

Ingénierie des Systèmes d'Information - UML

Elsa NEGRE
Université Paris-Dauphine
L3 APP - MIDO

Année 2024-2025

- **Objectif :**

- Introduire les **notions de base** de l'ingénierie des systèmes d'information et d'UML

- **Séances :**

10 séances dont 5 TDs + 1 examen

Plan du cours

- PARTIE I :

Présentation des systèmes d'information

- PARTIE II :

UML

Partie I

Présentation des systèmes d'information ?

Définition (1)

- Un **système d'information** (SI) est un ensemble organisé d'éléments qui permet de regrouper, de classifier, de traiter et de diffuser de l'information sur un phénomène donné.

Définition (2)

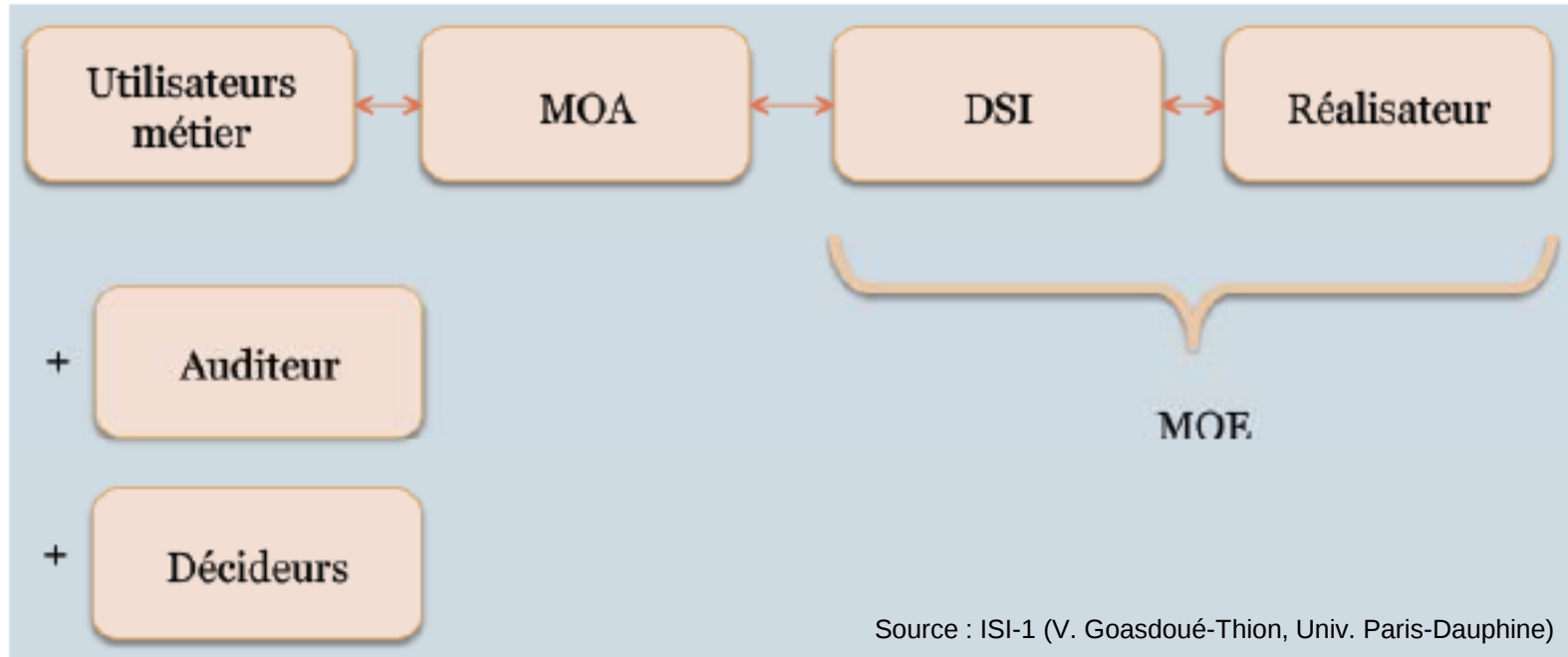
- Le système d'information est le véhicule de la communication dans l'organisation.

- Sa structure est constituée de l'ensemble des ressources (les hommes, le matériel, les logiciels) organisées pour :
 - collecter,
 - stocker,
 - traiter
 - et communiquer les informations.

Définition (3)

- Le système d'information coordonne ainsi grâce à l'information les activités de l'organisation et lui permet donc d'atteindre ses objectifs.

Qui intervient dans l'ingénierie des SI?



Les SI : MOA et MOE

MOA et MOE – ce sont des sigles couramment utilisés pour Maîtrise d'OuvrAge et Maîtrise d'Œuvre. Ce sont des entités organisationnelles.

Vocabulaire issu du BTP... MOA et MOE sont des termes empruntés au secteur des travaux publics.



MOA en construction - personne (morale) pour laquelle sont réalisés les travaux, entité porteuse des besoins.



MOE en construction - personne (morale) chargée par le maître d'ouvrage de concevoir le programme de restauration, de diriger l'exécution des marchés de travaux, et de proposer le règlement des travaux et leur réception.

Source : ISI-1 (V. Goasdoué-Thion, Univ. Paris-Dauphine)

La MOA

MOA (Maîtrise d'OuvrAge) – Entité responsable de l'organisation et des méthodes de travail autour du SI, responsable de la bonne compréhension entre les métiers et la DSI.

⇒ La MOA se place “entre les métiers et la MOE”.

La MOA a entre les mains la décision, le financement, la structuration du projet métier. Mais à ne pas confondre avec les utilisateurs. La MOA est donneur d'ordre de la DSI.

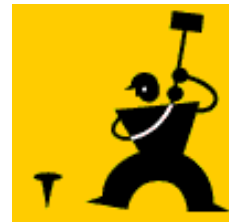
Rôles MOA : décrit les besoins, le cahier des charges, établit le financement et le planning général des projets, fournit au MOE les spécifications fonctionnelles générales (le « modèle métier ») et valide la recette fonctionnelle des produits, coordonne les instances projets entre les utilisateurs métiers et la MOE, assure la responsabilité de pilotage du projet dans ses grandes lignes, adapte le périmètre fonctionnel en cas de retard dans les travaux, pour respecter la date de la livraison finale.



Source : ISI-1 (V. Goasdoué-Thion, Univ. Paris-Dauphine)

La MOE

MOE (Maîtrise d'Œuvre) - Réalisateur technique du projet, elle en conçoit la solution informatique.. On peut voir la MOA comme son client. Elle est généralement composée de la DSI de l'entreprise et du réalisateur (des prestataires, sociétés de services, éditeurs et constructeurs).



Source : ISI-1 (V. Goasdoué-Thion, Univ. Paris-Dauphine)

SI : Pourquoi ?

- Pour mieux utiliser les données, les informations qui constituent la première richesse de toute organisation.

Les grandes familles de SI

- Les systèmes de conception : calcul numérique CAO/CFAO
- Les systèmes industriels ou embarqués
- Les S.I de gestion (majoritaires)
 - CAO : Conception Assistée par Ordinateur
 - CFAO : Conception et Fabrication Assistée par Ordinateur

Le SI en 2011

- Une révolution : le réseau Web
- Une nouveauté : ASP
 - **APPLICATION SERVICE PROVIDER**
(fournisseur de service d'application)
ou fournisseur d'applications hébergées

ASP (1)

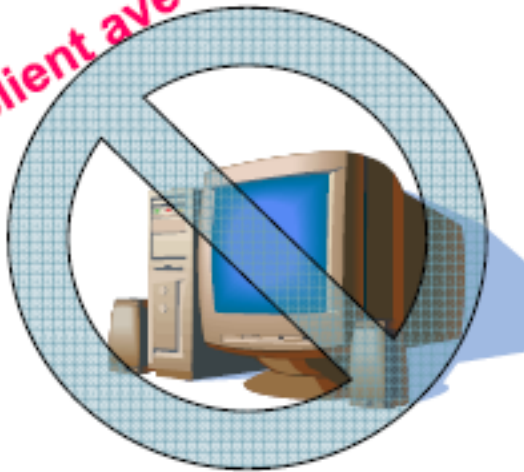
- Définition : C'est une organisation qui fournit des services informatiques à ses clients au travers d'un réseau (Internet)
- Avantage 1 : Simplicité
 - fournir un accès à des applications particulières en utilisant le protocole standard http.
 - Il faut disposer simplement d'un navigateur Web

ASP (2)

- **Avantage 2 : Réduction des coûts**
 - Pas de mise en place d'une infrastructure technique (serveurs réseau, logiciels d'accès,...)
 - Coûts partagés et ventilés sur une communauté de clients
 - Tous les clients profitent des mises à jours (upgrades)

ASP (3)

PC client avec application



Aller partout, ... de partout

ASP (4)

Malheureusement...

- Inconvénient 1 : La confidentialité des informations
 - Le client confie ses données à son ASP : confidentialité, pérennité du fournisseur, à prendre en compte dans une démarche de maîtrise des risques

ASP (5)

- Inconvénient 2 : L'intégration des données
 - Les flux d'entrée/sortie et les interfaces avec les autres S.I (car un S.I est rarement unique) sont plus délicats...

- Inconvénient 3 : La disponibilité
 - Qui dit réseau, dit « disponibilité et performance du réseau ; temps d'accès aux informations » : à prendre en compte pour l'évaluation de la robustesse des applications du S.I.

Une véritable révolution : le CLOUD computing

- Définition : déportation sur des serveurs distants des traitements informatiques traditionnels localisés sur un poste utilisateur.

ASP, CLOUD,...nomadisme

- Il faut donc penser le S.I différemment en intégrant :
 - L'externalisation (ses avantages et ses risques)
 - Aller partout,...de partout
 - Fin du poste de travail dédié

Composition du S.I (gestion)

- Un passage souvent « obligé » :
 - l'ERP (Entreprise Ressource Planning)
 - ou P.G.I (Progiciel de Gestion Intégré)
- L'ERP regroupe tous les systèmes informatisés permettant de soutenir le fonctionnement de l'entreprise.

E.R.P (1)

- Un E.R.P comprend généralement plusieurs modules :
 - Ressources humaines
 - Finances
 - Compta client
 - Compta fournisseurs
 - ...

E.R.P (2)

■ Pourquoi ?

- Aucune organisation ne part de zéro....
- Aucune organisation n'a les moyens de tout remettre à plat.
- Il en résulte alors des systèmes et des bases de données qui sont conservés, d'autres qui sont ajoutés, d'autres qui sont améliorés ou complétés...
- Et il faut faire avec

E.R.P (3)

■ Quelle finalité ?

- Avoir une base de données commune...
 - Cela est très difficile.
- De même, un seul ERP dans une organisation est souvent un objectif irréalisable. Dans la réalité, il existe souvent différents systèmes qui doivent cohabiter et s'interfacier entre eux.

E.R.P (4)

Donc :

- La problématique à prendre en compte dès la mise en place d'un ERP c'est :
 1. la cohabitation entre les systèmes
 2. la prolifération des bases de données ressources
 3. les interfaces avec les autres systèmes de l'organisation (qui peuvent aussi être externalisées...)
 4. ne pas oublier la production d'éléments décisionnels

N'oublions pas la donnée...

