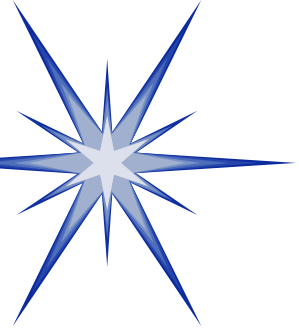


Unified Modeling Language

Modélisation Objet

Pr. Omar El Beqqali

omar.el-beqqali@insa-lyon.fr
http://www.fsdmfes.ac.ma/oelbeqqali
http://www710.univ-lyon1.fr/~obekkali

EMSI 2009 / 20010

Spécification et conception du logiciel.

Plan

Le processus de développement du logiciel

- *Les enjeux du Génie Logiciel.*
- *Le processus de développement du logiciel.*

(GL)

Le langage de modélisation unifié UML.

- Présentation générale.
- Méthodologie Objet en spécification et en conception
- Concepts fondamentaux.
- **Diagrammes UML**
  - -Diagramme des cas d'utilisation
  - -Diagramme de séquences.
  - - Diagramme de collaboration
  - - Diagramme de classes.
  - - Diagramme d'objets
  - - Diagrammes d'états-transitions
  - - Diagramme d'activités
  - - Diagramme de composants.
  - - Diagramme de déploiement.
- **UML & Bases de données**
- **Génération du code**
- Rétro-ingénierie (reverse engineering)
- Mise en œuvre d'UML : étude de cas

(2004 : 4 nouveaux diagrammes)

Spécification et conception du logiciel.
Méthodologie Objet : UML.
Objectifs

⇨ *Comprendre les enjeux du Génie Logiciel.*

⇨ *Comprendre l'importance des activités de spécification et de conception (G.L)*

⇨ **Connaître la notation UML.**

⇨ **Savoir comment utiliser UML pour spécifier et concevoir un logiciel.**

⇨ **Mettre en œuvre UML (cas)**

La modélisation

- Elle est essentielle pour :

  -Comprendre le fonctionnement d'un système

  -Maîtriser la complexité

  -Faciliter la communication au sein de l'équipe

- Et particulièrement en génie logiciel :

  -Être un facteur de réduction des coûts et des délais,

  -Être un facteur d'accroissement de la qualité du produit,

  -Permettre d'assurer une maintenance facile et efficace,

  -Permettre de contrôler l'avancement d'un projet.

Modèles et techniques utilisés par les méthodes objets

- Les méthodes de modélisation ‘classiques’ sont basées sur :
  - Un modèle de données et un modèle des traitements séparés
  - Une modélisation de flots de données

  Insatisfaisantes pour modéliser des systèmes objet

- *Aucune méthode ne couvre toutes les étapes du cycle de développement*

- **Triple perception du système d’information**
  - **Dimension statique: objets**
  - **Dimension dynamique: événements/états**
  - **Dimension fonctionnelle : flux/processus**

Conception par objets

- **Démarche itérative plus que descendante** ***(cycle itératif)***
- Grandes étapes
  - Identifier les entités du domaine
  - Structurer le domaine en analysant les propriétés de ces entités et leurs relations
  - Identifier les opérations que savent effectuer ces entités
  - Décrire précisément ces opérations en les reliant à des messages
  - Décrire le lancement du programme

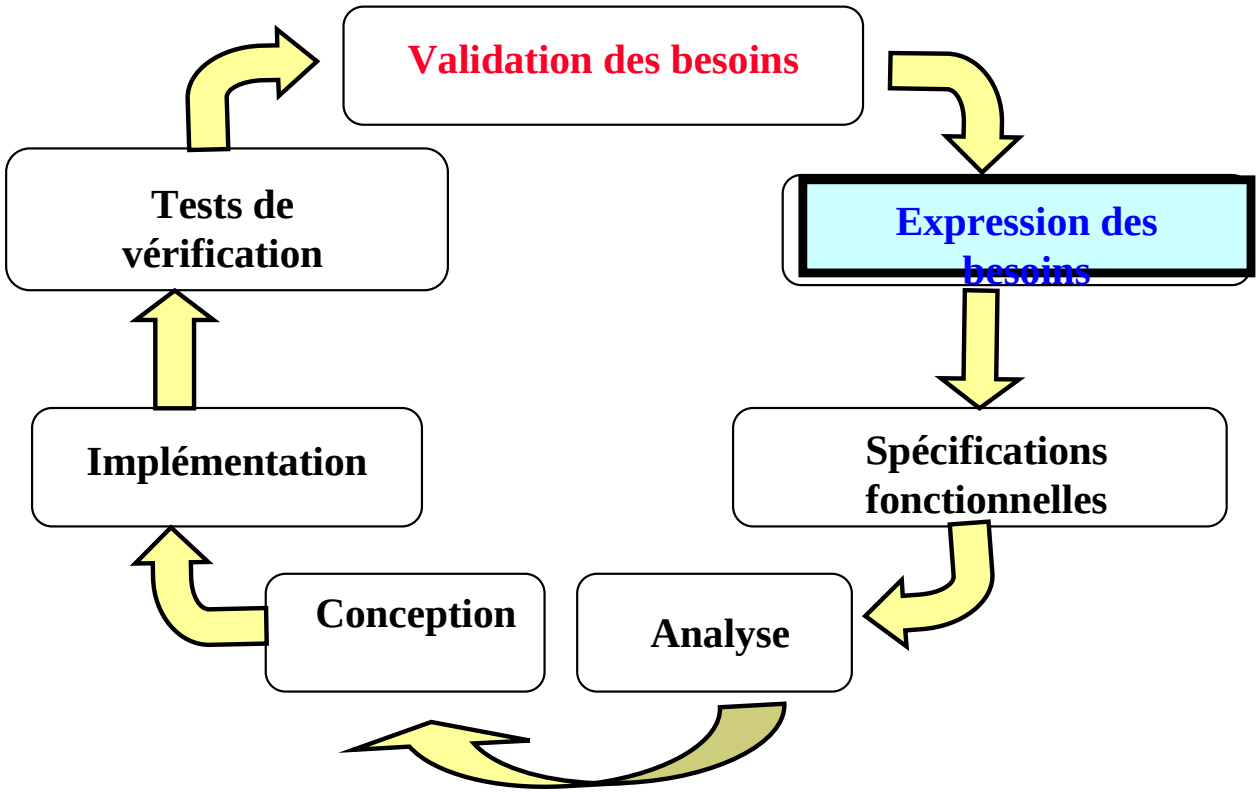
Avantages de la conception objet

- ***Avantages de l'utilisation de l'approche objet au niveau conceptuel***
  - Réduction de la « distance » entre langage utilisateur et langage conceptuel
  - Regroupement de l'analyse des données et des traitements
  - Simplification des transformations entre niveau conceptuel et niveau physique
  - Abstraction forte
  - Orienté vers la réutilisation : notion de composants, modularité, extensibilité, adaptabilité, souplesse.

Le cycle de vie Objet

(G.L)

Un cycle itératif : ***ce cycle s'appuie sur l'analyse des risques*** ***(adéquat pour la conception objet)***

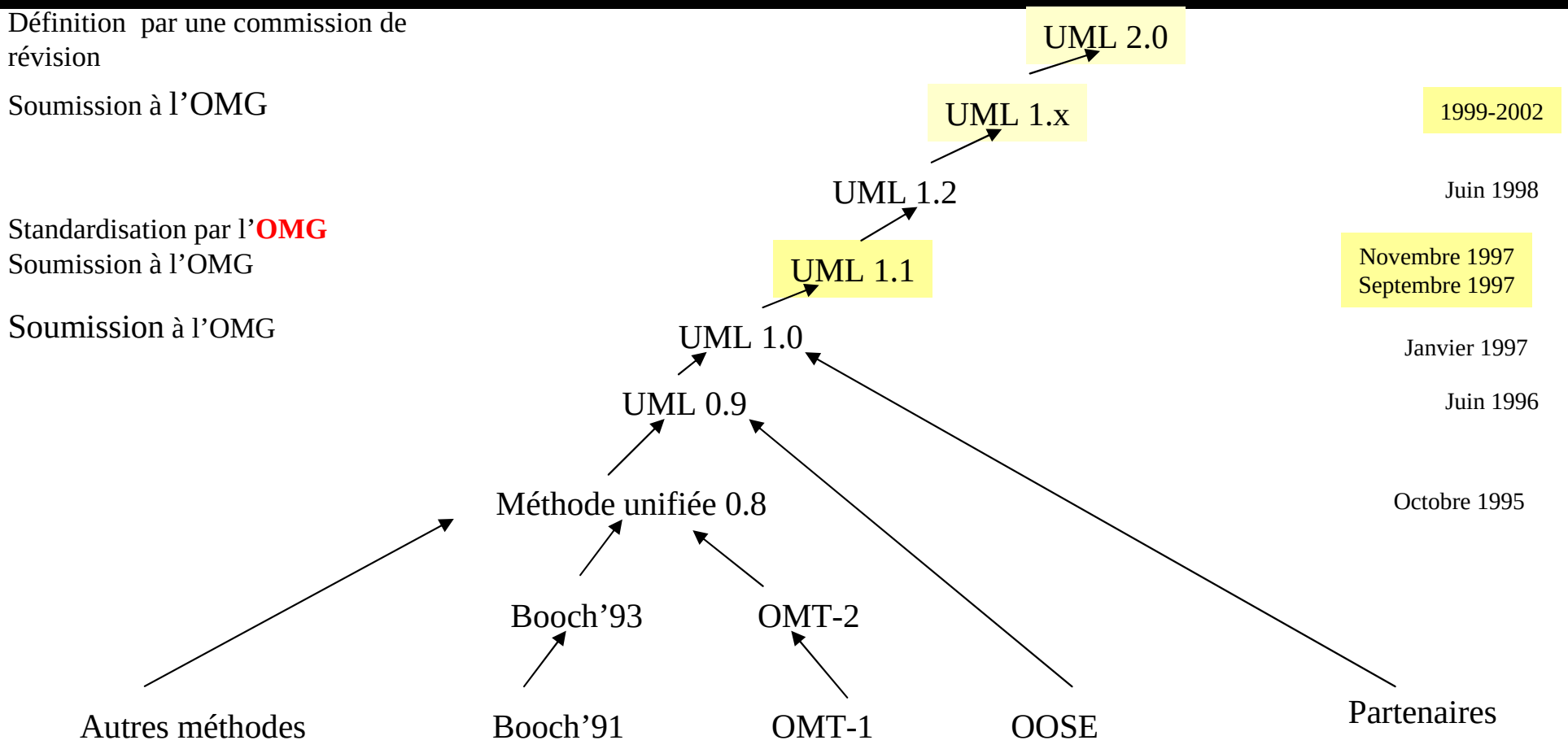


Méthodes objets (historique)

OOD	Object Oriented Design (G. Booch)	1991
HOOD	Hierarchical Object Oriented Design (Delatte & al.)	1993
OOA	Object oriented analysis (Schlaer ,Mellor)	1988, 92
OOA/OOD	(Coad & Yourdon)	1991
OMT	Object Modeling Technique	1991
OOSE	Object oriented software engineering (Jacobson & al.)	1992
OOM	Object oriented Merise (Bouzeghoub & Rochfeld)	1993
Fusion	(Coleman & al.)	1994

- Théorie développée par Ivar JACOBSON
- Reprise par de nombreuses méthodes : OMT, ROOM, Fusion, Booch, ..
- Elle repose sur une analyse **centrée utilisateur** pour déterminer les besoins du système.

Historique UML



1997: présentation de UML à l'OMG (Object Management Group),
UML 1.1 est adopté *(**http://www.omg.org**)*

1999: UML 1.3

2004: UML 2.1 (pas encore très utilisée)

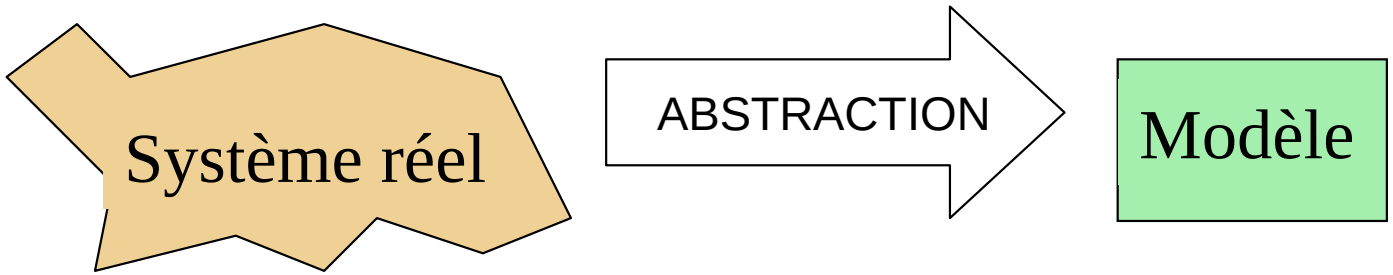
UML : langage de modélisation graphique et textuel

- UML unifie
  - Les concepts, quels que soient le domaine d'application
  - Les notations et concepts orientés objet
- UML est indépendant
  - Du type du système-logiciel, matériel, organisation..
  - Du domaine métier : gestion, ingénierie, finance…
- UML permet de :
  - Comprendre et de décrire les besoins,
  - Concevoir et construire des solutions,
  - Documenter un système tout au long du cycle de développement,
  - Communiquer entre les membres de l'équipe de projet.

UML : objectifs

- *Montrer les limites d'un système et ses fonctions principales* à l'aide des cas d'utilisation et des acteurs.
- Illustrer les réalisations de Cas d 'Utilisation à l'aide de diagrammes d'interaction.
- *Représenter la structure statique d'un système* à l'aide de diagrammes de classes, associations, contraintes.
- *Modéliser la dynamique, le comportement des objets* à l'aide de diagrammes états/transitions.
- *Révéler l'implantation physique de l'architecture* avec des diagrammes de composants et de déploiement.
- **Un langage utilisable par l'homme et la machine : permettre la génération automatique de code.**

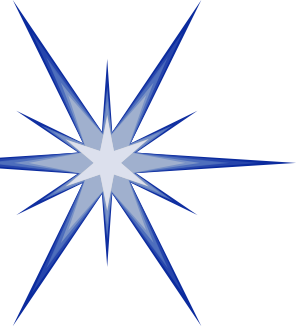
Modélisation en diagrammes



- Modèle = ensemble d'éléments de modélisation vus dans un ensemble de diagrammes
- Diagramme

-Support graphique de modélisation, chaque diagramme propose un point de vue différent.

-mise en œuvre d'un ensemble d'éléments de visualisation représentant des éléments de modélisation (graphe)



Vues générales des diagrammes et mécanismes de base d'UML

UML : les diagrammes (1)

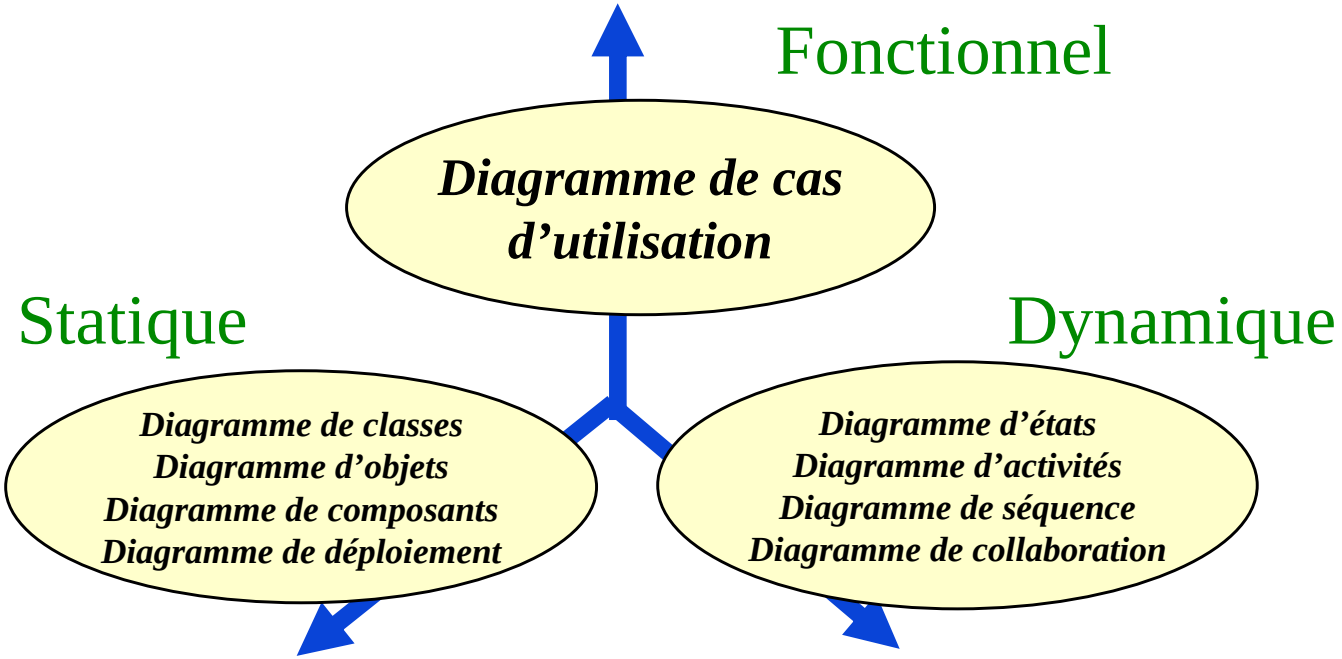
- Modèles manipulés au moyen de vues graphiques : 9 diagrammes **(4 autres nouveaux diagrammes en 2004)**
  - **Diagrammes des cas d'utilisation** : Fonctions du système du point de vue des utilisateurs
  - **Diagrammes de séquence** : Représentation temporelle des objets et de leurs interactions
  - **Diagrammes de collaboration** :Représentation spatiale des objets, des liens et des interactions

UML : les diagrammes (2)

- **Diagrammes de classes** : Structure statique en terme de classes et relations qui les lient
- **Diagrammes d'objets** : Représentation des objets et de leurs relations (liens)
- **Diagrammes d'états-transitions** : Comportement d'une classe en terme d'états, lié au cycle de vie des objets
- **Diagrammes d'activités** : Représentation du comportement d'une opération, d'un cas d'utilisation, ou d'un processus métier en terme d'actions
- **Diagrammes de composants** : Composants physiques d'une application, dépendance entre ces composants
- **Diagramme de déploiement** : Déploiement des composants sur les dispositifs matériels, modes de connexion..

Classification des diagrammes

L'ensemble des 9 diagrammes peut être réparti sur les trois axes de modélisation



Mécanismes de base

•**Stéréotype** : mécanisme permettant de classer les éléments similaires du modèle en familles << stéréotype>>

quelques stéréotypes : << besoin >>, << responsabilité >>

Exemple : classes Fenêtre, Icône, Bouton

→ valeurs communes (afficher/masquer),

→ stéréotype << **visuel** >>

•**Contraintes** : exprime un lien sémantique entre des éléments du modèle

notation : {}

•**Note** : commentaire rattaché à un diagramme

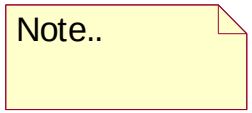
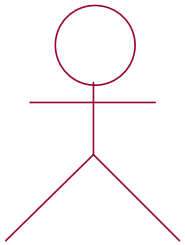


Diagramme de Cas d'utilisation

Cas d’utilisation : Introduction

- Le concept de cas d’utilisation introduit par Ivar Jacobson dans la méthode Object-Oriented Software Engineering (OOSE).
- Les fonctionnalités du système sont décrites comme un ensemble de cas d’utilisation.
- Chaque cas représente un flot spécifique d’événements vers le système.
- La description du cas d'utilisation définit ce qui arrive dans le système lors de sa réalisation.

