

UML

Unified Method Language

Langage de modélisation pour la
Conception Orientée Objet



UML – Plan du cours

- ✚ Introduction – Historique – Grands Principes
- ✚ Analyse, Conception et Ingénierie
- ✚ Diagrammes Structurels (vue statique)
- ✚ Diagrammes Comportementaux (vue dynamique)
- ✚ Conclusion

Introduction



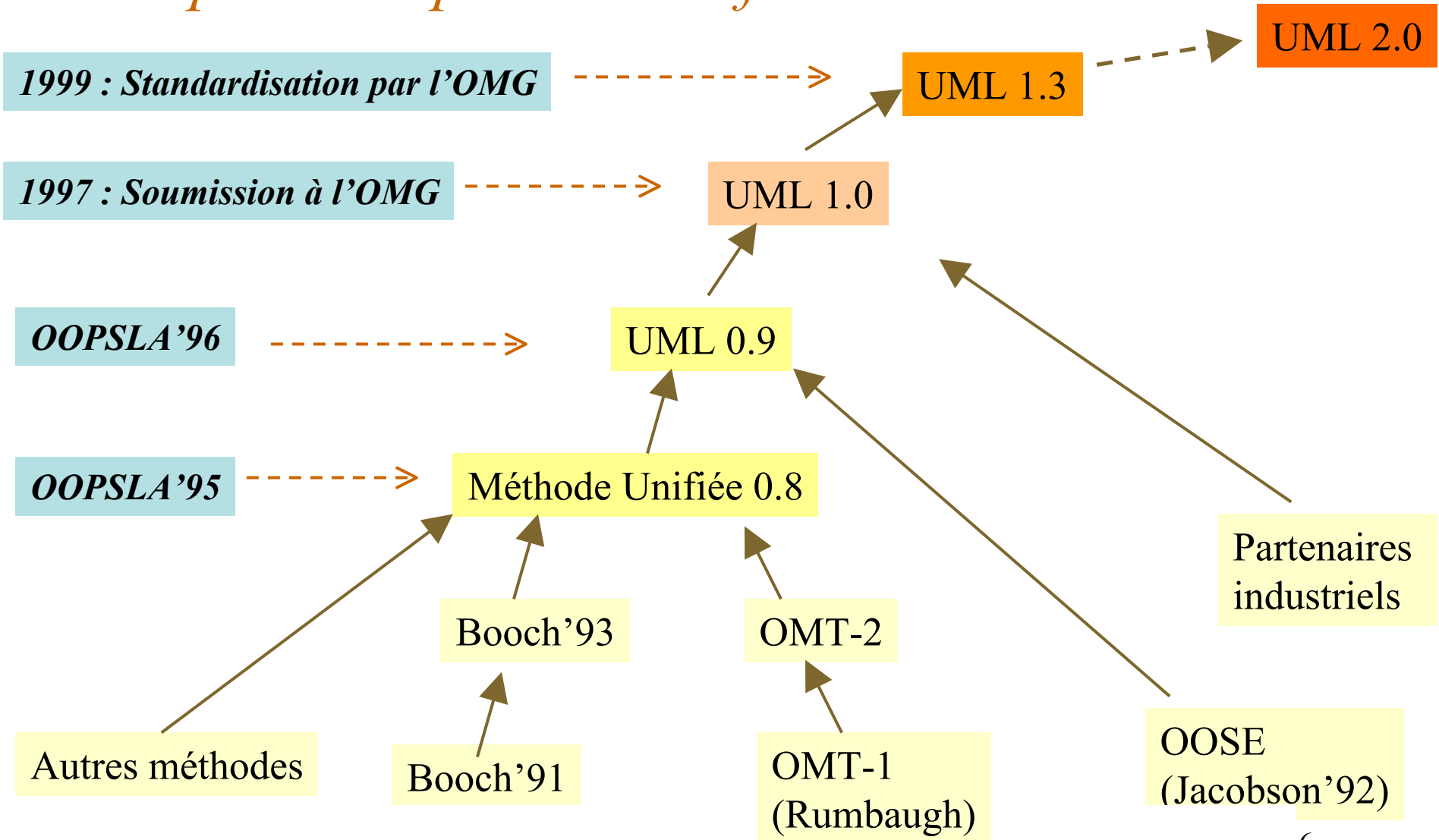
Introduction

- ✿ UML est un ***Langage de modélisation*** permettant de décrire un ***projet et ses composants*** à la fois d'un point de vue ***statique*** et d'un point de vue ***dynamique***.
- ✿ Il est issu de diverses méthodes d'analyse et de conception Orientée Objet
 - ▣ Rumbaugh (OMT)
 - ▣ Booch (OOD)
 - ▣ Jacobson (OOSE)

Historique



Principales Étapes de la définition d'UML





Historique

- ❁ Plusieurs méthodes d'analyse et de conception entre 1988 et 1996:
 - ❑ Sally Shlaer et Steve Mellor: 2 ouvrages sur analyse et conception (1989 et 1991)
 - ❑ Peter Coad et Ed. Yourdon: approches « orientées prototypes » (1991,1993 et 1995)
 - ❑ Conception pilotée par responsabilités (communauté smalltalk 1990) et cartes CRC (Class-Responsability-Collaboration) (1989)
 - ❑ Grady Booch (Rational software) développement de systèmes en ADA (1994, 1996)



Historique

- Jim Rumbaugh (laboratoires de recherche General Electric) travaille sur OMT (Object Modeling Technique) (1991 et 1996)
- Ivar Jacobson (communicateurs téléphonique Ericsson) introduit le concept de cas d'utilisation (use case) (1992 et 1995)



Historique

- ❖ Conférence OOPSLA'94 : Grand clivage entre les différentes communautés mais idée d'une standardisation
- ❖ 1994 Jim Rumbaugh quitte G.E. pour rejoindre Grady Booch chez Rational Software pour fusionner leurs méthodes.
- ❖ OOPSLA'95 version 0.8 de la méthode unifiée
- ❖ 1995 Ivar Jacobson rejoint l'équipe.



Historique

- ❖ OOPSLA'96 Présentation d'UML
- ❖ 1996 Mise en place d'un groupe de travail par l'OMG (Object Management Group)
- ❖ 1997 Soummision d'UML 1.0 à l'OMG
- ❖ 1999 UML 1.3 est rendu publique
- ❖ 2000 publication de la spécification complète

Les grands principes d'UML



Objectifs d'UML

- ❁ Langage visuel de modélisation
 - ❑ Exploitable par des méthodes d'Analyse et de Conception différentes
 - ❑ Adapté à toutes les phases du développement
 - ❑ Compatible avec toutes les techniques de réalisation
- ❁ Indépendant des langages de programmation
- ❁ Base formelle pour la compréhension du langage
- ❁ Encourage l'utilisation de la conception orientée objet
- ❁ Supporte les concepts de développement de haut niveau: patterns, composants, frameworks.



Diagrammes UML

- Différentes façon de représenter un système :
 - Point de vue dynamique
 - Point de vue statique

- En UML cela se traduit par 9 principaux digrammes:
 - 5 diagrammes structurels (point de vue statique)
 - 4 diagrammes comportementaux (point de vue dynamique)



UML : Diagrammes structurels

- ✚ Diagramme de Cas d'utilisation (use case)
- ✚ Diagramme de classes
- ✚ Diagramme d'objets
- ✚ Diagramme de composants
- ✚ Diagramme de Déploiement