- La donnée (information) est la nouvelle richesse de l'entreprise

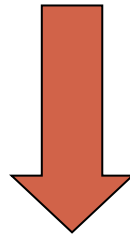
OUI.....Mais ...il faut savoir :

A condition de ... (1)

- Gérer les contenus (content Management)
 - Définition : gérer les informations brutes et savoir les transformer en données structurées
- Gérer les accès
 - Prendre en compte la gestion des flux et des protocoles d'échanges

A condition de ... (2)

- Ne pas oublier que le SI est souvent géré et pris en charge par des partenaires prestataires externes à l'organisation



EXTERNALISATION

Le Workflow :

le partage « nouvelle génération »

- Définition : partage des informations par tous les acteurs
- Remplacer le partage du papier (copies, stockage, envois,...)

Le Workflow : à condition de ...

- Développer les réseaux, les accès...
- Mais aussi :
 - de réussir la conduite du changement
 - de revoir les procédures
 - de former les personnes

Le Workflow :

pour la donnée, c'est tout gagnant !

- Cohérence
- Rapidité
- Fluidité

Le Workflow

- C'est un exemple flagrant de la nécessité d'opérer un changement d'organisation pour assurer une meilleure productivité.

L'accès aux données : l'Infocentre (1)

- Un peu d'histoire...
- L'Infocentre dans les années 70-80, consistait à mettre à la disposition des utilisateurs finaux :
 - La puissance de calcul d'un **ordinateur** en temps partagé, accessible via des **terminaux**, pour accéder à des **banques de données** grâce à des **programmes** spécifiques (Basic, Cobol, Fortran)

L'accès aux données : l'Infocentre (2)

- C'était l'époque du Mainframe et des terminaux passifs.
- Démodé, non ???
 - pas tant que ça !

L'accès aux données : l'Infocentre (3)

- Dans les années 90, l'Infocentre est devenu dynamique et a été peu à peu remplacé par l'informatique décisionnelle, le datawarehouse, le datamart (entrepôt de données), ...

L'informatique décisionnelle (1)

- DSS = décision support system
- ou BI = business intelligence
- désigne les moyens, les outils et les méthodes qui permettent de collecter, consolider, modéliser et restituer les données d'une organisation afin de permettre à ses responsables d'avoir une vision d'ensemble

L'informatique décisionnelle (2)

- L'entrepôt de données :
 - Il peut prendre la forme d'un Datawarehouse ou d'un Datamart (plus orienté METIER spécifique dans l'organisation)

L'informatique décisionnelle (3)

- Le tableau de bord :
 - pour contrôler et prendre la bonne direction
- Ce qu'il s'est passé -> REPORTING
- Pourquoi ?
- Ce qu'il se passe
- Ce qu'il va se passer
- Que devrait-il se passer ?

L'informatique décisionnelle (4)

■ Le reporting

- Collecter les données
- Sélectionner des données sur une période donnée
- Ordonner : trier, filtrer selon des critères
- Calculer
- Présenter

La collecte des données (1)

- Une donnée devient une information lorsqu'elle est porteuse de sens.
- La mise à disposition d'un grand volume de données ne doit pas être la seule préoccupation
- Qui peut le plus, peut le moins... ne marche pas en informatique
- Donc, il faut bien identifier les besoins des utilisateurs et savoir quelle donnée aura du sens pour quel utilisateur

La collecte des données (2)

- Les principales difficultés sont :
 - L'accessibilité des données en raison de l'hétérogénéité du S.I
 - Le nettoyage des erreurs présentes dans les bases
 - La consolidation

La collecte des données (3)

- Pourquoi ?
- Le S.I d'une entreprise est un ensemble de briques de données plus ou moins bien empilées....au cours des années.
- Le résultat : des données hétérogènes et peu fiables quand elles sont consolidées (*limites des datamarts qui constituent un datawarehouse...*)

La collecte des données : comment faire ?

- On part des données sources issues des systèmes transactionnels de production :
 - Au fil de l'eau : (achats, ventes,...)
 - C'est le rendu de l'activité de l'entreprise au fil du temps des opérations
 - À un instant donné (stock, commandes passées, contrats,...)

De MERISE....

- Les outils pour modéliser les données
 - Dans le domaine des SGBD, on utilise le plus souvent les modèles « entité-relation » qui a fait le succès de la méthode MERISE et de son MCD (modèle conceptuel de données)

... à UML

- dans UML, on retrouve cette approche dans le diagramme de classes...

Partie II

UML

BIBLIOGRAPHIE

- « UML par la pratique », Pascal ROQUES, ED. Eyrolles
- « UML en action », Pascal ROQUES, ED. Eyrolles
- « Modélisation objet avec UML », P MULLER, N. GAERTNER, ED. Eyrolles
- <http://uml.free.fr/>

QU'EST CE QU'UN MODÈLE ?

Modèle

- = abstraction de la réalité
- = ensemble de concepts, règles, un formalisme
- = vue subjective mais pertinente de la réalité

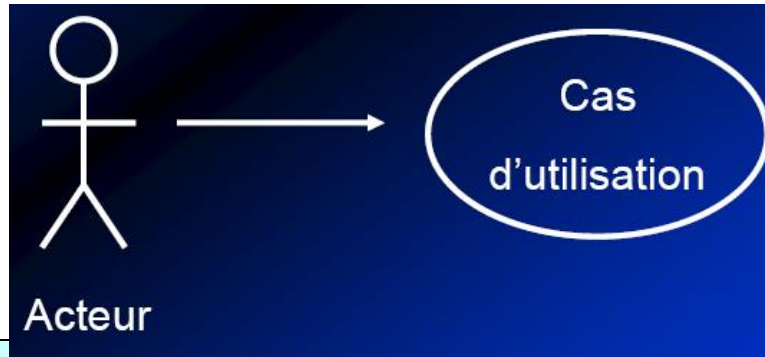
QU'EST CE QU'UN DIAGRAMME ? (1)

Diagramme

= Possibilité de visualiser et de manipuler (éditer) des éléments de modélisation

= Représentation graphique d'une séquence d'opérations ou de la structure d'un système

QU'EST CE QU'UN DIAGRAMME ? (2)

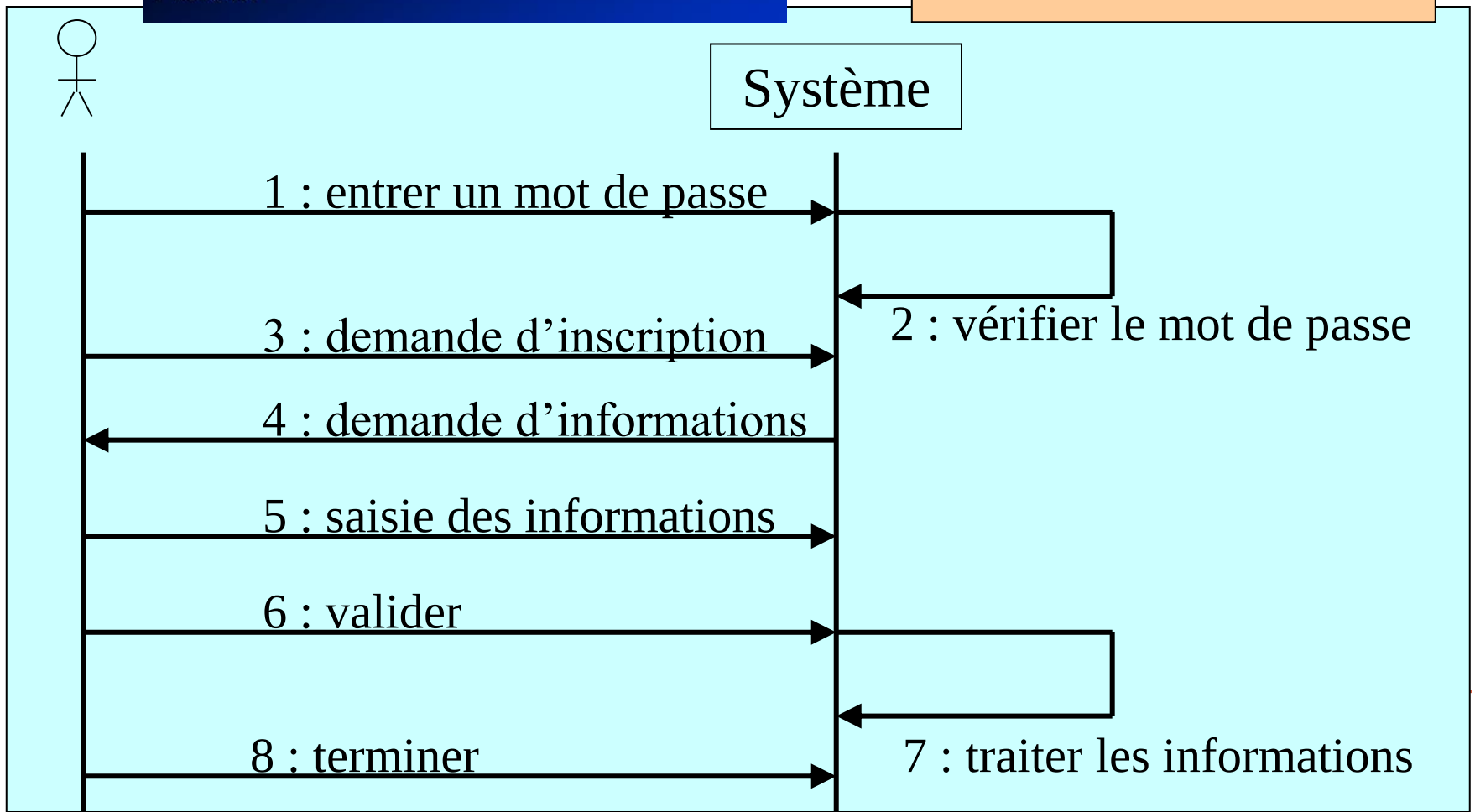


Personne

nom : chaîne

prénom : chaîne

date de naissance : date



Méthode d'analyse et de conception

- Une méthode = **4 éléments fondamentaux**
- Décrire une **DÉMARCHE** qui liste les tâches à effectuer
- Fournir un **MODÈLE** pour décrire la sémantique des données ainsi que leurs comportements
- Fournir un ensemble de **DIAGRAMMES** s'appuyant sur un **FORMALISME** de description (graphique ou textuel)

INTRODUCTION À LA NOTATION UML (1)

UML

= langage destiné aux phases amont de la réalisation d'un logiciel.

