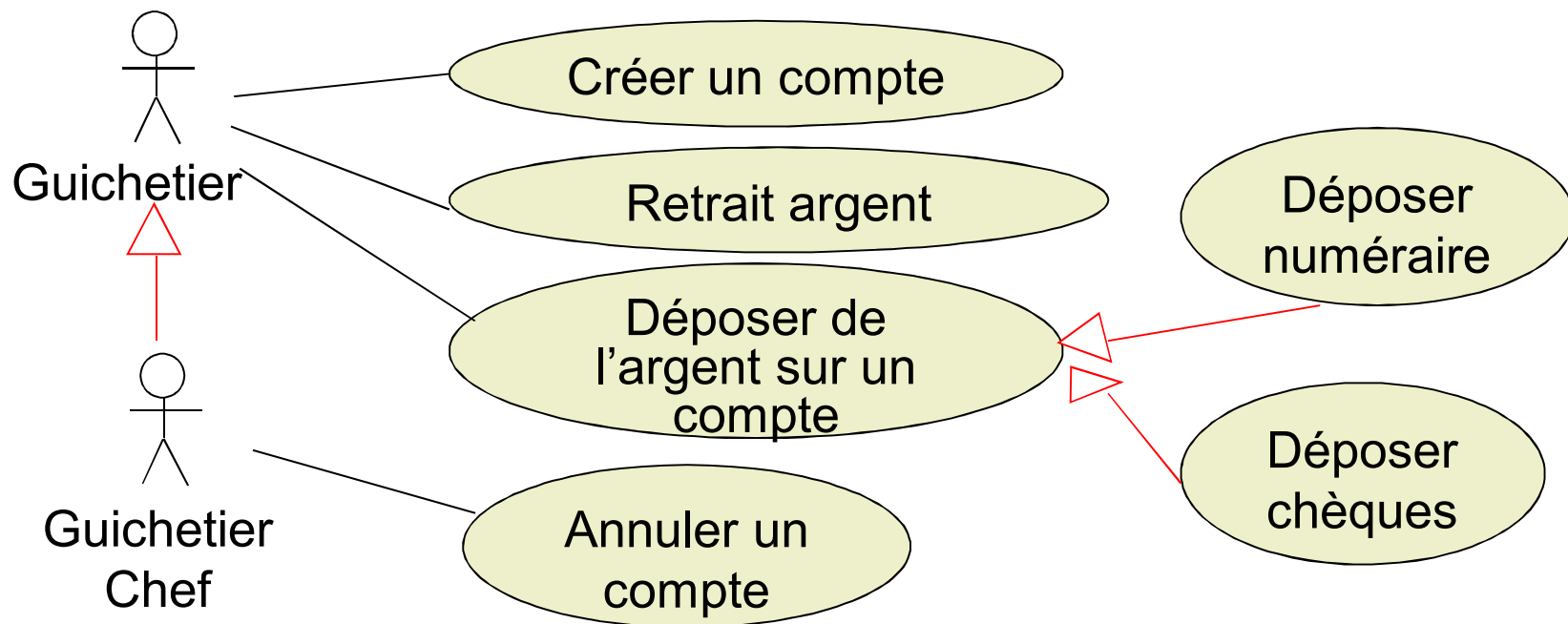


Exemple: Relation d'inclusion

Les cas d'utilisation "**Déposer de l'argent**", "**Retirer de l'argent**", "**Effectuer des virements**" et "**Consulter solde**" incorporent de façon explicite le cas d'utilisation "**S'authentifier**", à un endroit spécifié dans leurs enchaînements.



Exemple: Relation d'héritage



Un 'Guichetier Chef' est un 'Guichetier' spécialisé qui peut faire tout ce que peut faire un Guichetier et, en plus, il peut annuler un compte.

L'héritage simplifie le dessin (moins d'interactions à dessiner).

'Déposer chèques' et 'Déposer numéraire' sont 2 spécialisations de 'Déposer de l'argent sur un compte' (2 manières de faire).

Diagramme de cas d'utilisation

Il délimite **le domaine d'étude** en précisant:

- ❑ Ce qui est à la charge du système et
- ❑ En identifiant l'environnement extérieur au système étudié avec lequel ce dernier communique.



Méthodologie



Le processus unifié

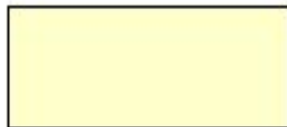
- (1) Définir le modèle de cas d'utilisation
 - (1) Introduire le système
 - (2) Trouver les acteurs
 - (3) Décrire brièvement chaque acteur
 - (4) Trouver les cas d'utilisation, exprimer les relations
 - (5) Décrire le modèle comme un tout
- (2) Définir les priorités entres CU
- (3) Détailler chaque CU (en fonction des priorités)

Description préliminaire du système

- **Choisir un identificateur**

- Baptiser le système, le plus tôt possible
- Risque d'être référencé dans toute la vie future de l'entreprise

- **Brève description textuelle** (quelques lignes max.)



CGDR24/7

Le système logiciel **CGDR24/7** ("Crédit Grenoblois Dans la Rue, 24h/24, 7j/7"), déployé sur un distributeur de billets de la gamme DB600, a pour but de contrôler l'ensemble des fonctions associées au distributeur en incluant son fonctionnement normal, mais aussi sa sécurité et sa maintenance.

Description des acteurs

- Pour chaque acteur :
 - choisir un identificateur représentatif de son rôle
 - donner une brève **description textuelle**



Guichetier

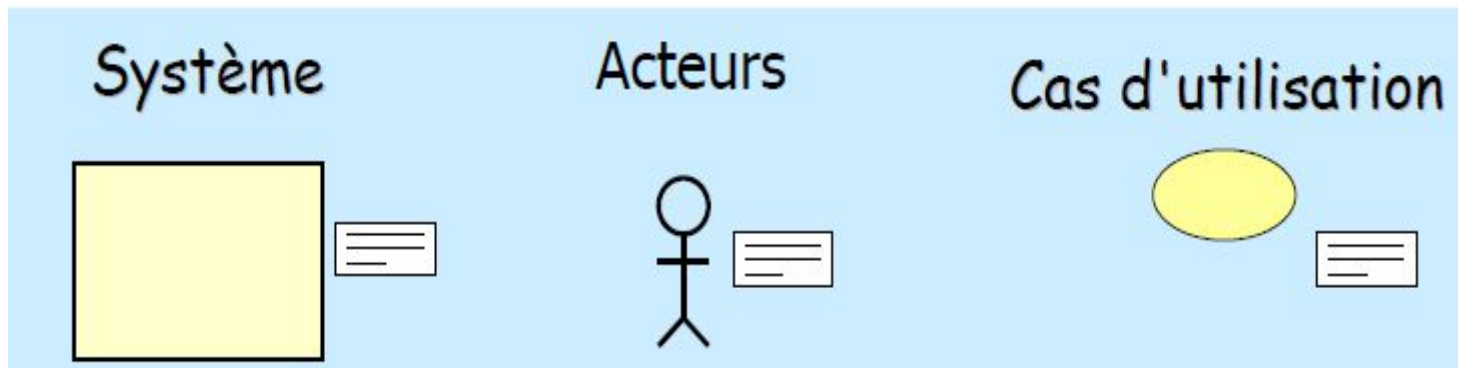
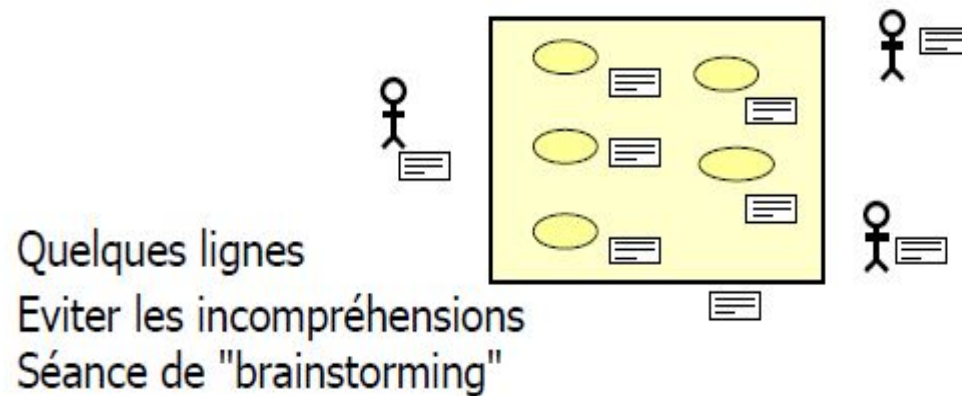
Un **guichetier** est un employé de la banque chargé de faire l'interface entre le système informatique et les clients qu'il reçoit au comptoir. Le guichetier peut réaliser les opérations courantes : création d'un compte, dépôt et retrait d'argent, etc.

- **identificateur représentatif** : *note de style* :
 - choisir une forme nominale décrivant un rôle
 - identification concise mais précise
 - terme provenant autant que possible du métier
 - utiliser par exemple le style MajMin

Description des Cas d'Utilisation

- **identificateur représentatif** : notes de style :
 - choisir une forme verbale décrivant une action
 - l'acteur est généralement le sujet
 - identification concise mais précise
 - éviter les connecteurs (et, ou, puis, ...)
 - terme provenant autant que possible du métier
 - utiliser par exemple le style MajMin
 - terme générique comme "Gérer" en cas de besoin
Gérer = Créer, Supprimer, Ajouter, Modifier, ...
Exemple: GérerLesDroits

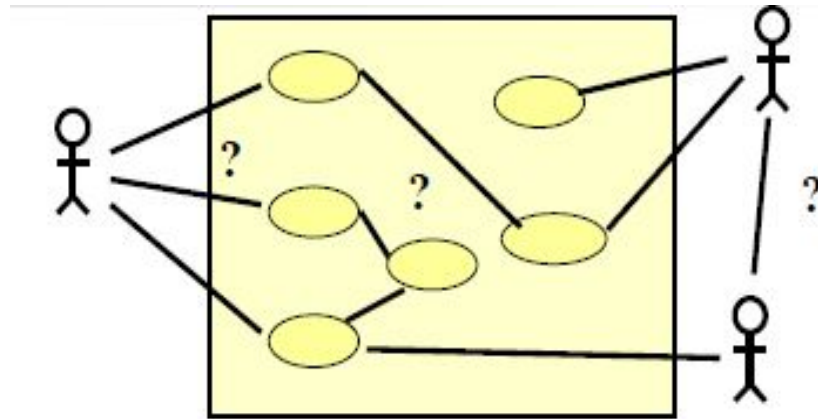
Description préliminaire de chaque élément



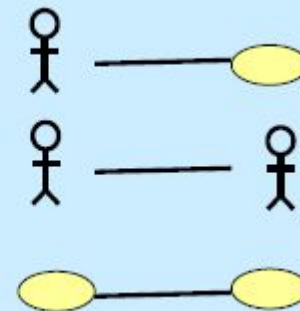
Le processus unifié

- (1) Définir le modèle de cas d'utilisation
 - (1) Introduire le système
 - (2) Trouver les acteurs
 - (3) Décrire brièvement chaque acteur
 - (4) Trouver les cas d'utilisation, exprimer les relations
 - (5) Décrire le modèle comme un tout
- (2) Définir les priorités entres CU
- (3) Détailler chaque CU (en fonction des priorités)

Relation entre éléments de base

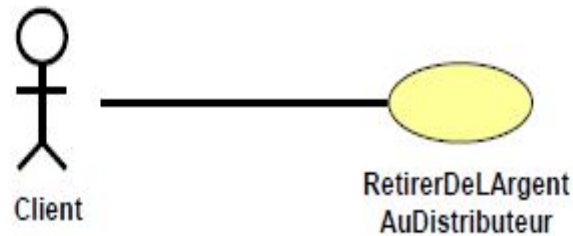


- Relations acteurs \leftrightarrow cas d'utilisation ?
- Relations acteurs \leftrightarrow acteurs ?
- Relations cas d'utilisation \leftrightarrow cas d'utilisation ?