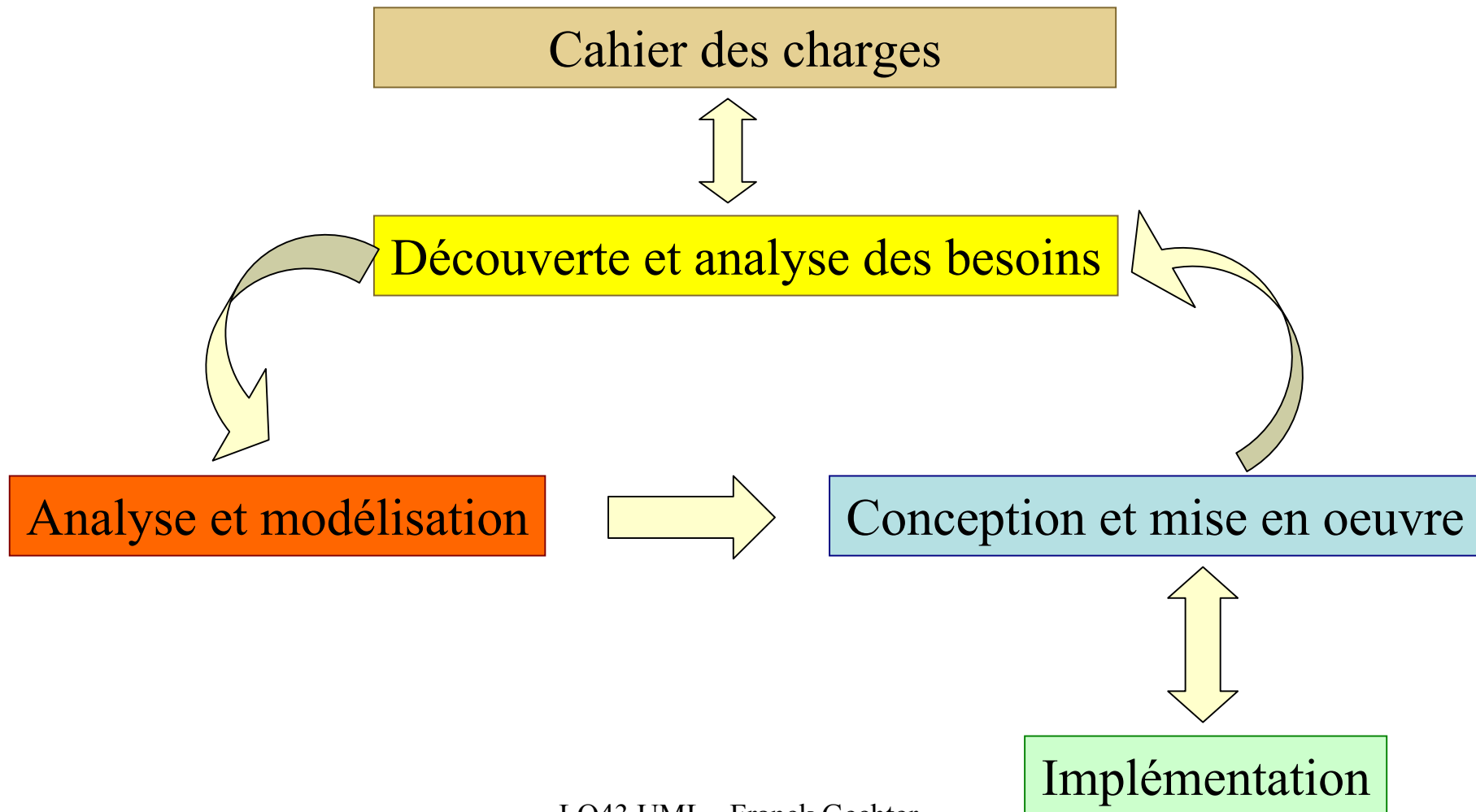
UML: Diagrammes comportementaux

- ✚ Diagramme de Séquences
- ✚ Diagramme d'Activités (États - Actions)
- ✚ Diagramme d'États – Transitions
- ✚ Diagramme de Collaboration

Processus d'ingénierie et UML



Processus d'ingénierie





Découverte et analyse des besoins

- ❖ **Diagramme des cas d'utilisation:**
décrit les fonctions du système selon le point de vue des utilisateurs (Jacobson)
- ❖ **Diagramme de séquence:**
représentation temporelles des interactions entre les objets (point de vue de l'IHM)



Analyse et modélisation

- **Diagramme des classes:** structure des données du système définie comme une ensemble de relation entre classes (association, composition, agrégation, spécialisation, implémentation,...)
- **Diagramme d'objets:** illustration macroscopique des objets et des relations qui les lient.
- **Diagramme de collaboration:** représentation des interactions entre les objets.
- **Diagramme États-Transitions:** comportement des objets d'une classe (états possibles et transitions entre ces états)
- **Diagramme d'activités:** structure temporelle d'une activité d'un objet



Conception et mise en oeuvre

- ❖ **Diagramme de séquence:** représentation temporelle des interactions entre objets pour une opération donnée
- ❖ **Diagramme de déploiement:** description de la répartition des composants (objets) sur la structure matérielle cible
- ❖ **Diagramme de composants:** architecture des composants physiques d'un application

Description statique

Les cas d'utilisation (use cases)



Analyse et représentation des besoins

✿ Analyse des besoins:

- ✦ Compréhension des besoins à couvrir par le logiciel
- ✦ Formalisation de ces besoins

✿ Représentation des besoins:

- ✦ Use cases + scénarii : utilisation du logiciel par les acteurs
- ✦ Diagrammes de séquences : analyse temporelle des scénarii
- ✦ Diagrammes de classe/objets : structure du système
- ✦ Diagramme de collaboration : interactions entre les éléments du système (acteurs et composants)

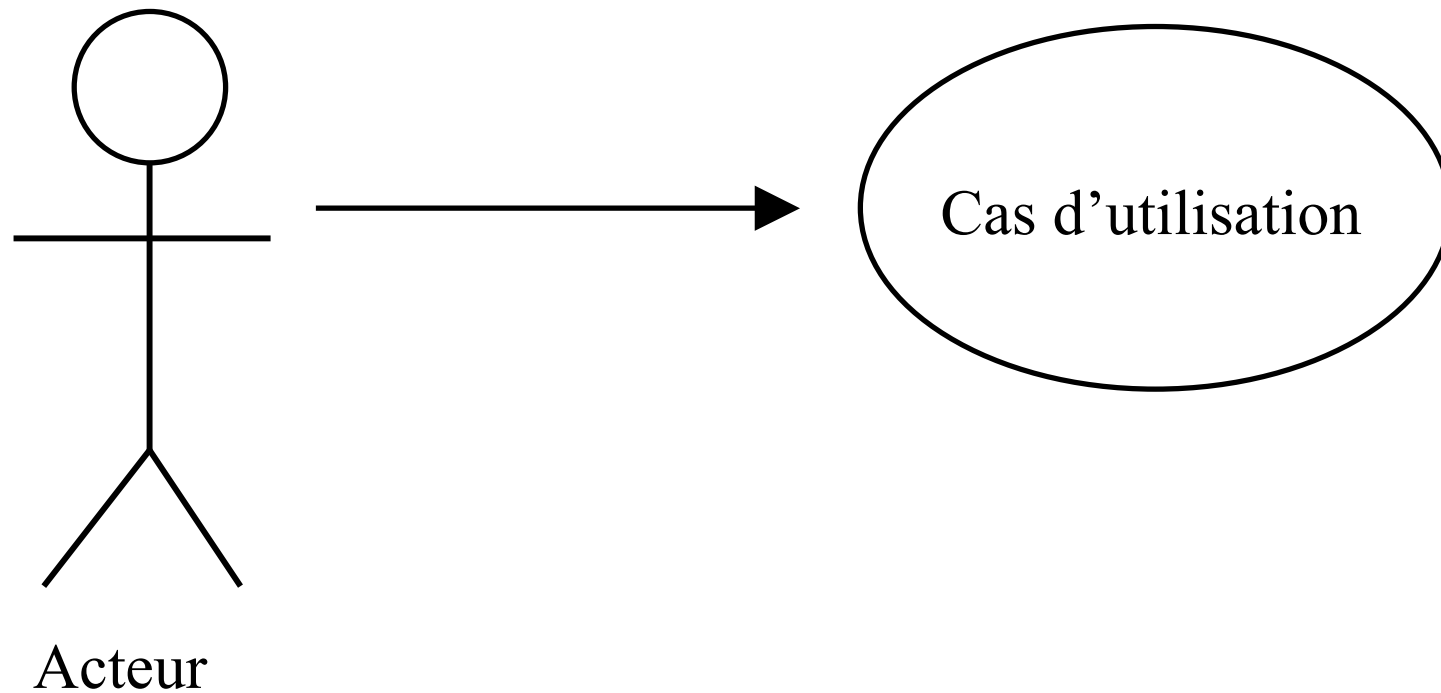


Cas d'utilisation

- ✿ Un système est conçu pour des utilisateurs
- ✿ Use cases
 - Description des comportements du système vis-à-vis de l'utilisateur
 - Facilitent structuration des besoins
 - Représentation simple et expressive
 - Expriment les limites et les objectifs du système
 - Simplifient les échanges avec le rédacteur du cahier des charges
 - C'est la représentation d'une fonctionnalité, réponse d'une stimulation de l'utilisateur



Cas d'utilisation





Cas d'utilisation : Définitions

● Acteur:

- Entité externe qui interagit avec le système
- Prend des décisions
- Possède un rôle vis-à-vis du système
- Il peut être un utilisateur et/ou un autre système

● Acteur $\neq$ Utilisateur

- Un utilisateur peut jouer plusieurs rôles
- Plusieurs utilisateurs peuvent jouer un même rôle
- Un acteur peut être un autre système



Cas d'utilisation : Définitions

✚ Cas d'utilisation:

- ✚ Ensemble des actions réalisés par le système en réponse à une action donnée
- ✚ Suite d'interaction entre acteurs et système
- ✚ Correspond à une fonction visible à l'utilisateur
- ✚ Formalisation l'obtention d'un objectif
- ✚ Les cas d'utilisation doivent être mutuellement exclusifs



Les cas d'utilisation: exemple

- Donner les différents cas d'utilisation des différents acteurs d'une banque



Relation entre les cas d'utilisation

- ❊ Ne correspondent pas à des communications entre les cas.
- ❊ Permettent une structuration des cas d'utilisation
 - ❑ Utilisation d'un autre cas (*uses* ou *include*): permet l'extraction d'un comportement commun à plusieurs cas d'utilisations (exemple: donner son code de carte bleu à un GAB)
 - ❑ L'extension (*extends*) : extraction de scénarii communs ou optionnels (exemple: retirer de l'argent avec débit différé *est une extension de* retirer de l'argent)

Les scénarii



Les scénarii : liens avec les use cases

- Le système correspond à l'ensemble des use cases sans les acteurs
- Un use case = un chemin d'exécution possible
- Un scénario =
 - un chemin particulier d'exécution
 - une séquence d'événements / instructions.
 - instance des cas d'utilisation



Les scénarii

- ❁ Il s'agit en général d'un descriptif sous forme de texte des actions à effectuer pour un cas d'utilisation donné
- ❁ Il peut être décomposé en plusieurs sous scénarii:
 - ❑ L'un correspondant au scénario optimal
 - ❑ Les autres correspondants aux alternatifs (tests d'erreurs, mauvais renseignements, ...)