= technique de modélisation unifiée issue de méthodes orientées objet plus anciennes :

- OMT (James Rumbaugh),
- OOSE (Ivar Jacobson),
- OOD (Grady Booch)

Intérêt des standards = faciliter la communication

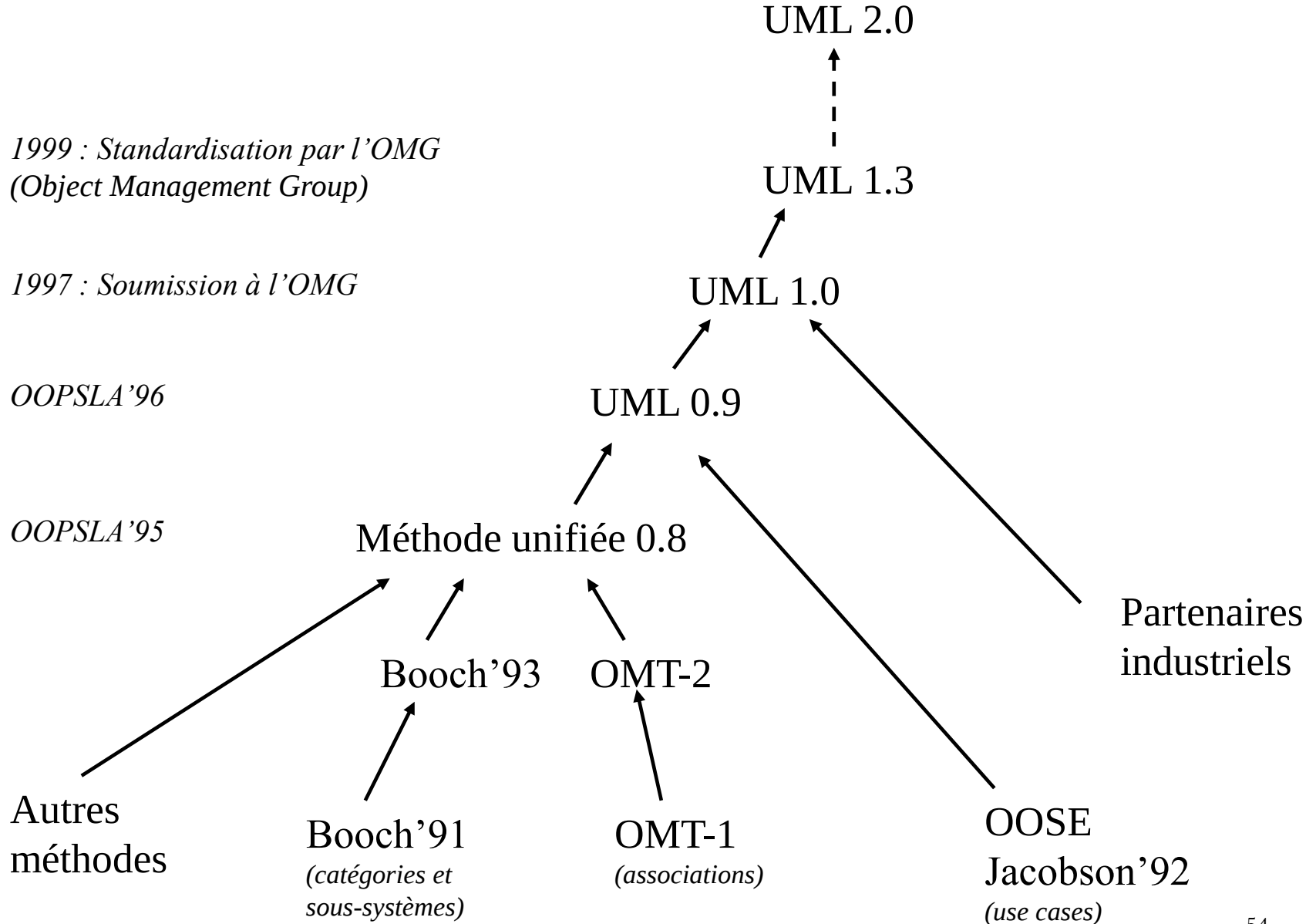
INTRODUCTION À LA NOTATION UML (2)

*1999 : Standardisation par l'OMG
(Object Management Group)*

1997 : Soumission à l'OMG

OOPSLA'96

OOPSLA'95



OBJECTIFS D'UML

- Représenter des systèmes entiers
 - Prendre en compte les facteurs d'échelle
 - Créer un langage de modélisation
 - utilisable à la fois par les hommes et les machines
 - compatible avec toutes les techniques de réalisation
 - adapté à toutes les phases du développement
 - Indépendant des langages de programmation
-

UML EST UN LANGAGE POUR ...

■ VISUALISER

- chaque symbole graphique a une sémantique

■ SPÉCIFIER

- de manière précise et complète, sans ambiguïté

■ CONSTRUIRE

- les classes, les relations, ...

■ DOCUMENTER

- les ≠ diagrammes, notes, contraintes, exigences seront présentés dans un document
-

Mais attention... (1)



UML est un **langage** → pas une méthode !

savoir parler ne signifie pas qu'on sait à qui parler, à quel moment engager la conversation, et quoi dire

- UML ne résout pas tous les problèmes de communication :
 - précision → diagrammes complexes → difficiles à comprendre
 - compréhensibilité → diagrammes plus simples → sujets à interprétation (reste une part de subjectivité)
 - comme avec tout langage, on peut « mal s'exprimer » (erreur, imprécision, incohérence, ...)

Mais attention... (2)

-  UML ne "guide" pas :
- ❑ pas de conduite de projet,
 - ❑ pas de gestion de la qualité,
 - ❑ pas de gestion du changement,
 - ❑ pas de description des rôles,
 - ❑ ...

POURQUOI L'APPROCHE OBJET ?

- Possibilité de réutiliser les éléments d'un développement à un autre
- Simplicité du modèle de représentation qui ne fait appel qu'à des concepts fondateurs :
 - Les objets
 - Les messages
 - Les classes
 - La généralisation
 - Le polymorphisme

Les diagrammes UML 1.4

Le langage UML 1.4 propose 9 diagrammes :

- 4 statiques (structure - données)

- Diagramme d'objets
- Diagramme de classes
- Diagramme de composants
- Diagramme de déploiement

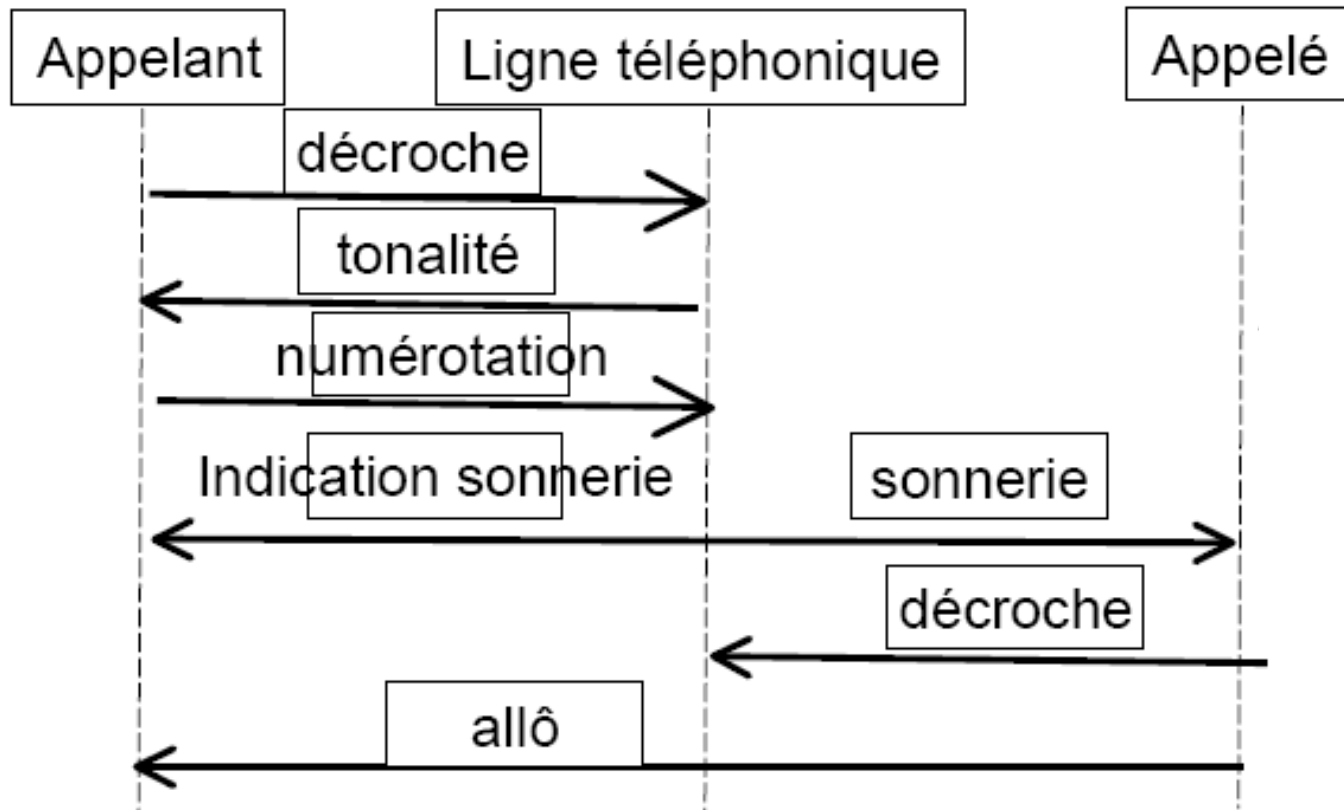
- 5 dynamiques (comportement)

- Diagramme de cas d'utilisation
- Diagramme de collaboration
- Diagramme de séquence
- Diagramme d'états-transition
- Diagramme d'activités

RELATION ENTRE DIAGRAMMES ET ÉTAPES DU PROCESSUS DE DÉVELOPPEMENT (1)

- *Découverte et analyse des besoins :*
 - Diagramme de cas d'utilisation : décrit les fonctions du système selon le point de vue de ses futurs utilisateurs
 - Diagramme de séquence : représentation temporelle des objets et de leurs interactions
 - Diagramme d'activités: pour des représentations grossières, ils représentent le comportement d'une méthode ou d'un cas d'utilisation.