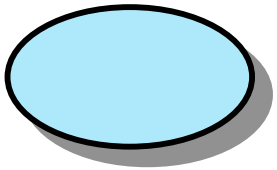
Cas d'utilisation : Notation (1)



Acteur

Entité qui agit sur le système; représente un ensemble cohérent de rôles qu'un utilisateur peut effectuer.

Use Case



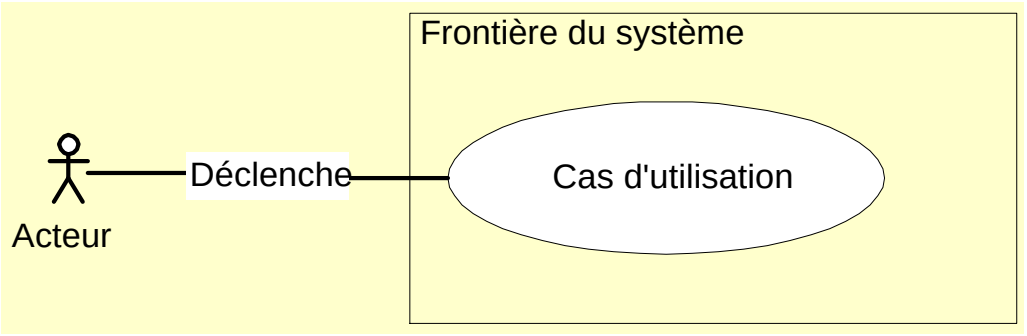
Cas d'utilisation

Ensemble cohérent de fonctionnalités fournies par le système ou un sous-système en réponse à une action de l'utilisateur. Ce ci se traduit par l'échange de messages entre le système et les acteurs.

Elaboration de CU

questions à se poser :

- quelles sont les tâches de l'acteur ?
- quelles informations l'acteur doit-il créer, sauvegarder, modifier, détruire, ou simplement lire ?
- l'acteur doit-il informer le système de changements externes ?
- On s'intéresse au domaine du '*quoi faire*', pas du '*comment*' *(sinon on rentre dans la phase de conception)*
- on doit rester au niveau de l'interaction acteur/système



Cas d'utilisation : Démarche

- Recherche des acteurs externes
- Pour chaque acteur les cas d'utilisation
- Pour chaque cas d'utilisation :
  - rechercher les interactions
  - rechercher les objets manipulés
- Faire la maquette de chaque cas d'utilisation

Remarque :Les diagrammes des cas d'utilisation se retrouveront à tous les stades du projet.

C.U caractéristiques (1)

- Le cas d'utilisation utilise une description textuelle
- Le cas d'utilisation est un cadre pour l'élaboration des différentes fins possibles pour le cas d'utilisation
- L'analyse est complète lorsque tous les cas d'utilisation sont étudiés
- Un cas d'utilisation décrit l'échange standard entre un acteur externe et un système; décrit une famille de scénarios (voir plus loin) incluant les cas d ’erreur.
- L'acteur est l'initiateur de l'échange, il peut être:
  - Personne
  - équipement
  - système externe

Acteur

- Rôle joué par une personne ou une chose qui interagit avec un système
- La même personne physique peut jouer le rôle de plusieurs acteurs
- Plusieurs personnes peuvent jouer le même rôle
- Nom de l'acteur = Rôle joué par l'acteur

- **Détermination des acteurs ==> précision des limites du système de manière progressive**

Différentes catégories d'acteurs

Catégories

Acteurs principaux: personnes utilisant le système (à qui va servir le système)

•**Acteurs secondaires**: qui administrent le système (paramètrent le système en lui fournissant les informations nécessaires ou effectuent des m.a.j).

Exemple : bibliothèque => *l'administrateur est un A.S, le membre est un A.P.*

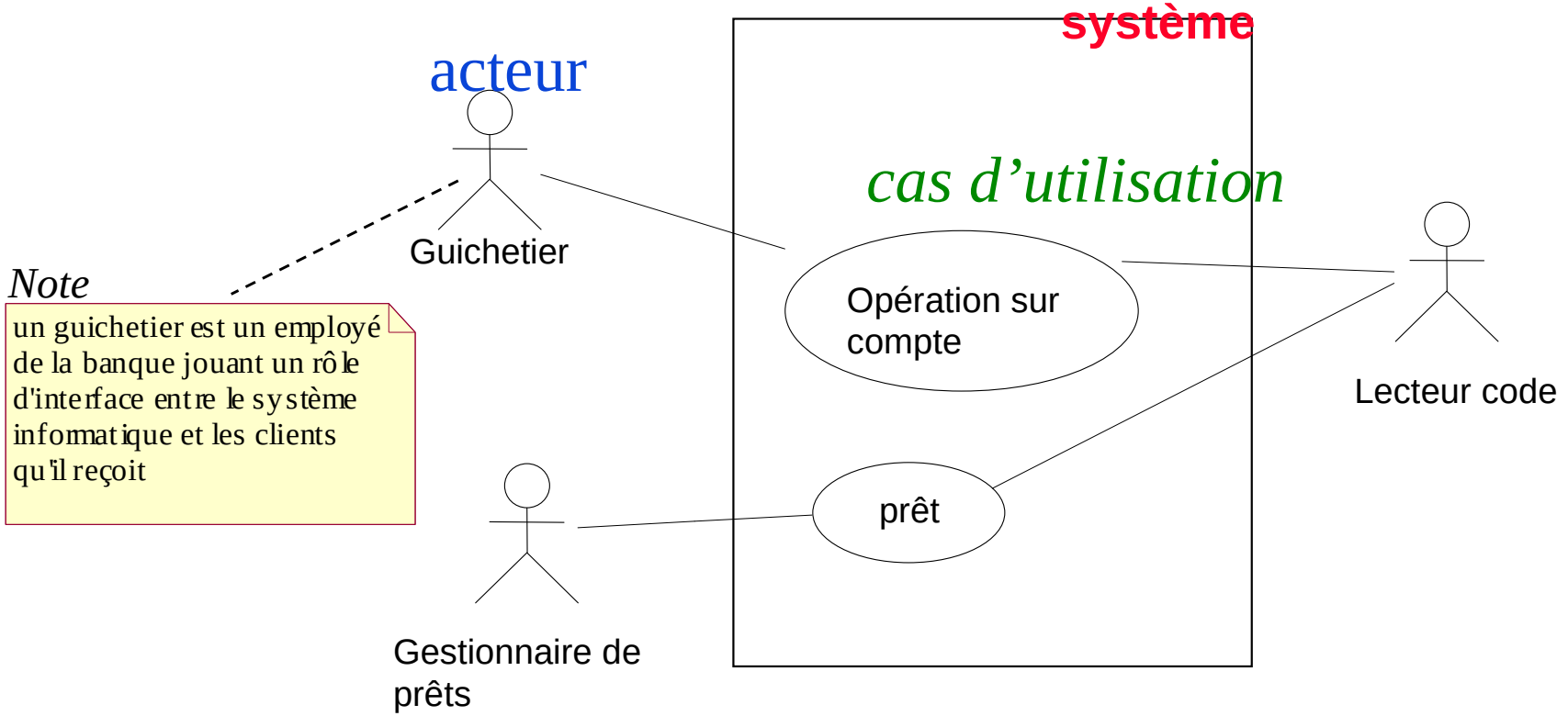
Types

Matériel externe: dispositifs matériels faisant partie du domaine de l'application.

•**Humain**s : utilisateurs du système.

•**Logiciels, robots** : qui exploitent les données du système.

Acteurs et cas d’utilisation

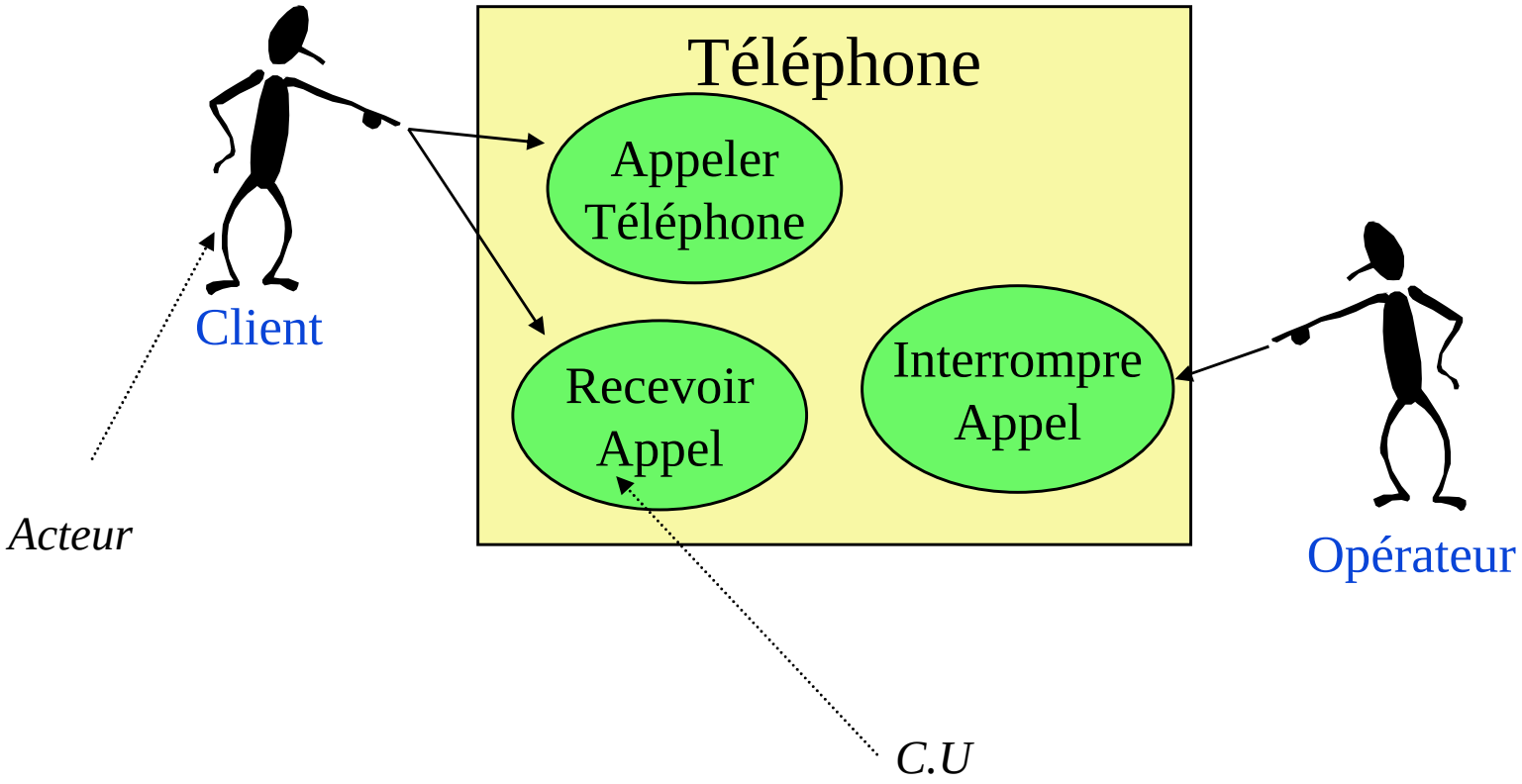


Cas d’utilisation : description générique d’une transaction complète entre l ’acteur et le système (claire et précise).

Remarque : pas d ’interactions entre acteurs

C.U : exemple

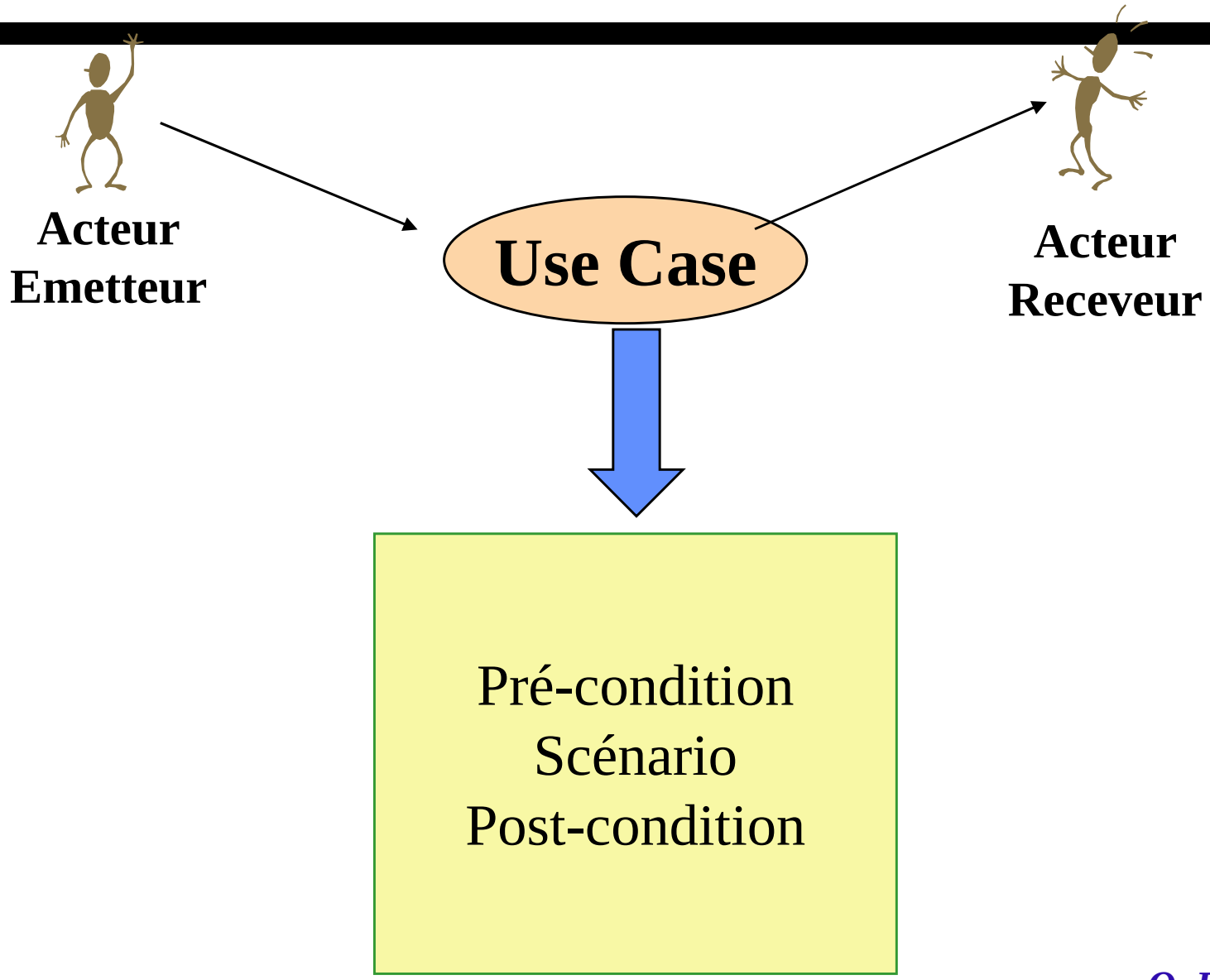
Diag. C.U Système téléphonique

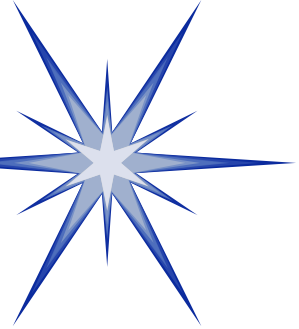


CU
caractéristiques (2)

- Identification d'une finalité de l'utilisateur
- Un stimulus de départ
- Une *pré-condition* du système au déclenchement
- Un enchaînement d'interactions
- Une *post-condition* du système à la fin du cas d’utilisation
- Enchaînement d’actions à effectuer
- Une fin normale (conditions d’exécution éventuelles)

Le cas d'utilisation





Cas d'utilisation : Exemple (1)

- Nom du cas d'utilisation:

  Attribution d'une place sur un vol

- Un stimulus de départ : ***Client présente sa réservation***
- Une pré-condition: ***Guichet ouvert & réservation sur un vol***
- Un enchaînement d'interactions: ***Scénarios***
- Une post-condition: ***Place affectée au passager & Fin-réservation***

Cas d'utilisation : Notation (2)

Les relations entre cas d'utilisation :

-Extend

-Include

Include →

Capture comment une ou plusieurs descriptions d'un « use case » intègrent la description d'un autre use case.

(Équivalent à l'agrégation pour les classes, voir plus loin)

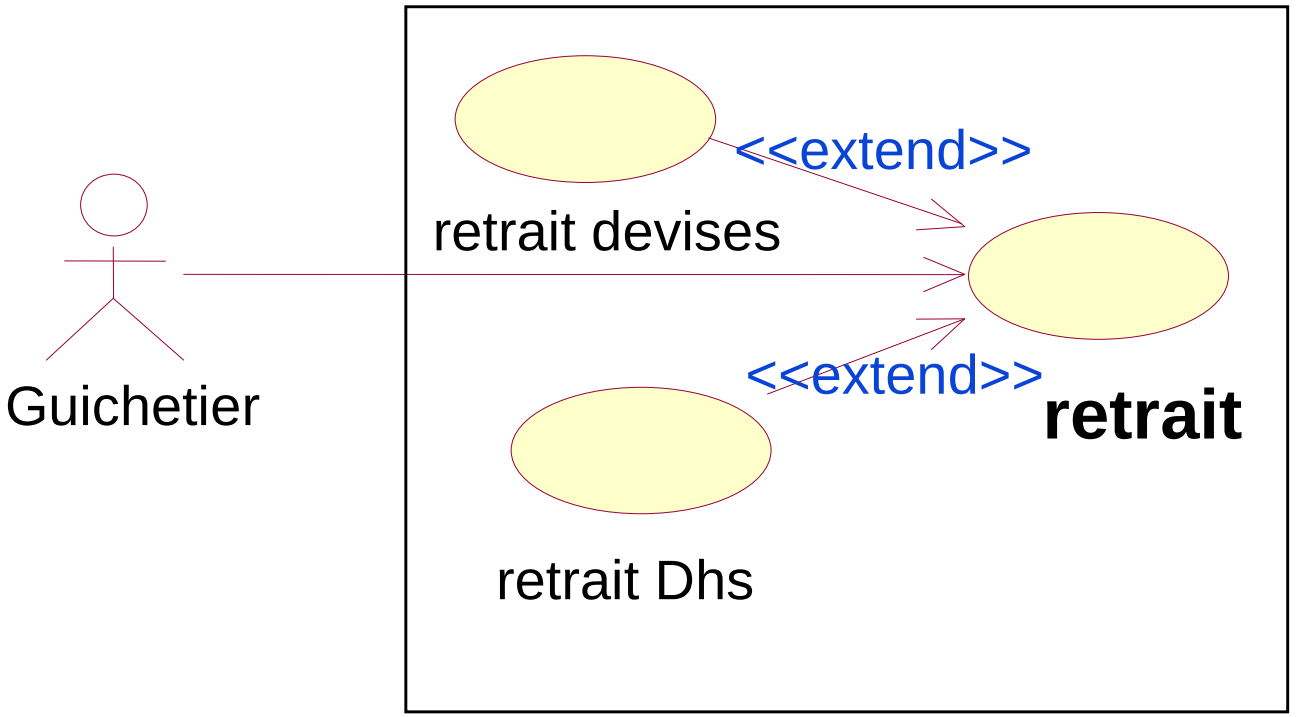
Extend →

Représente l'insertion d'un « use case » dans un autre use case.Équivalent à la généralisation/spécialisation pour les classes *(voir plus loin)*.