Les scénarii: exemple

- ✚ Détaillez le scénario correspondant à un retrait d'argent dans un GAB

Diagramme de classes

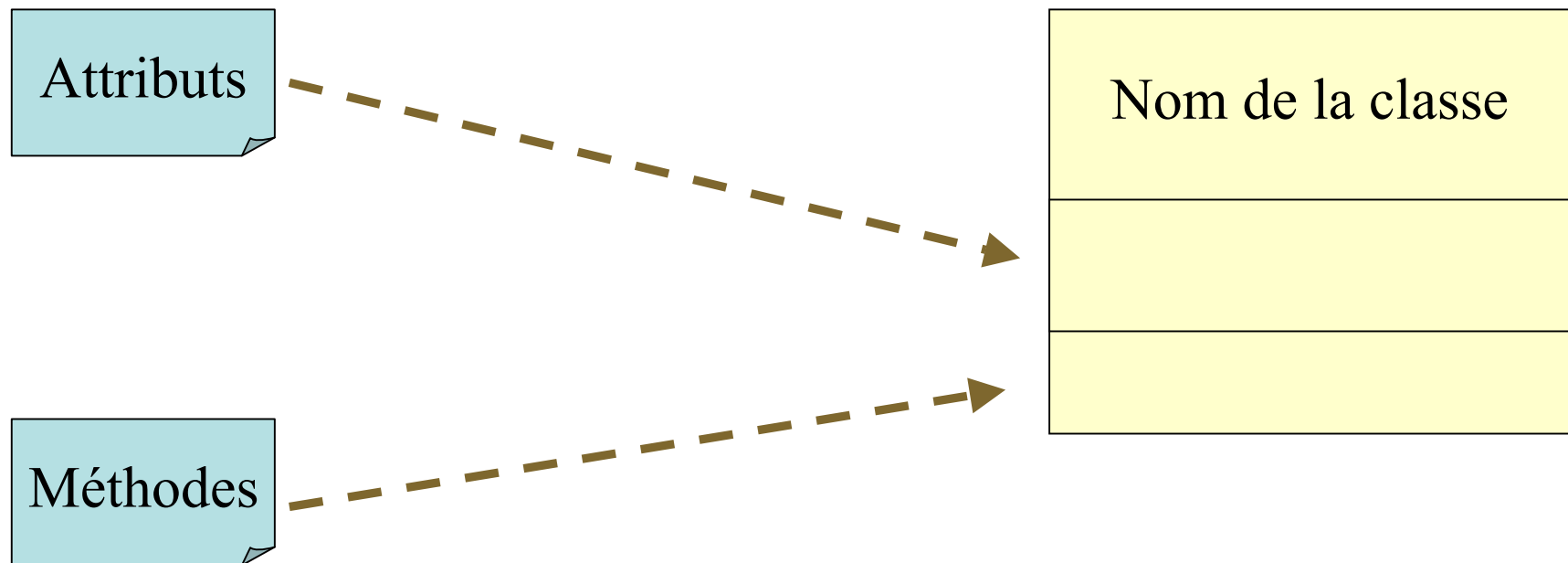


Diagramme de classes

- Une classe est une description abstraite permettant de définir des objets ayant:
 - Des propriétés,
 - Des comportements,
 - Des relations,
 - Des sémantiques...
- ...Communes



Représentation d'une classe UML



Les différents champs, hormis le nom, sont optionnels



Les attributs

Nom de la classe
Nom attribut : type = <i>valeur initiale</i>

- Les attributs définissent une propriété commune
- Chaque nom est unique
- La valeur de l'attribut dépend de chaque objet

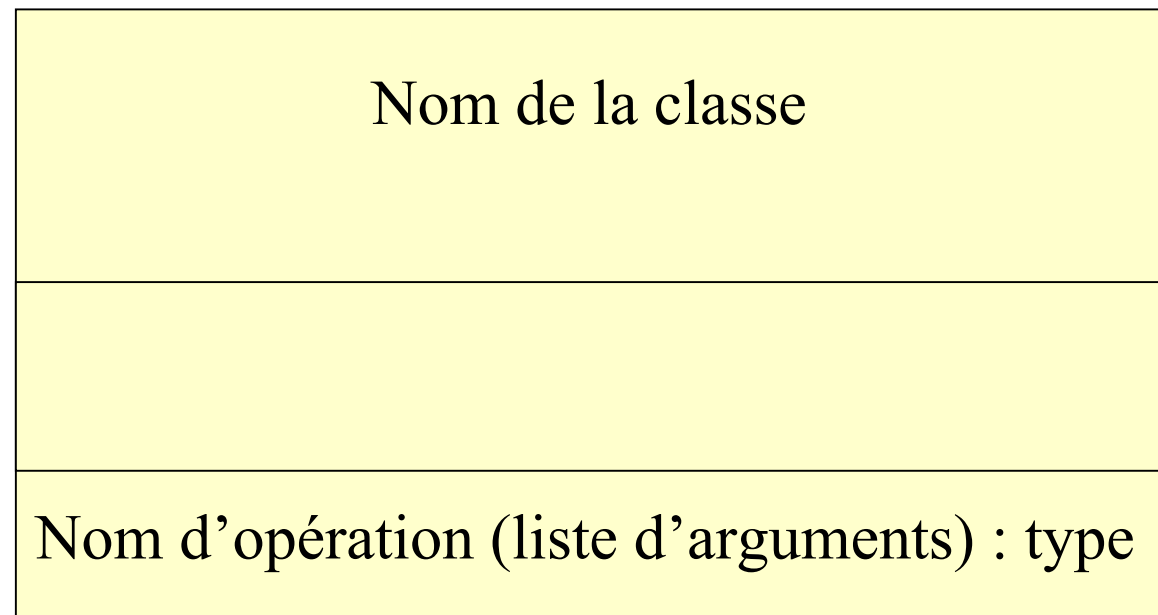


Exemple

CD
Artiste : chaîne Titre : chaîne Numero_ISBN : entier Stock : entier PrixU_fournisseur : réel PrixU_vendu : réel



Les méthodes





Propriétés d'accessibilité des attributs et des méthodes

● On retrouve les trois niveaux de protections classiques:

■ Privé (-)

■ Protégé (#)

■ Publique (+)

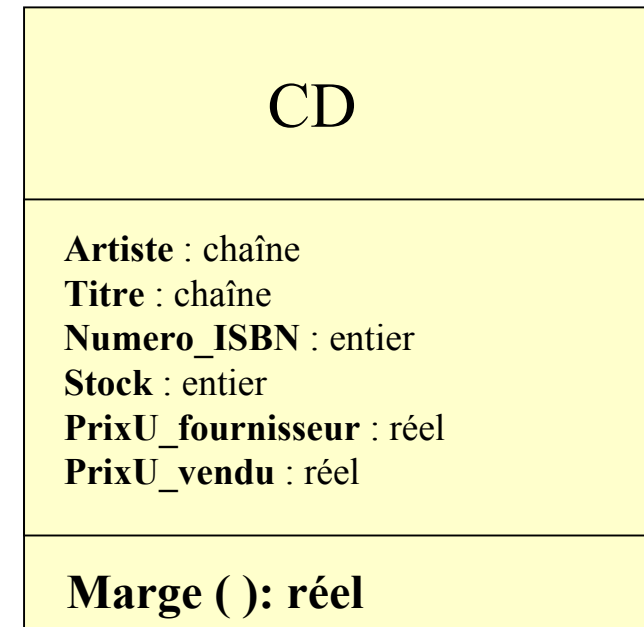
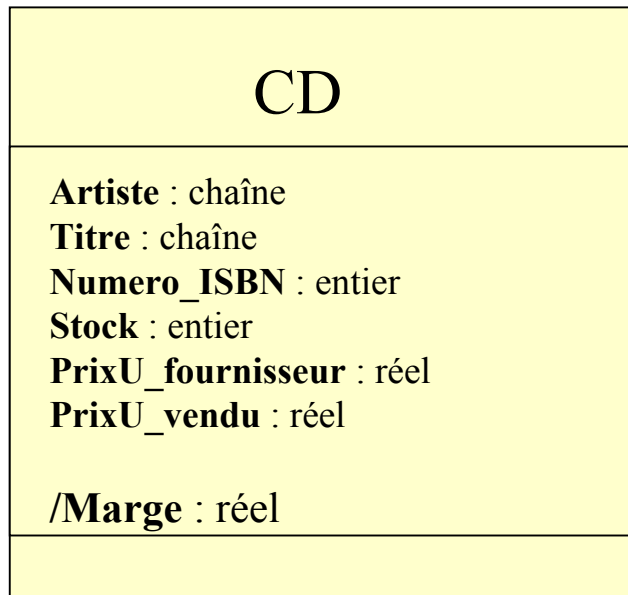


Les attributs dérivés

- ✿ Un attribut dérivé permet d'indiquer clairement qu'un attribut découle d'autres propriétés (*exemple: la surface d'un rectangle dépend de sa largeur et de sa longueur*)
- ✿ Il sont noté **/nom attribut** et ont des valeurs calculées à partir des autres attributs