Exemple de diagrammes de séquence :

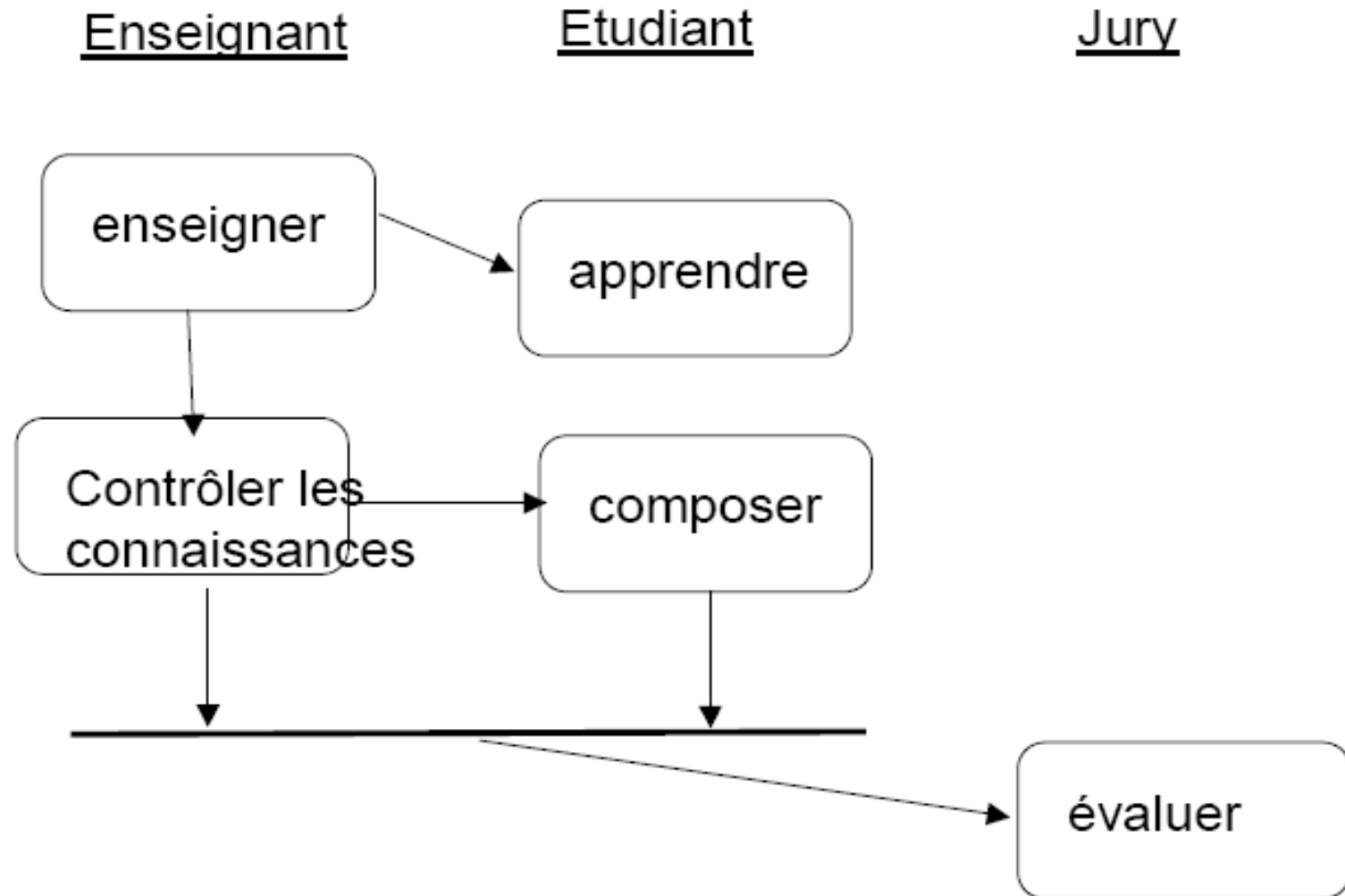


RELATION ENTRE DIAGRAMMES ET ÉTAPES DU PROCESSUS DE DÉVELOPPEMENT (2)

■ *Spécification :*

- Diagramme de classes : structure des données du système définies comme un ensemble de relations et de classes
- Diagramme d'objets : illustration des objets et de leurs relations. Ce sont des diagrammes de collaboration simplifiés sans représentation des envois de messages.
- Diagramme de collaboration : représentation des interactions entre objets
- Diagramme d'états-transitions : représentation du comportement des objets d'une classe en terme d'états et de transitions d'états
- Diagramme d'activités: pour des représentations grossières, ils représentent le comportement d'une méthode ou d'un cas d'utilisation.

Exemple de diagrammes d'activités :

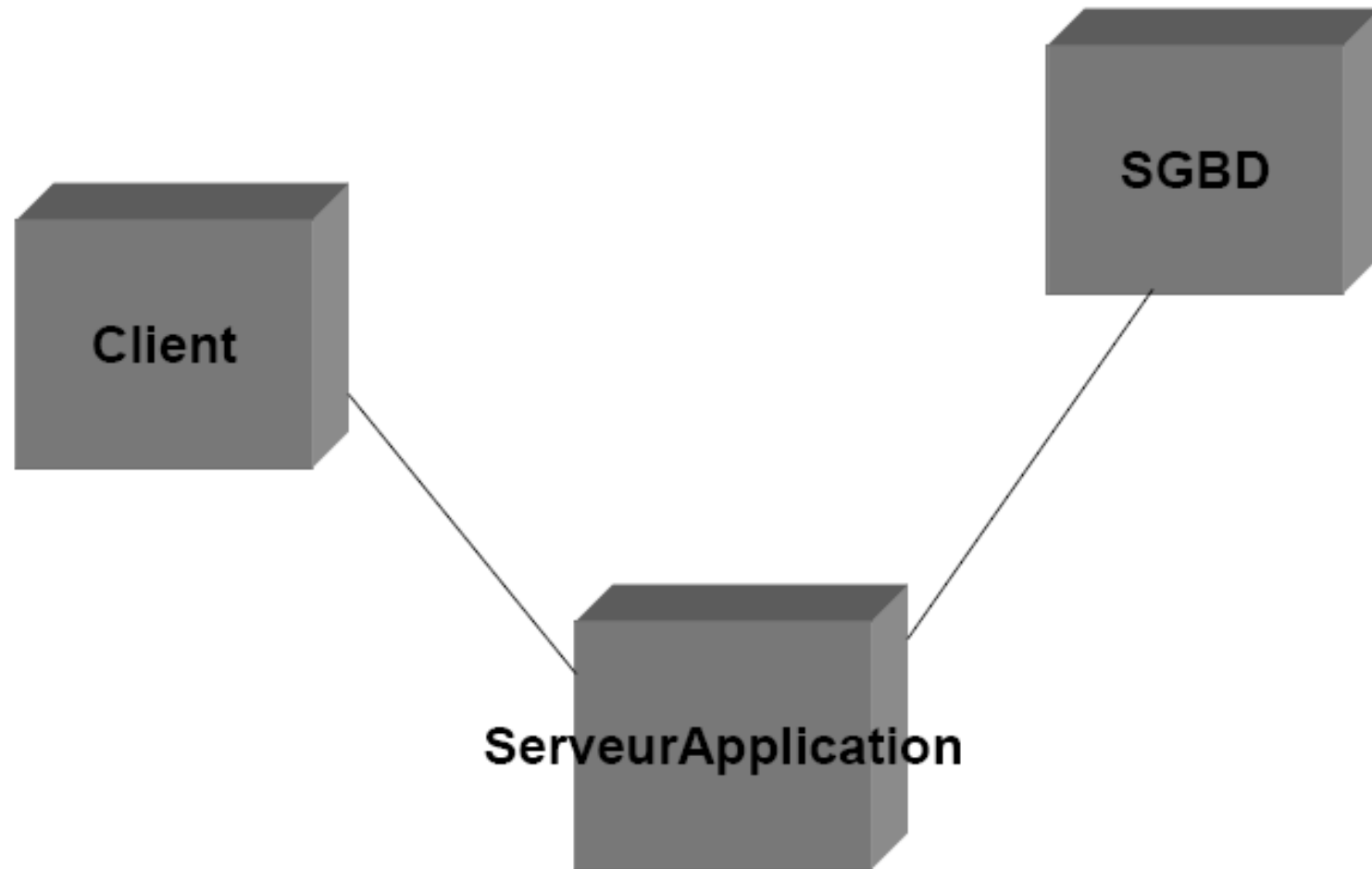


RELATION ENTRE DIAGRAMMES ET ÉTAPES DU PROCESSUS DE DÉVELOPPEMENT (3)

■ **Conception :**

- Diagramme de séquence : représentation des interactions temporelles entre objets dans la réalisation d'une opération
- Diagramme de composants : architecture des composants physiques d'une application
- Diagramme de déploiement : description du déploiement des composants sur les dispositifs matériels

Exemple de diagramme de déploiement :



Attention !

- Un même type de diagramme (e.g. diagramme d'activité)
 - peut être utilisé :
 - pour modéliser des concepts différents
 - à des moments différents du processus de développement (donc pour des objectifs différents)
 - à différents niveaux d'abstraction
 - peut ne pas être utilisé du tout dans le processus de développement

DIAGRAMMES DE CAS D'UTILISATION

(USE CASES)

DIAGRAMMES DE CAS D'UTILISATION (1) (Interaction d'un acteur extérieur au système avec le système : Use Cases)

Constat : Le système existe pour servir ses utilisateurs

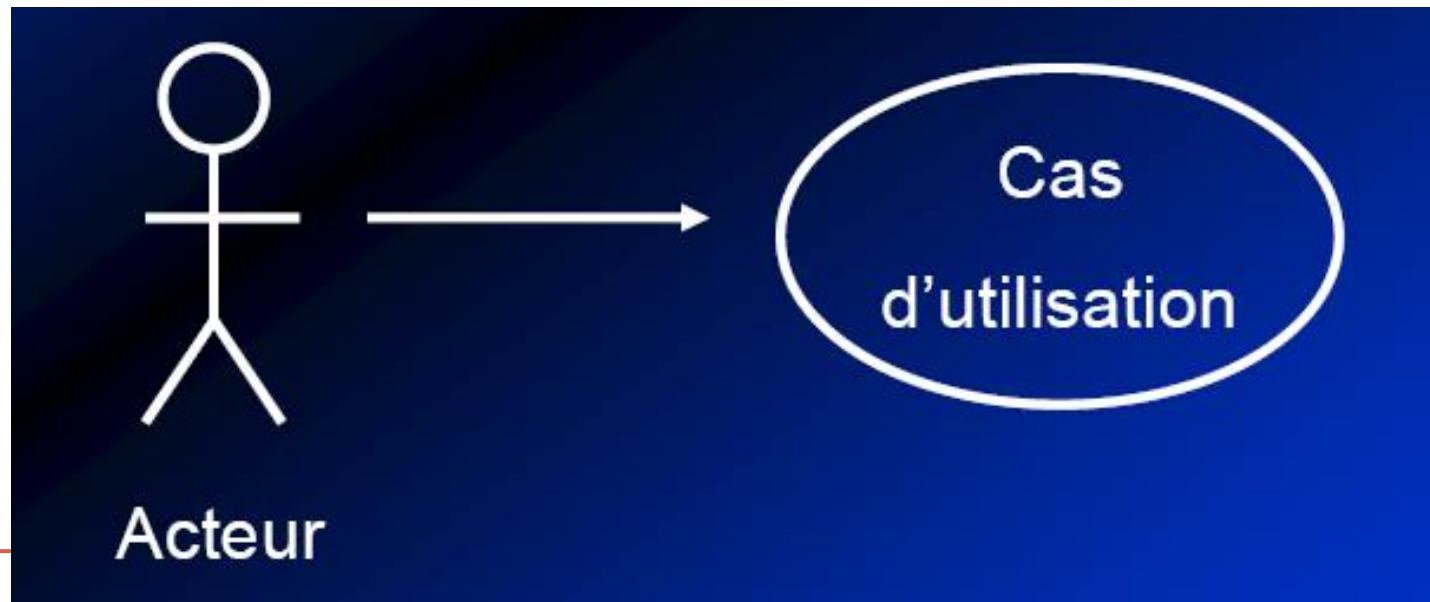
Cas d'utilisation (use cases) [Jacobson 92]