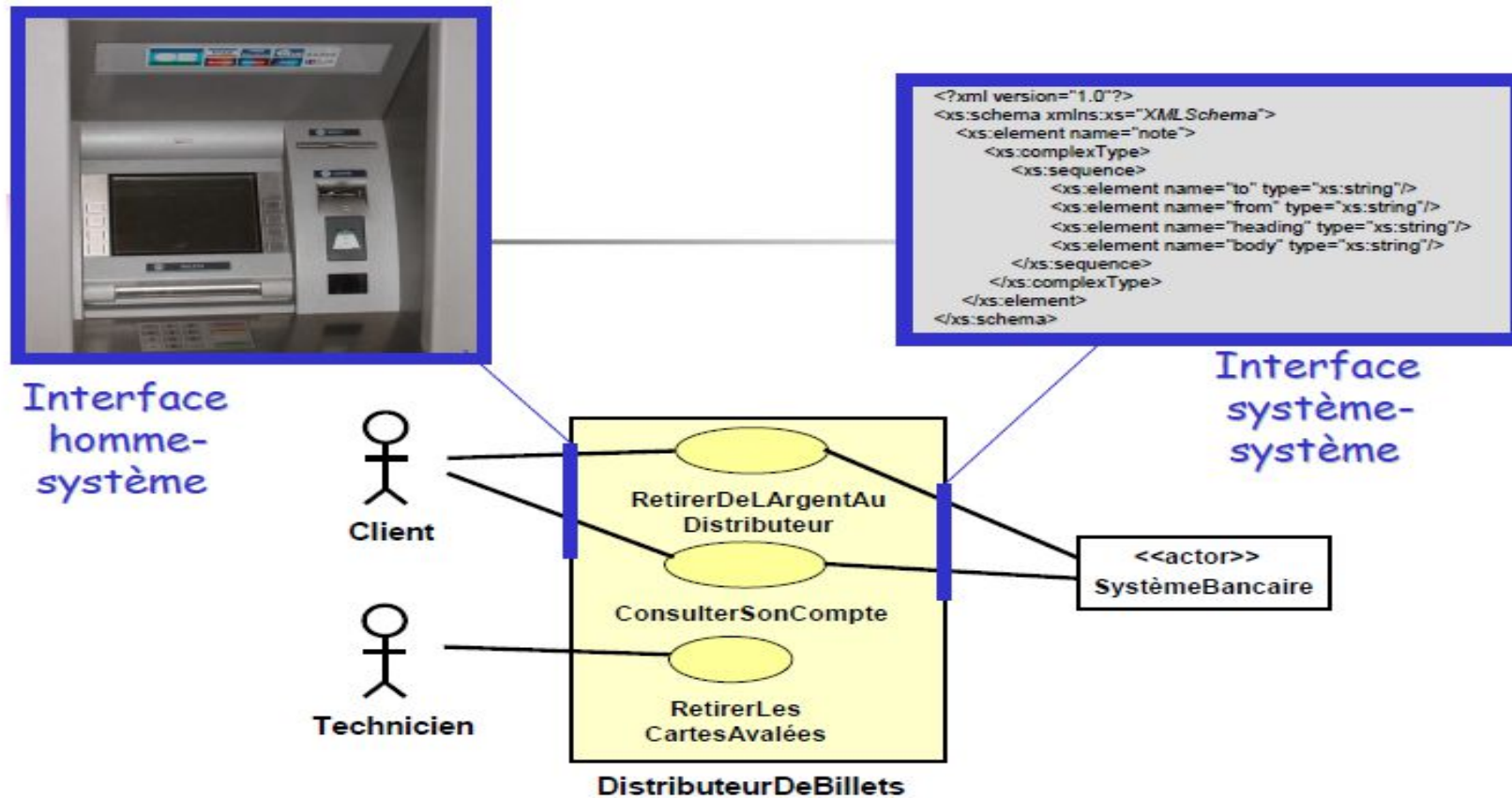
Relation acteur-Cas d'utilisation

- Échange de messages potentiellement dans les 2 sens



- Sera raffiné par la suite (spécifications externes)
 - Si l'acteur est un humain : interface homme – système
 - Si l'acteur est un logiciel : interface logicielle

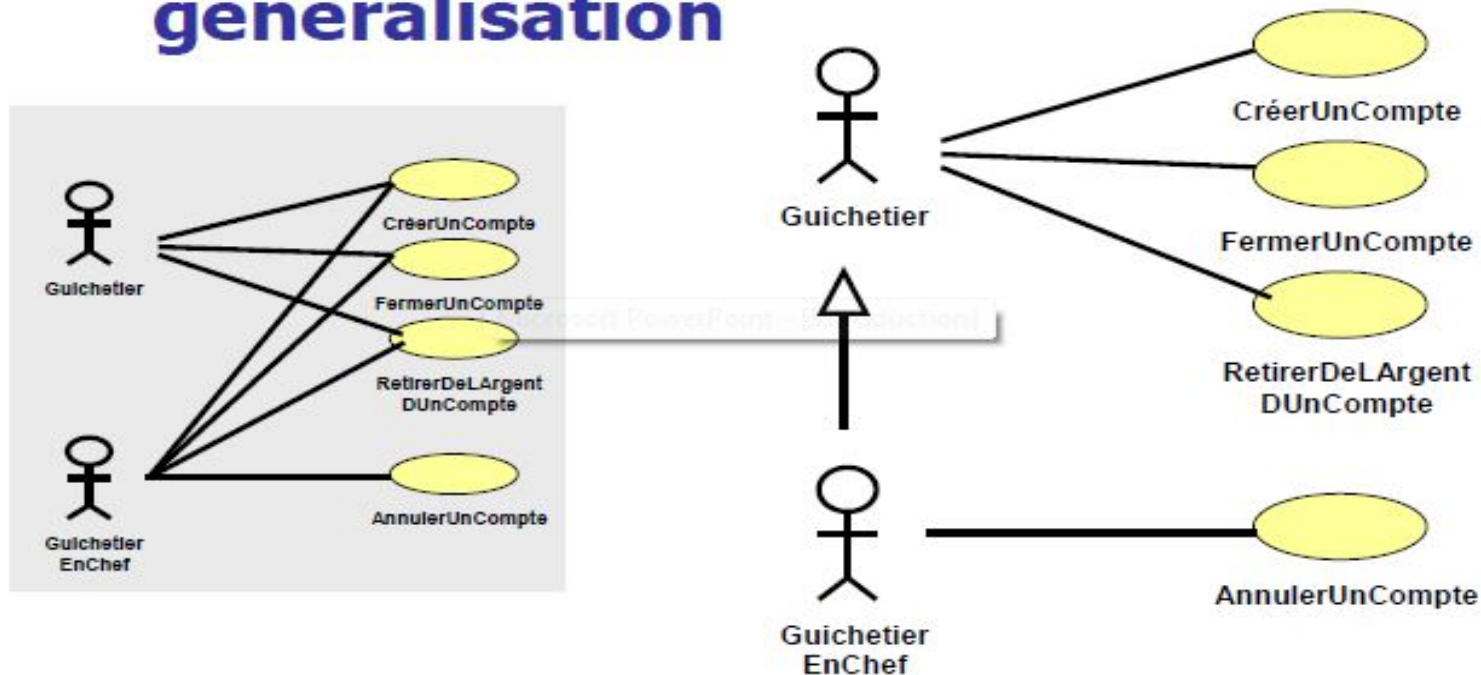
Spécifications: Interfaces



Description (par la suite) dans les documents de **spécification externes**

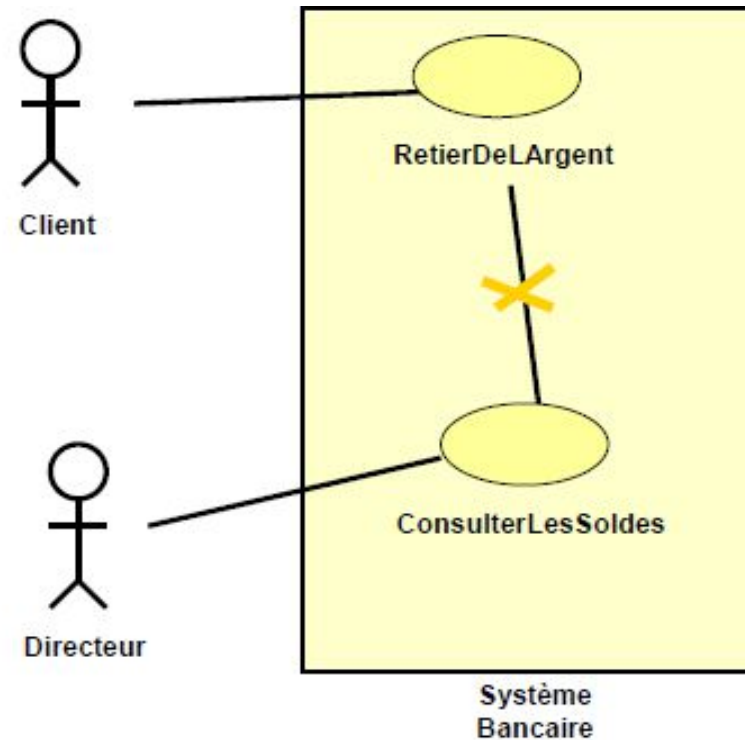
Relation entre acteurs

- La seule relation entre acteur est la **généralisation**



Communication entre CU

- Pas de communication entre cas d'utilisation
- UML se concentre sur les interactions système - extérieur



Le processus unifié

(1) Définir le modèle de cas d'utilisation

- (1) Introduire le système
- (2) Trouver les acteurs
- (3) Décrire brièvement chaque acteur
- (4) Trouver les cas d'utilisation, exprimer les relations
- (5) Décrire le modèle comme un tout

(2) Définir les priorités entres CU

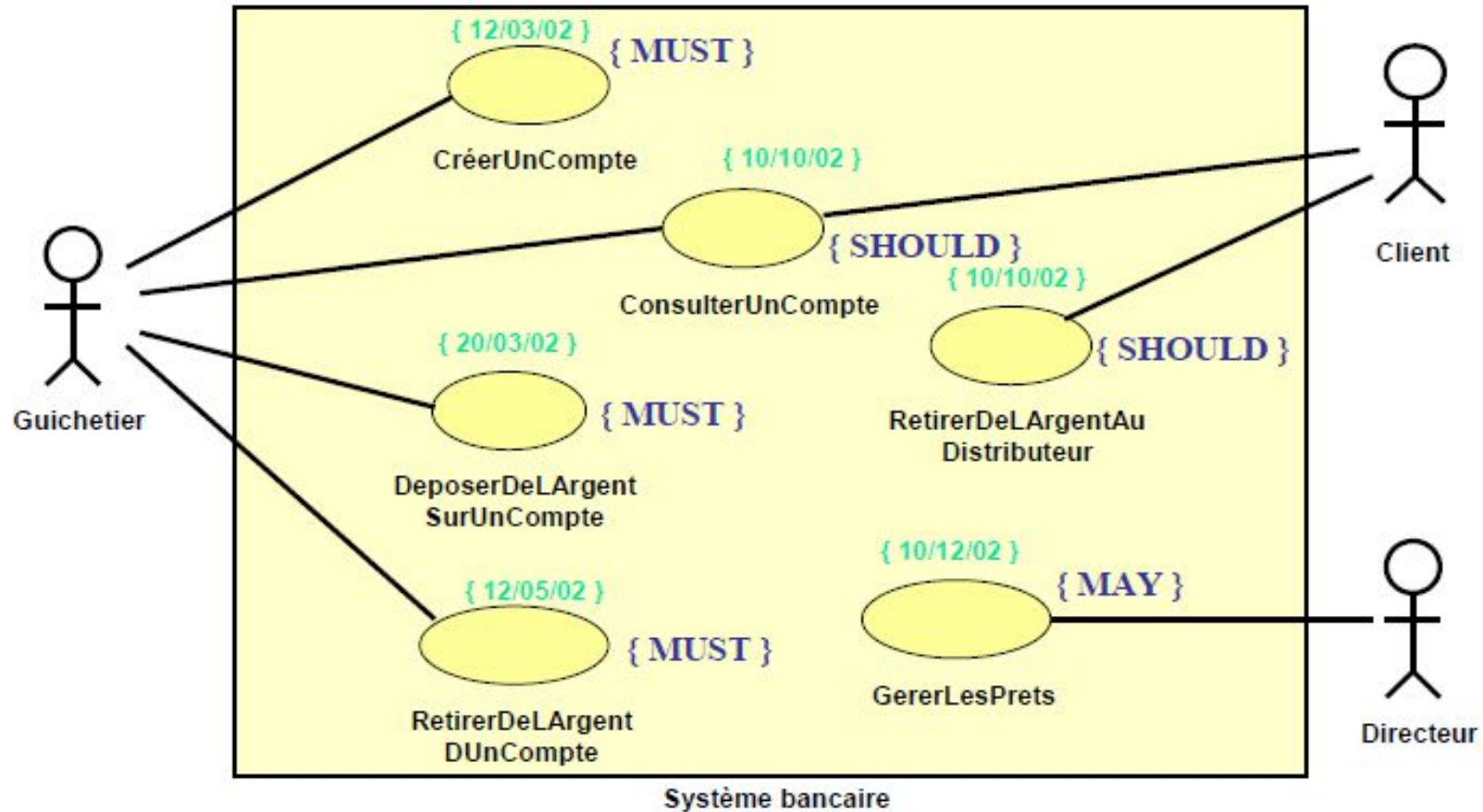
(3) Détailler chaque CU (en fonction des priorités)

Exemple 1

Se servir des cas d'utilisation pour définir les itérations :

Cas d'utilisation	Risque	Priorité	Itération
Traiter une commande	Moyen	Moyenne	5
Gérer les infos clients	Bas	Moyenne	6
Consulter les en-cours	Haut	Moyenne	3
Gérer la facturation	Bas	Basse	8
Suivre les règlements	Bas	Basse	8
Planifier une mission	Haut	Haute	2
Suivre une mission	Moyen	Haute	4
Réaliser l'inventaire	Bas	Moyenne	7
Manipuler les colis	Bas	Moyenne	7
Définir le plan de transport	Haut	Haute	1
Gérer les ressources	Moyen	Haute	3
Gérer les profits	Bas	Moyenne	9
S'authentifier	Bas	Moyenne	9