Exemple



Analyse

Conception



Relation entre les classes

✚ Association

✚ Agrégation

✚ Composition

✚ Spécialisation

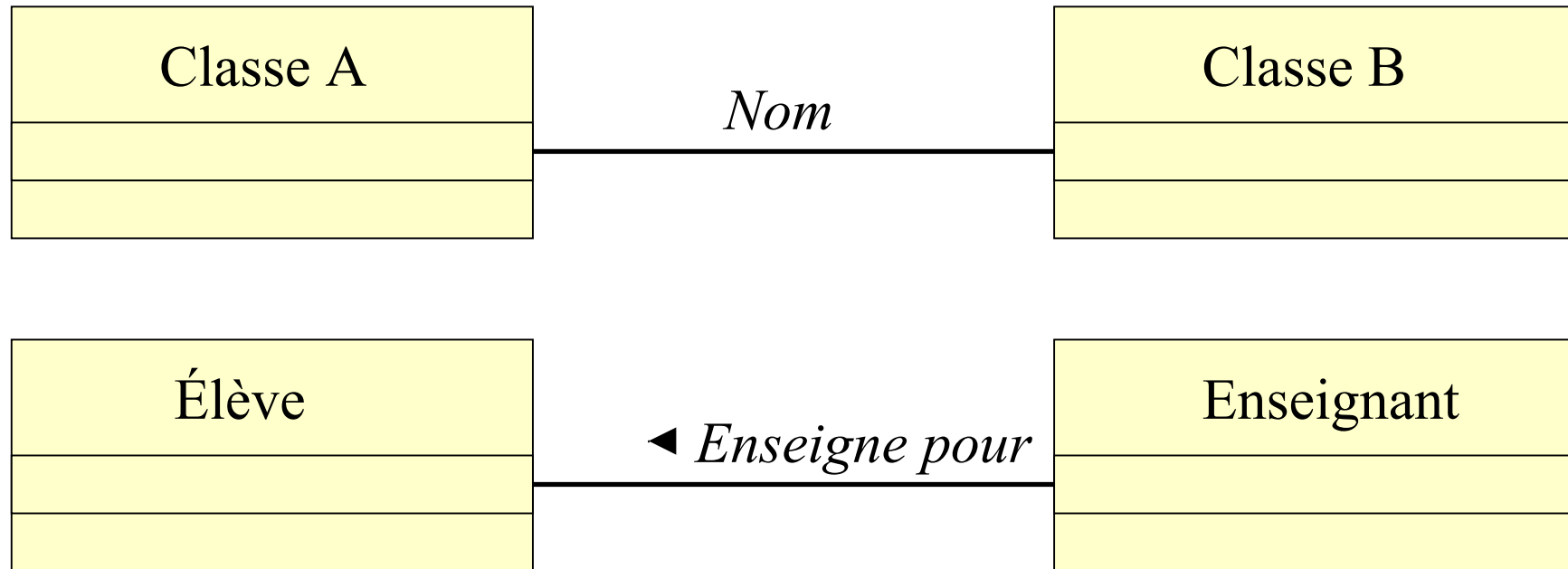


Les associations

- ❖ Représente une association structurelle entre 2 classes.
- ❖ Ces associations peuvent être nommées.
- ❖ Elles peuvent également posséder des cardinalités



Les associations





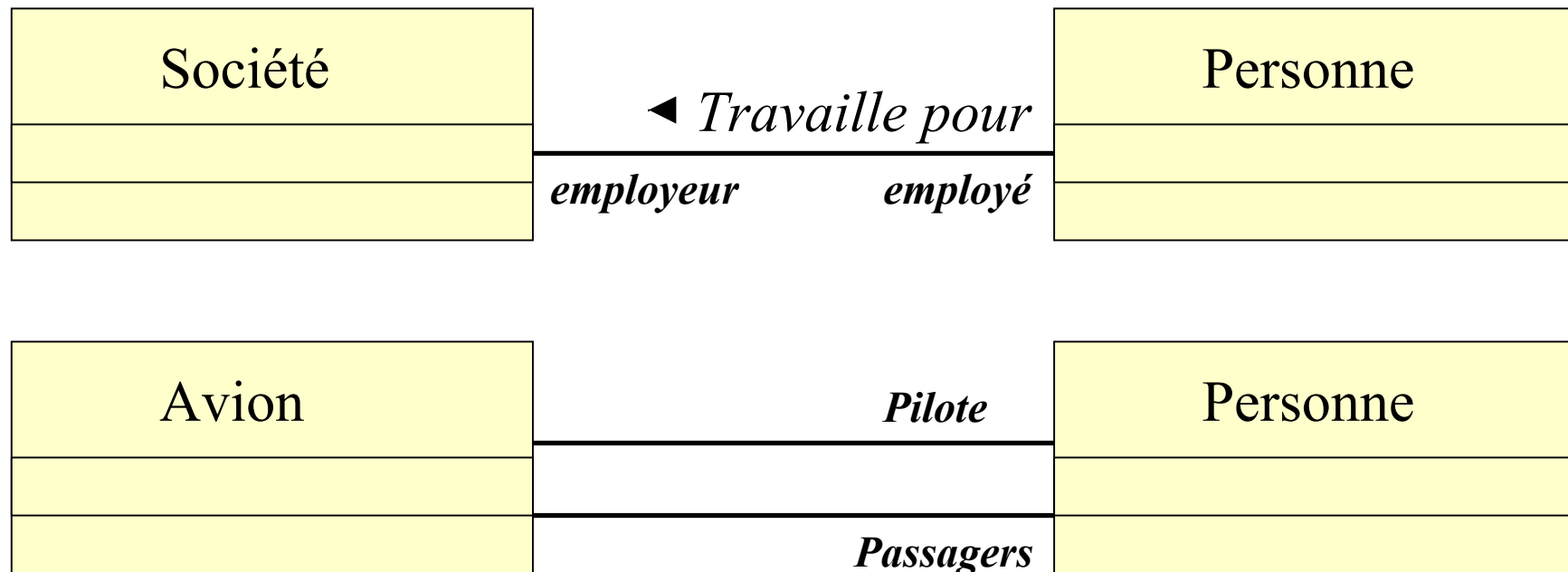
Les associations

- ❖ Toute association binaire possède 2 rôles.
- ❖ Un rôle définit la manière dont une classe intervient dans une relation.
- ❖ Le nommage des associations et celui des rôles ne sont pas mutuellement exclusifs
- ❖ L'intérêt des rôles réside dans le cas où plusieurs associations lient deux classes.
- ❖ **Attention:** Lorsqu'il y a trop d'associations entre deux classes c'est suspect...



Les associations

Exemple:





Cardinalité et multiplicité des associations

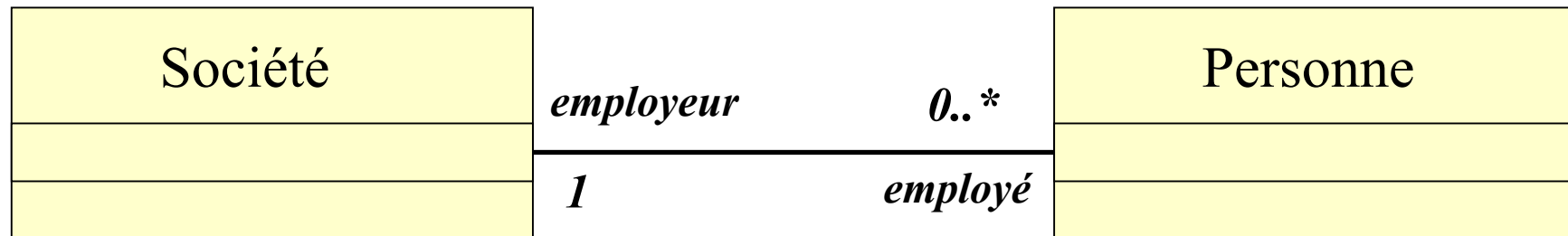
- La cardinalité est une information qui quantifie le nombre nécessaire de fois pour la participation d'un objet à une instance.

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



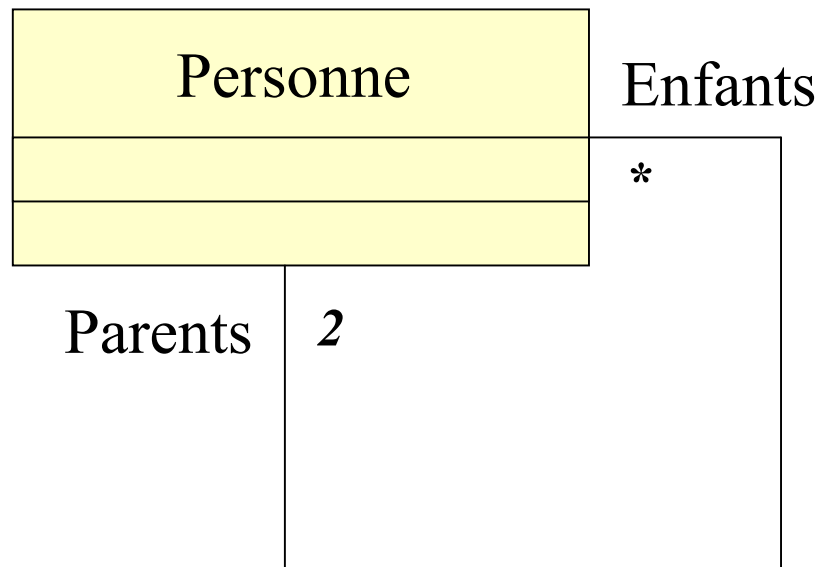
Cardinalité et multiplicité des associations

Exemple:





Les associations réflexives



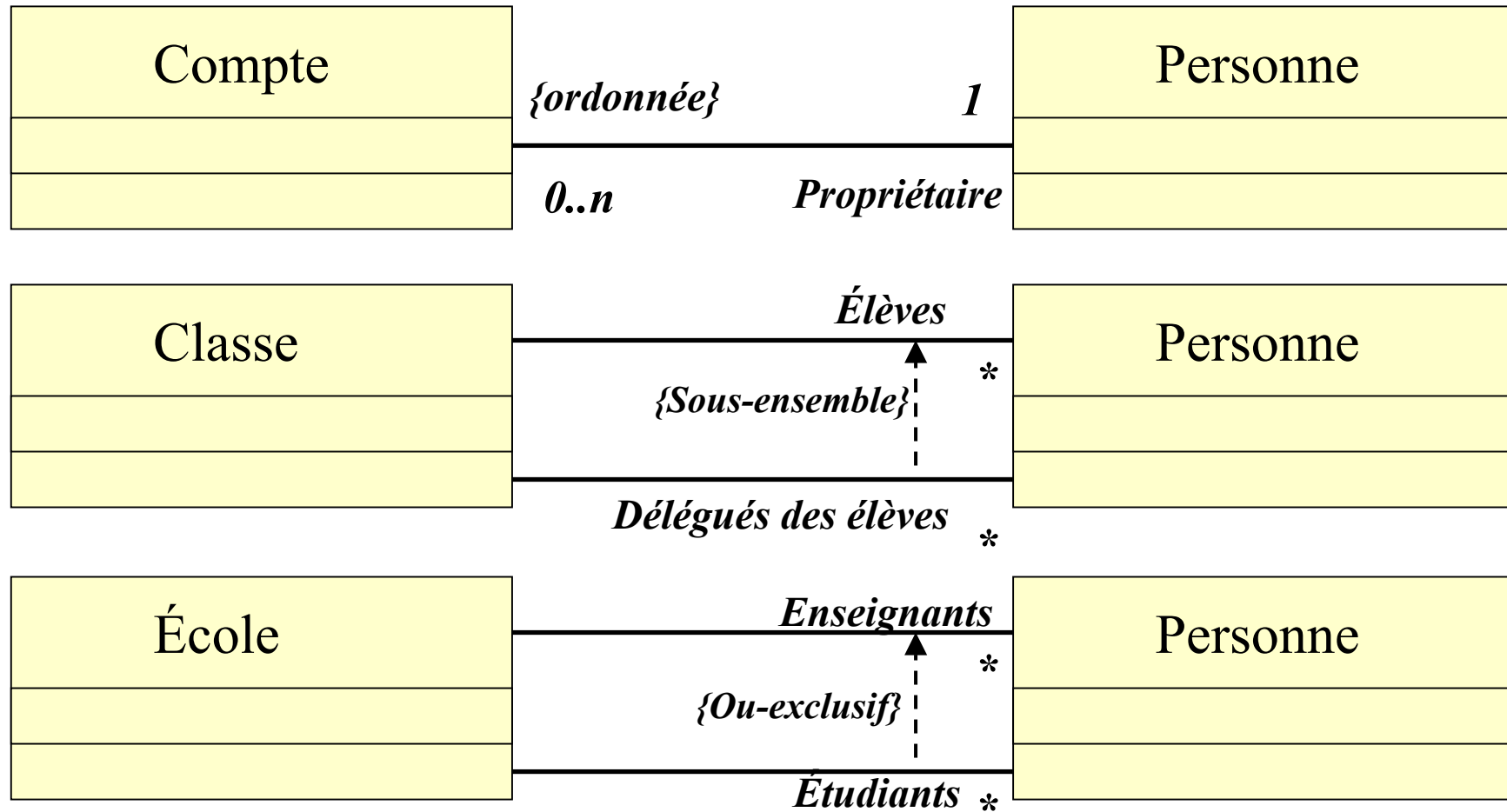


Les contraintes sur les associations

- ⊗ Contraintes qui portent sur une relation ou un groupe de relation (en général mise entre accolades {})
- ⊗ Contraintes de sous ensemble:
 - ⊠ Collection incluse dans une autre collection
 - ⊠ Une association parmi un groupe d'associations est possible



Les contraintes sur les associations



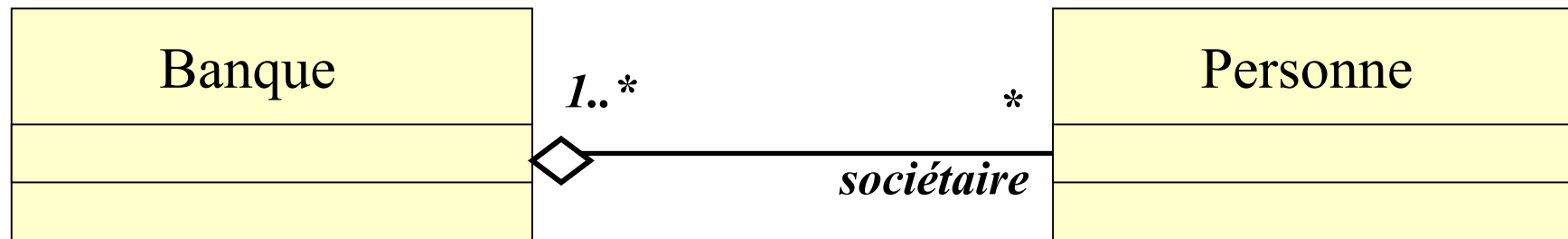


Agrégation

- ⊗ Une agrégation est une relation non symétrique.
- ⊗ Cette relation est à mettre en rapport avec la notion d'agrégation que nous avons vu en C++
- ⊗ Cas d'utilisation :
 - Une classe B « fait partie » d'une classe A
 - Les valeurs d'attributs de la classe B se propagent dans A
 - Une action sur A implique une action sur B
 - Les objets de B sont subordonnés aux objets de A
- ⊗ Il suffit que l'un de ces critères soit vérifié



Agrégation exemple



- Une agrégation peut avoir une cardinalité
- Une agrégation peut également avoir un rôle

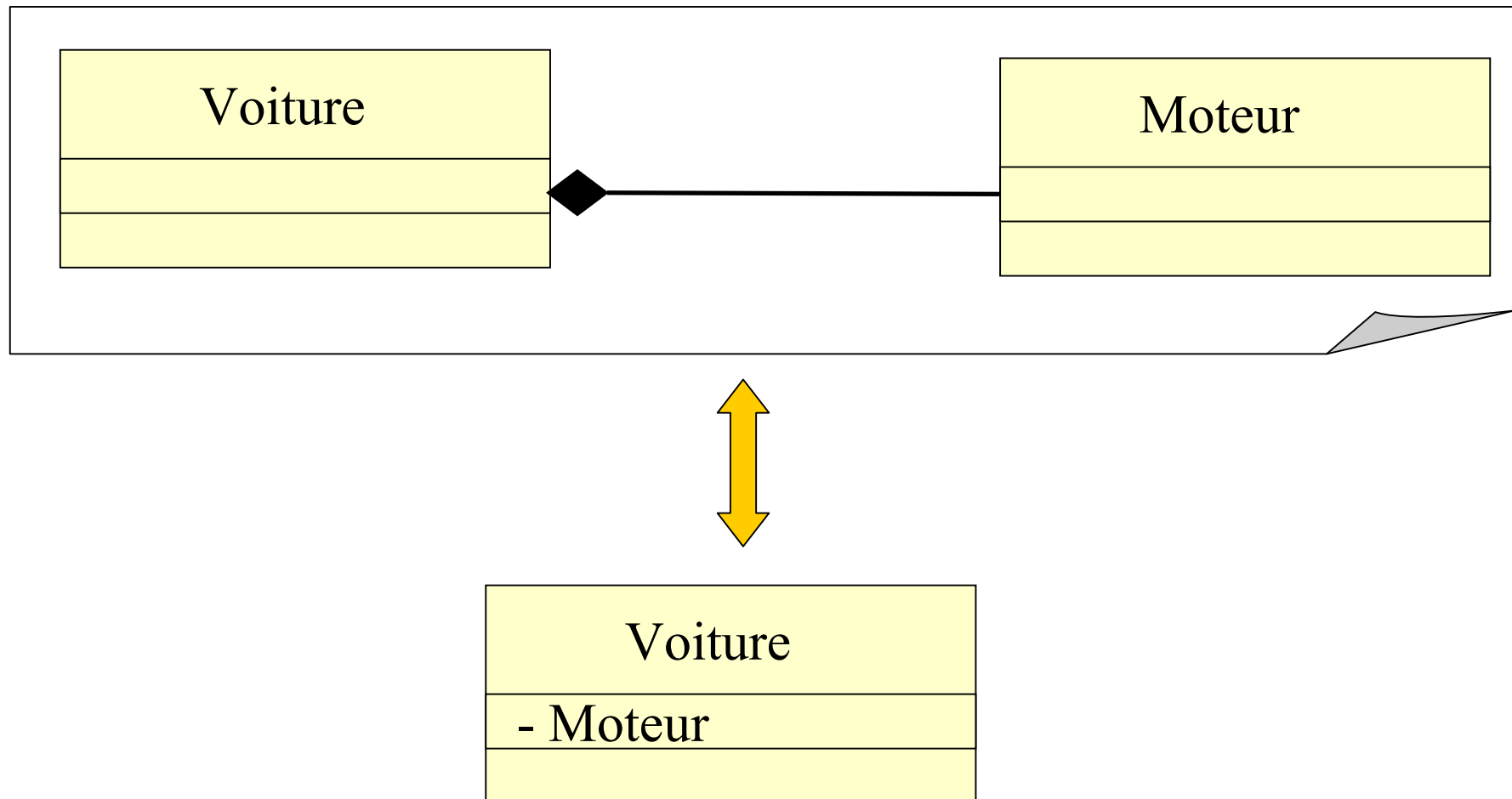


Composition

- ✿ C'est un cas particulier d'agrégation
- ✿ Peut être modélisée au moyen d'attributs.
- ✿ Le composant est physiquement présent dans l'agrégat
- ✿ Implique une contrainte sur la valeur de cardinalité coté agrégat (0 ou 1)
- ✿ 0 implique un attribut non renseigné
- ✿ La composition doit être retenue lorsqu'un attribut participe à la relation