= Idée : **description du comportement du système** du point de vue de son utilisateur (facilite l'expression des besoins)

Comportement = {Actions}+{Réactions}

DIAGRAMMES DE CAS D'UTILISATION (2)

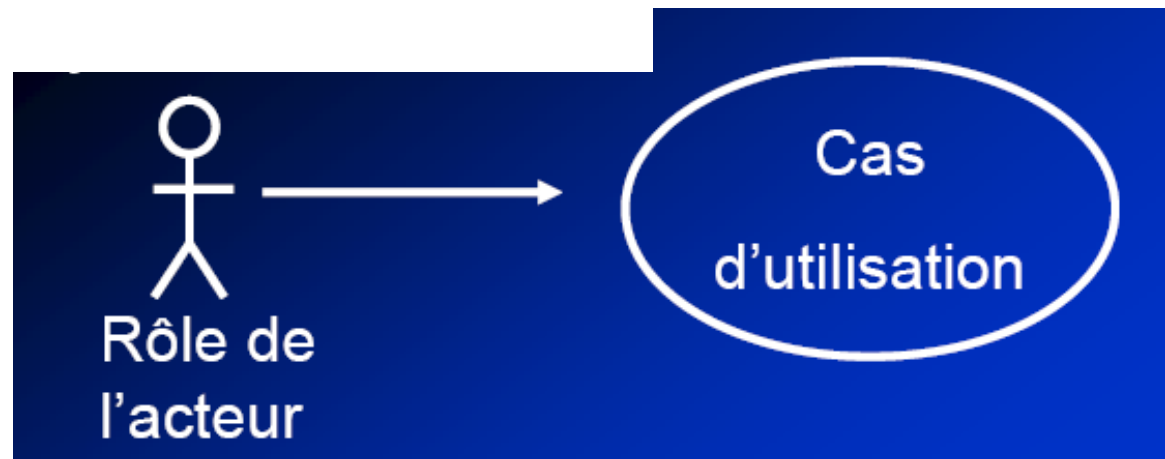
- Un cas d'utilisation correspond à une manière spécifique d'utiliser le système



DIAGRAMMES DE CAS D'UTILISATION (3)

Cas d'utilisation : définitions

- **Acteur** : entité externe qui agit sur le système



DIAGRAMMES DE CAS D'UTILISATION (4)

Acteurs vs utilisateurs

- **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...
- **Types d'acteurs :**
 - Utilisateur\$ **principaux**
 - Utilisateurs **secondaires**
 - **Périphériques externes**
 - **Systèmes externes**

DIAGRAMMES DE CAS D'UTILISATION (5)

Définition des Acteurs

- Pour chaque acteur :
 - **choisir un identificateur représentatif du rôle**
 - **éventuellement accompagné d'une brève description textuelle :**



guichetier

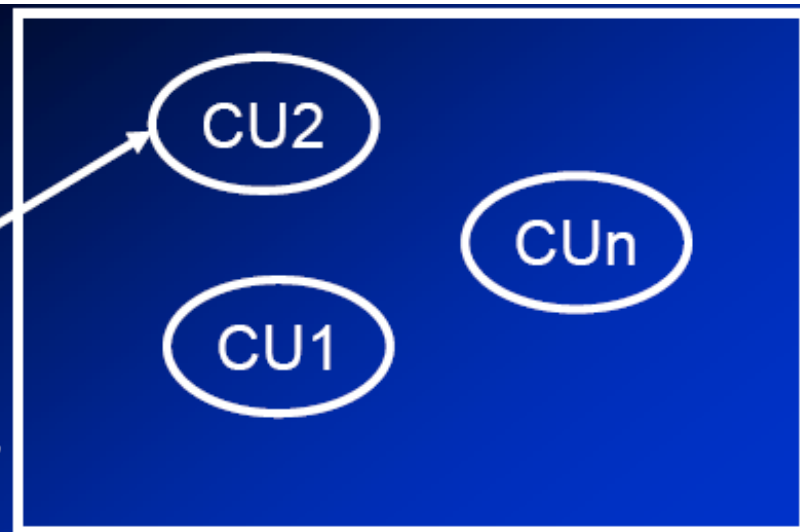
Un guichetier est un employé de la banque jouant un rôle d'interface entre le système informatique et les clients qu'il reçoit au comptoir

DIAGRAMMES DE CAS D'UTILISATION (6)

Cas d'utilisation : définitions

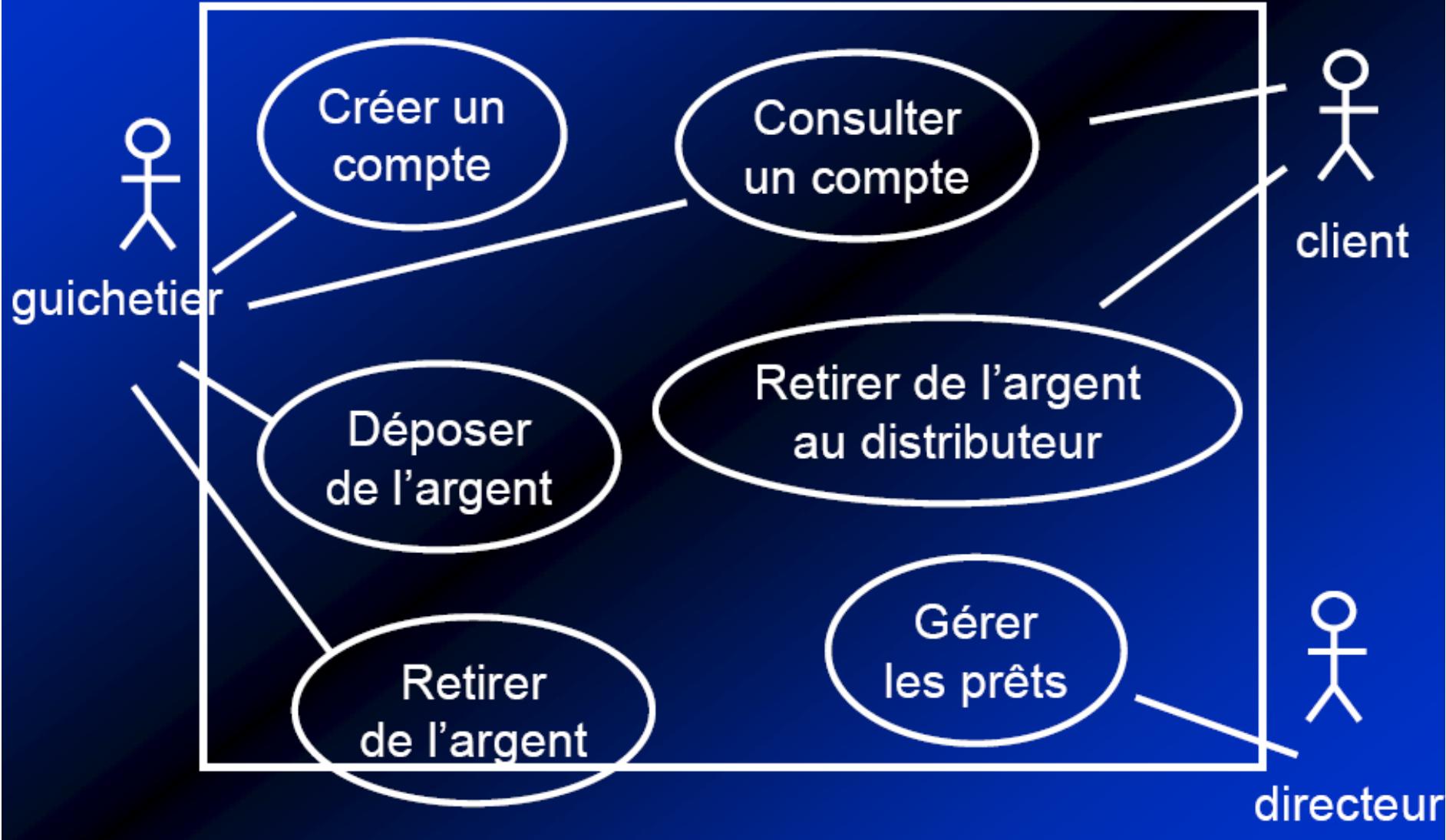
Cas d'utilisation: ensemble des actions réalisées par le système en réponse à une action d'un acteur

• les cas d'utilisation ne doivent pas se chevaucher



DIAGRAMMES DE CAS D'UTILISATION (7)

Cas d'utilisation : exemples de diagrammes de CU



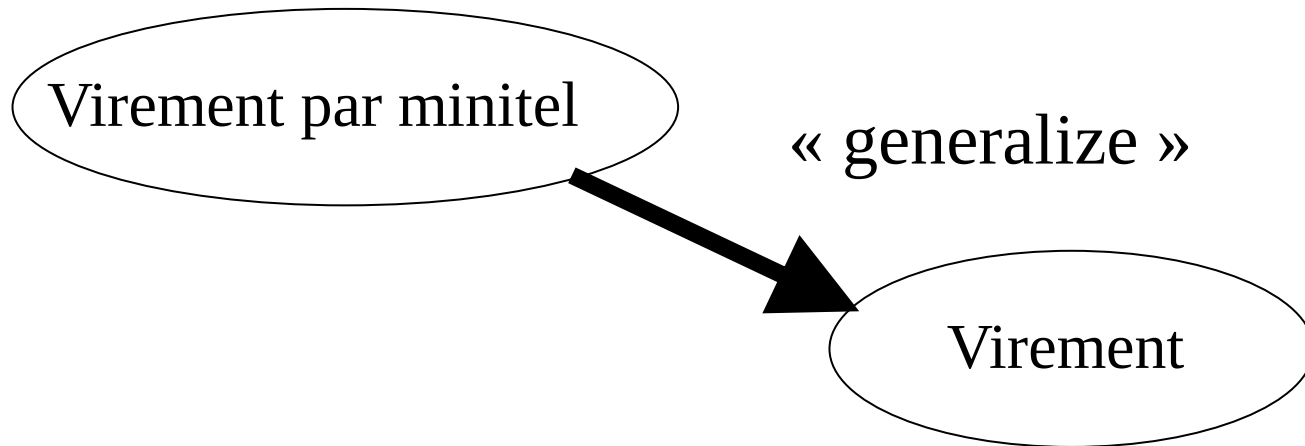
Relations entre cas d'utilisation

- La généralisation
- L'inclusion
- L'extension

DIAGRAMMES DE CAS D'UTILISATION (9)