Cas d'utilisation : La notation (1)

La relation « extend » :

- dénote un comportement optionnel,
- indique que tous les UC ‘fils’ **héritent** du UC ‘mère’ (UC pointé)

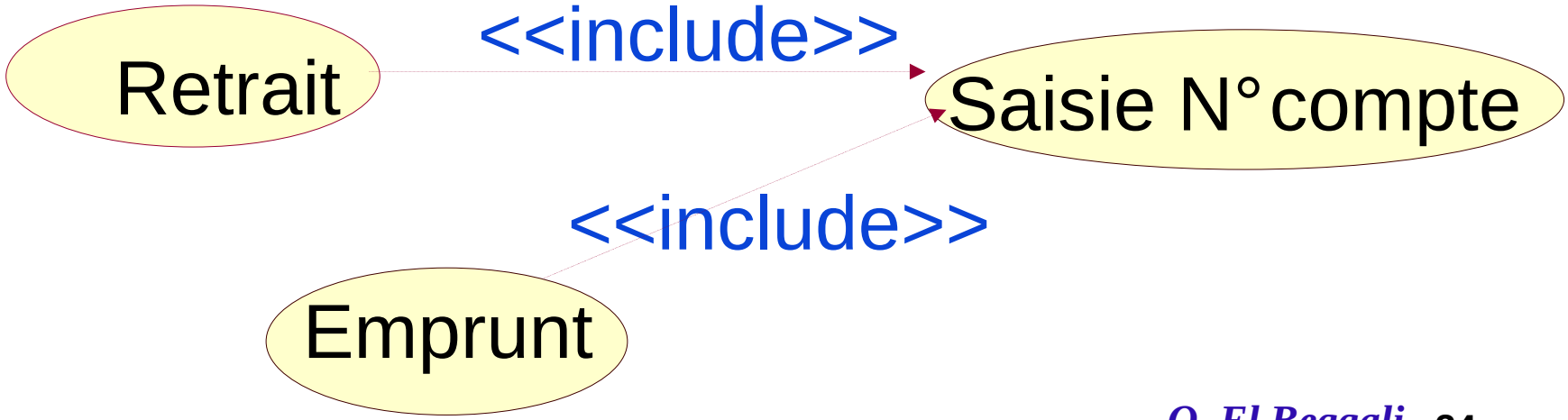


Cas d'utilisation : La notation (2)

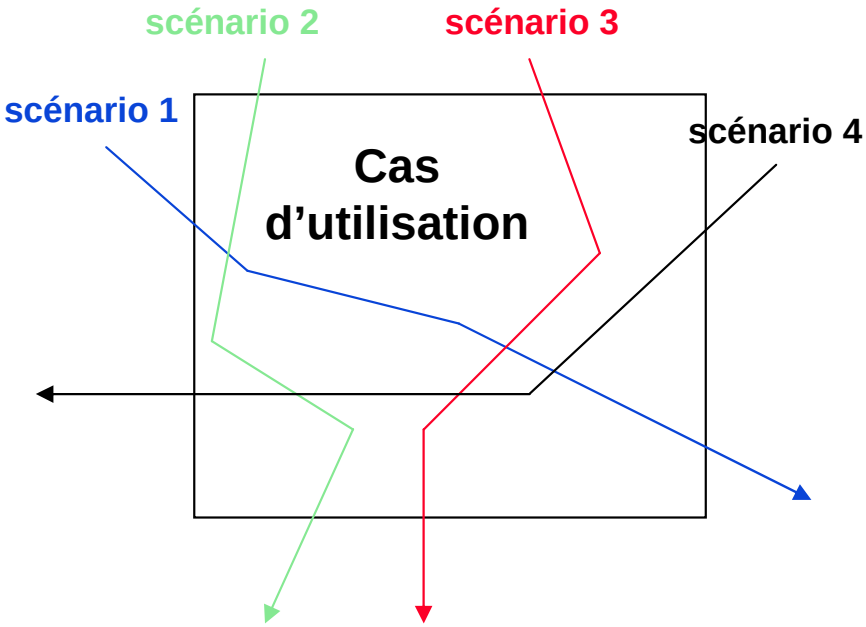
La relation « include» indique que le UC pointé par la flèche est une sous partie de l'autre UC .

⇒ 'factorisation' des UC dont les fonctionnalités servent fréquemment dans des différents UC

(comportement communs à plusieurs UC).



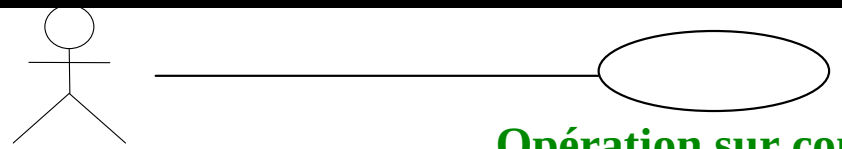
Cas d’utilisation et scénarios



- Scénario = chemin dans le CU
- Description du CU
  - ensemble des scénarii
  - document avec flot d’événements
    - du point de vue de l’acteur
    - détaille ce que le système doit fournir à l’utilisateur quand le CU est exécuté
      - flot normal des événements (80 %) ***-Nominal-***
      - flots d’événements alternatifs
      - flots d’exceptions (quand le CU ne ne termine pas correctement)

CU : scénarios

Exemple : GAB



Description générale C.U : ce cas d'utilisation concerne les opérations que le client peut réaliser sur son compte sur un distributeur bancaire.

Description des scénarios :

Le cas d'utilisation commence quand l'utilisateur se connecte. Le système lui propose :

a/ de retirer du liquide
b/ de déposer un chèque

Si le client choisit a/, le système lui propose divers montants. L'utilisateur choisit une somme. Le système vérifie que le compte est suffisamment approvisionné, et délivre l'argent et un reçu. Sinon, le système informe l'utilisateur de l'échec de l'opération, et lui rend sa carte.

Le cas d'utilisation se termine quand l'utilisateur retire sa carte bancaire.

Cas d'utilisation et interactions

- Le diagramme des CU présente une vue du système *de l'extérieur*
- Une interaction décrit comment les cas d'utilisation (classes, opérations) sont réalisés comme interactions dans une « société » d'objets (communication entre objets)
- Vue dynamique du système : deux types de diagrammes d'interaction
  - diagrammes de collaboration
  - diagrammes de séquences

Cas d'utilisation : scénarios

Exemple

- Nom du cas d'utilisation:

  Attribution d'une place sur un vol

- Le passager présente sa réservation à l'hôtesse
- qui lui demande son nom,
- le passager fournit l'information,
- le système trouve sa réservation [*ou ne la trouve pas (autre scénario)*],

  ...

- le passager accepte la place proposée par le système.

Scénarios - exemples

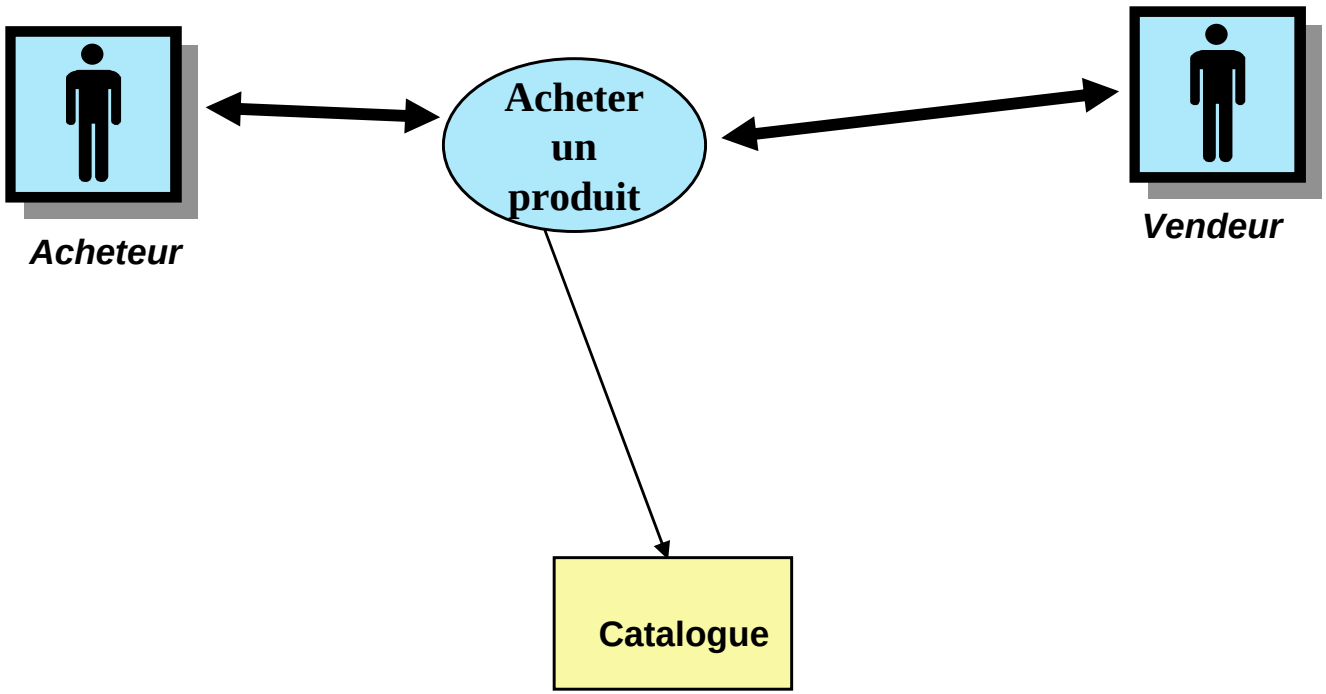
- Cas d'utilisation :

1/ Acheter un produit

2/ Passer les tourniquets du RER

Cas d'utilisation: un besoin

Exemple : activité commerciale



Cas d'utilisation : Le scénario

Exemple Vente

- Le scénario est une application parmi d'autres d'un cas d'utilisation.
- Pour décrire complètement un cas, plusieurs scénario sont nécessaires.
- Il apporte une compréhension précise sur une application du cas d'utilisation

Scénario pour Négocier avec succès

Acheteur demande le prix
Vendeur interroge le catalogue
Vendeur donne le prix de base
Acheteur s'étonne
Acheteur se plaint
Vendeur fait une proposition
...

Scénario pour Négocier avec échec

Acheteur demande le prix
Vendeur interroge le catalogue
Vendeur donne le prix de base
Acheteur s'étonne
Acheteur se plaint
Vendeur fait une proposition
Acheteur change de produit

Scénario pour Négocier avec condition

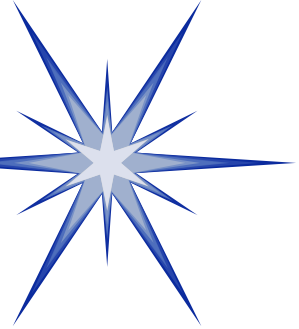
Acheteur demande le prix
Vendeur interroge le catalogue
Vendeur donne le prix de base
Acheteur s'étonne

Cas d'utilisation : Le scénario

Exemple 'RER'

- Passer les tourniquets du RER
  - Présenter le ticket au tourniquet
  - Le tourniquet avale le ticket
  - Le tourniquet contrôle le ticket
  - le tourniquet redonne le ticket
  - Passer le tourniquet

…..Autres Scénarii….



Diagrammes de séquences

Les diagrammes de séquence (1)

•Le diagramme de séquence montre quels sont les objets qui participent à l'exécution du use-case et quels sont les messages qu'ils échangent : description des interactions entre les objets d'un point de vue temporel.

•Chaque bloc ou objet participant dans le processus est représenté par une barre verticale.

Remarque *: l'ordre dans lequel apparaissent les barres n'a pas d'importance (lisibilité).*

Diagrammes de séquence (2)

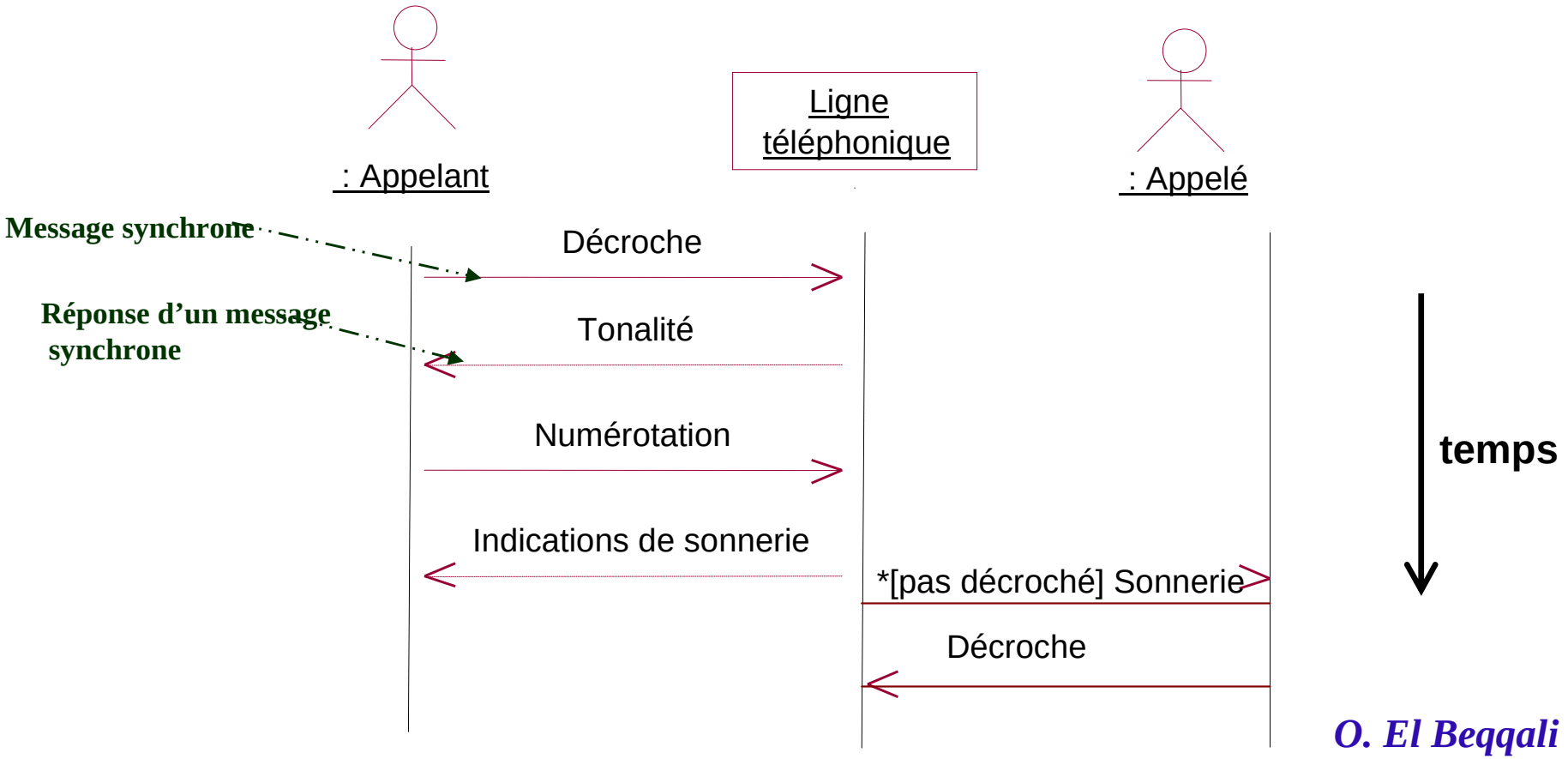
- Interactions entre objets selon un point de vue temporel
  - 2 utilisations
    - Documentation des cas d'utilisation
      - Evénements qui surviennent dans le domaine de l'application
    - Représentation précise des interactions entre objets
      - Echanges de messages

Remarque : possibilité de représentation de la structure de contrôle, la création et la destruction des objets, la récursion, les branchements conditionnels.

Diagrammes de séquence :