Composition: exemple



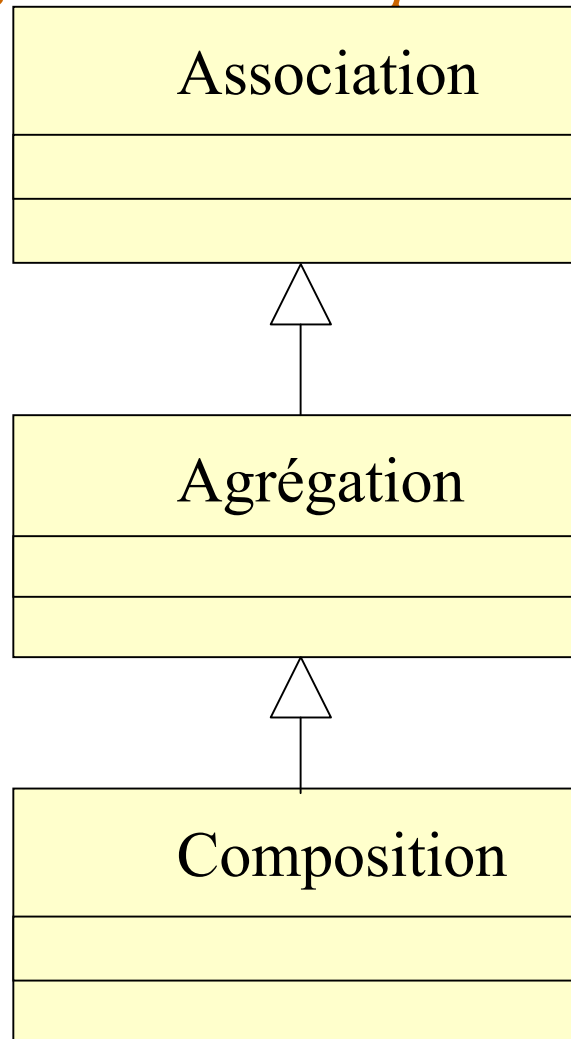


Généralisation-spécialisation-héritage

- La relation de généralisation – spécialisation est une sorte d'héritage public comme on l'a vu en C++.
- Cette relation peut contenir des contraintes:
 - Selon plusieurs critères (véhicule → motorisation ou véhicule → milieu)
 - Inclusif ou exclusif



Exemple de spécialisation : relation entre Association, Agrégation et Composition



Les diagrammes d'objets



Les diagrammes d'objets

- ✿ Liens structurels entre les instances des classes des projets.
- ✿ Facilite la compréhension de structures complexes
- ✿ On peut représenter les instances de trois façons:

Nom de l'objet

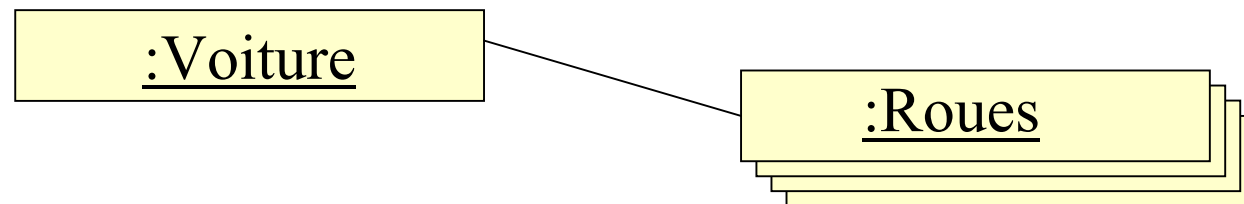
Nom de l'objet:Nom de Classe

:Nom de Classe



Les diagrammes d'objets

- Les objets **composés** d'autre objets peuvent être visualisés
- Les valeurs des attributs sont optionnels
- Les valeurs des liens sont optionnels
- Les liens réflexifs des diagrammes de classes peuvent être explicités en terme d'instances
- Les liens de cardinalité supérieur à deux peuvent être représentés de la façon suivantes



Description dynamique

Diagramme de séquence



Diagramme de séquence et scénario

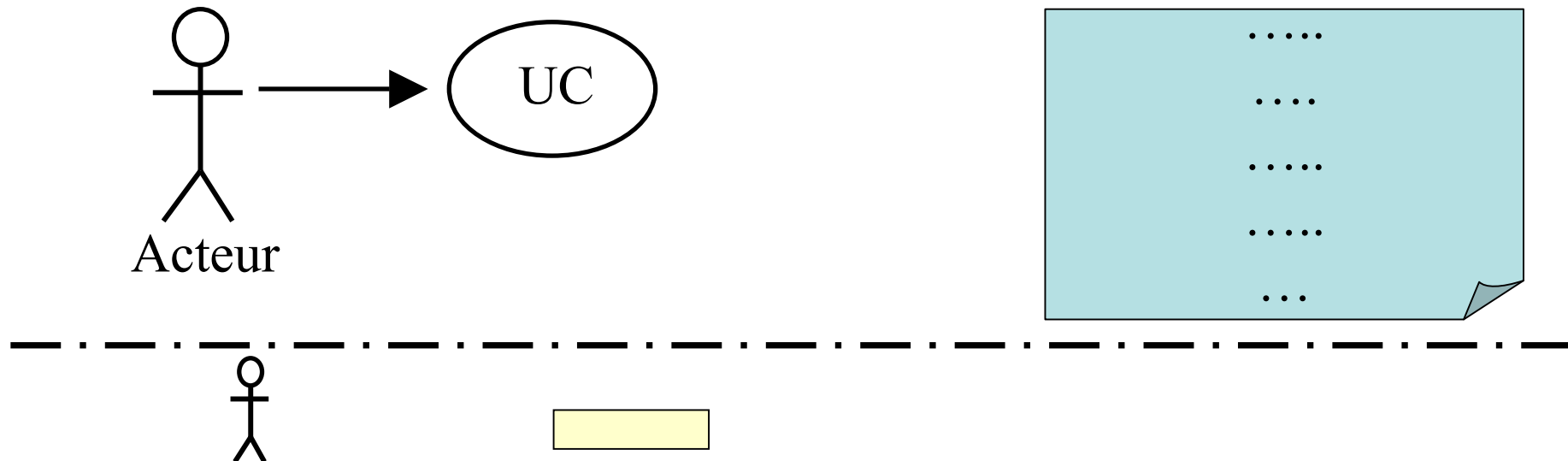


Diagramme de séquence



Diagramme de séquence

- ✿ Modélisation des interactions entre les différents objets suite à un événement externe au système.
- ✿ Mise en place d'un aspect temporel
 - **Synchrone**: l'émetteur de la requête est bloqué jusqu'à la fin du traitement
 - **Asynchrone**: l'émetteur peut poursuivre son exécution.
- ✿ Il est possible d'effectuer des échanges conditionnels
- ✿ On peut matérialiser la création et la destruction d'objets