Relations entre cas d'utilisation

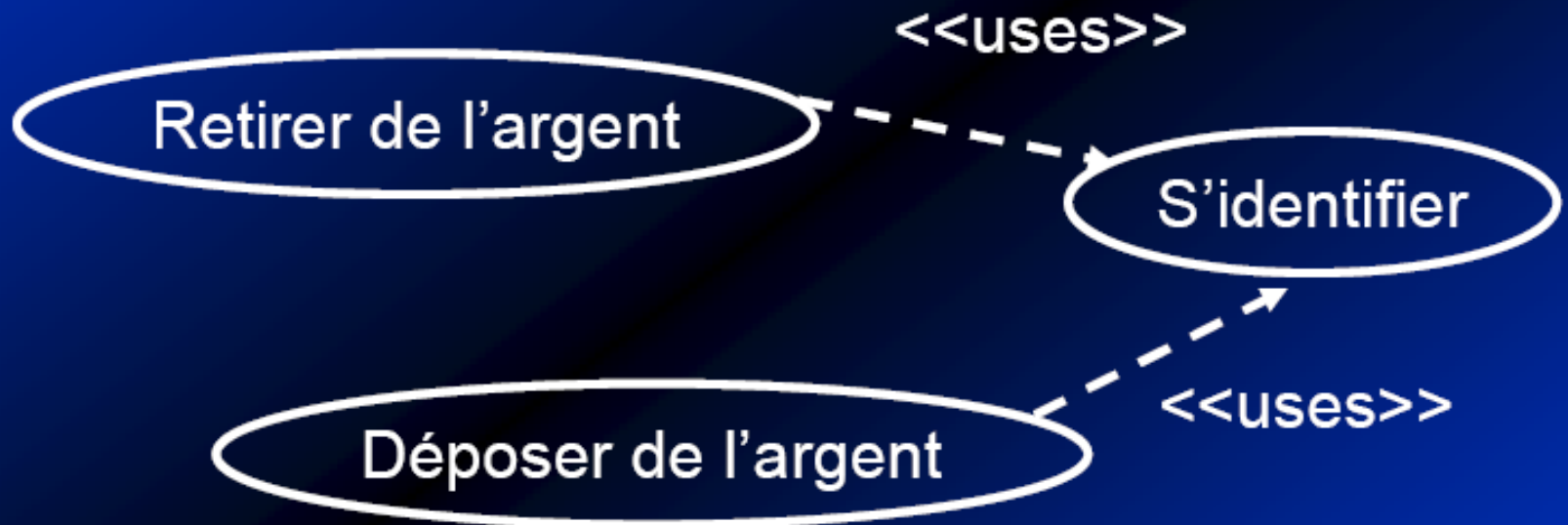
Relation ***generalize*** entre cas d'utilisation



DIAGRAMMES DE CAS D'UTILISATION (10)

Relations entre cas d'utilisation

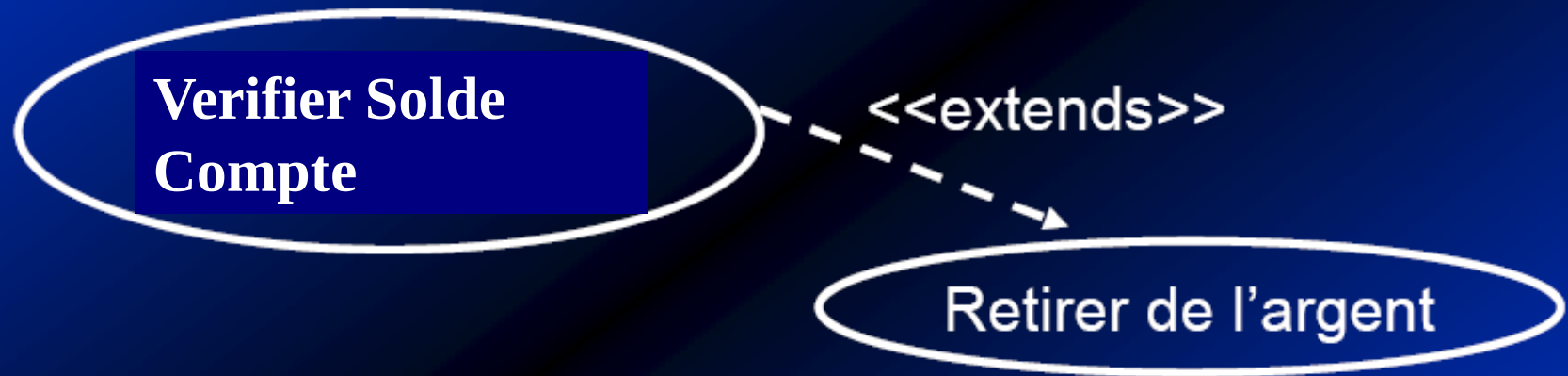
Relation *uses* entre cas d'utilisation



DIAGRAMMES DE CAS D'UTILISATION (11)

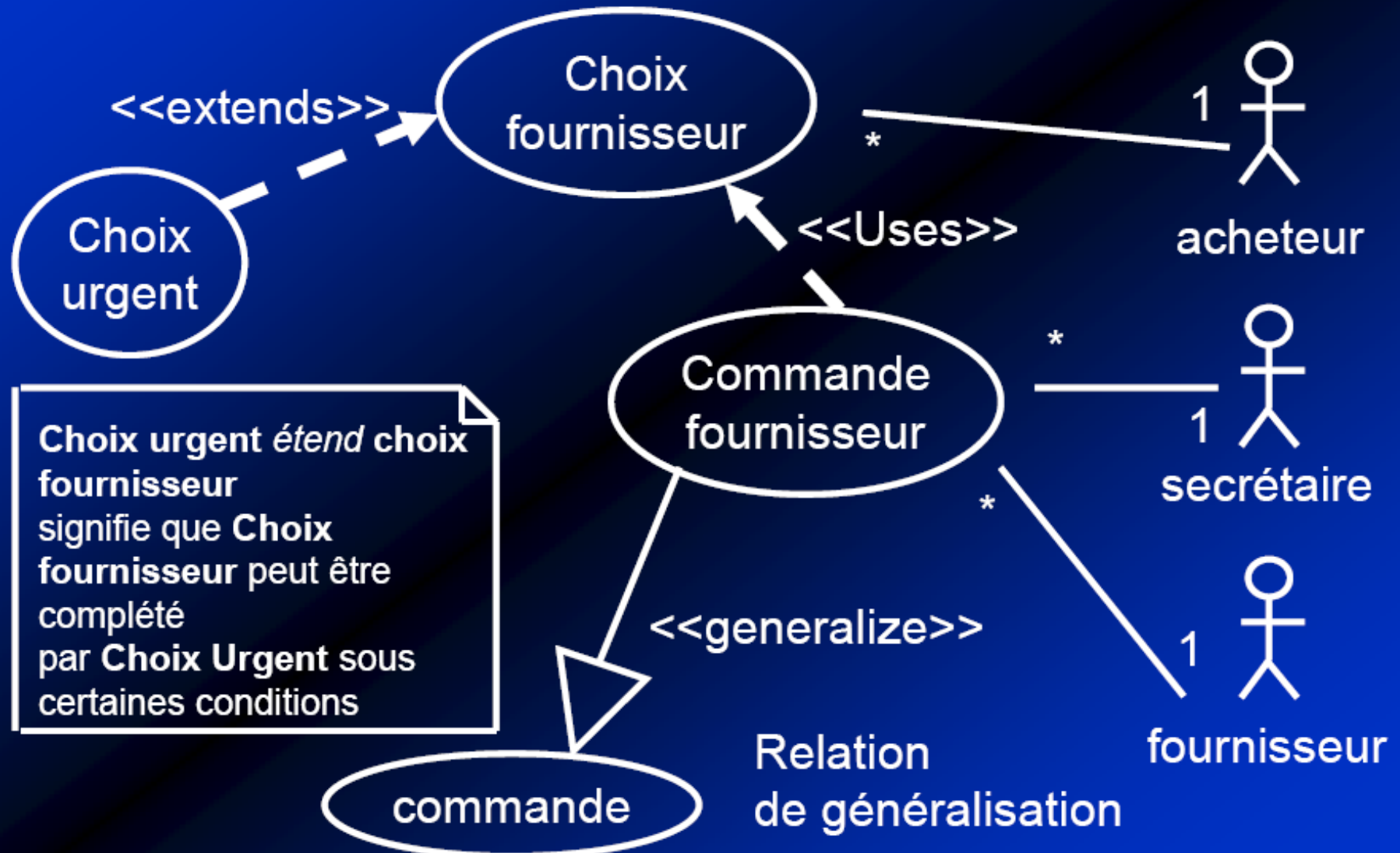
Relations entre cas d'utilisation

Relation ***extends*** entre cas d'utilisation



DIAGRAMMES DE CAS D'UTILISATION (12)

Exemple de relations



DIAGRAMMES DE CAS D'UTILISATION (13)

Cas d'utilisation et scénario

- le **système** = **ensemble** de **cas d'utilisation**
- le système possède les **cas d'utilisation** mais pas les acteurs
- Un **cas d'utilisation** = ensemble de « chemins d'exécution » possibles
- Un **scénario** = un chemin particulier d'exécution
= une séquence d'événements
- Un **scénario** = Instance de cas d'utilisation
- Une **instance d'acteur crée** un **scénario**

Cas d'utilisation et scénario

- **spécification exhaustive de tous les scénarios**
difficile, voire impossible
- **sélection des scénarii les plus intéressants**
 - **scénario optimal** : décrit l'interaction la plus fréquente
 - **scénarios dérivés** : décrit certaines alternatives importantes non décrites dans le scénario optimal

Attention !

- Attention au niveau de détail : un CU n'est pas une fonction
- Remarques importantes (pour le DCU mais valable pour tous les autres types diagrammes)
 - il existe des variantes (*notation des acteurs par exemple*), des compléments (*multiplicité*), d'autres composants, etc
 - on ne définit pas « le » diagramme mais UN diagramme (tient compte de votre vision et de vos objectifs)

DIAGRAMMES DE SÉQUENCE

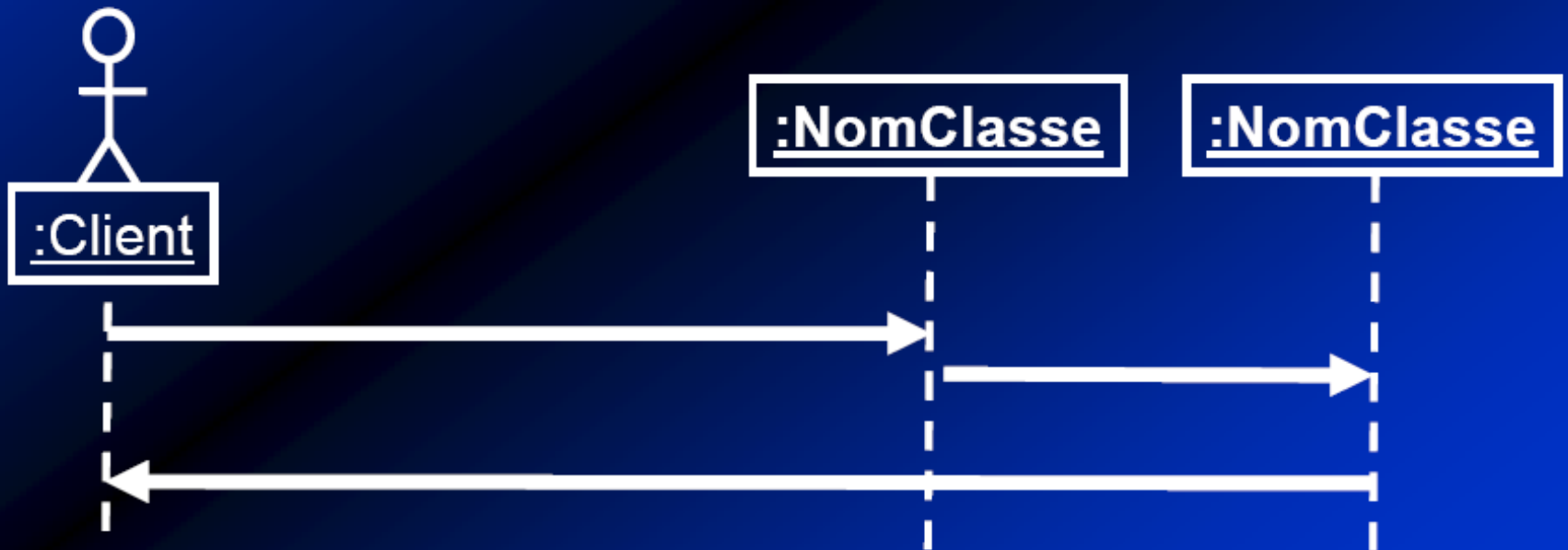
DIAGRAMMES DE SÉQUENCE (1)