exemple : système d'appel téléphonique

- Description de scénarios typiques et des exceptions…



Diagrammes de séquence

exemple : système de gestion de comptes bancaires

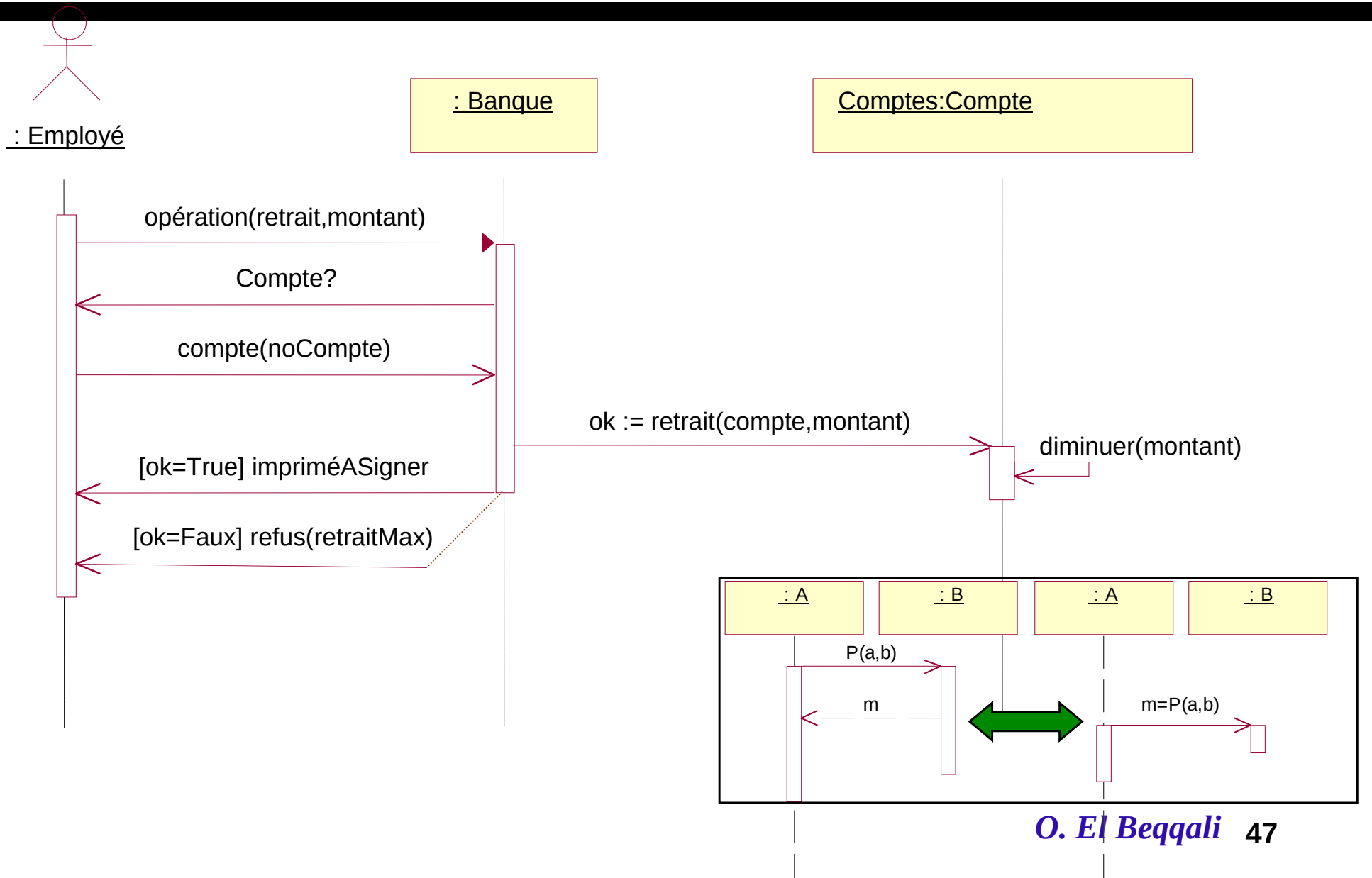
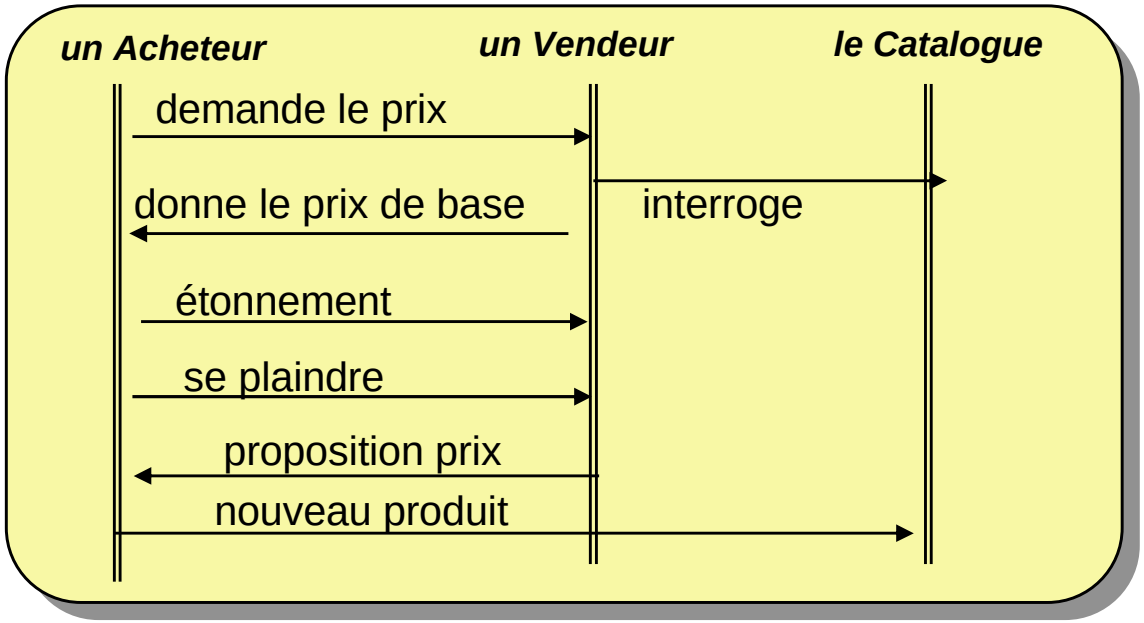


Diagramme de séquence

le suivi des événements (trace event)

Exemple1 (Vente)

- Le suivi d'événements est un diagramme qui présente la séquence des événements d'un scénario ainsi que les objets qui émettent et reçoivent les événements.
- Le temps augmente en partant du haut vers le bas, mais l'espace entre les événements n'est pas proportionnel au temps.



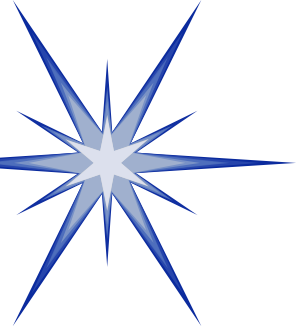


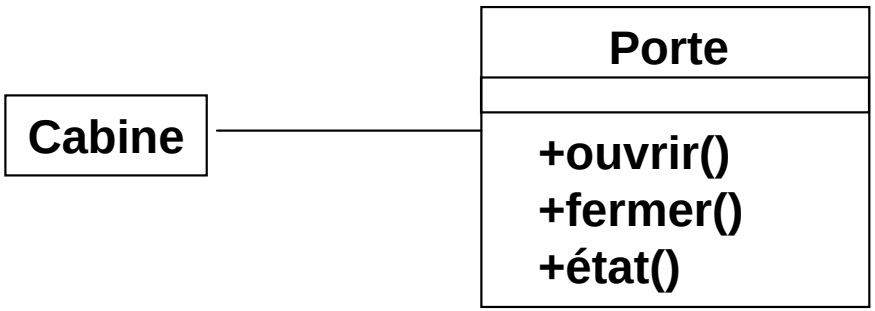
Diagramme de collaboration

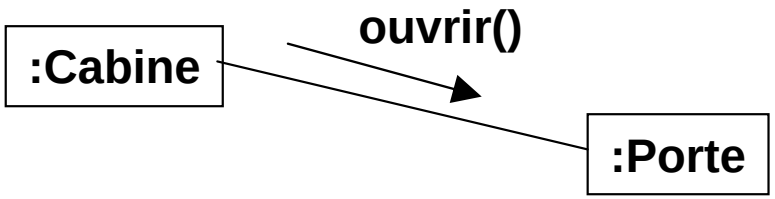
Diagrammes de collaboration

- Représente les interactions entre objets et relations structurelles permettant celles-ci.
- Description:
  - Du comportement collectif d'un ensemble d'objets
  - Des connexions entre ces objets
  - Des messages échangés par les objets
- Interaction réalisée par un groupe d'objets qui collaborent en échangeant des messages
- Temps non représenté de manière implicite *(numérotation des messages)*

Les interactions

- Eléments d'une interaction
  - *instances*
  - *liens* (support messages)
  - *messages* déclenchant les opérations
  - *rôles* joués par les extrémités de liens



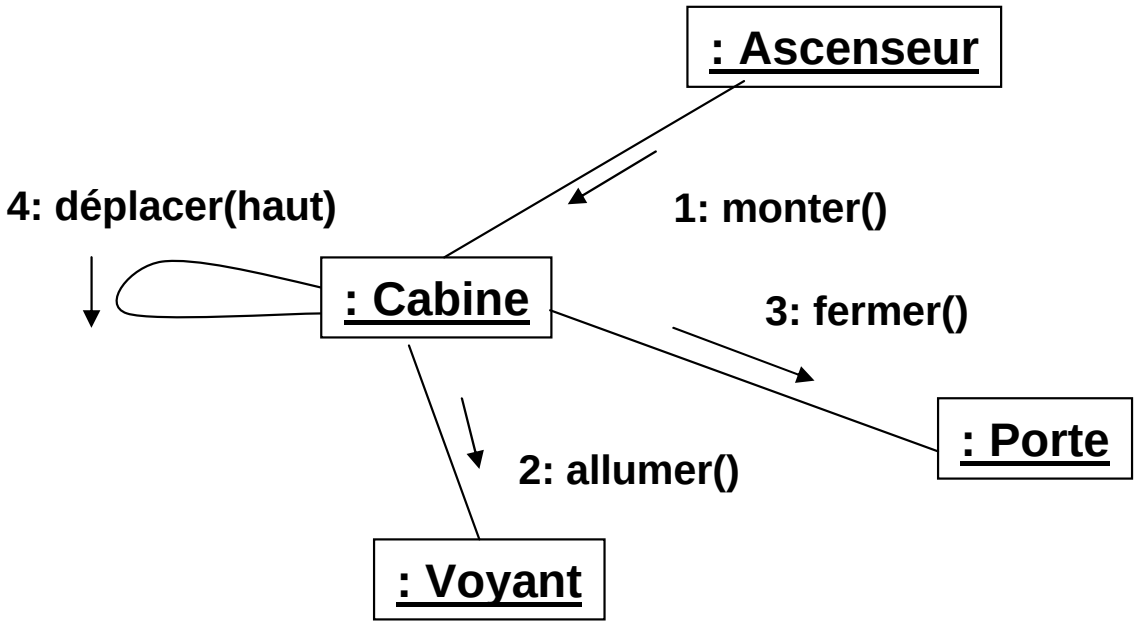


Interaction entre 2 objets instances des classes ***Cabine*** *et* ***Porte***

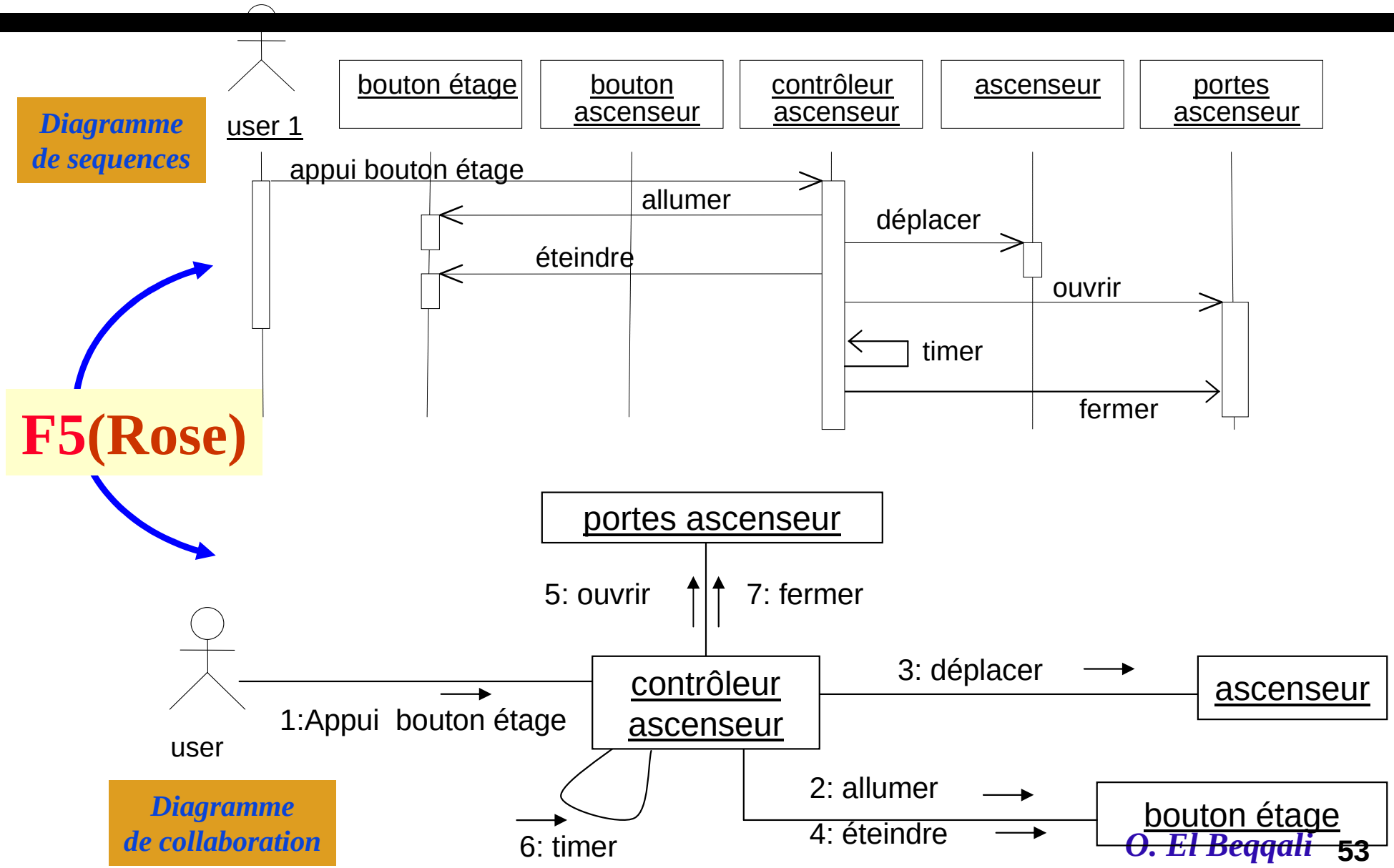
Modélisation d'une cabine d'ascenseur qui 'demande' à une porte de s'ouvrir

Diagrammes de collaboration

Représentation de l'ordre des envois de messages pour l'exemple De l'ascenseur :



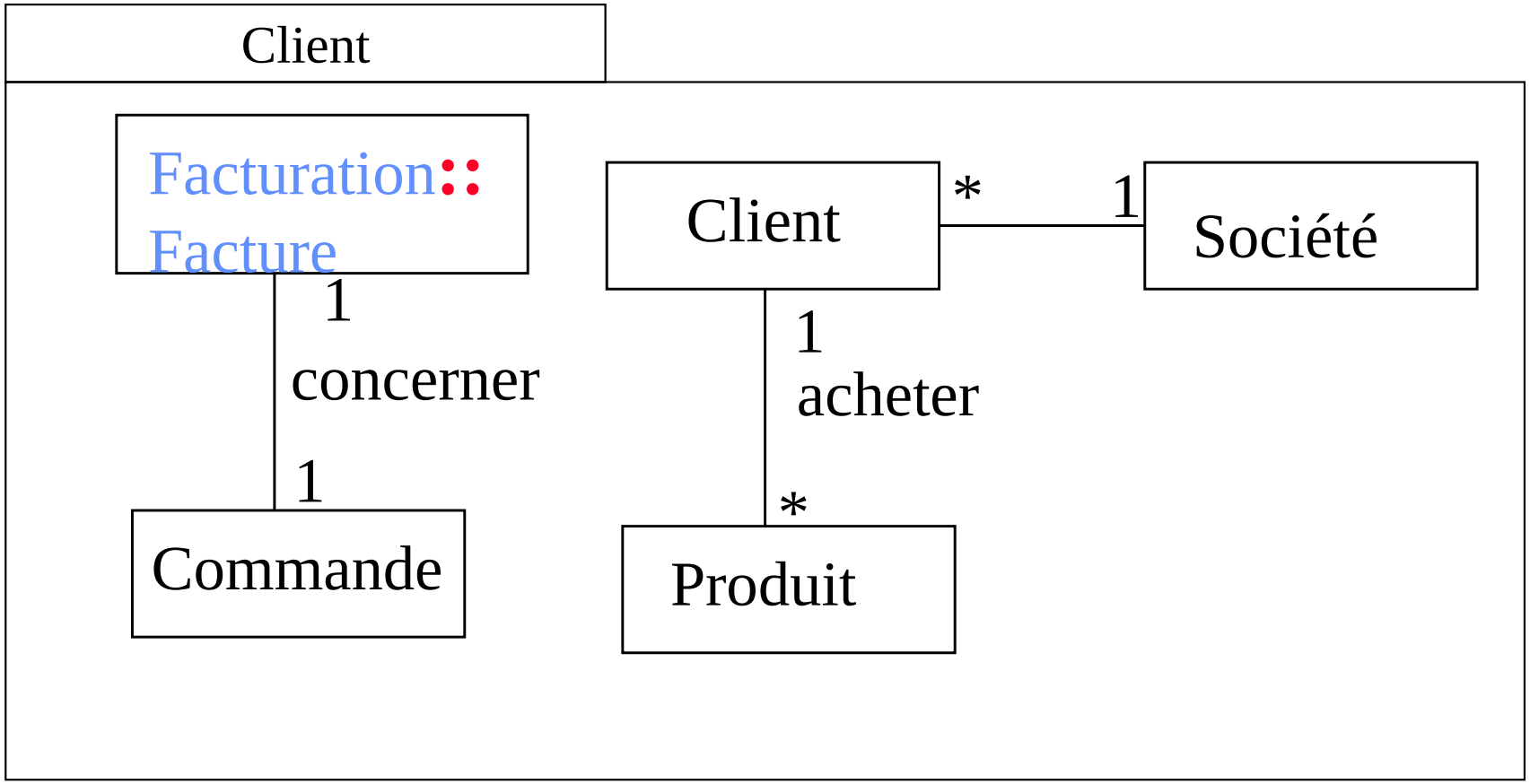
Équivalence diagrammes de séquence / collaboration



Paquetages *(package)*

- regroupement logique d'éléments de diagramme qui entretiennent entre eux des relations étroites
- clarté / partage du travail dans une équipe
- un paquetage possède des éléments de modélisation et peut en importer.
- forme générale du système : hiérarchie de paquetage + relations de dépendances entre paquetages (<< importe >>, << accède >>)
- *sous-système* : spécifie le comportement collectivement offert par ses éléments (en terme d'opérations)