Diagramme de séquence: exemples

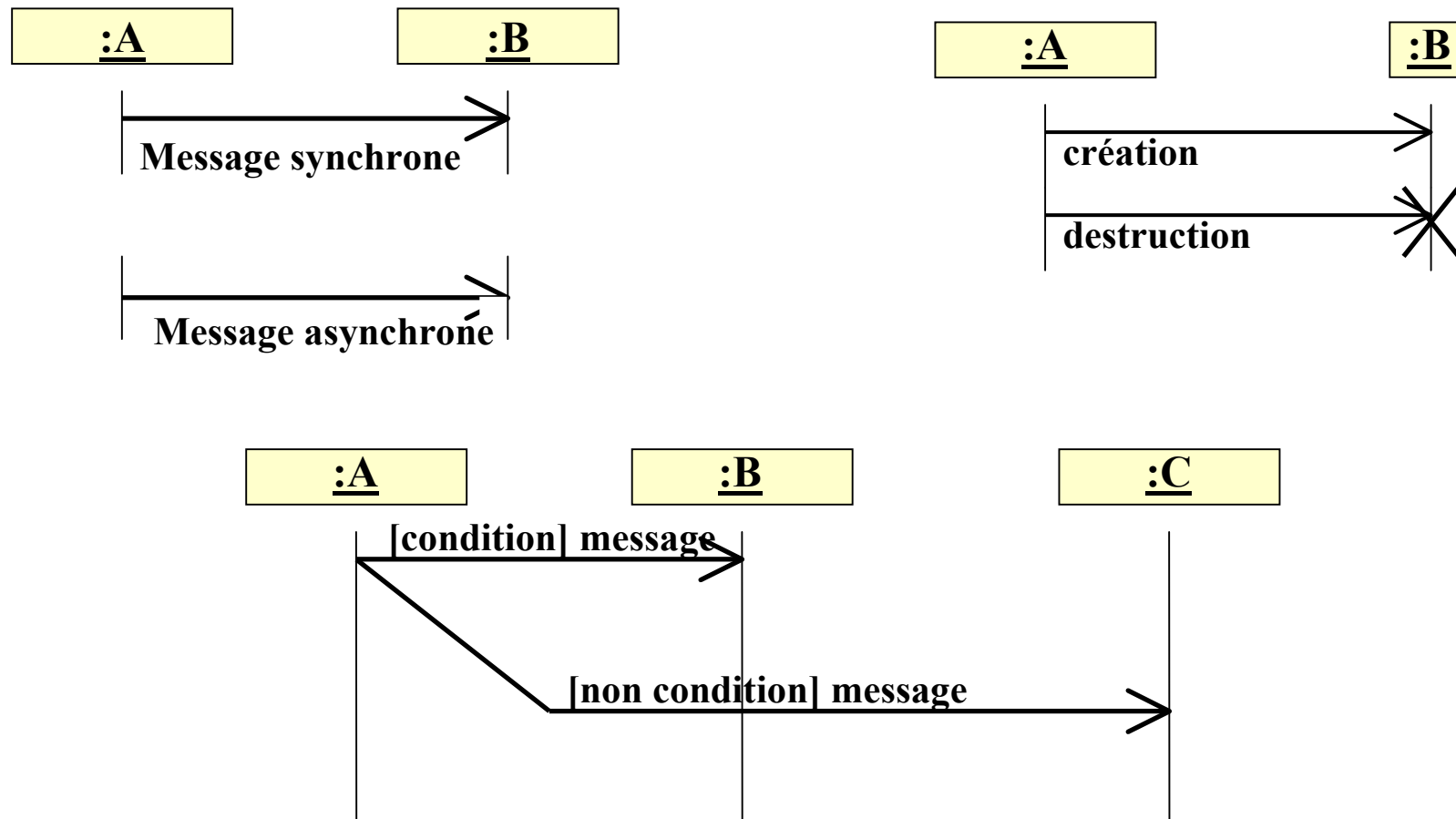




Diagramme de séquence: exemple

- Utilisation d'un téléphone pour la communication entre deux personnes

Diagramme d'états transitions



Diagramme d'états transitions

- ❁ Décrit le comportement des objets d'une classe au moyen d'un automate à états finis.
- ❁ Comportement d'un objet = graphe
 - ▣ Nœuds → états de l'objet: étape dans le cycle de vie d'un objet.
 - Satisfait certaines conditions
 - Réalisation potentiel de certaines actions
 - Attente d'événements.
 - ▣ Arc → Transitions entre états: exécution d'une action ou réaction de l'objet



Diagramme d'états transitions

- ⊗ Chaque objet est dans un état particulier à un instant donné.
- ⊗ Chaque état est identifié par un nom
- ⊗ Un état est stable et durable (attente d'événements pour changer d'état)
- ⊗ Chaque diagramme d'état transition comprend:
 - Un état initial unique pour un niveau de hiérarchie donné.
 - Des états intermédiaires
 - Éventuellement un (ou plusieurs) état final (finaux) (un système ne s'arrêtant pas, n'a pas d'état final)



Diagramme d'états transition

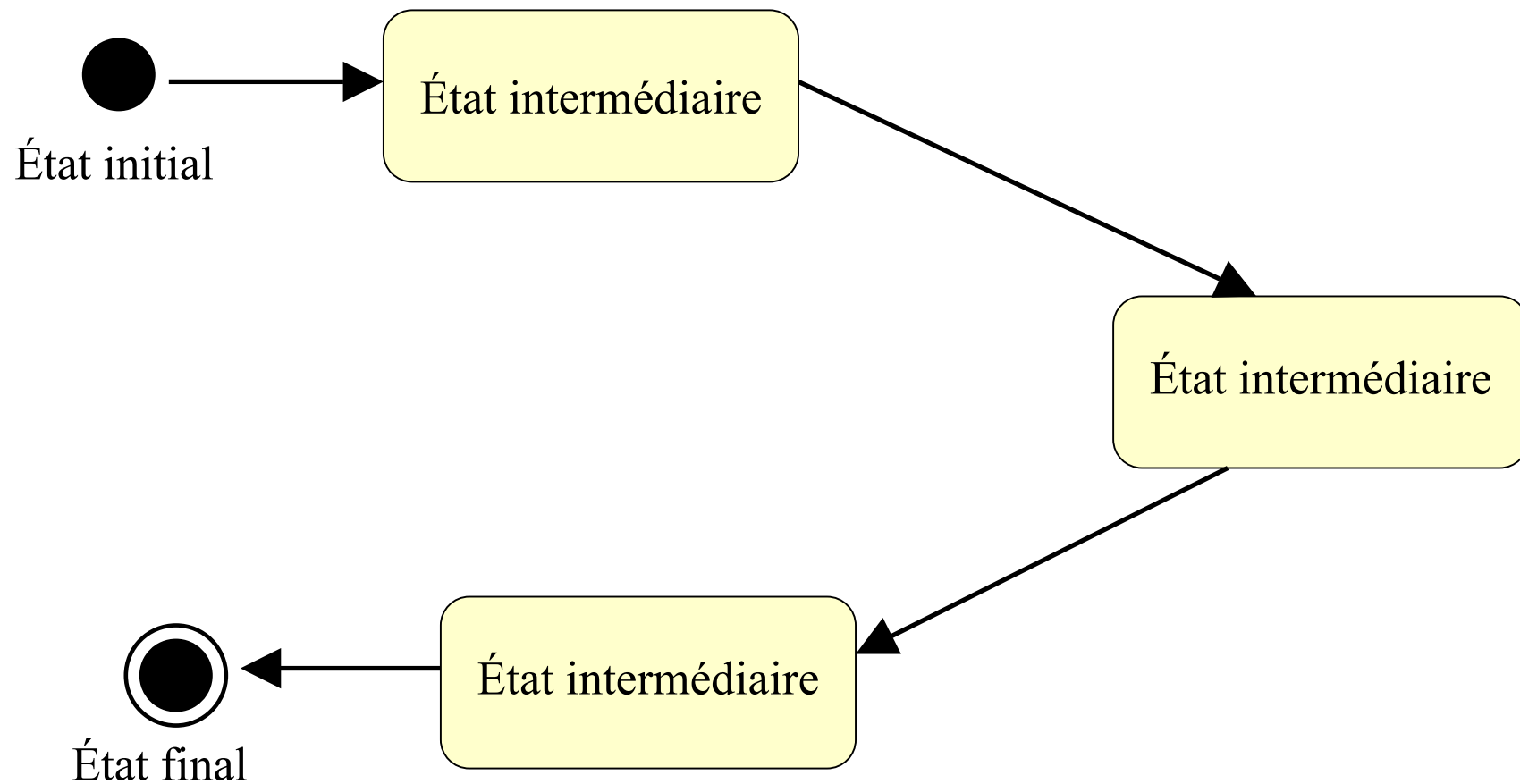




Diagramme d'états transition

⊕ Transitions

- ▣ Les transitions sont unidirectionnelles
- ▣ Le diagramme d'état transition est un graphe orienté

⊕ Événements

- ▣ Un événement correspond à l'occurrence d'une situation donnée
- ▣ L'événement est déclencheur de la transition inter-état
- ▣ Les événements peuvent être conditionnels