Exemple 2: DCU décoré



Le processus unifié

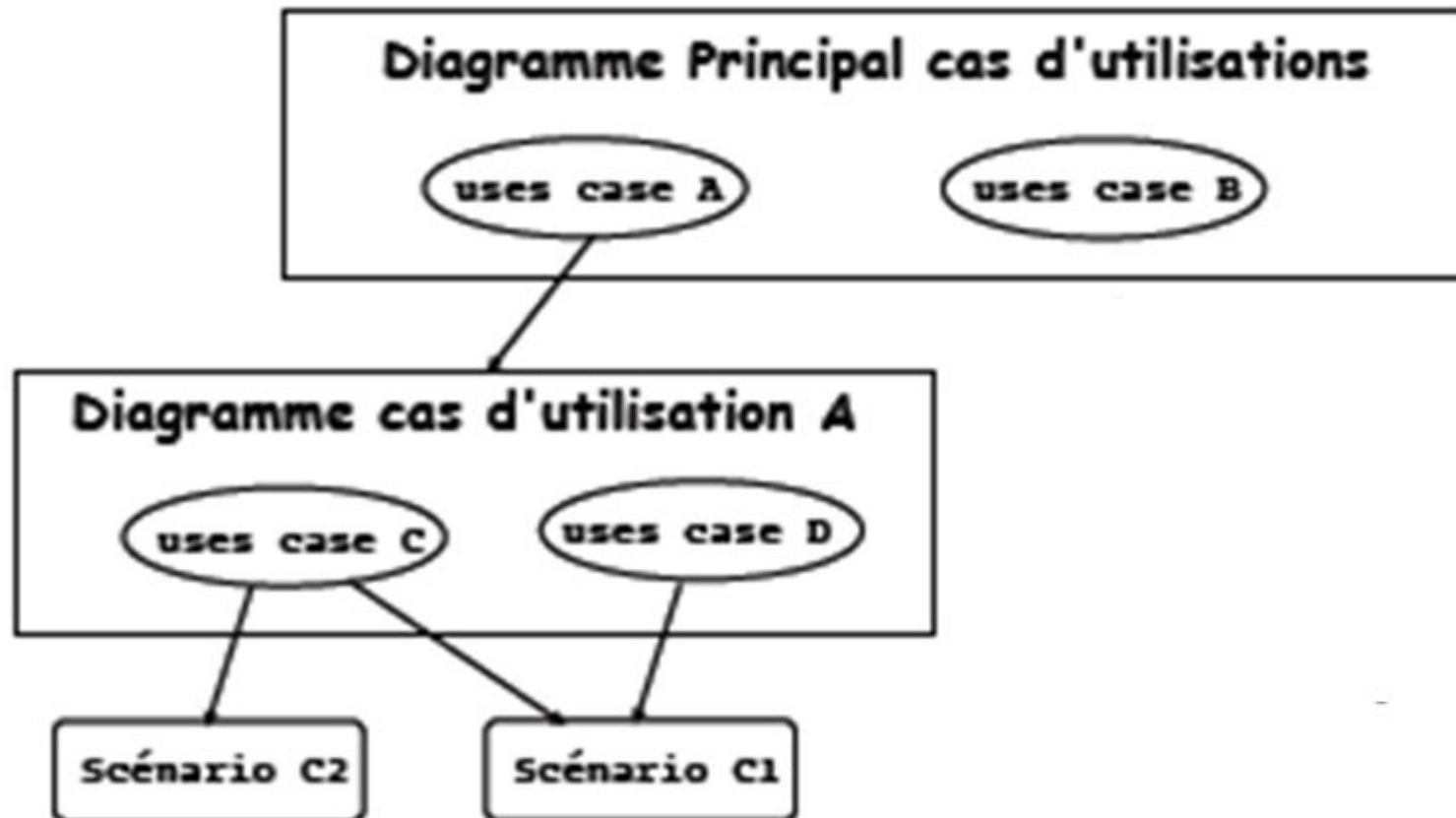
- (1) Définir le modèle de cas d'utilisation
 - (1) Introduire le système
 - (2) Trouver les acteurs
 - (3) Décrire brièvement chaque acteur
 - (4) Trouver les cas d'utilisation, exprimer les relations
 - (5) Décrire le modèle comme un tout
- (2) Définir les priorités entres CU
- (3) Détailler chaque CU (en fonction des priorités)

Niveau de raffinement des DCU

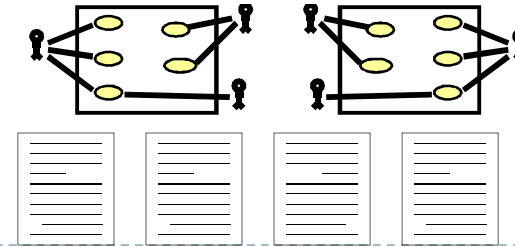
- ▶ Un DCU, comme tout diagramme UML, permet de décrire une réalité selon **différents niveaux de raffinement**.
- ▶ Il convient d'indiquer le « niveau d'abstraction »:
 - ▶ Niveau stratégique: Présente le contexte général, les grandes fonctions du système (processus métier): **DCU principal ou général**.
 - ▶ **Niveau raffiné**: Ce sont des cas d'utilisation qui participent au bon déroulement de cas d'utilisation: **DCU raffiné**

Diagramme de cas d'utilisation

- Un cas d'utilisation peut être **raffiné** par des diagrammes de cas d'utilisations, formant ainsi une **arborescence**.



Description du modèle de cas d'utilisation



- ▶ **Un modèle de cas d'utilisation** peut contenir
 - ▶ Plusieurs diagrammes
 - ▶ Plusieurs descriptions textuelles
- ▶ **Permet d'avoir une vision globale du système**
- ▶ Permet de définir des priorités entre CU
- ▶ **Utile pour la planification détaillée**

Description des cas d'utilisation

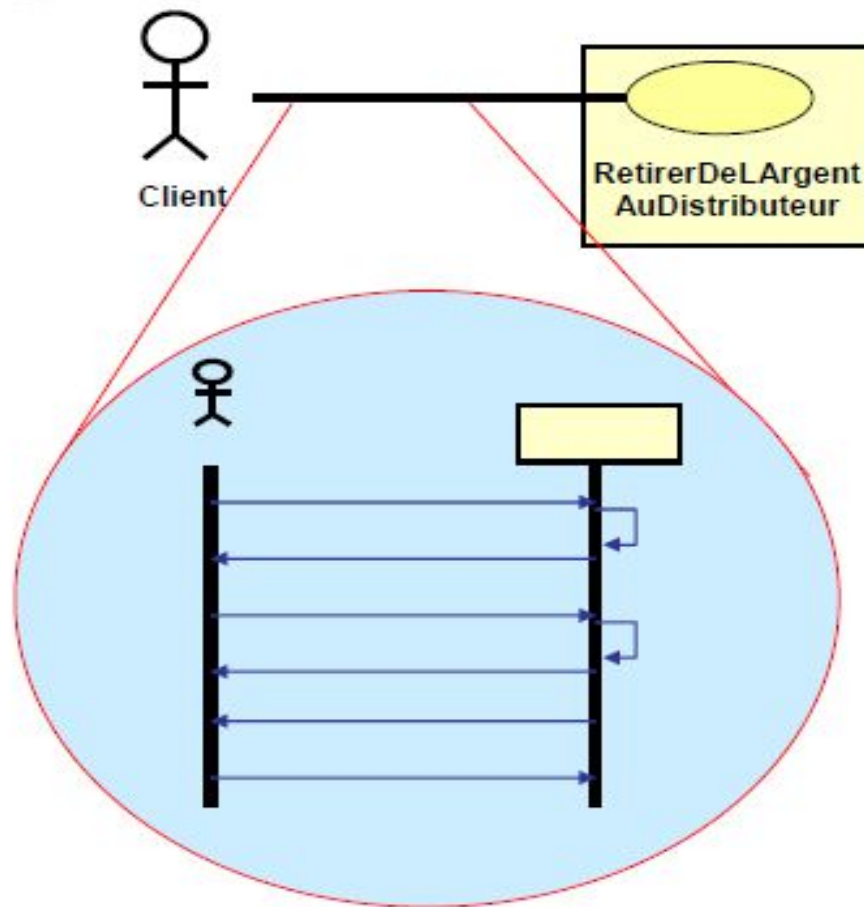
- ▶ Le diagramme de cas d'utilisation décrit les grandes fonctions d'un système du point de vue des acteurs.
- ▶ Mais il n'expose pas de façon détaillée le dialogue entre les acteurs et les cas d'utilisation.
- ▶ **Nécessité de décrire ce dialogue**

Description des cas d'utilisation

Deux façons sont couramment utilisées pour décrire les cas d'utilisation :

- ▶ Description textuelle,
- ▶ Description à l'aide d'un diagramme de séquence ou diagramme d'activité

Description de CU



Description via des
**diagrammes de
séquences**
ou **textuelle**

Description textuelle de CU

- ▶ Les cas d'utilisation sont développés **par du texte** dans **des scénarios** non représentés dans le diagramme.
- ▶ **CU = ensemble de scénarios.**
- ▶ Un scénario= une exécution particulière d'un CU.

Description textuelle de Cas d'utilisation

Les cas d'utilisation peuvent être spécifiés avec différents **niveaux de détail**:

- Format de **haut niveau**:
 - Abrégé, peu détaillé
- Format **étendu**:
 - Très détaillé

Description textuelle de haut niveau

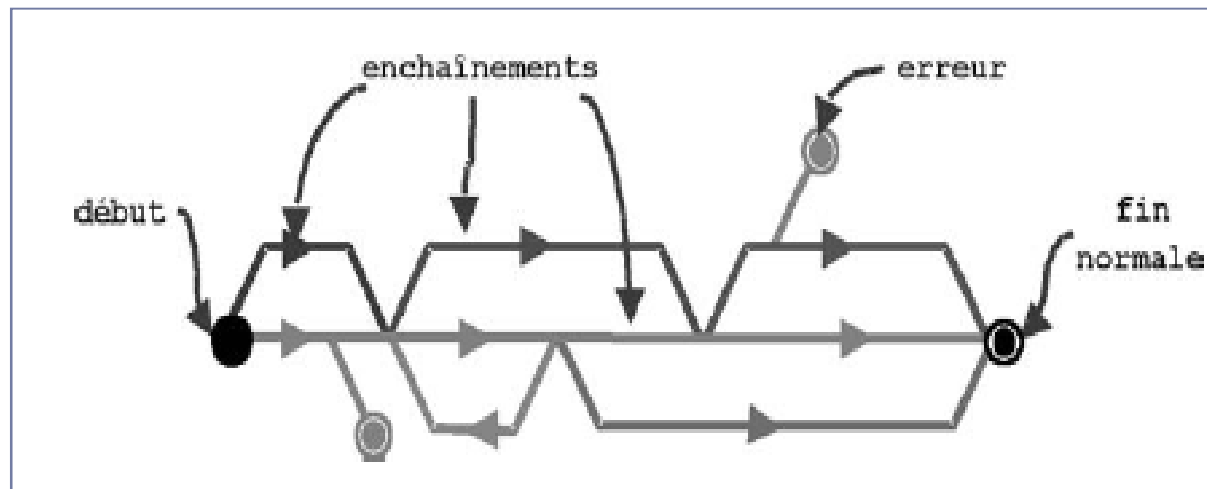
Un exemple de usecase pour le guichet automatique:

Cas d'utilisation:	Effectuer un retrait
Acteur(s):	Client
Type:	Primaire
Description:	Un client se présente au guichet automatique, s'identifie, retire un montant d'argent et quitte avec son argent.

Description textuelle étendue

1. Identification : Titre, Résumé, Acteurs.
2. Description des scénarios:
 1. Nominal,
 2. Alternatifs,
 3. D'erreur ou exception
 4. Préconditions + postconditions
3. Optionnel : **Exigences non-fonctionnelles** (fiabilité, performance, intégrité, ...) **et besoins IHM.**

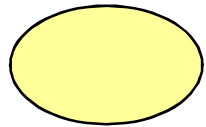
Scénarios



Description textuelle des cas d'utilisation

- ▶ **Les pré-conditions** : Conditions garantissant que le cas d'utilisation peut démarrer correctement. *Ex: Disposer d'un numéro de réservation pour acheter un ticket de cinéma via le distributeur (Ex4 du TD).*
- ▶ **Des scénarios** : On distingue
 - ▶ **le scénario nominal**, qui se déroule quand il n'y a pas d'erreur,
 - ▶ **les scénarios alternatifs** qui sont les variantes du scénario nominal
 - ▶ Branchement dans la séquence nominale (on y revient)
 - ▶ **les scénarios d'exception ou d'erreurs** qui décrivent les cas d'erreurs.
 - ▶ Le séquençement nominal s'interrompt, sans retour à la séquence nominale.
- ▶ **Les post-conditions** : Ensemble de conditions qui doivent être satisfaites à la fin du cas (scénario nominal) pour garantir que le système est dans un état cohérent.

Exemple de description détaillée d'un CU: sommaire d'identification



**Retirer
de l'argent**

Titre : Retirer de l'argent

Résumé : CU qui permet à un client de la banque de retirer de l'argent

Acteurs : client

Précondition :

Le distributeur contient des billets, il est en attente d'une opération, il n'est ni en panne, ni en maintenance

Début : lorsqu 'un client introduit sa carte bancaire dans le distributeur.