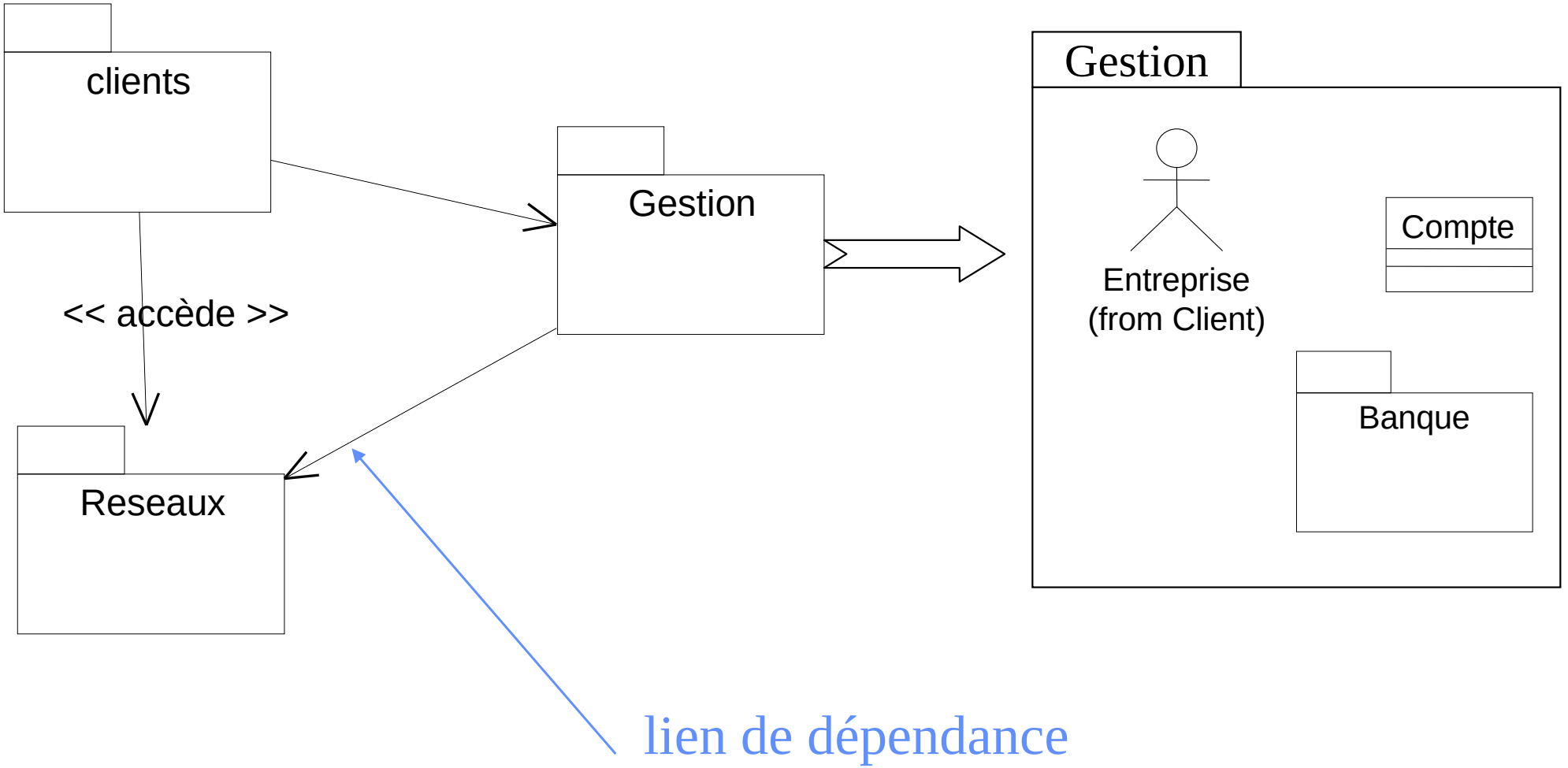
Paguetages: un espace de nommage



nomDuPackage : : NomDeLaClasse

Paquetages : exemple

But : organiser un modèle objet



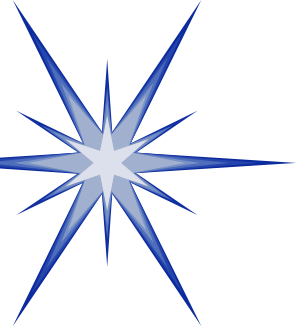


Diagramme de Classes

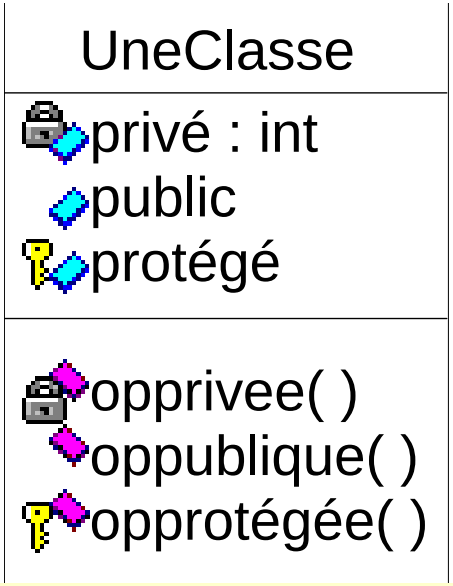
Diagrammes de classes

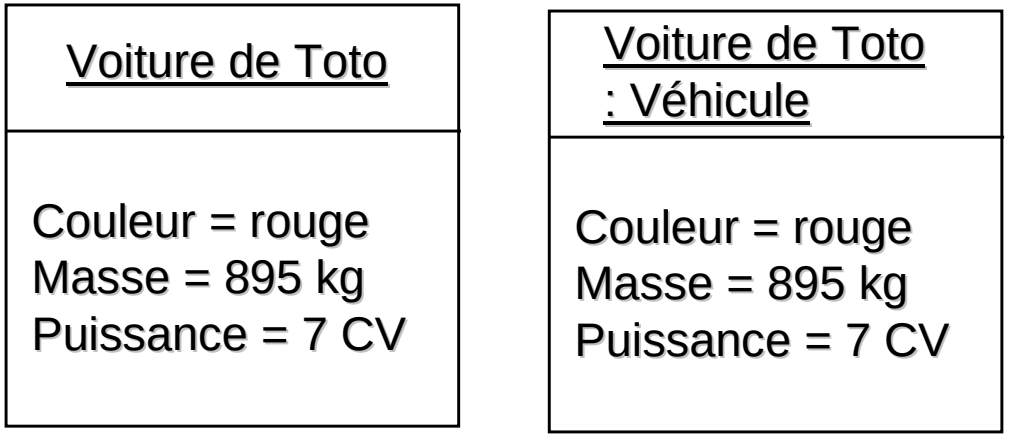
- Vue logique / statique / structurelle d'un système : les classes et leur relations
- Dans UML
  - Classes : structures et comportements
  - relations : associations, agrégations, dépendances, généralisation/spécialisation, noms de rôles
  - contraintes
  - indicateurs de multiplicité et de navigation
- Mais : toujours difficile de déterminer les classes → il faut des méthodes
  - Modèle du domaine stable
  - Objets dans les diagrammes de séquence et de collaboration,

Diagramme d'objets

- **Objet** : unité atomique formée de l'union d'un état et d'un comportement
- **Etat d'un objet**: valeurs instantanées de tous ses attributs(l'état d'un objet à un moment donné est la conséquence des comportements passés).

(Rose)





Nom objet : classe

Visibilité et **portée** des **attributs** et des **opérations :**

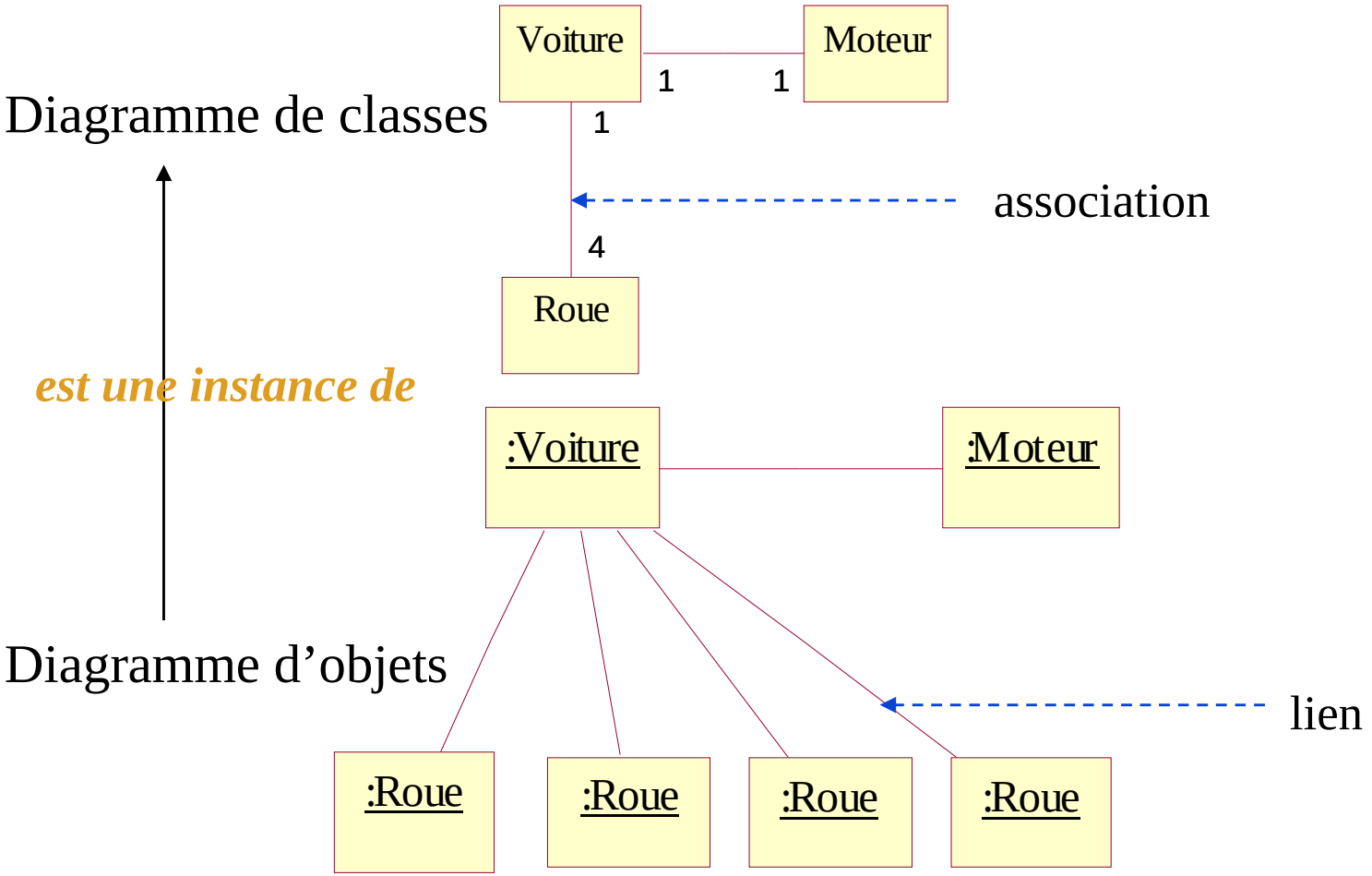
Public : l'élément est visible pour tous les clients de la classe

Protégé ***(protected)***: l'élément est visible pour les sous-classes de la classe

Privé ***(private)***: l'élément est visible pour la classe seule

Diagramme de classes & objets

Exemple



Liens entre objets / *relations* entre classes

Liens et relations

- Possibilité de communication entre objets :

  interactions entre objets (cf. diagrammes)

  → relation entre classes

- 3 grands types (voir plus loin)
  - ***Associations*** ("est produit par", "est affilié à", "se trouve à", "est conduit par"…) : dépendance entre classes, communication entre objets

    Ex: « un lecteur lit un livre » (forme verbale)
  - ***Agrégation/composition*** ("fait partie de") : objets composites / composants

    Ex : « un train est composé de wagons »
  - ***Généralisation/spécialisation*** ("est une sorte de") : héritage entre objets,

    Exemple : « un chat *est une sorte* d'animal »

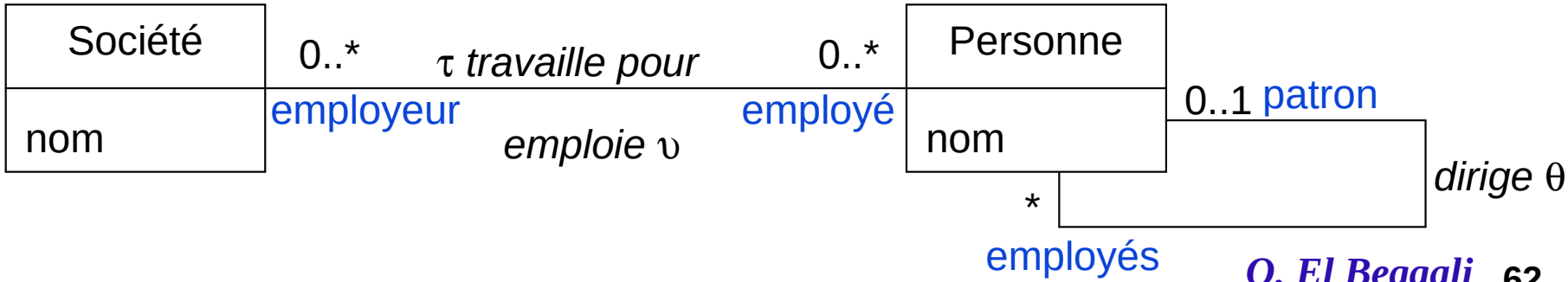
Associations

- Connexion bidirectionnelle entre classes
- Notation générale :

Nom : forme verbale, sens de lecture avec flèche

Rôles : forme nominale, identification extrémité association

Multiplicité : 1, 0..1, M..N, *, 0..*, 1..*

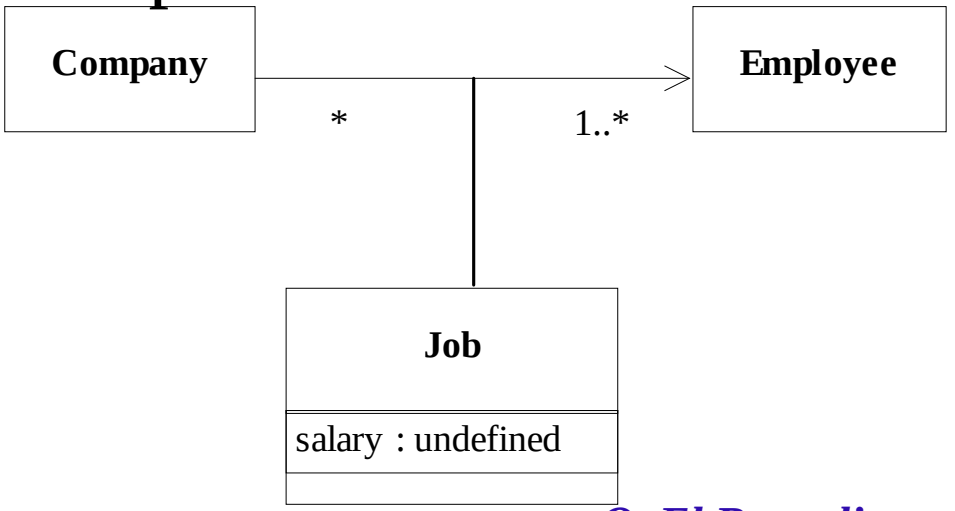


Multiplicité des associations

1	Un et un seul
0..1	Zéro ou 1
N	N (entier naturel)
M..N	De M à N (entiers naturels)
0..*	De zéro à plusieurs
1..*	De 1 à plusieurs

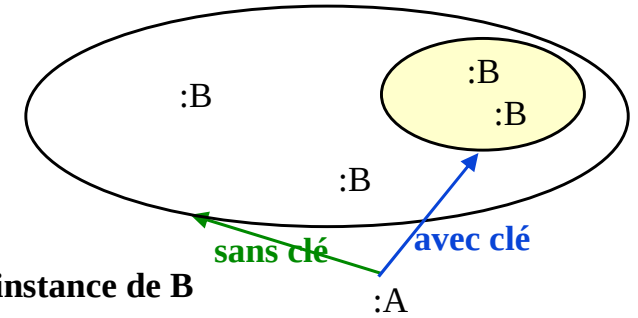
Classes-Associations

- Une classe-association est une association qui est aussi une classe.
- Les classes-associations sont utilisées lorsque les associations doivent porter des informations
- Il est toujours possible de se passer des classes-associations.

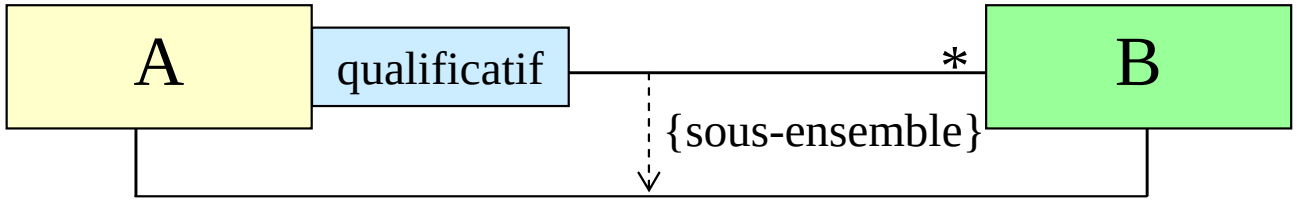


Associations qualifiées (1)

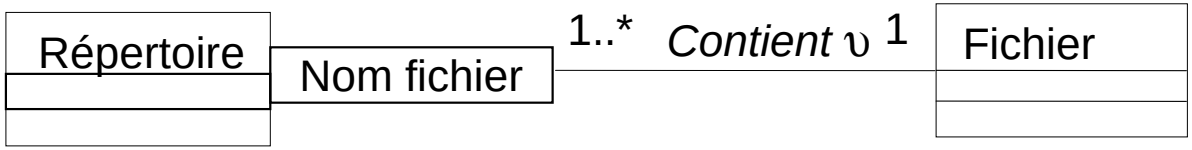
- Restrictions : sélection d'un sous-ensemble des objets qui participent à l'association à l'aide d'une clé

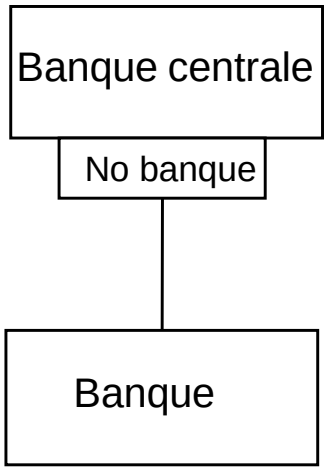


{L'instance de A , valeur du qualificatif } identifie le sous ensemble des instance de B



Exemple :





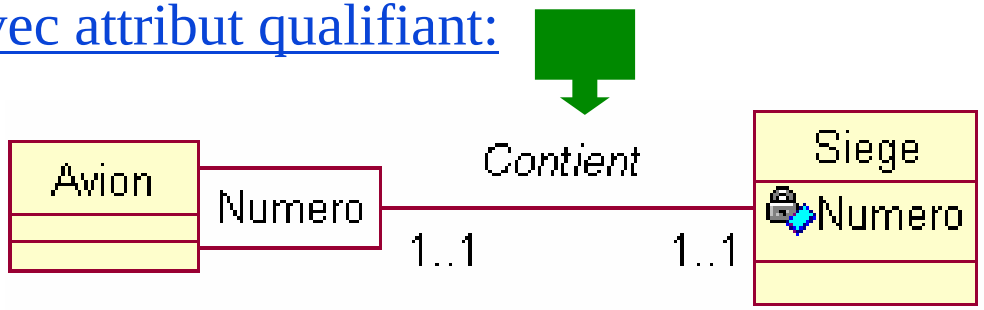
Remarque : Le rôle d'un qualificateur est de réduire la cardinalité d'une association et joue le rôle semblable à une clé primaire dans une BDR.

Associations qualifiées (2)

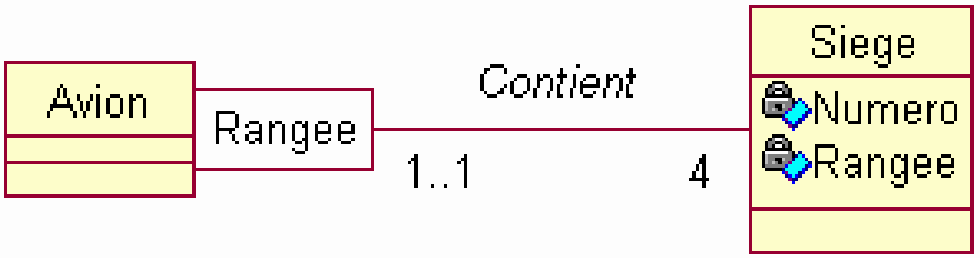
exemple

ce schéma ne permet pas de dire que chaque siège a un numéro qui est unique pour chaque avion.

Avec attribut qualifiant:



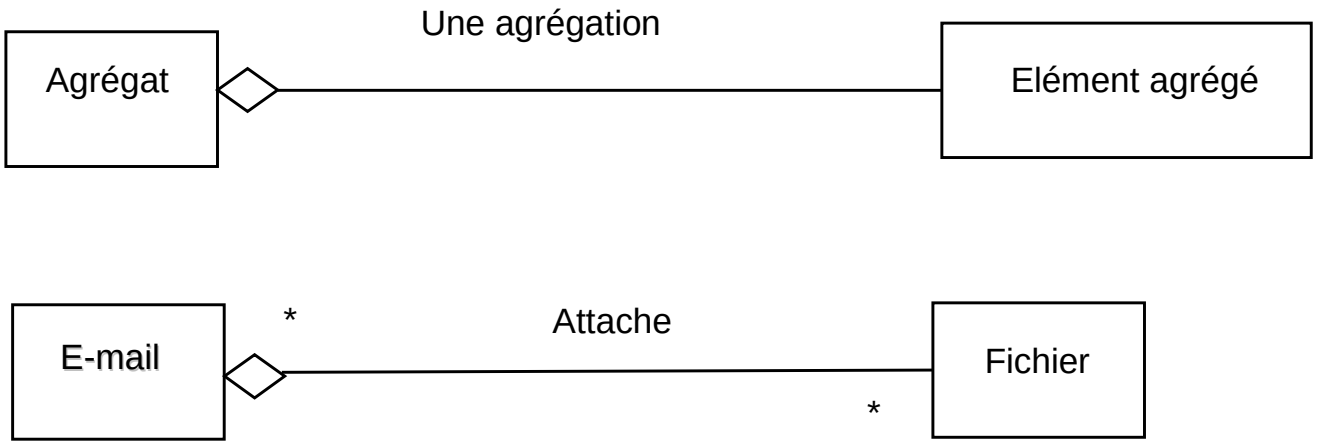
« un avion contient un siège pour un numéro donné ».



dans un avion, pour une rangée donnée, il y a 4 sièges.