(séquence d'enchaînement de messages pour décrire une interaction entre objets)

Introduction

Diagramme de séquence :

- Modélisation des **interactions entre objets** suite à un **événement externe**
- **Aspect temporel : messages asynchrones ou synchrones**



DIAGRAMMES DE SÉQUENCE (2)

Catégories de messages

2 catégories de messages :

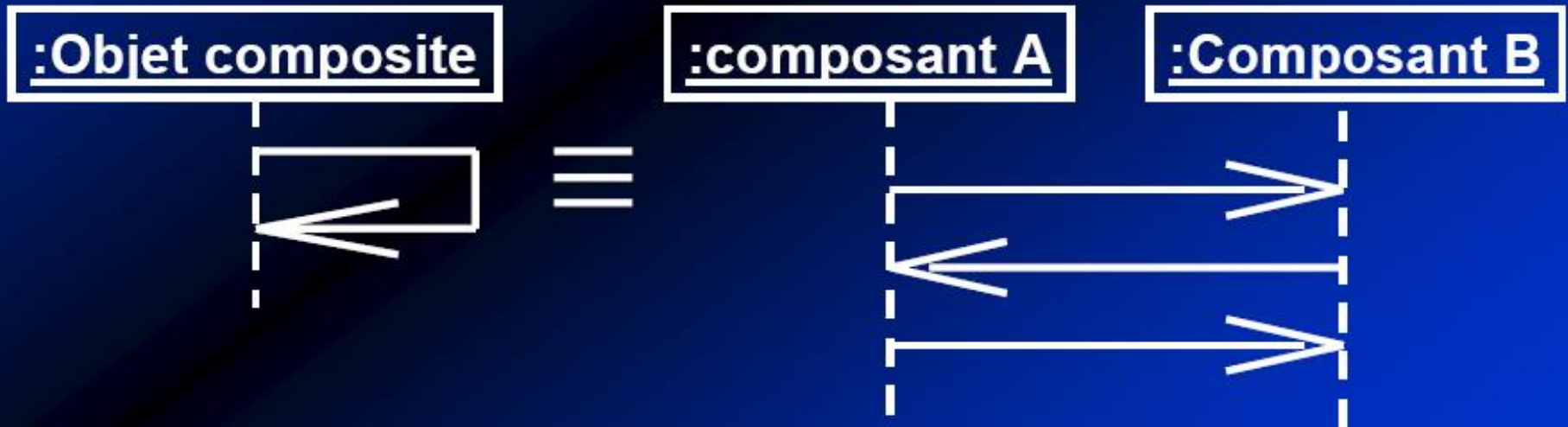
- **synchrone** : l'émetteur est bloqué jusqu'au traitement effectif du message
- **asynchrone** : l'émetteur n'est pas bloqué, il peut poursuivre son exécution



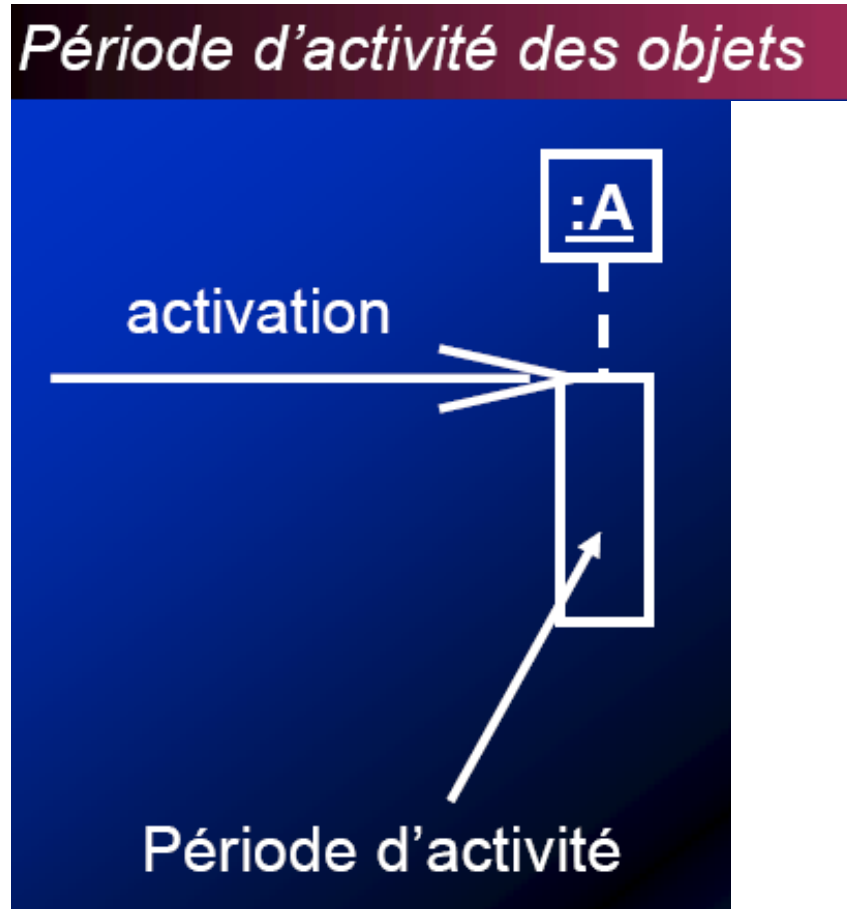
DIAGRAMMES DE SÉQUENCE (3)

Envoi de messages d'un objet sur lui même

Un objet peut s'envoyer des messages à lui-même :

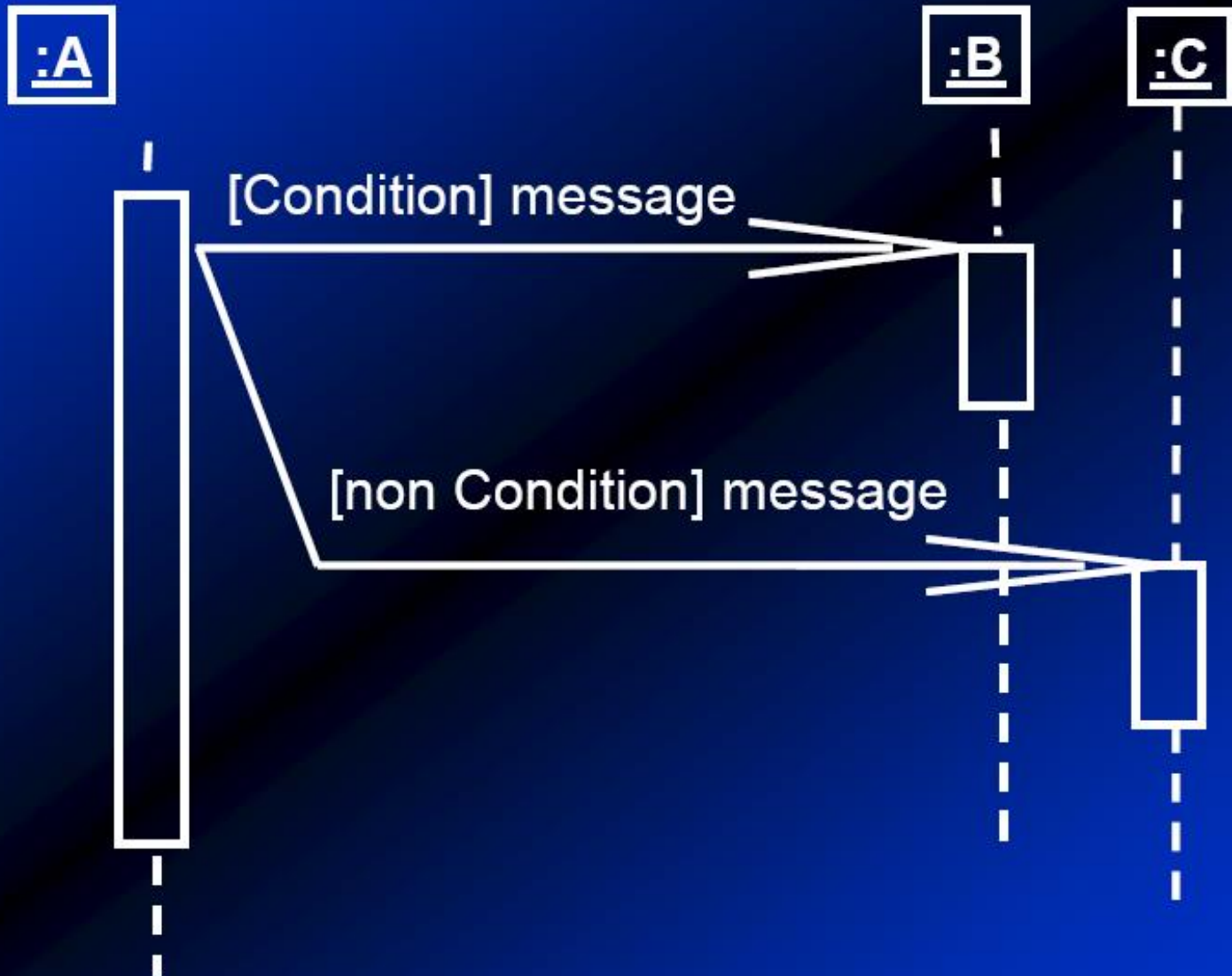


DIAGRAMMES DE SÉQUENCE (4)