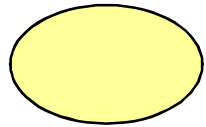
Fin : lorsque la carte bancaire et les billets sont sortis.

Postcondition :

Le GAB contient moins de billets qu'au début du CU.

Transaction de retrait enregistrée par le GAB.

Exemple de description détaillée d'un CU: description des scénarios



**Retirer
de l'argent**

Scénario nominal = déroulement normal :

- (1) le *client* introduit sa carte bancaire
- (2) le *système* lit la carte et vérifie si la carte est valide
- (3) le *système* demande au client de saisir son code
- (4) le *client* saisit son code confidentiel
- (5) le *système* vérifie que le code correspond à la carte
- (6) le *client* choisit une opération de retrait
- (7) le *système* demande le montant à retirer

...

Scénarios alternatifs

A1.Montant demandé supérieur au solde du compte (branchement à l'étape 7)

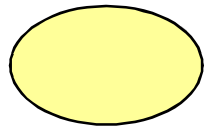
A2.Code erroné (pour la 1^{ère} ou 2^{ème} fois) (branchement à l'étape 3)

Scénarios d'erreurs (se terminent brutalement) :

E1.Carte invalide : au cours de l'étape (2) si la carte est jugée invalide, le système affiche un message d'erreur, rejette la carte et le cas d'utilisation se termine.

E2.Code erroné (pour la 3^{ème} fois) : au cours de l'étape (5) ...

Exemple de description détaillée d'un CU: exigences non-fonctionnelles



**Retirer
de l'argent**

Exigences non-fonctionnelles :

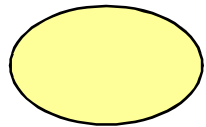
(A) *Performance* : le système doit réagir dans un délai inférieur à 4 secondes, quelle que soit l'action de l'utilisateur.

(B) *Résistance aux pannes* : si une coupure de courant ou une autre défaillance survient au cours du cas d'utilisation, la transaction sera annulée, l'argent ne sera pas distribué. Le système doit pouvoir redémarrer automatiquement dans un état cohérent et sans intervention humaine.

(C) *Résistance à la charge* : le système doit pouvoir gérer plus de 1000 retraits d'argent simultanément

...

Exemple de description détaillée d'un CU: besoins IHM

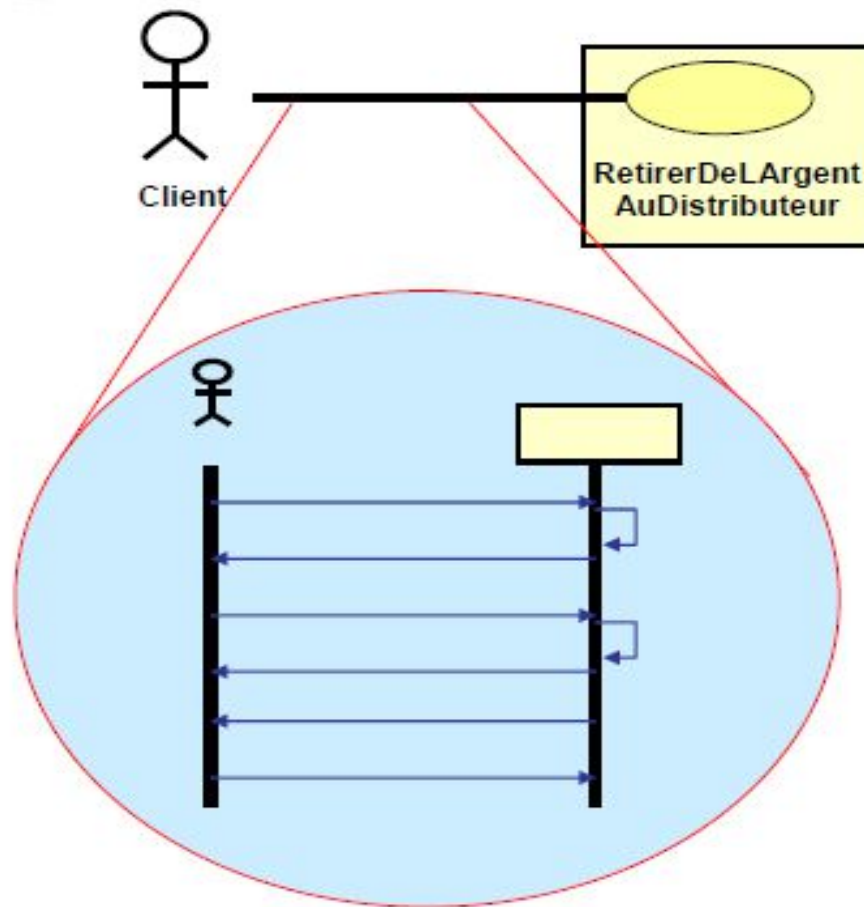


**Retirer
de l'argent**

Besoins d'IHM : dispositifs d'entrée-sortie à la disposition de l'acteur principal:

- Lecteur de carte bancaire
- Clavier numérique
- Ecran pour l'affichage des messages du GAB
- Touches autour de l'écran
- Distributeur de billets
- Distributeur de tickets

Description de CU



Description via des
**diagrammes de
séquences
ou textuelle**

Description graphique des scénarios

- ▶ Les scénarios peuvent être illustrés par des diagrammes de séquence ou diagramme d'activité.
- ▶ Ces diagrammes sont issus de l'axe dynamique d'UML

Diagramme de séquence

Intérêt : Déroulement chronologique d'un scénario

Entités : Les acteurs et le système

Messages : Echanges d'informations entre entités

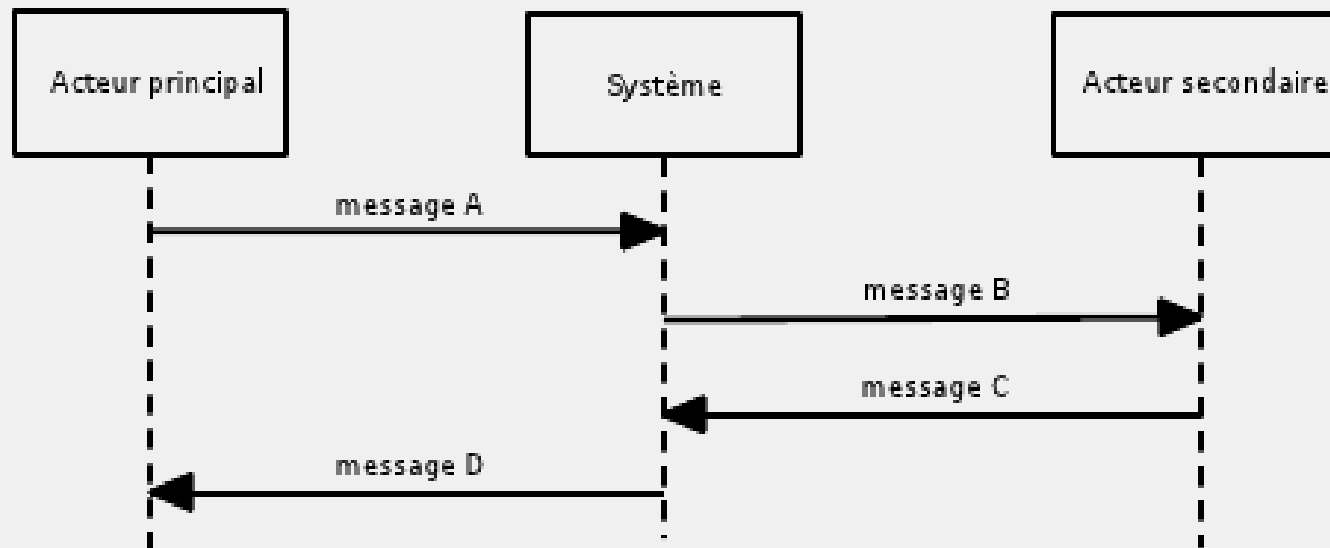
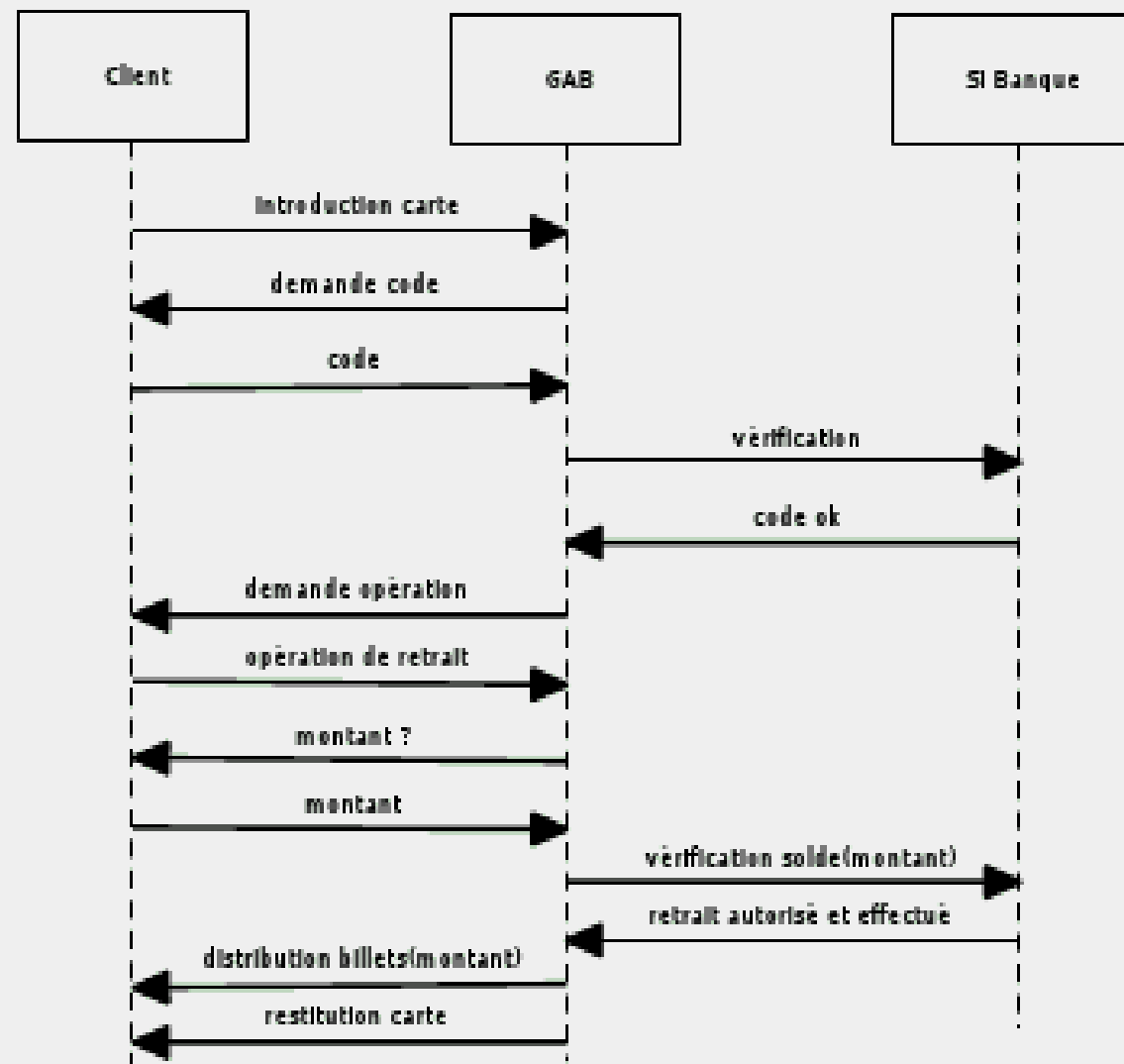


Diagramme de séquence « système »

- ▶ Un diagramme de séquence « système »
 - traite le système comme une « boîte noire »
 - représente les interactions entre les acteurs et le système
 - illustre la succession temporelle des événements qui influencent le fonctionnement du système.