L’agrégation

- Association non symétrique
  - Une des extrémités joue un rôle prédominant par rapport à l’autre

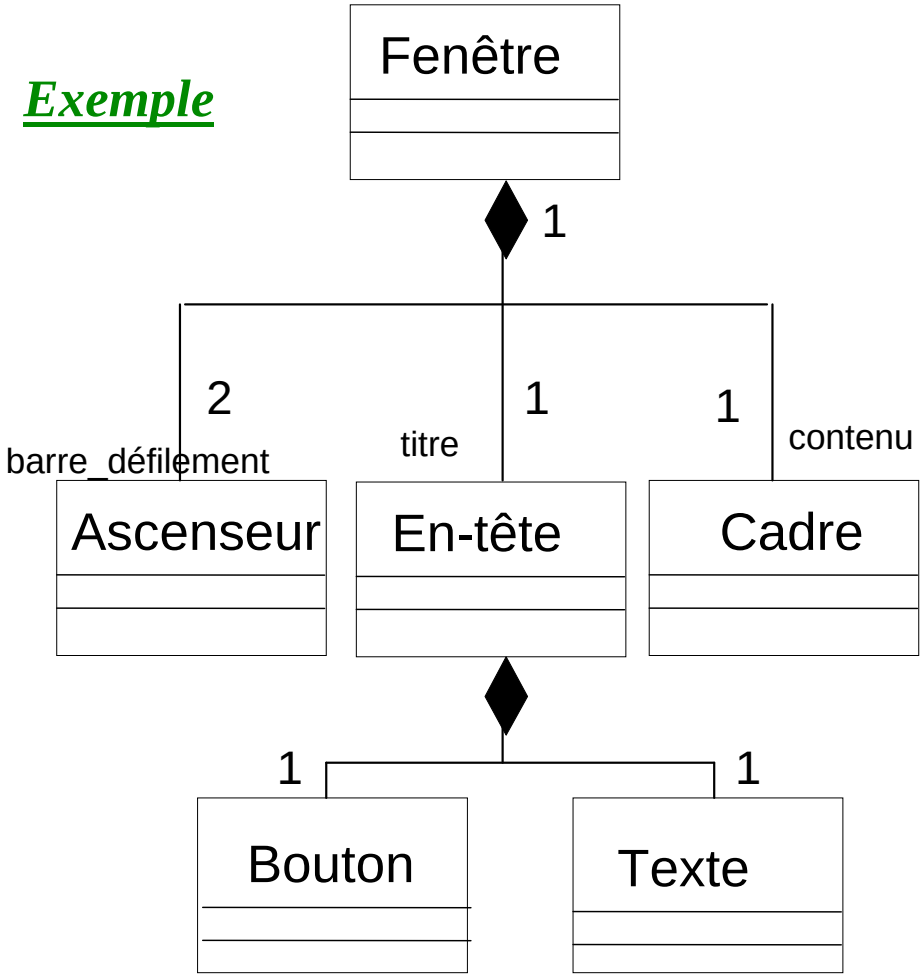


Composition d'objets

- Cas particulier d'agrégation ⟹ contenance physique

« Si destruction objet composé ⟹ destruction des composants »

Exemple



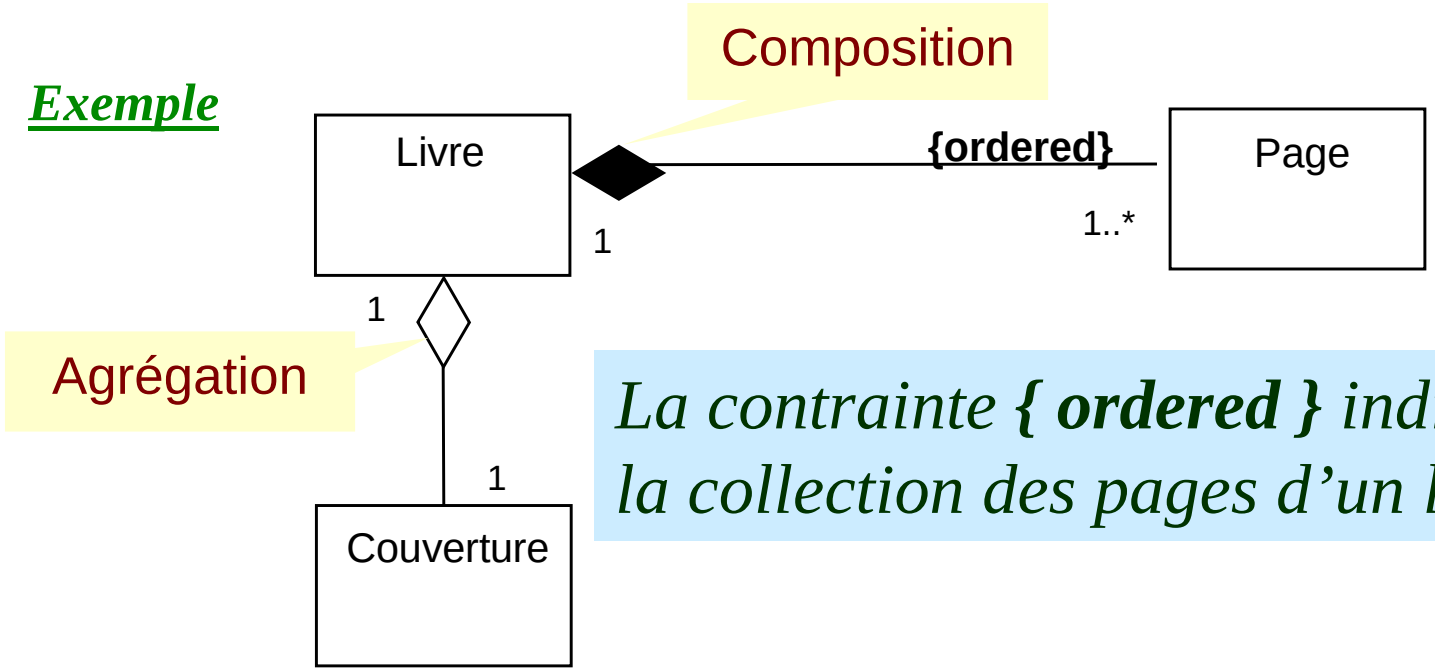
Fenêtre
barre_défilement : ascenseur titre : en-tête contenu : cadre

La composition

- Agrégation forte
- Destruction du composite (x est une partie de y?)

  => Destruction automatique de tous ses composants

Exemple



*La contrainte **{ ordered }** indique que*
la collection des pages d'un livre est ordonnée

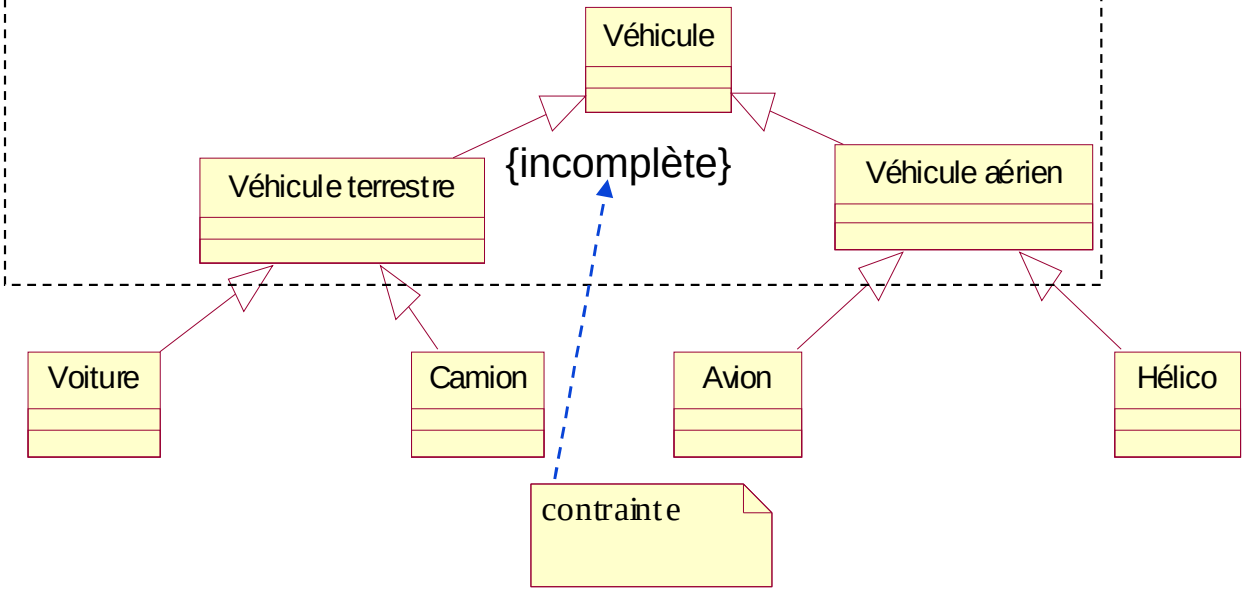
L’héritage

- Moyen de réaliser la classification ou l’organisation en classes → vocabulaire à réserver à l’implantation
- Principe de substitution : toutes les propriétés de la classe ‘parent’ doivent être valables pour les classes ‘enfant’
- Héritages: simple / multiple

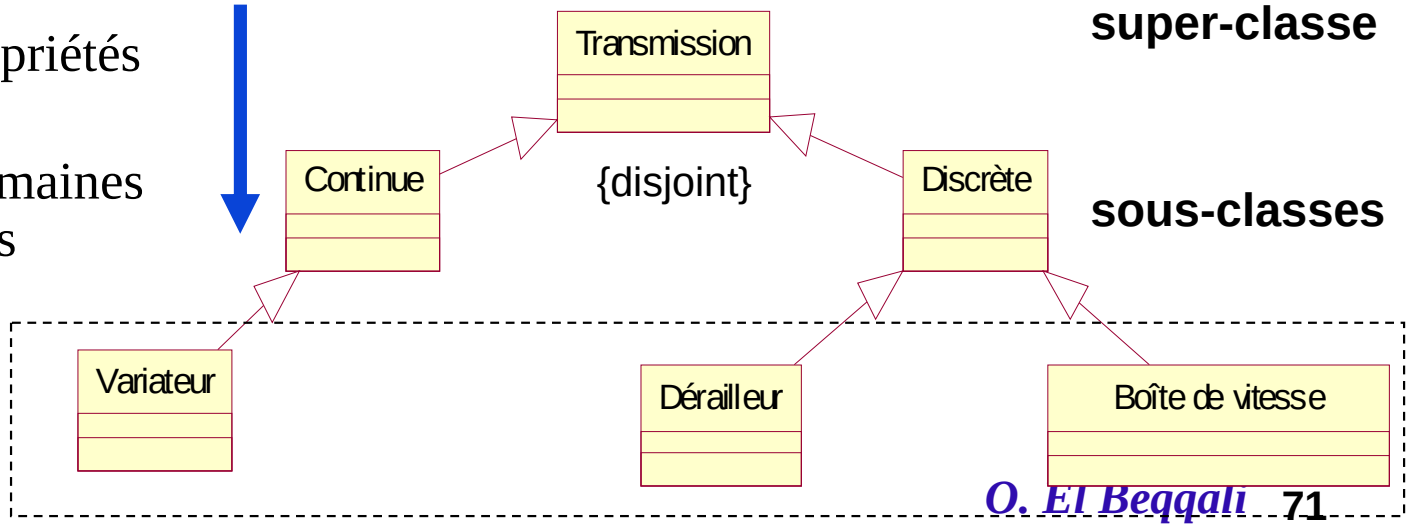
Diagramme de classes

Généralisation et spécialisation (1)

- ***Généralisation*** : regroupement, « est-un », « sorte-de »



- ***Spécialisation*** (symétrique généralisation): raffinement de classe
  - par extension de propriétés (ajouter un attribut)
  - par restriction de domaines de valeurs d'attributs

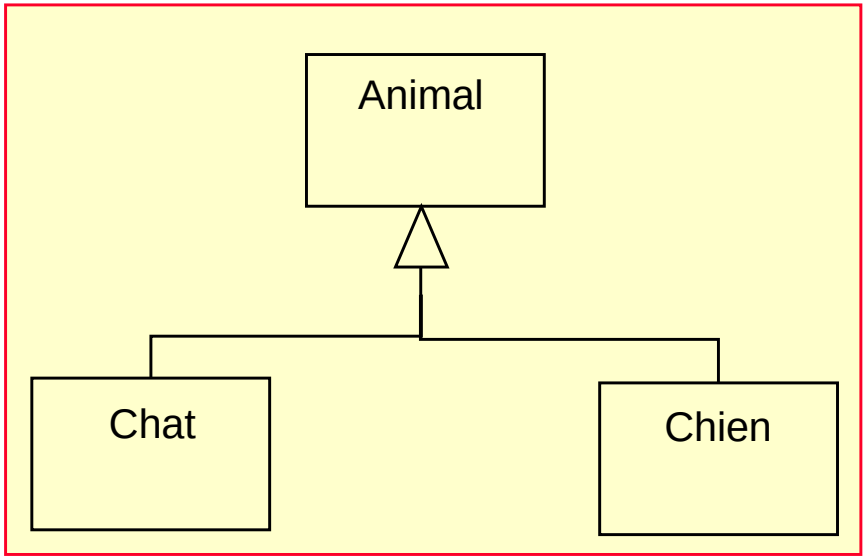


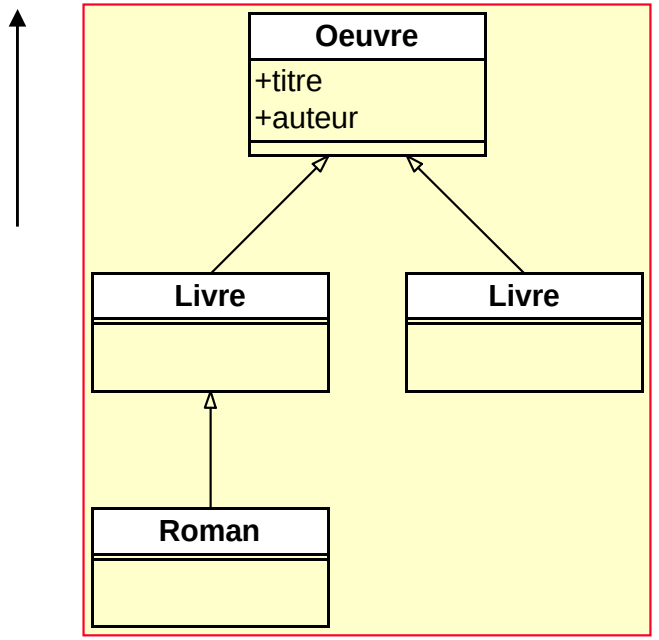
Généralisation/spécialisation (2)

- Relation de classification entre un élément plus général et un élément plus spécifique

  ‘est un’ / ‘est une sorte de’ (‘***is a***’ / ‘***kind of***’)

Exemples (héritage) :