Diagramme états-transition: exemple

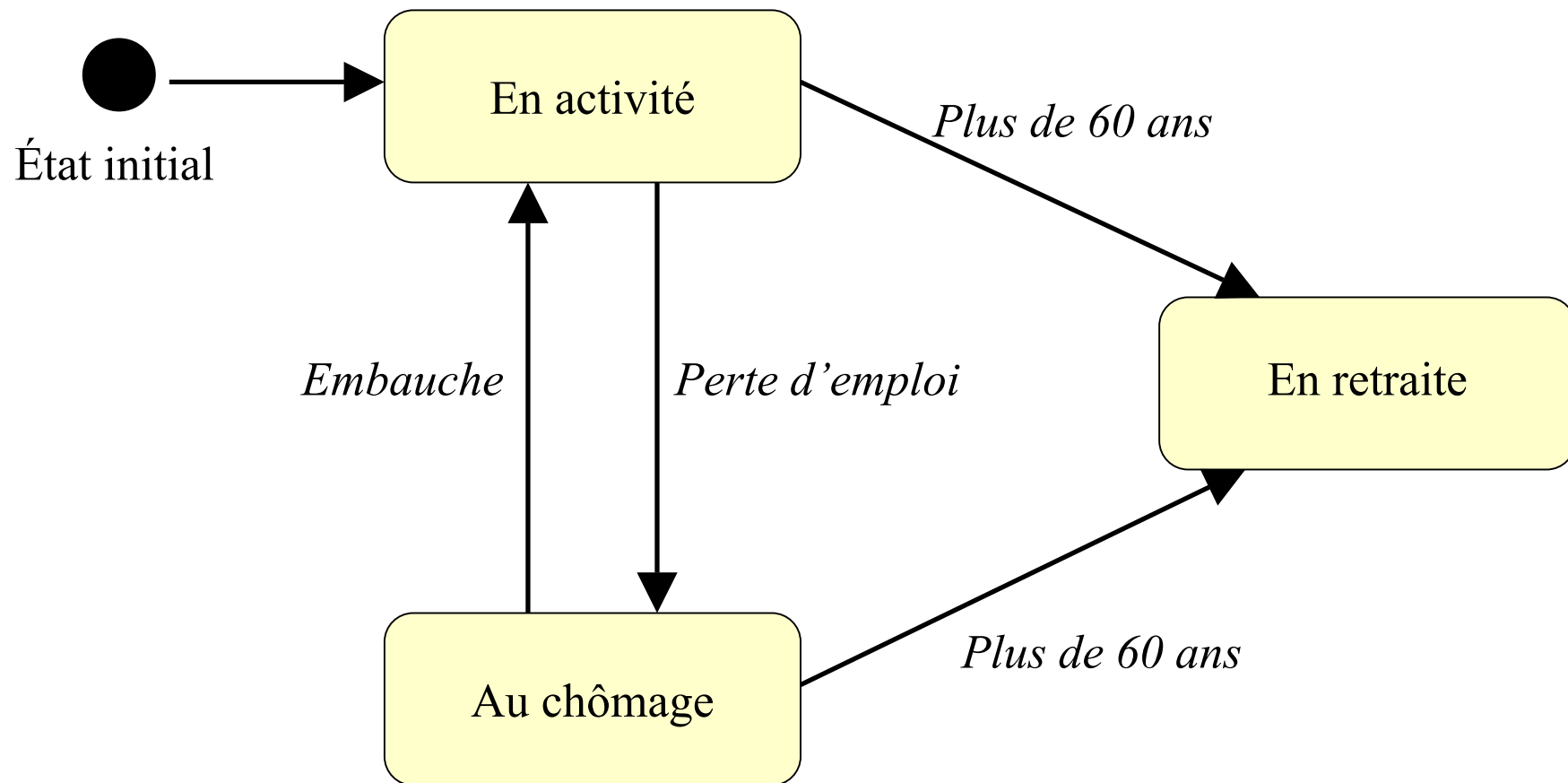


Diagramme d'activités (états-actions)



Diagramme d'activités

- ✿ C'est une variante des diagrammes états-transition.
- ✿ Mets l'accent sur les activités, leurs relations et leurs impacts sur les objets.
- ✿ Une activité mets en œuvre un ou plusieurs objets.
- ✿ Les transitions entre activités peuvent être conditionnées par des expressions booléennes.



Diagramme d'activités : un premier exemple

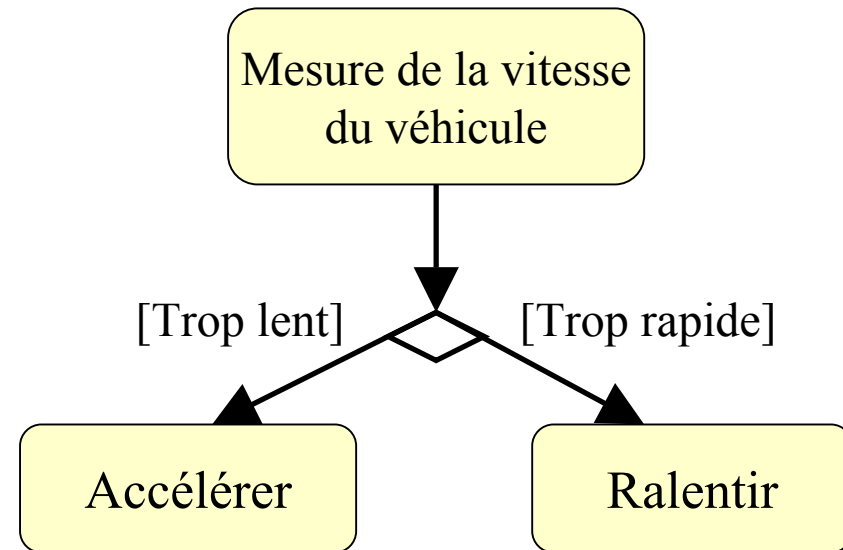
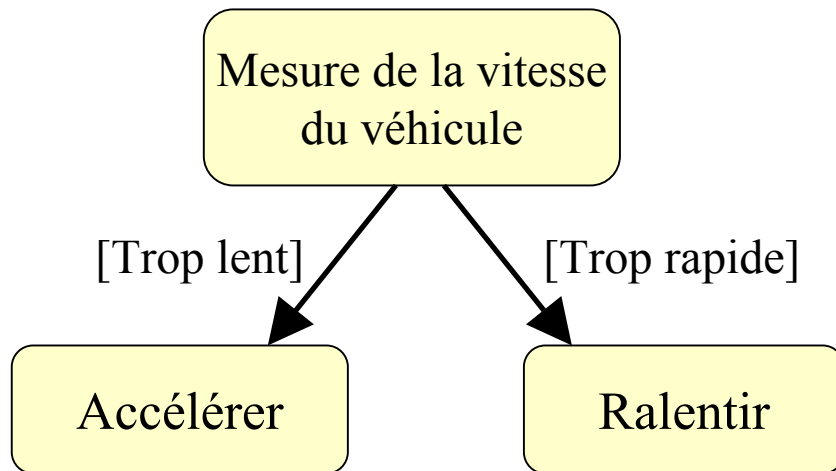


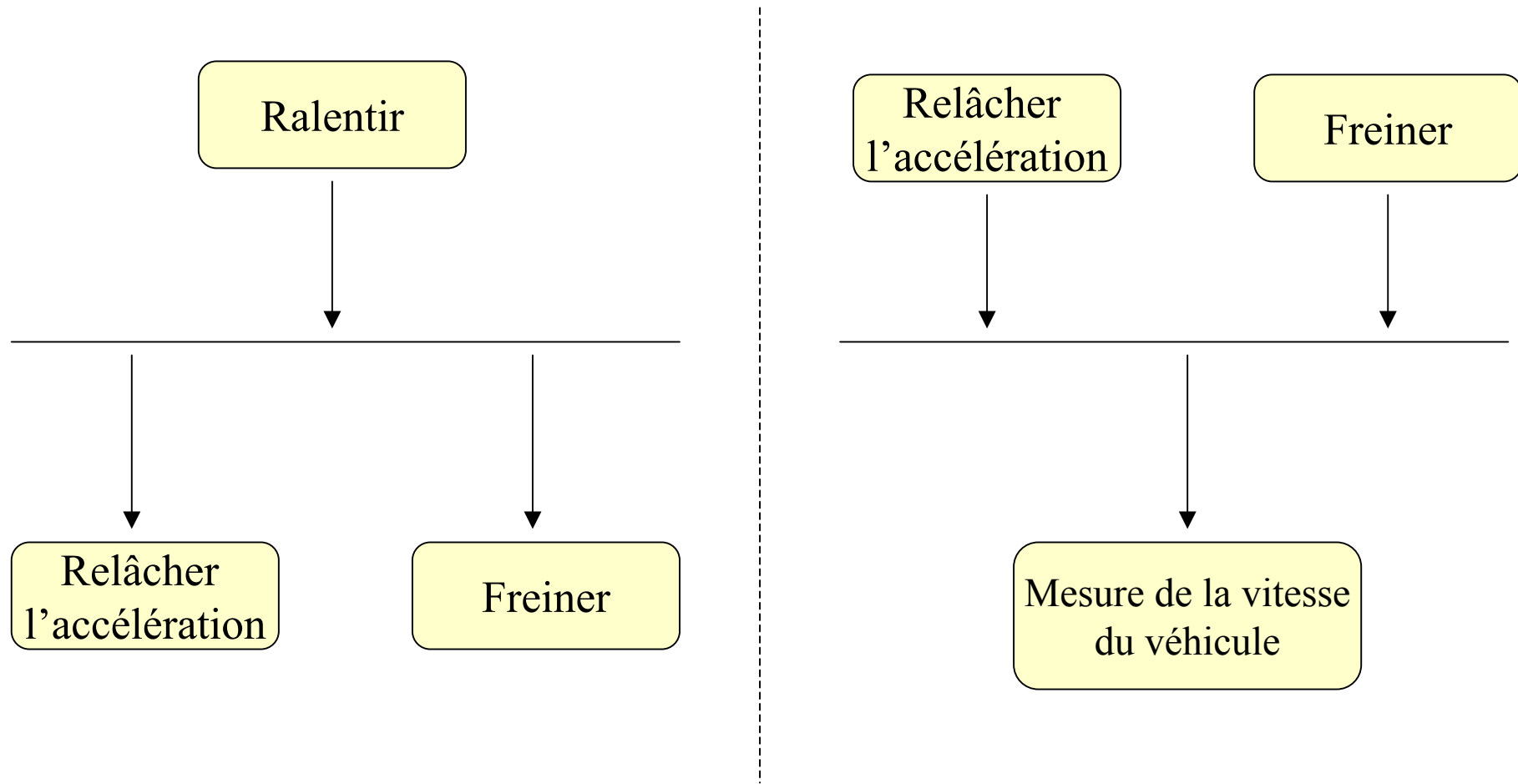


Diagramme d'activités et synchronisation

- ✚ On peut éventuellement synchroniser deux activités déclenchés par la même transition
- ✚ On peut également démarrer une activité en synchronisant l'achèvement de deux autres.



Diagramme d'activités et synchronisation





Conclusion

- ❖ Tous les diagrammes ne sont pas nécessaire pour chaque projet.
- ❖ Ils servent à illustrer et détailler des éléments du projet.
- ❖ L'étape d'analyse et de conception se fait **toujours** avant l'implémentation.
- ❖ Cependant, il peut y avoir des retours sur la conception après l'implémentation (processus récursif)



Bibliographie

- ✿ Cours de Frédéric Julliard - Université de Bretagne Sud
- ✿ UML Martin Fowler – CampusPress 2002
- ✿ Cours de Jacques Lonchamp(rubrique enseignement): <http://www.loria.fr/~jloncham>
- ✿ <http://uml.free.fr/>