Diagramme de séquence

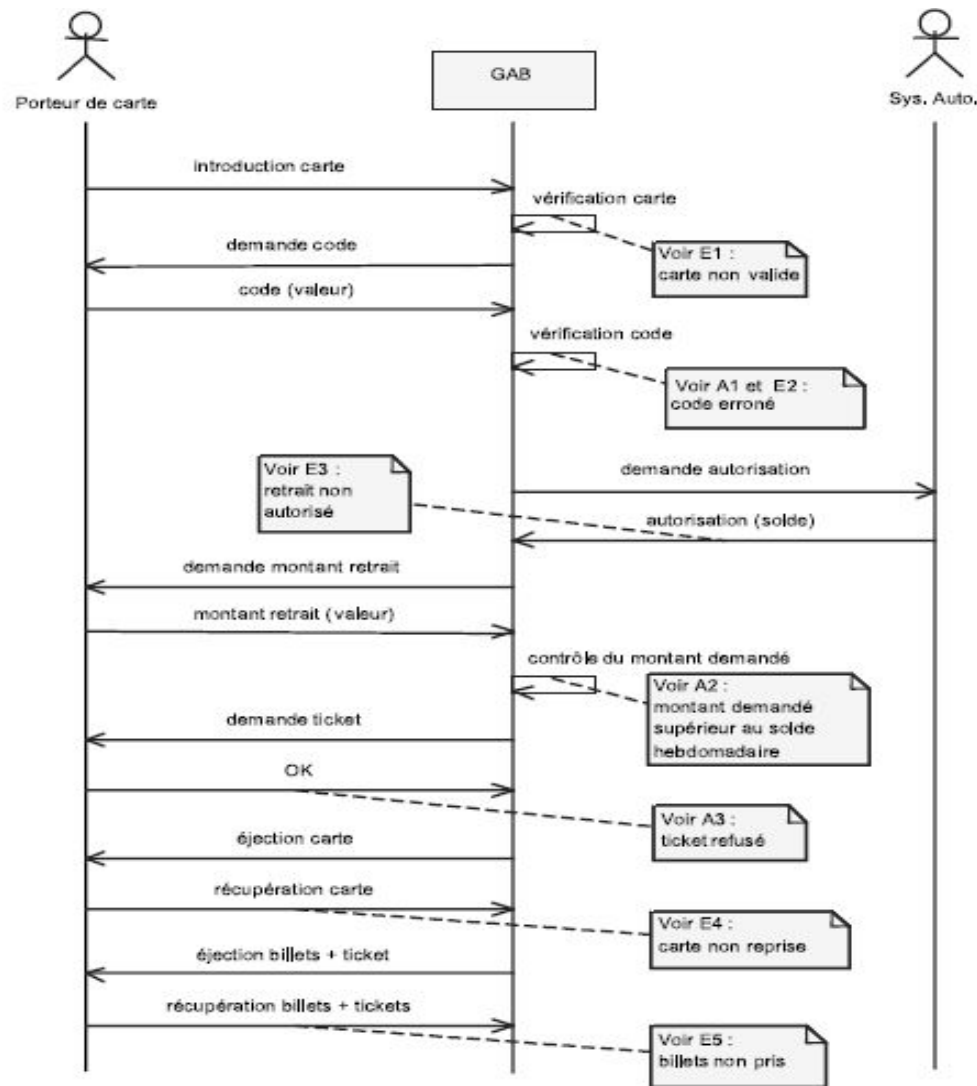
Une possibilité :



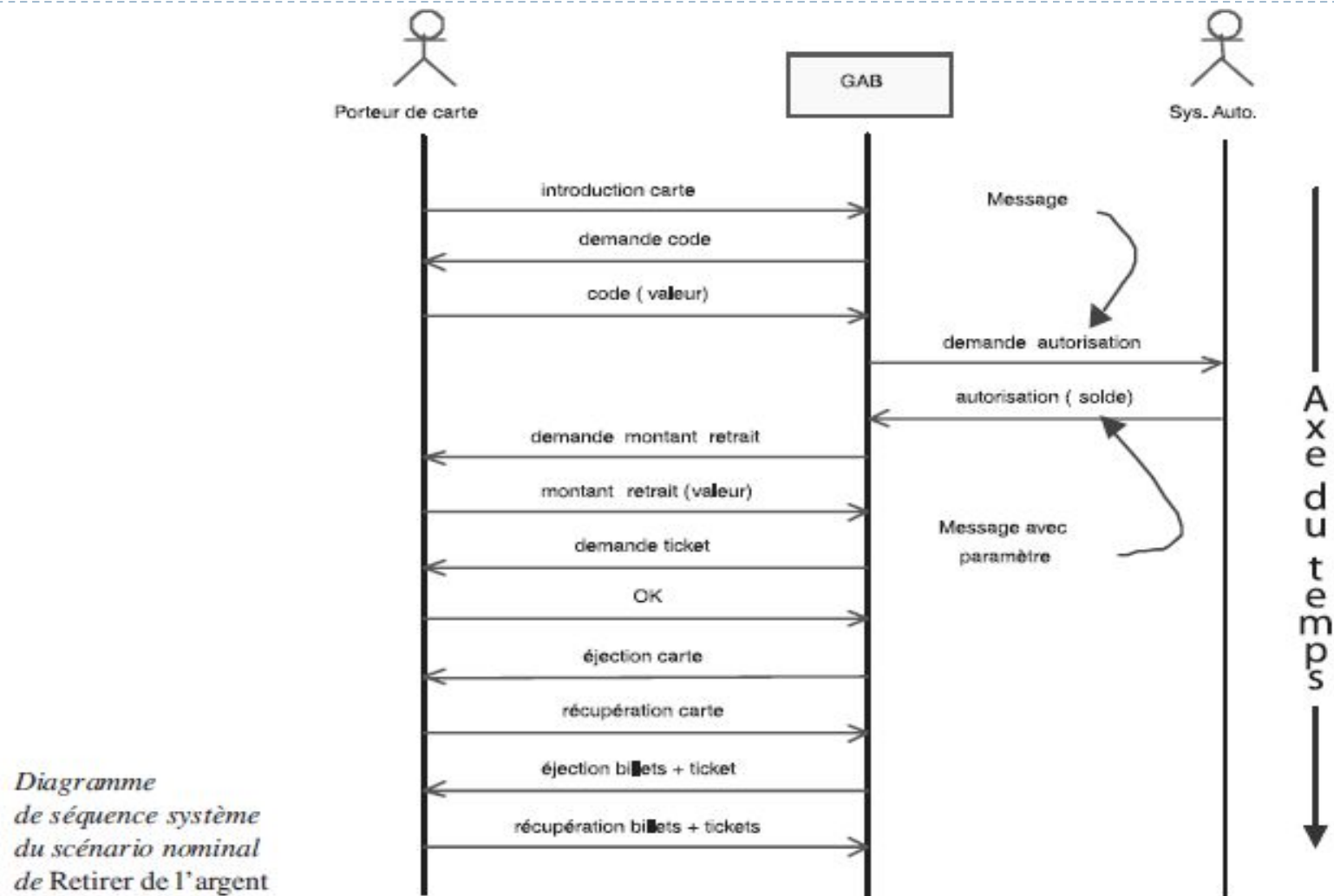
Est-ce que ce diagramme de séquence est correct?

Version 1 du diagramme de séquence système

*Diagramme de séquence
système enrichi du
scénario nominal de
Retirer de l'argent*



Version 2 du diagramme de séquence système



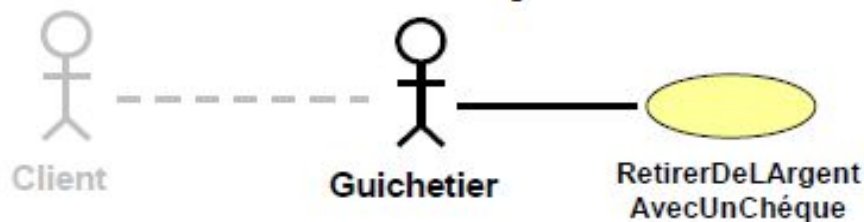


Problèmes récurrents

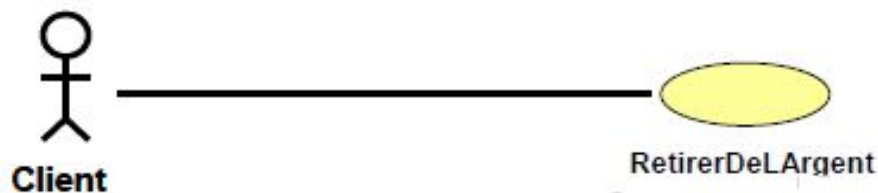


Problème des intermédiaires

- Représentation des intermédiaires entre le système et l'intéressé ?
- **Différents points de vue**

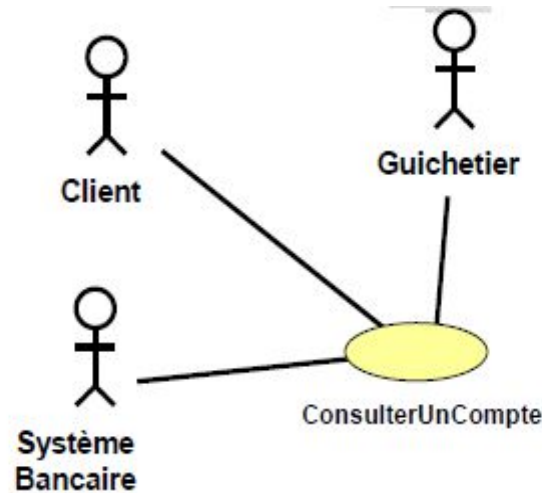


On insiste sur le lien de communication, l'échange de messages et l'interface



On insiste sur les objectifs et on masque complètement les aspects liés à l'interface

Une notation peu informative



L'association "communique" est peu informative :

qui réalise le cas d'utilisation ? qui collabore à son déroulement ?

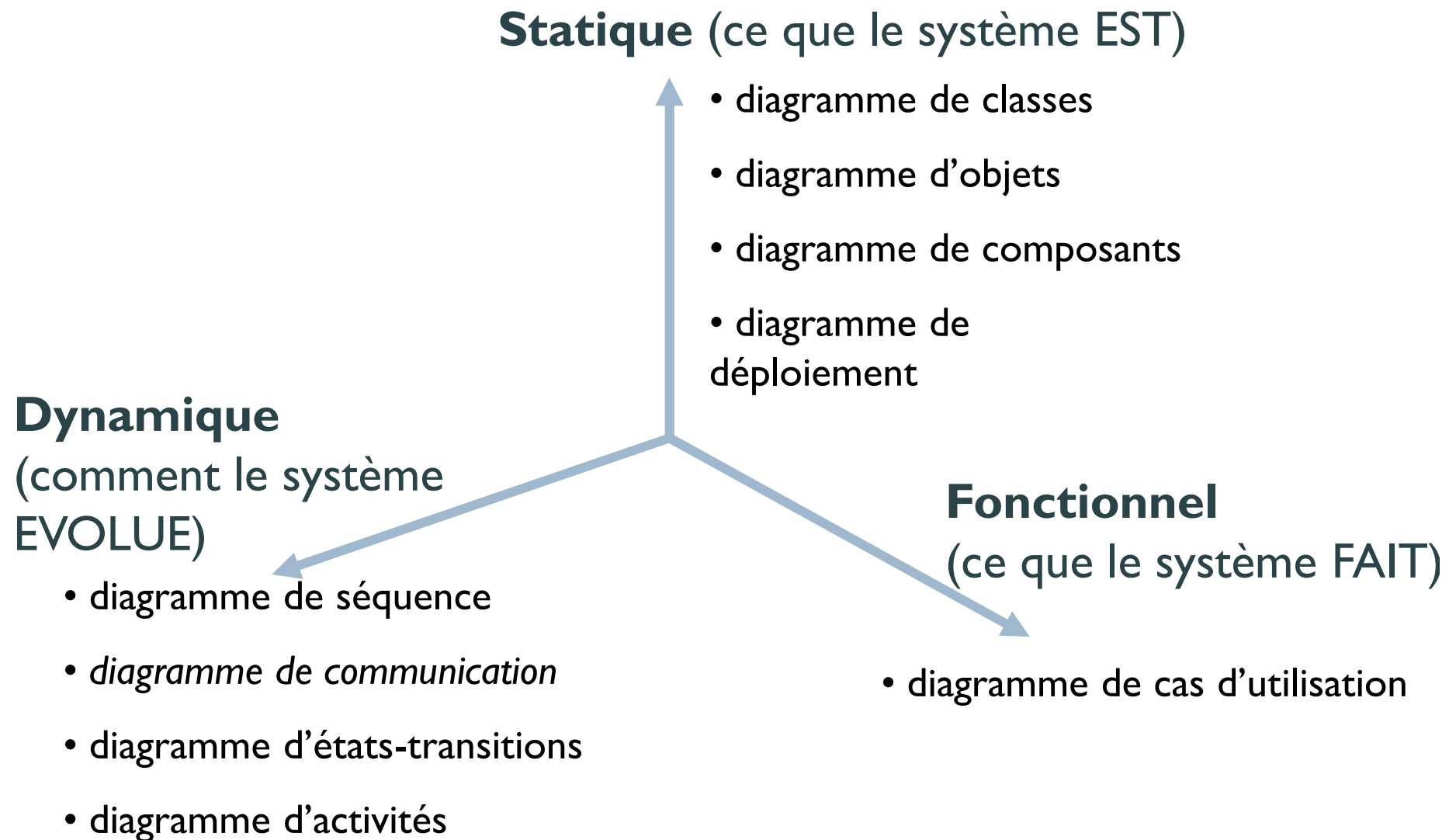
quels acteurs peuvent participer à un même scénario simultanément ?

Pas de notation standard pour exprimer les réponses

On peut ajouter des notes textuelles pour lever toute ambiguïté

Chapitre 3: Diagrammes statiques

3 Axes de modélisation d'un système



Les activités du projet

- ▶ Etude préalable : Analyse du contexte
- ▶ Analyse et spécification des besoins
- ▶ Analyse du problème ou métier ou système
- ▶ Conception de la solution

Analyse métier

- ❑ Les besoins sont analysés selon le point de vue des développeurs (et non pas celui du client).
- ❑ L'analyse se focalise sur ce qu'on doit réaliser (QUOI).
- ❑ L'analyse permet d'obtenir une vue logique du système.
- ❑ Modéliser le système afin de le rendre plus compréhensible.