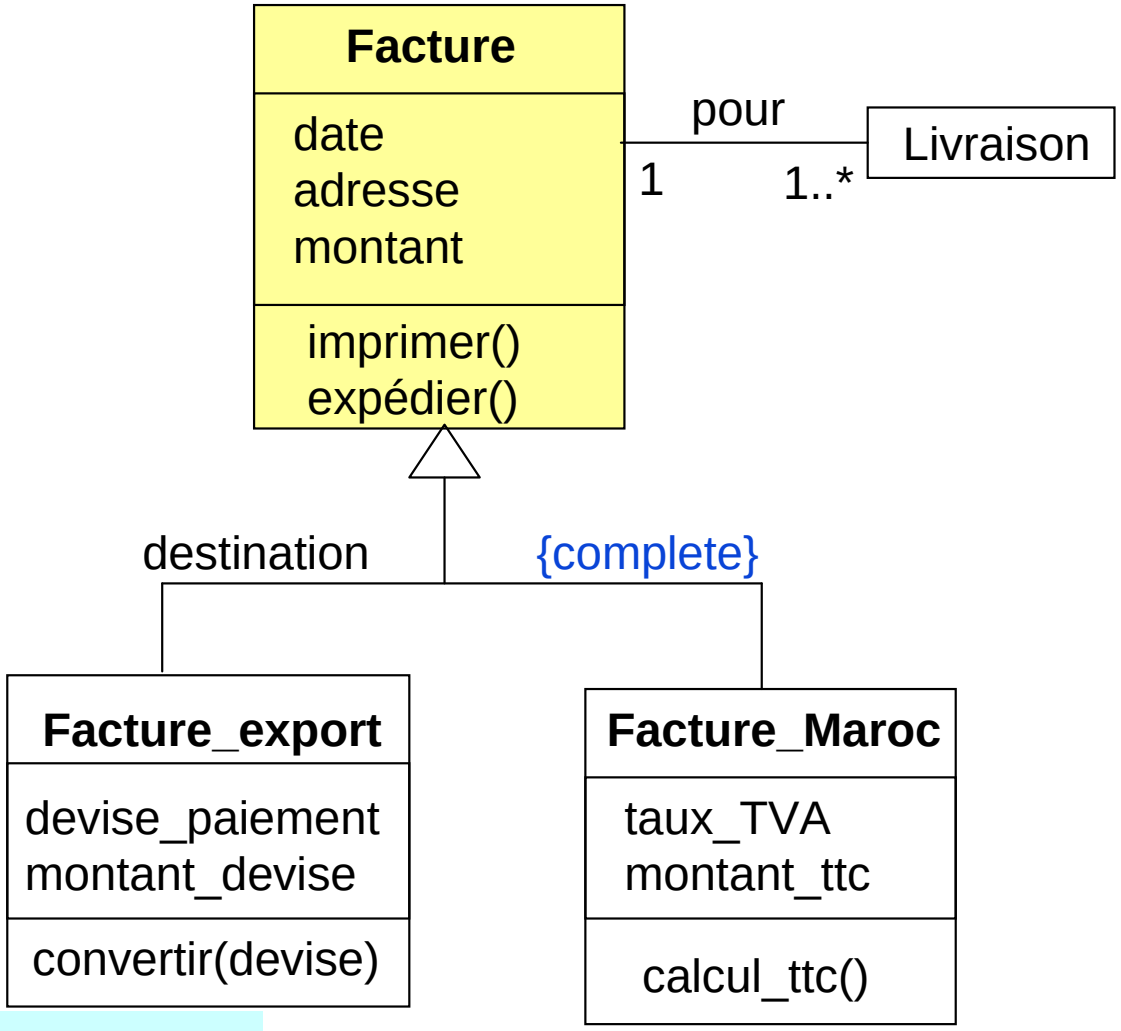




Hiérarchies de classes

- Contraintes :
  {complète},
  {incomplète},
  {disjoint},

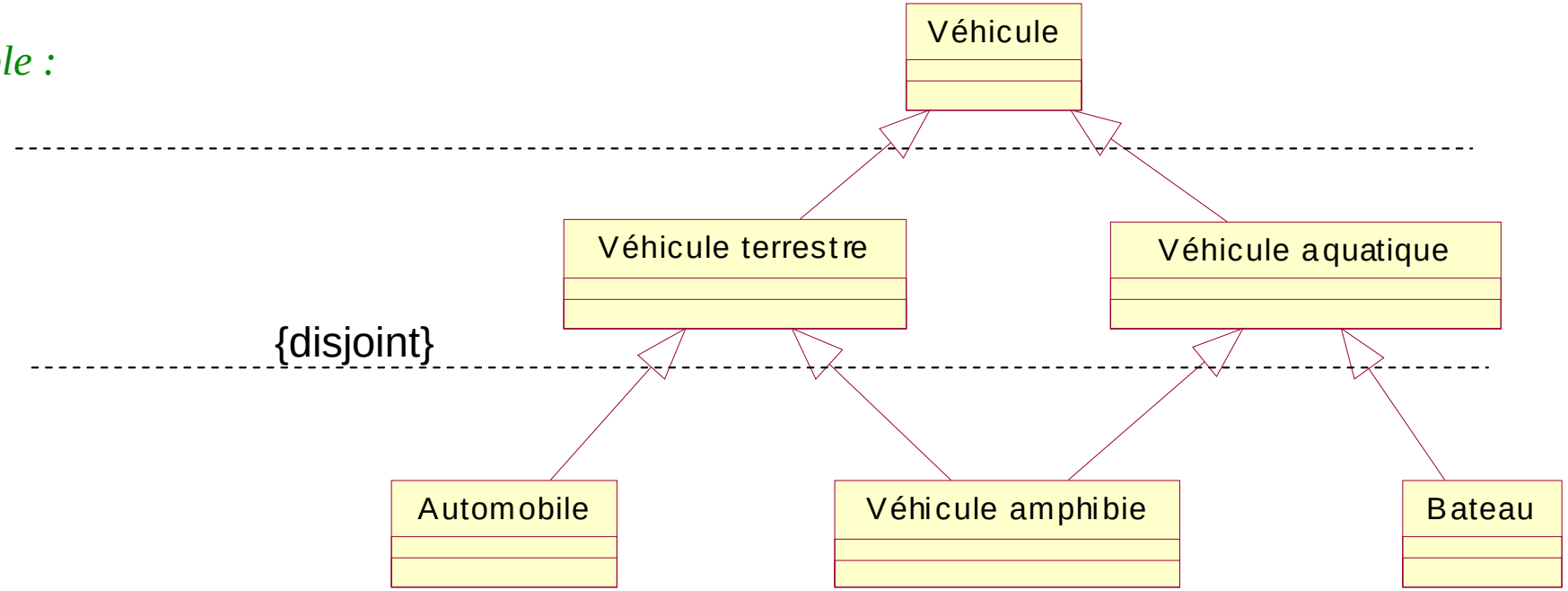
- Attributs / opérations communs sont montrées au niveau le plus haut.



La contrainte {complète} indique que la généralisation est terminée et qu'il n'est plus possible de rajouter des sous-classes.

Héritage multiple

Exemple :



- Problème : conflits d'héritage (un attribut vitesse_max en km/h, en Nœuds → qu'hériter ?)
- En général, très difficile à utiliser : certains langages négligent l'héritage multiple

Construction du diagrammes états – transition *Statecharts (1)*

- Attaché à une classe ou à un cas d'utilisation, il doit présenter une classe par rapport à ses états possibles et aux transitions qui le font évoluer.

Lorsque [évènement] => changement d'état
L'action peut être conditionnelle : [condition] action.

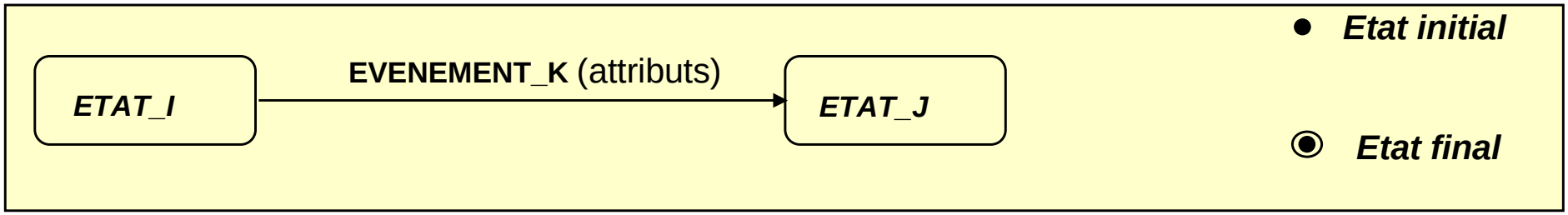
- **Un état** = image de la conjoncture instantanée des valeurs des attributs d'un objet
  & Présence ou non de ses liens à d'autres objets.

- **Une transition** représente un changement d'état ; elle est déclenchée par un événement.

  - Transitions
    - peuvent être automatiques
    - peuvent être conditionnées
  - Evènements
    - déclenchent les transitions

Construction du diagrammes états – transition *Statecharts (2)*

Un ***Statechart*** est un graphe orienté dont les nœuds sont des ***états*** et les arcs sont des *transitions*.



- Entrée : les diagrammes de séquence et les diagrammes de classes
- Sortie : les diagrammes états-transitions
- Transformation mise en œuvre

une classe => un diagramme états-transitions

Statecharts (3)

- Dans un état, une activité est une opération séquentielle qui se termine d'elle-même au bout d'un certain temps, ou opération continue qui dure indéfiniment (peut être interrompue par l'arrivée d'un événement).

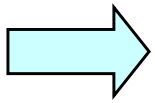
- Contrôle des opérations:
  - Une ***activité*** est une opération qui nécessite un certain temps d'exécution; elle est associée à un état.
  - Une ***action*** est une opération instantanée (ou de durée non significative); elle est associée à un événement.

• **Actions associées à un état :**

action d'entrée **:** exécution instantanée de l 'action lors de l'entrée dans l'état.

action de sortie **:** exécution de l 'action au moment de quitter l'état.

actions internes **:** exécution d'une action sur *événement* sans changer d'état.



ETAT

faire : activité_1
entrée / action_2
sortie / action_3
événement / action_4

Diagrammes d'états-transitions - *Exemple*

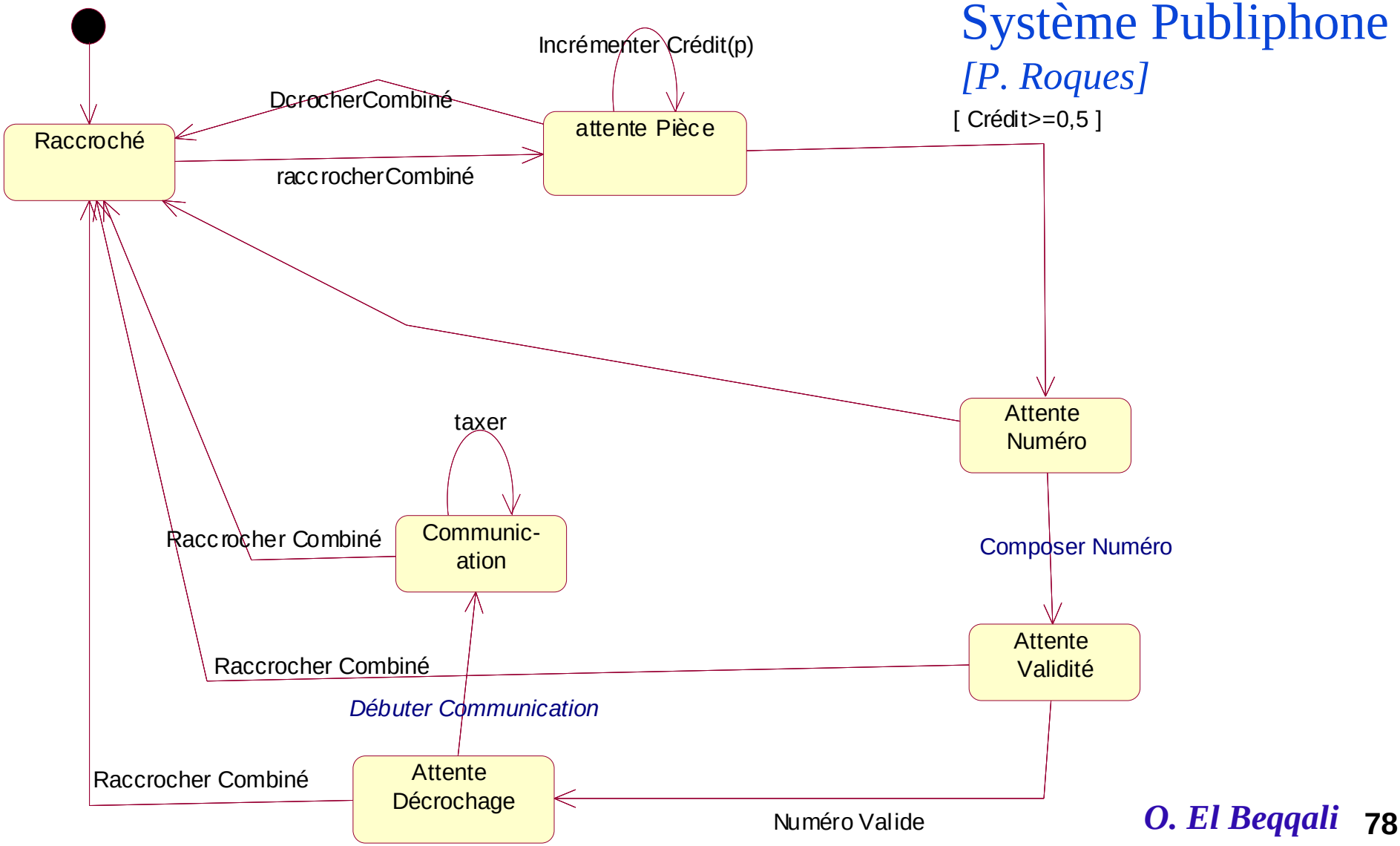


Diagramme d'activités

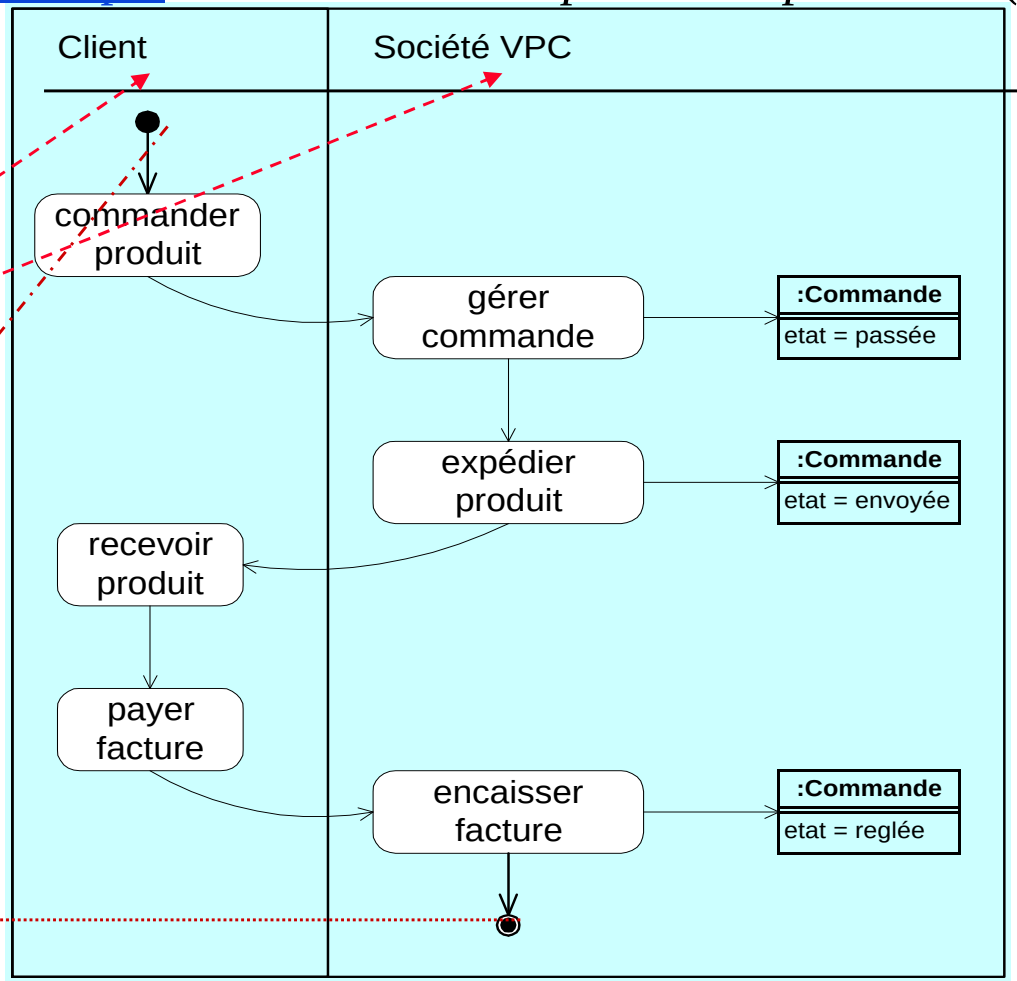
On s'intéresse plus aux actions qu'aux états, il montre l'activité et le fonctionnement d'une opération d'une classe, par exemple.

Exemple : *activité de vente par correspondance (VPC)*

***Montre le cycle de vie d'une classe donnée.**

***En général divisé en couloirs d'activité**

(Swimlanes)



Début

Fin

Diagramme de composants (1)

- Un diagramme de composant représente les composants logiciels d'un système
- Diagramme utilisé dans la phase de conception / réalisation : éléments physiques constituant le système et leurs relations
- Décrivent les éléments physiques et leurs relations dans l'environnement de réalisation sous forme de modules.
- Modules : toutes sortes d'éléments physiques (fichiers sources/données, exécutables,bibliothèques)

Diagramme de composants (2)

Exemple :

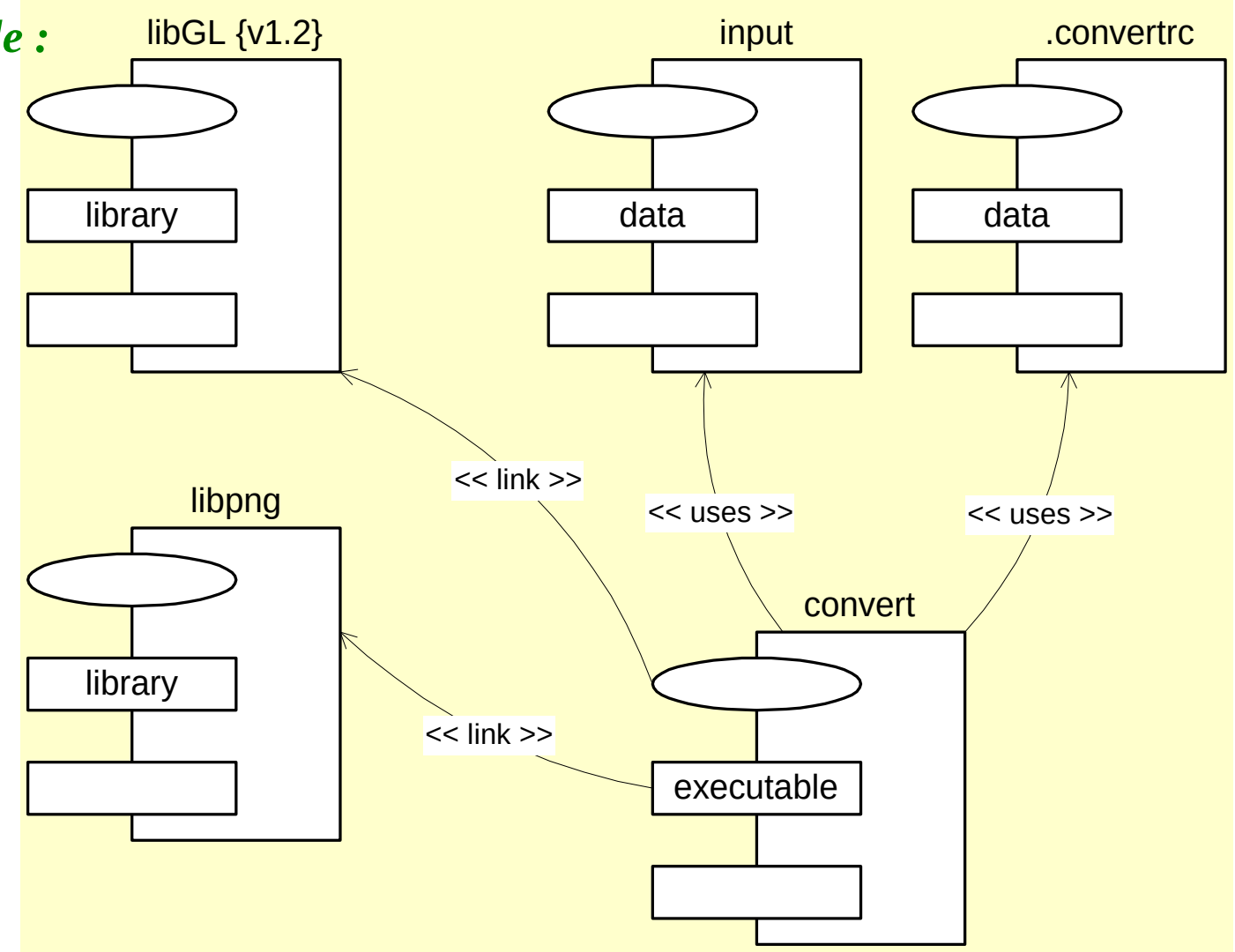


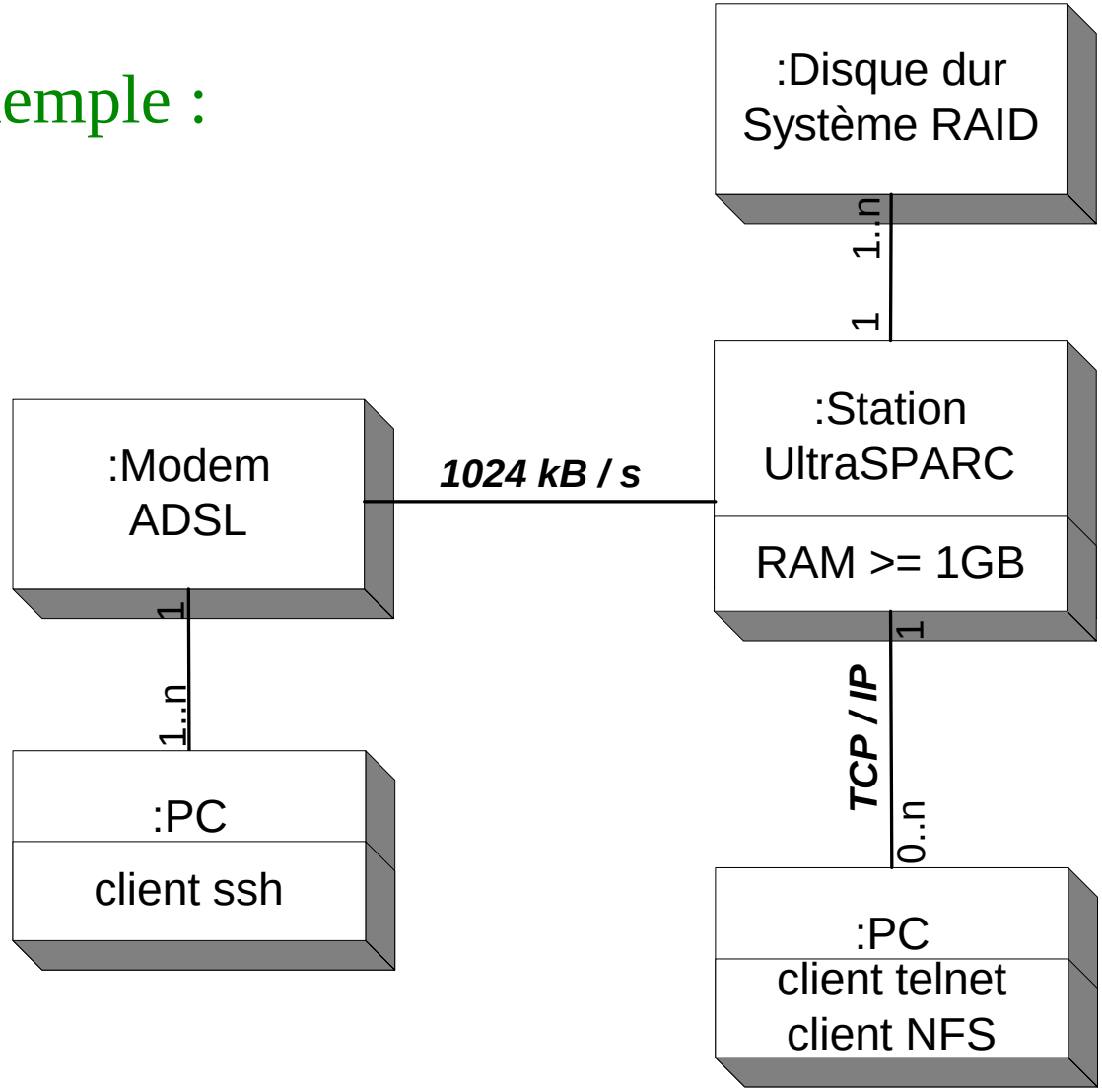
Diagramme de déploiement (1)

Montrent la disposition et l'organisation physique des différents matériels qui entrent dans la composition d 'un système et la répartition des programmes exécutables sur ces matériels.