UML

- Point de vue fonctionnel
- Point de vue Statique:
 - Diagrammes de classe
- Point de vue dynamique

Modélisation O.O: Les objets

- ▶ Les objets informatiques définissent une représentation abstraite des entités du monde réel.
- ▶ Un objet: Une entité concrète ou une entité virtuelle.
- ▶ Une abstraction met en avant les caractéristiques essentielles et qui dissimulent les détails.
- ▶ Une abstraction se définit par rapport à un point de vue.

Exemples de caractéristiques

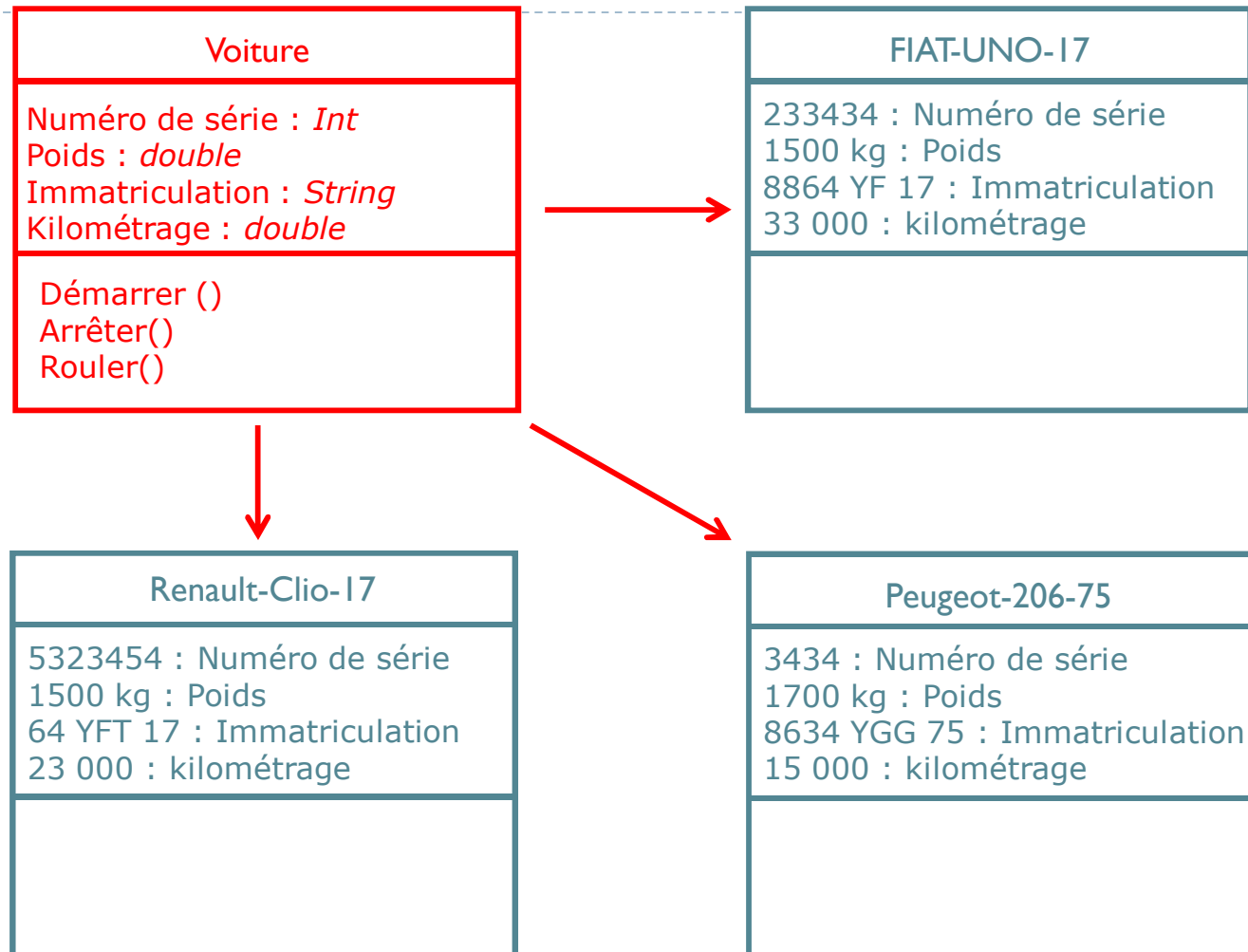


- Pour une voiture, **les caractéristiques** sont: la marque, la puissance, la couleur, le nombre de places assises, ...
- **Plusieurs points de vue possibles:** propriétaire, mécanicien, vendeur, ...

Les Classes

- ▶ Classe d'objets
- ▶ La **classe** sert à regrouper sous un même terme générique les objets partageant les mêmes caractéristiques et le même comportement.
- ▶ On dit qu'un **objet** est une instance d'une classe.

Modélisation objet



Modélisation objet

Une classe est composée:

➤ attributs

données dont les valeurs représentent l'état de l'objet

➤ méthodes ou opérations

opérations applicables aux objets



NB. UML fait la distinction entre opération et méthode.

Une méthode est une implémentation d'une opération. Plusieurs méthodes peuvent donc implémenter une même opération...

Les classes

Nom de classe

Nom de classe
Attributs

Nom de classe
Attributs
Opérations

Nom de classe

Diagramme de classe: relations

- ▶ Les relations inter-classe qui sont couramment utilisées sont :
 - ▶ Association
 - ▶ Héritage
 - ▶ Agrégation
 - ▶ Composition

Diagramme de classe: Associations

- ▶ Une association indique que deux classes communiquent entre elles (dans un sens ou dans les deux sens).
- ▶ Une association ne sous-entend aucune hiérarchie dans la relation.

Diagramme de classe: Associations

- Représentation UML : trait reliant les deux classes en relation.



« Tout client a toujours 0 ou plusieurs comptes »
« Tout compte a toujours 1 et 1 seul titulaire »

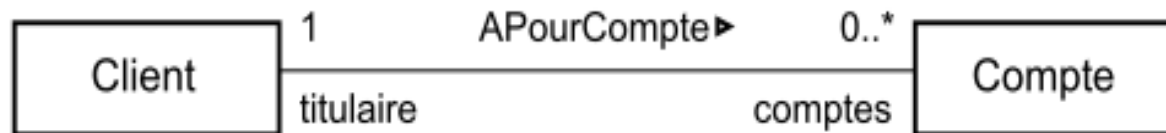
Diagramme de classe: Associations

Cardinalités d'une association

Précise combien d'objets peuvent être liés à un seul objet source

Cardinalité minimale et cardinalité maximale ($C_{min}..C_{max}$)

Doivent être des constantes



« Tout client a toujours 0 ou plusieurs comptes »

« Tout compte a toujours 1 et 1 seul titulaire »

Cardinalité ou multiplicité des associations

Notation abrégée des multiplicités :

- 1** \Leftrightarrow 1..1 (exactement 1)
- *** \Leftrightarrow 0..* (0 ou plusieurs)
- n** \Leftrightarrow n .. n (exactement n)
- 1..*** \Leftrightarrow 1 ou plusieurs (1 ou plus)
- 0..1** \Leftrightarrow 0 ou 1 (au plus un)
- 1..100** \Leftrightarrow entre 1 et 100
- 2,4,5** \Leftrightarrow 2, 4 ou 5

Diagramme de classe: Associations

Association : nommage

- Les associations peuvent être nommées c.-à-d. identifiées par un texte unique décrivant la sémantique de l'association.



- Note : utiliser une **forme verbale** active (“travaille pour”) ou passive (“employé par”).
- Si ambiguïté, indiquer le sens de lecture avec les signes ◀ ou ▶ (par défaut lecture de gauche à droite).



Diagramme de classe: Associations

Association : rôles des extrémités

- Les extrémités des associations peuvent être qualifiées par des rôles.
- Un rôle indique comment une classe *Source* voit une classe *Destination*.



ASSOCIATION : caractéristiques

- ▶ Un objet (dit *objet utilisateur*) peut utiliser les **services** d'un autre objet (dit *objet utilisé*) selon le sens de navigation indiqué par l'association.
- ▶ Pour utiliser un service d'un objet, l'objet utilisateur envoie un **message** à l'objet utilisé.

Association à navigabilité restreinte

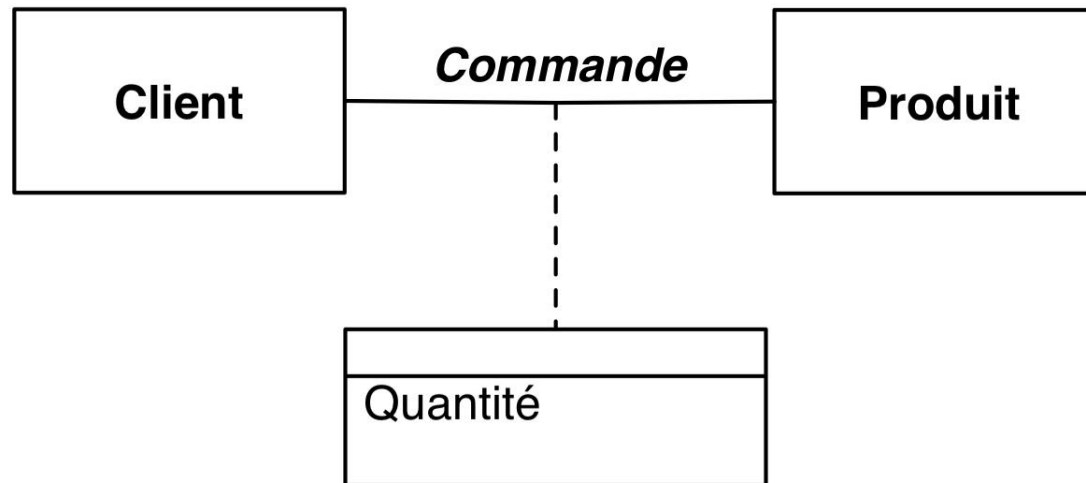
- Par défaut, une association est **navigable dans les deux sens**.
- Indique qu'un objet de la classe cible peut être atteint par un objet de la classe source au travers de l'association.
- Si la relation est entre les classes A et B et que seulement B est navigable, **alors on pourra accéder à B à partir de A mais pas vice versa**.
- **Exemple:**



- *On doit être en mesure de savoir le client qui a fait la commande et non toutes les commandes d'un client.*
- **La classe commande doit avoir un champ faisant référence à la classe client.**

Classe association

- Une association peut avoir des attributs = **classe-association**



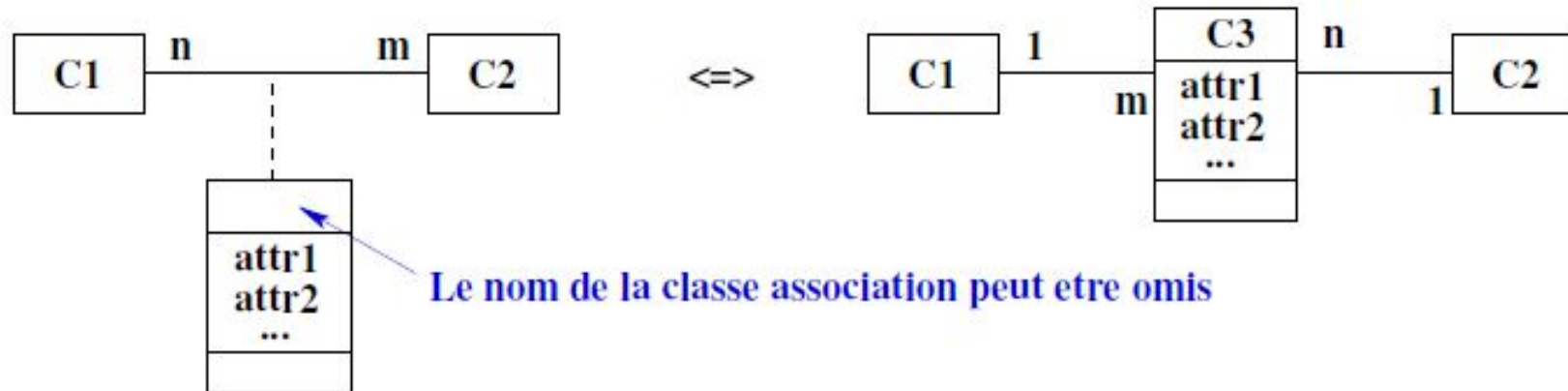
- Les classes association sont utiles quand il y a **des attributs qui sont pertinents à l'association, mais à aucune des classes impliquées.**

Classe association

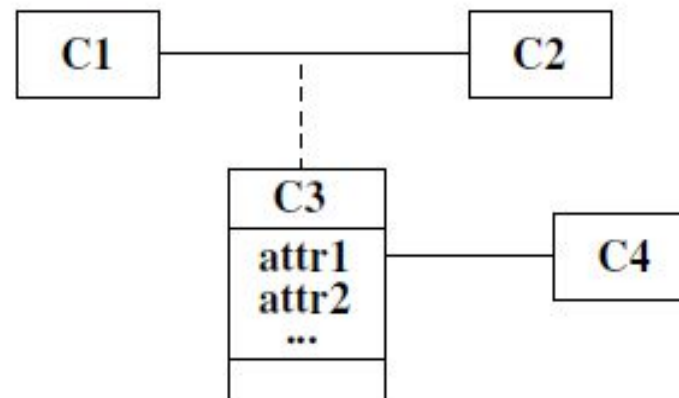
- ▶ Ce concept UML avancé *n'existe pas dans les langages de programmation objet*, il faut donc le traduire en le transformant en classe normale,

Classes-associations

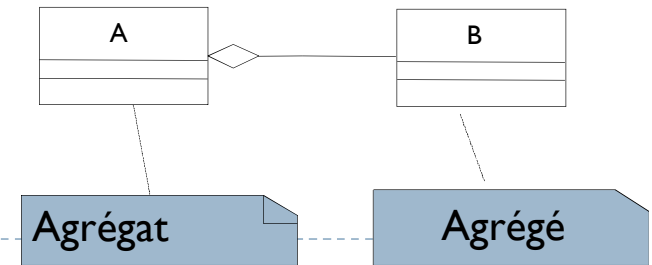
Association attribuée :



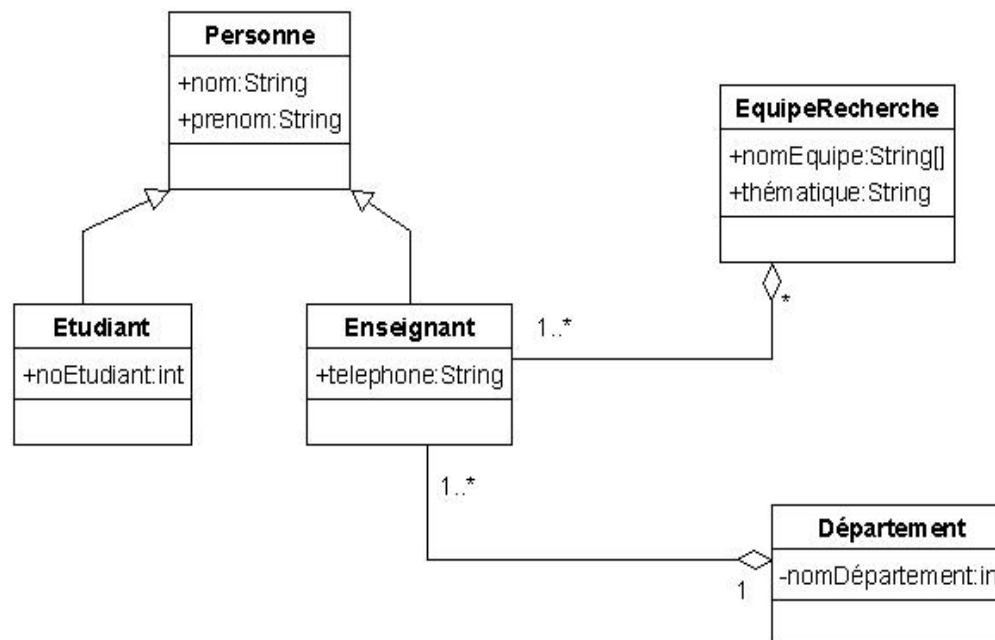
Une classe association peut participer à d'autres associations :



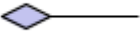
Relation d'Agrégation

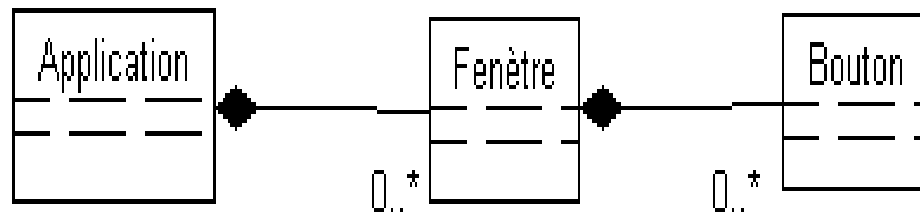


- ▶ L'agrégation peut être assimilée à une appartenance - faible -.
- ▶ La suppression de A n'implique pas la suppression de B.
- ▶ L'élément agrégé peut être partagé.
- ▶ Représente une association asymétrique: une des classes joue le rôle d'agrégat et l'autre classe joue le rôle d'agrégé.
- ▶ Cardinalité quelconque.



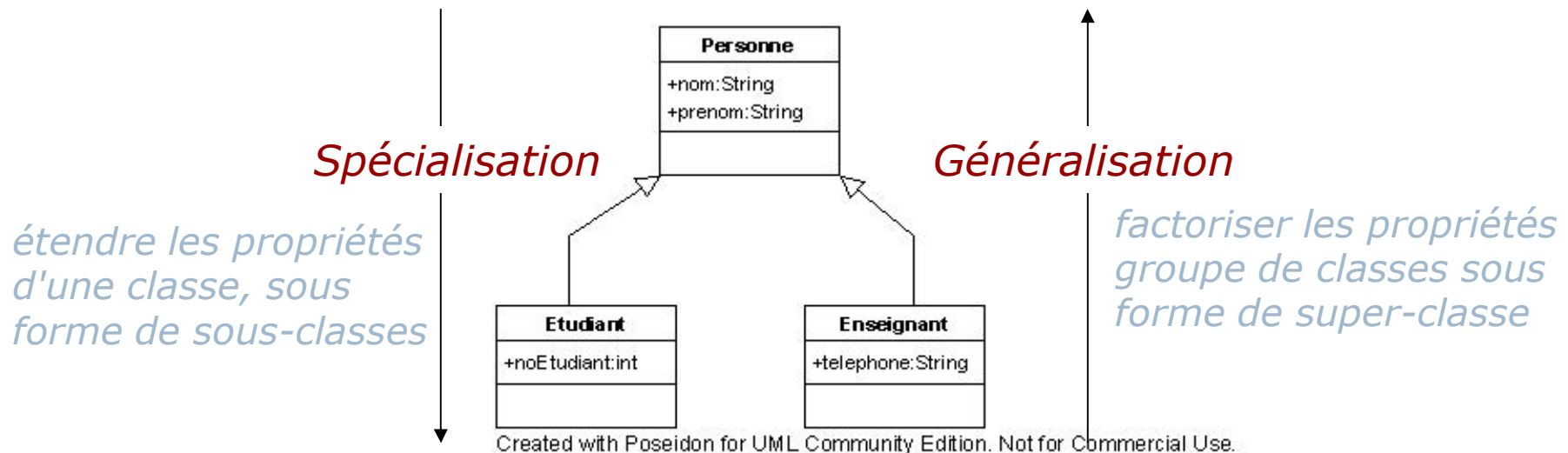
Relation de Composition

- Il s'agit d'une **appartenance forte**. **La vie de l'objet composant est liée a celle de son composé**. Aussi appelée « **agrégation forte** ».
- Une relation de composition est indiquée par une ligne avec un losange rempli. 
- Les cycles de vies **des composants** et de **l'agrégat** sont liés : si l'agrégat est détruit , ses composants le sont aussi.
- A un même moment, une instance de composant **ne peut être liée qu'à un seul agrégat**.
- la non-présence des valeurs de multiplicités est synonyme de 1..1.



Relation d'Héritage

- Principe: hiérarchies de classes
- permet de gérer **la complexité**, en ordonnant les objets au sein d'arborescences de classes,
- Elle présente une classe spécifique comme descendante d'une classe plus générique.

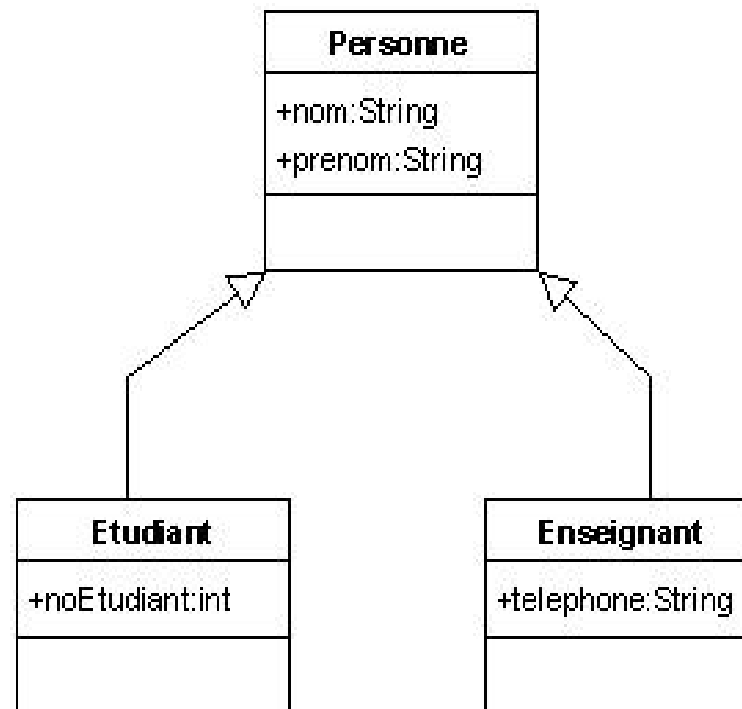


En UML, un héritage est représenté par une ligne en tirets, terminée par une flèche évidée.

Relation d'Héritage

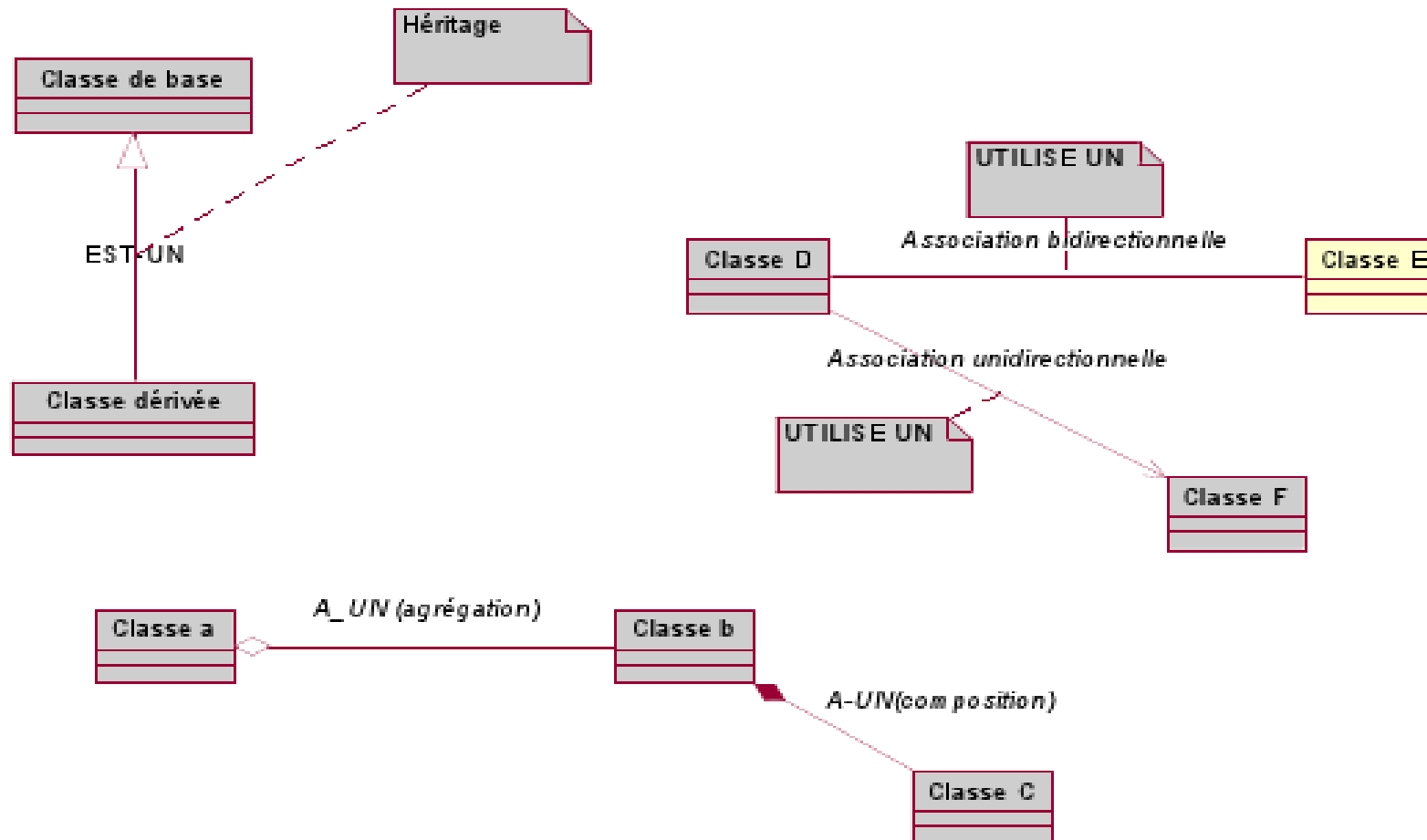
- ▶ **super-classe** : classe plus générale reliée à une ou plusieurs autres classes spécialisées (**sous-classes**) par une relation de **généralisation**
- ▶ Les sous-classes **héritent** des propriétés de leur super-classe et peuvent comporter des propriétés spécifiques supplémentaires

EXEMPLE



Created with Poseidon for UML Community Edition. Not for Commercial Use.