➡ La façon pour déployer les différentes éléments d'un système

Diagramme de déploiement (2)

Exemple :



Démarche d'application D'UML

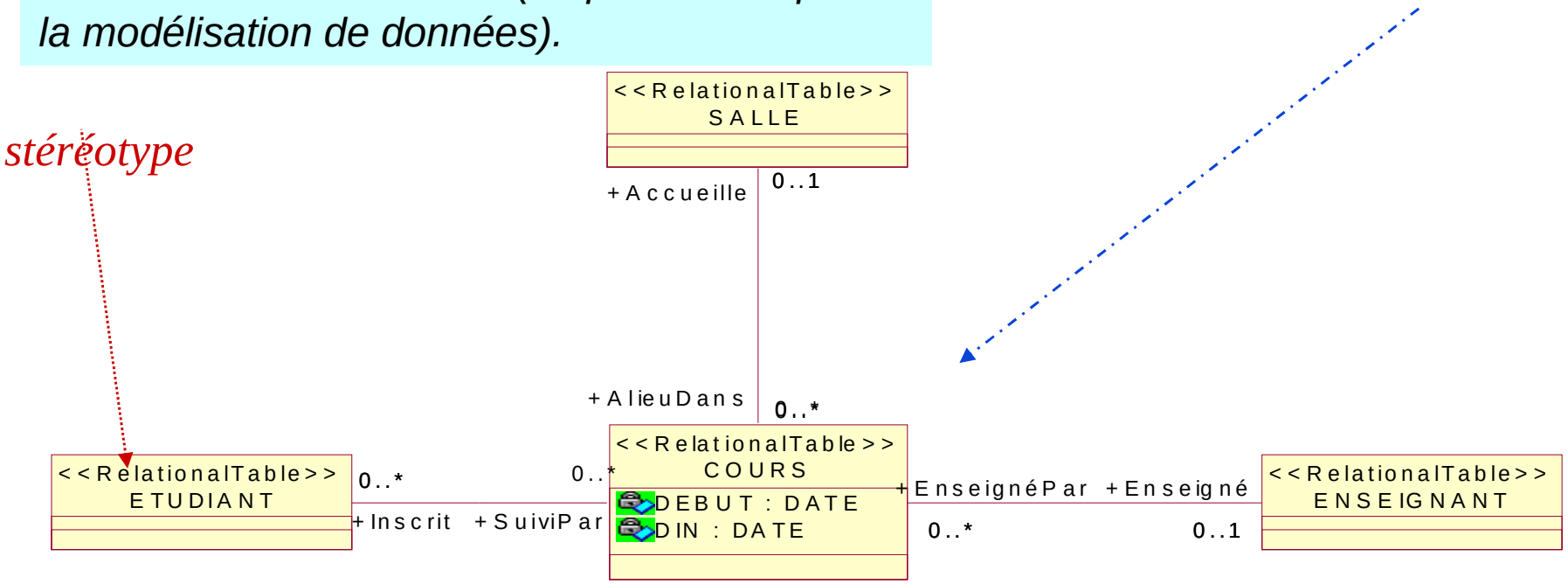
Étape 1 : **élaboration d'un diagramme de contexte du système à étudier:**
Précision du champ du système à étudier
Étape 2 **: identification et représentation des cas d'utilisation :**
Identification des fonctions du système
Étape 3 : **description et représentation des scénarios:**
chaque cas d'utilisation se traduit par un certain nombre de scénarios.
Chaque scénario fait l'objet d'une description textuelle. Chaque scénario est ensuite décrit sous forme graphique à l'aide du diagramme *de séquence* et/ou *diagramme de collaboration.*
Étape 4 : **identification des objets et classes**
Étape 5 : **élaboration du diagramme de classes**
Étape 6 : **élaboration du diagramme état-transition** :
pour chaque classe importante c'est à dire présentant un intérêt pour le système à modéliser, un diagramme état-transition est élaboré.
Étape 7 **: consolidation et vérification des modèles:**
Itération des étapes 3, 4, 5 et 6 => niveau suffisant pour la description du système.

UML & Bases de données (1)

2 approches

-DataModeler de Rose (seul ou intégré) , outil de développement d'applications et de modélisation métier *(en plein essor pour la modélisation de données).*

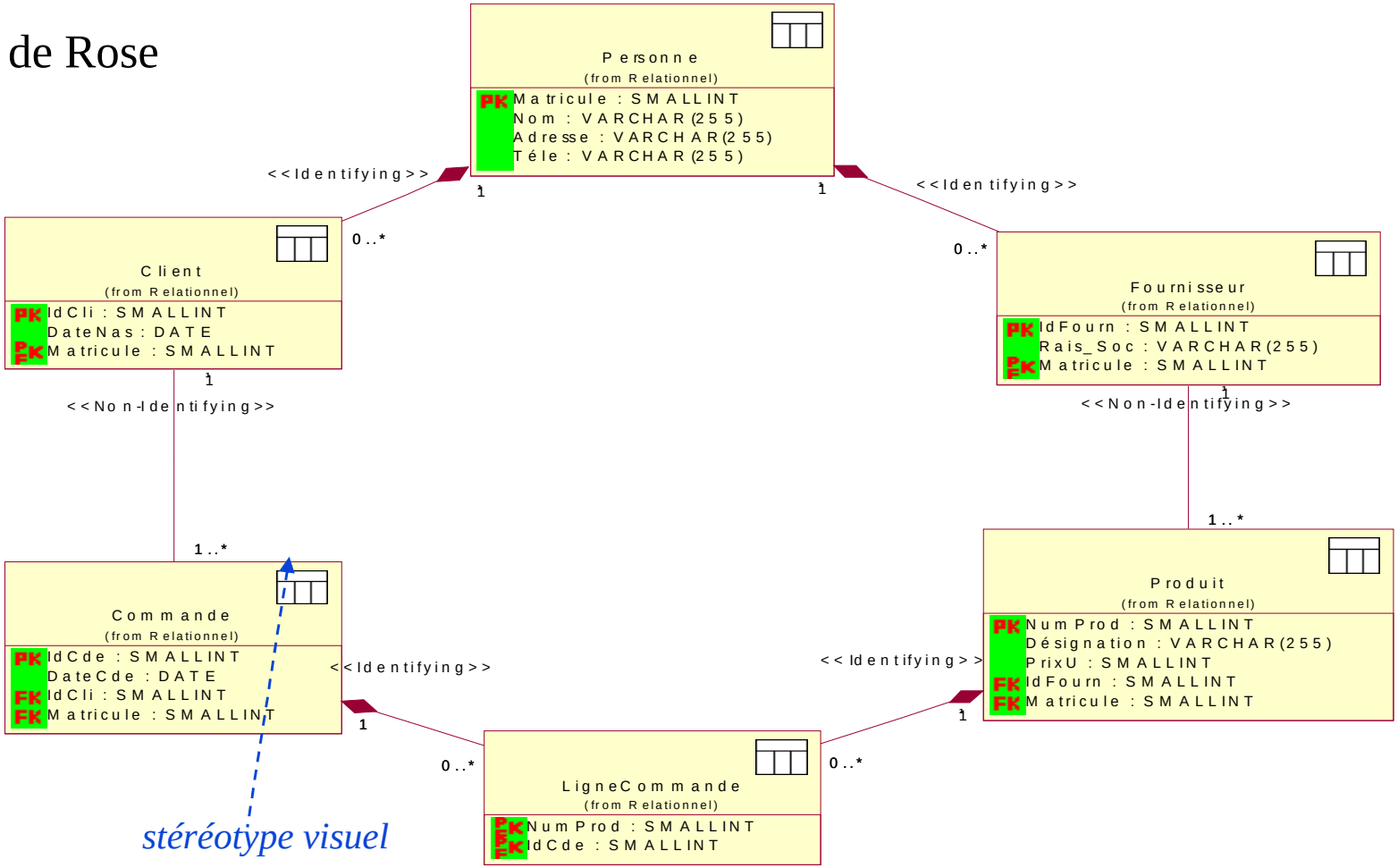
-Assistant de création de schémas de BD *(Oracle)*



UML & Bases de données (2)

DataModeler de Rose

schéma de BD généré à partir d'un diagramme de classes (package)



générateurs UML -> code

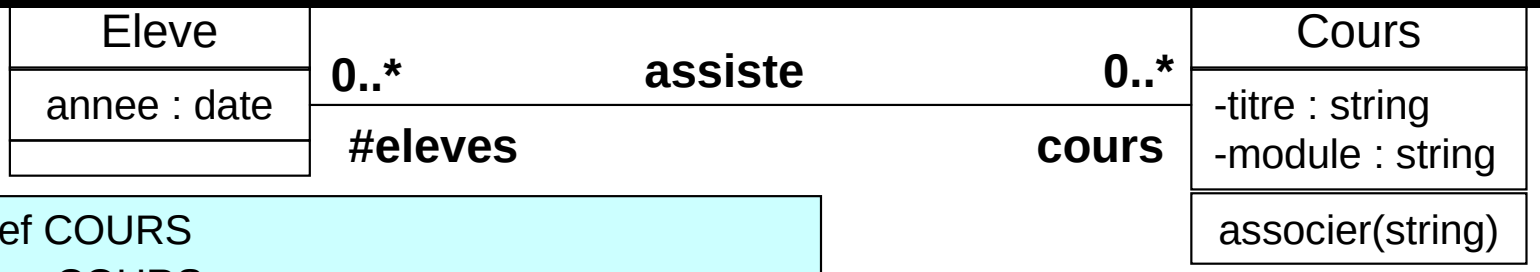
- Différents types :
  objet , bases de données
- Automatique ?
  - information dans les modèles insuffisante
  - paramétrage des outils : traduction des associations, méthodes engendrées automatiquement
  - degré de traduction dépend du langage cible
- Modèle structurel largement traduit
  - squelette des classes (à compléter)
  - héritage, associations,

Outils standards

- Rational Rose
  - Outil de plus important du marché
  - http://www.rational.com (racheté par IBM)

- Together
  - Outil fortement couplé avec Java
  - http://www.togethersoft.com
  - Racheté par Borland
- ArgoUML
  - Outil Open Source
  - http://argouml.tigris.org
- Visio
  - Outil non complet de microsoft
  - http://www.microsoft.com/office/visio

Traduction en objet (C++)



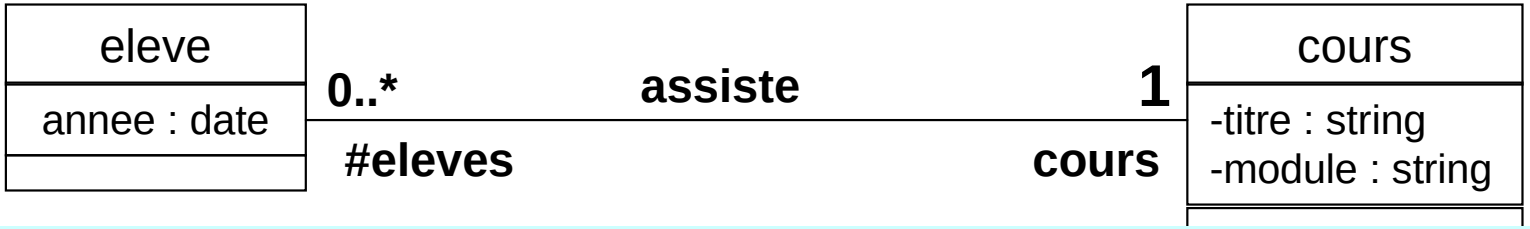

```
#ifndef COURS
#define COURS
class Eleve
class Cours {
	public:
		void associer(un_module : string) ;
		void set_titre(const string value) ;
		const string get_titre() const ;
		void set_module(const string value) ;
		const string get_module() const ;
	protected:
		Eleve* Eleves ;
	private:
		string titre ;
		string module ;
}
#endif
```

```
#include "Eleve.h"
#include « Cours.h"

void Cours::associer(un_module : string)
{
	/* A ECRIRE */
}

void Cours::set_titre(const string value)
	{ titre = value ; }
const string Cours::get_titre()
	{ return titre ; }
void Cours::set_module(const string value)
	{ module = value ; }
const string Cours::get_module()
	{ return module ; }
```

Traduction en relationnel

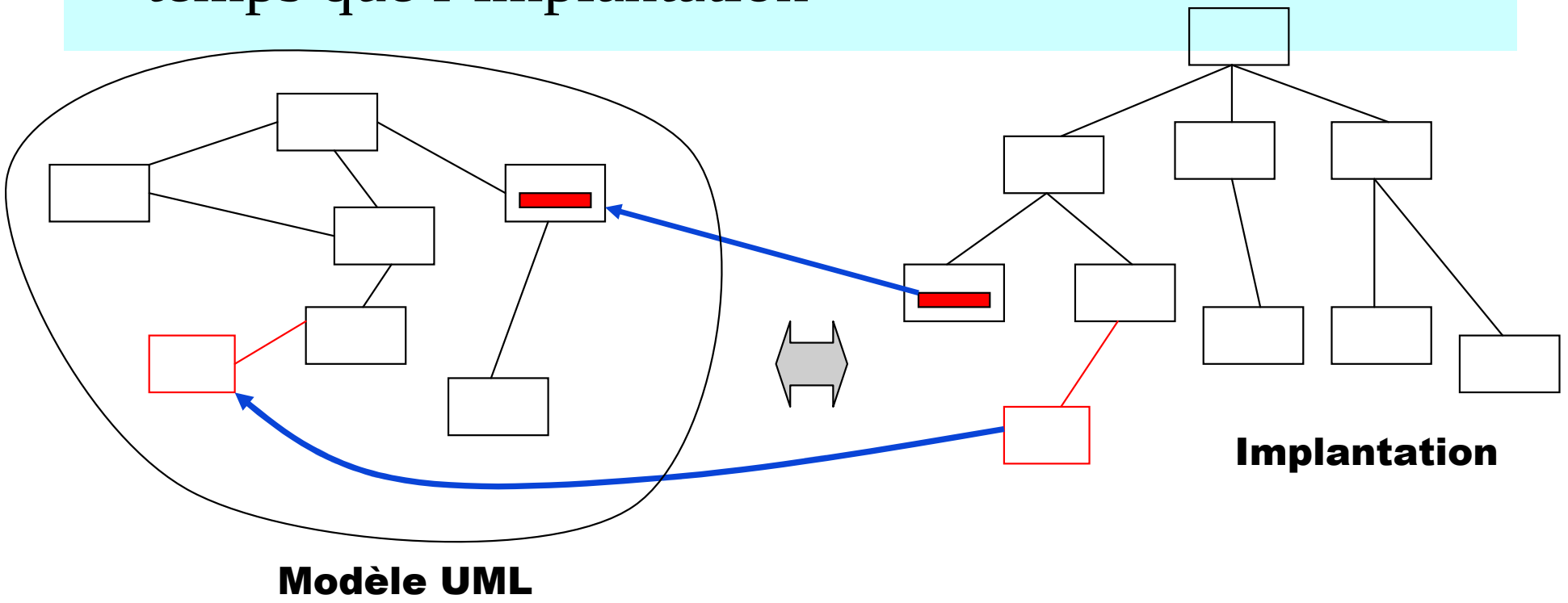



```
CREATE TABLE eleve (
      eleve_id NUMBER (5) ,
      annee DATE,
      PRIMARY KEY (eleve_id)
) ;

CREATE TABLE cours (
      cours_id NUMBER (5) ,
      titre CHAR (128) ,
      module CHAR(48) ,
      PRIMARY KEY (cours_id),
      eleve_id NUMBER (5)  REFERENCES eleve(eleve_id)
) ;
```

Nécessité de la rétro-ingénierie (reverse engineering)

Objectif ⇒ Faire évoluer le modèle en même temps que l'implantation




Conclusion

Conclusions

- Propriétés d'UML
  - Unification des concepts de modélisation
  - Puissance d'expression
    - Nombreux formalismes (issus de méthodes existantes)
    - Mécanismes d'extension inclus (stéréotypes)
  - Description par un méta-modèle *(syntaxe et sémantique des modèles)*
  - Compromis formalisation / niveau d'abstraction

  (impression qu'UML est une méthode)

- UML est devenu un standard international : adopté un peu partout (universel)
  - les modèles sont simples, faciles à lire et à communiquer
  - difficulté pour des systèmes très complexes, d'où la nécessité d'outils
- Les outils puissants sont nombreux (**Rose**, Rhapsody, Select, Objectoring....)

Bibliographie

Présentation d'UML non exhaustive , pour la description exacte de toute la syntaxe et la sémantique :

Références :

- www.omg.org, www.rational.com
- A. Muller : « Modélisation objet avec UML », Eyrolles, 2000
- P. Roques, « UML par la pratique », Eyrolles, 2004
- J. Rumbaugh, I. Jacobson, and E. Booch, « The Unified Modeling Language Reference Manual ». Reading, MA: Addison-Wesley, 1999.
- Web….