Types de relation entre classes



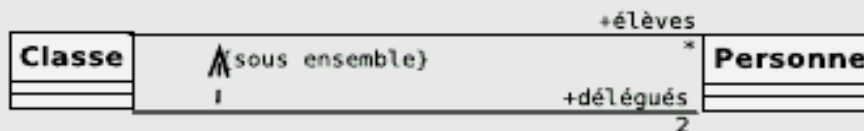
Les contraintes

- Une contrainte porte sur une relation ou un groupe de relations
- Notation : {*description contrainte*}

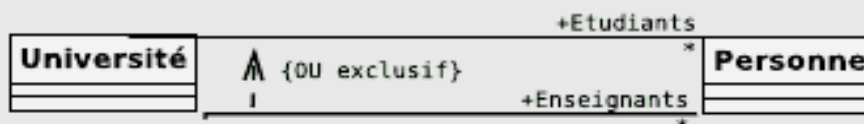
Exemple : Ordre



Exemple : Sous-ensemble

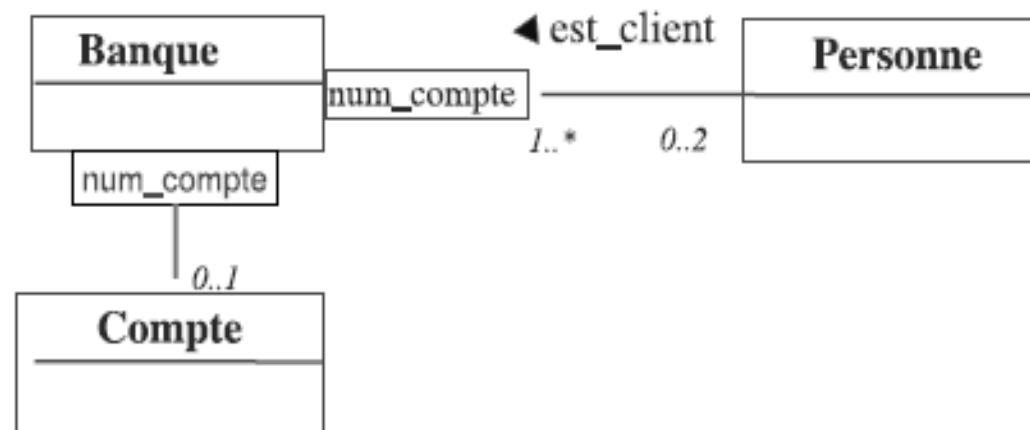


Exemple : Exclusivité



Relation- Qualification

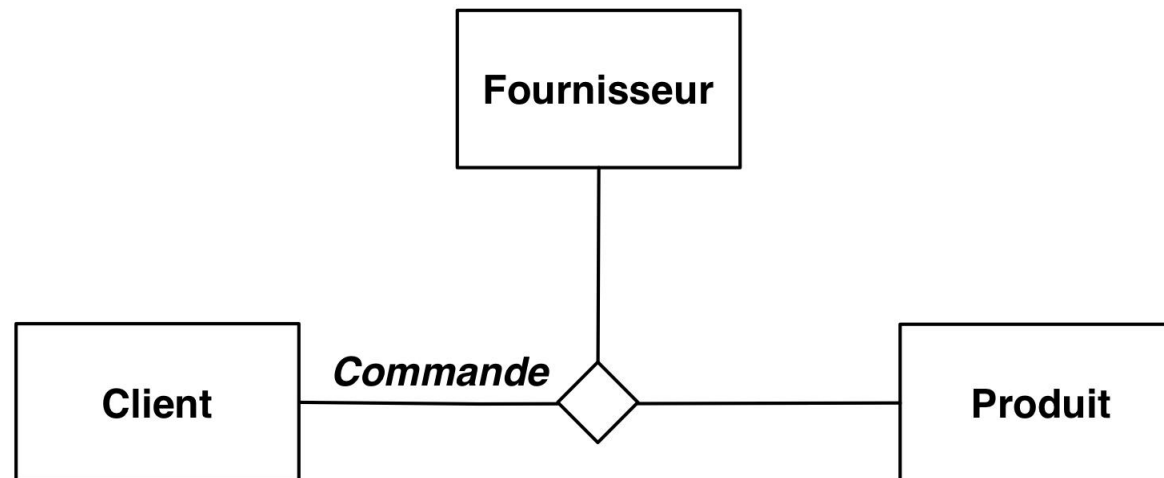
- ▶ Un qualificatif sur une association permet de sélectionner un sous-ensemble dans l'association (cela n'a de sens que pour une cardinalité *).
- ▶ Par exemple, à partir d'une banque et d'un numéro de compte, on sélectionne un unique compte.
- ▶ Ou bien, à partir d'une banque et d'un numéro de compte, on sélectionne au plus 2 personnes.



- ▶ La qualification se représente par un rectangle placé au niveau de la classe source du qualificatif.

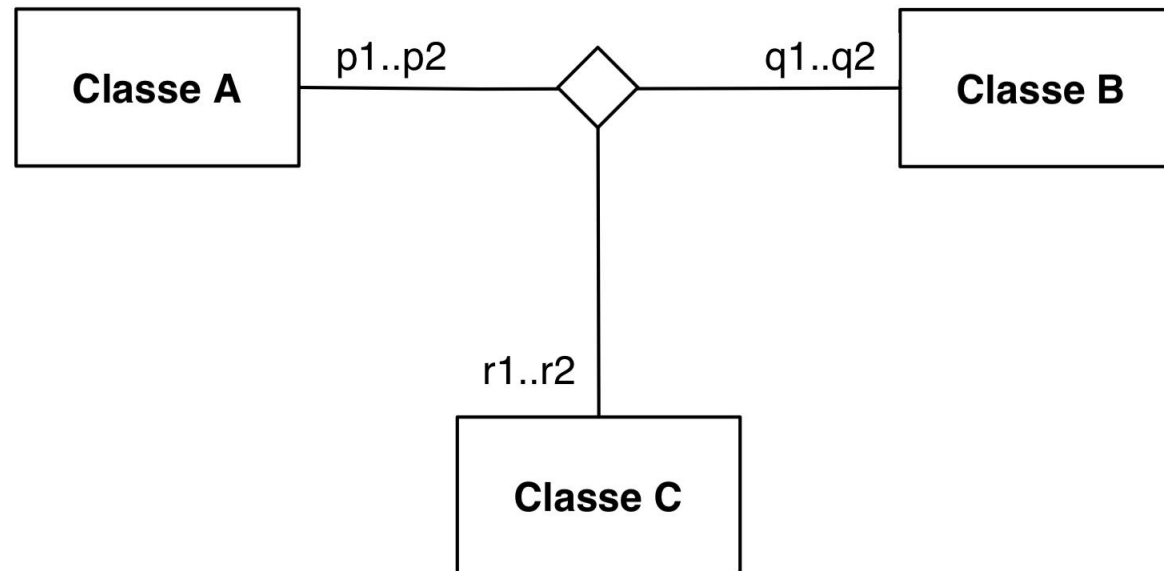
Associations n-aires

- **degré ou arité d'une association** = nombre de classes participantes
 - Association **unaire** : relie 2 instances d'une classe
 - association **binaire** : relie 2 classes
 - association **ternaire** : relie 3 classes
 - association **n-aire** : relie n classes



Associations n-aires

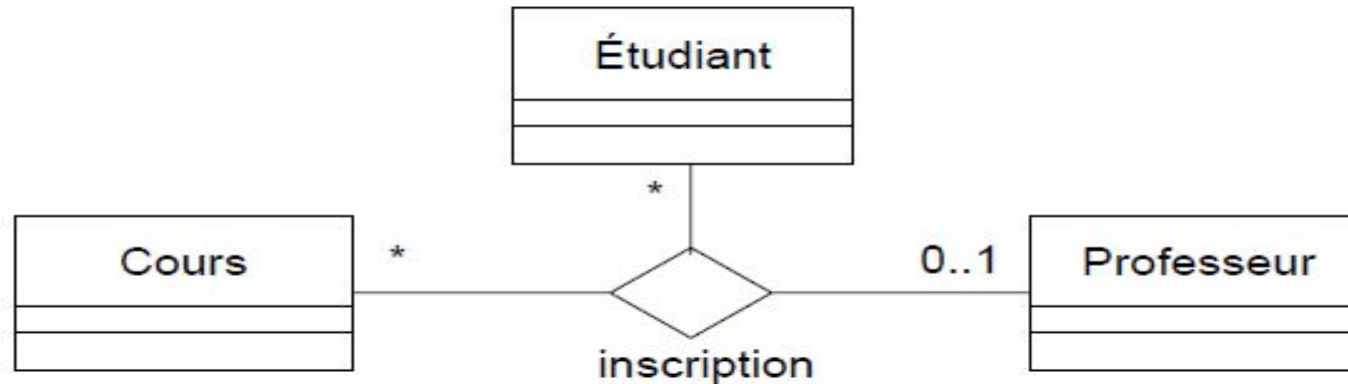
■ Exemple ternaire



- Pour un couple d'instances de la classe A et de la classe B, il y a au min. **r1** instances de la classe C et au max. **r2** instances,
- ...
- ...

Association

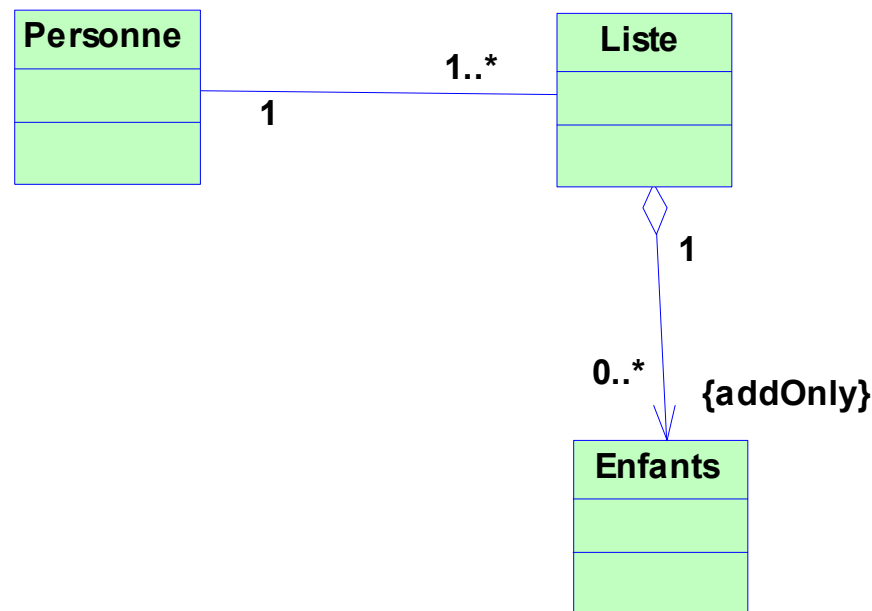
Cardinalité – association n-aire



- *Pour un cours et un étudiant donné, il peut y avoir un (l'étudiant est inscrit) ou aucun professeur (l'étudiant n'est pas inscrit).*
- *Un professeur peut donner plusieurs cours suivis par le même étudiant.*
- *Dans un cours, un professeur enseigne à plusieurs étudiants.*

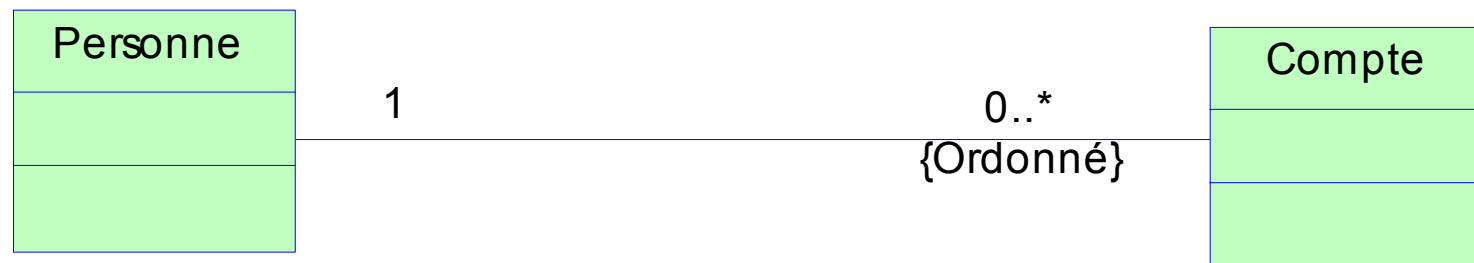
Contraintes sur les associations: contrainte {addOnly} (UML 1)

- ▶ La contrainte prédéfinie **{addOnly}** autorise l'ajout de nouveaux objets, mais pas leur suppression ni leur mise à jour.
- ▶ instances ajoutables mais non retirables.



Contraintes sur les associations: contrainte {ordonné-ordered}

Indique que les objets **seront ordonnés à chaque opération de création, modification, suppression, ...**



Les comptes d'une personne sont ordonnés

Contraintes sur les associations: contrainte {frozen} (UML 1)

- ▶ La contrainte prédéfinie {frozen} (**readonly UML 2**) interdit l'ajout, la suppression ou la mise à jour de **valeurs d'attribut ou de liens** d'un objet vers les objets de la classe associée, après l'initialisation du premier.