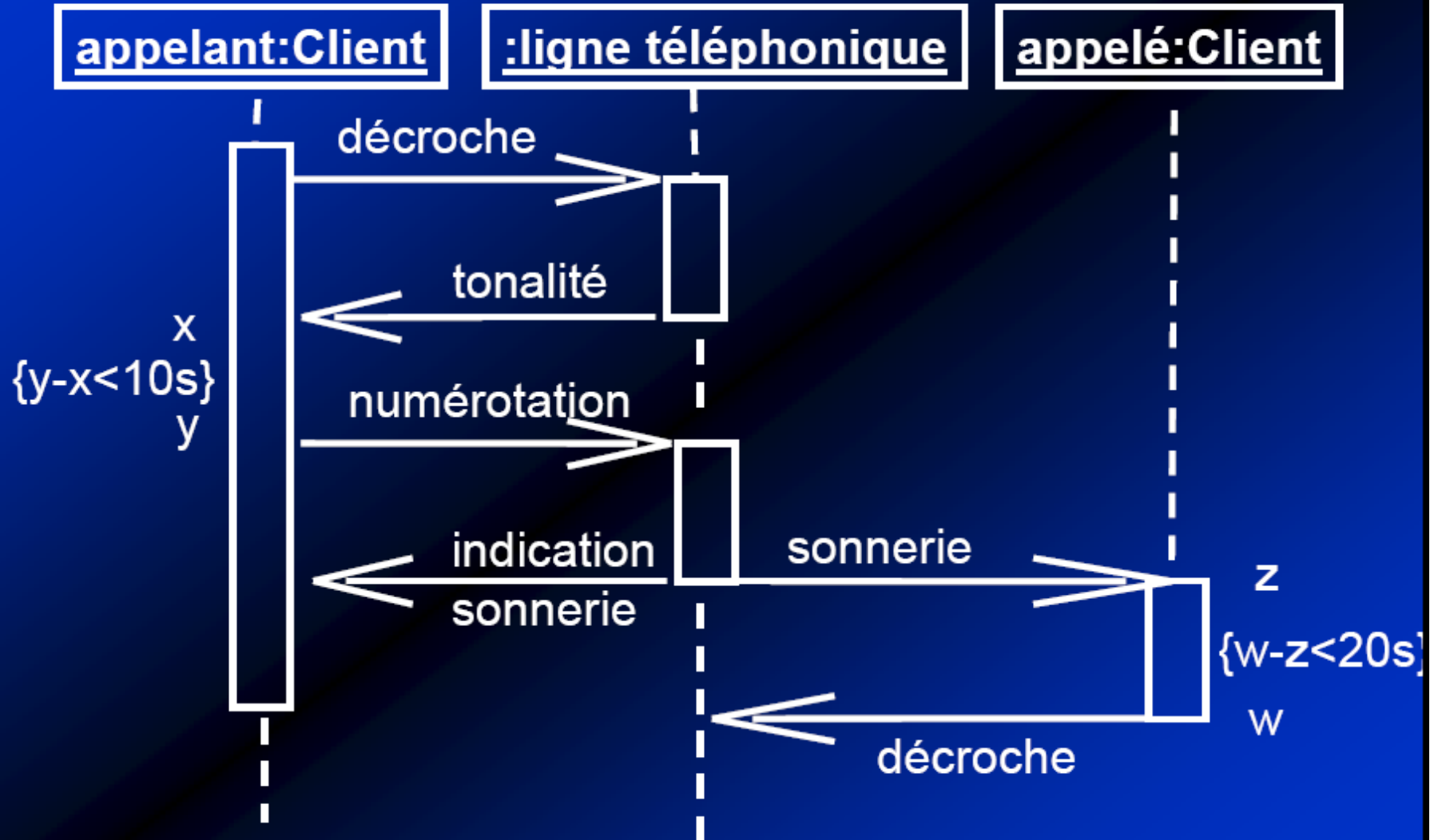
DIAGRAMMES DE SÉQUENCE (5)

Envoi conditionnel de message



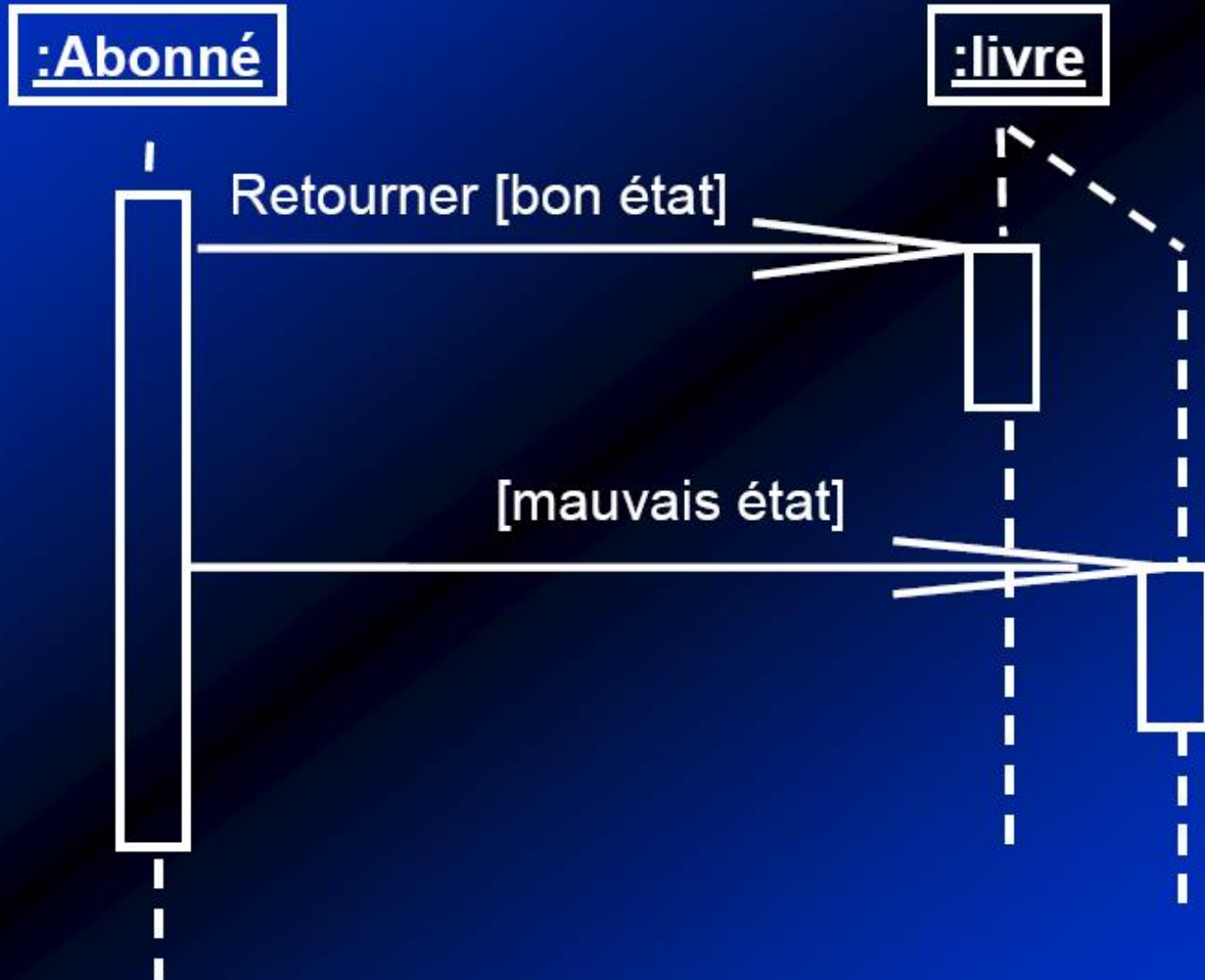
DIAGRAMMES DE SÉQUENCE (6)

Contraintes temporelles



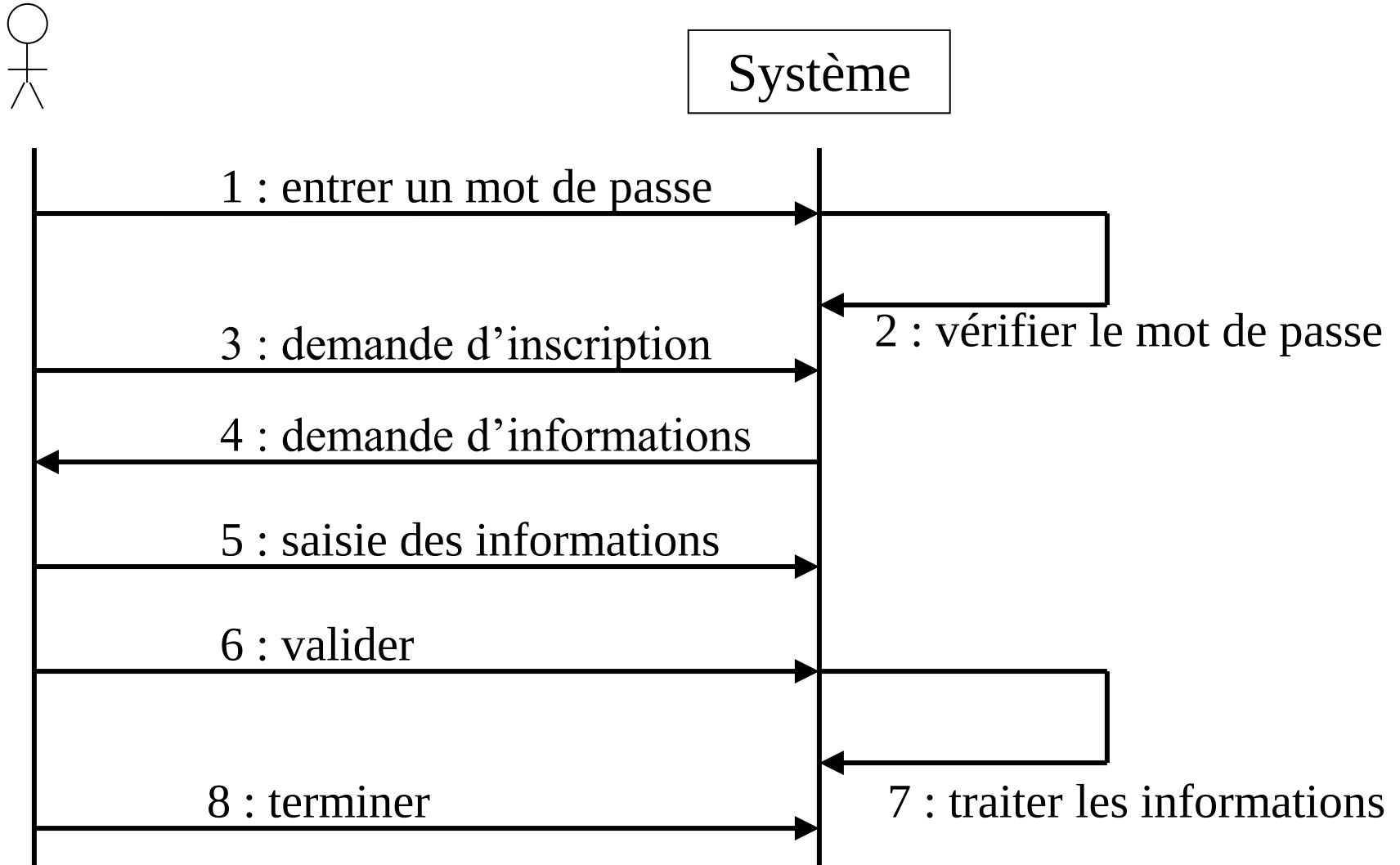
DIAGRAMMES DE SÉQUENCE (7)

Traitement conditionnel d'un message reçu



DIAGRAMMES DE SÉQUENCE (8)

Exemple :



DIAGRAMMES DE CLASSES

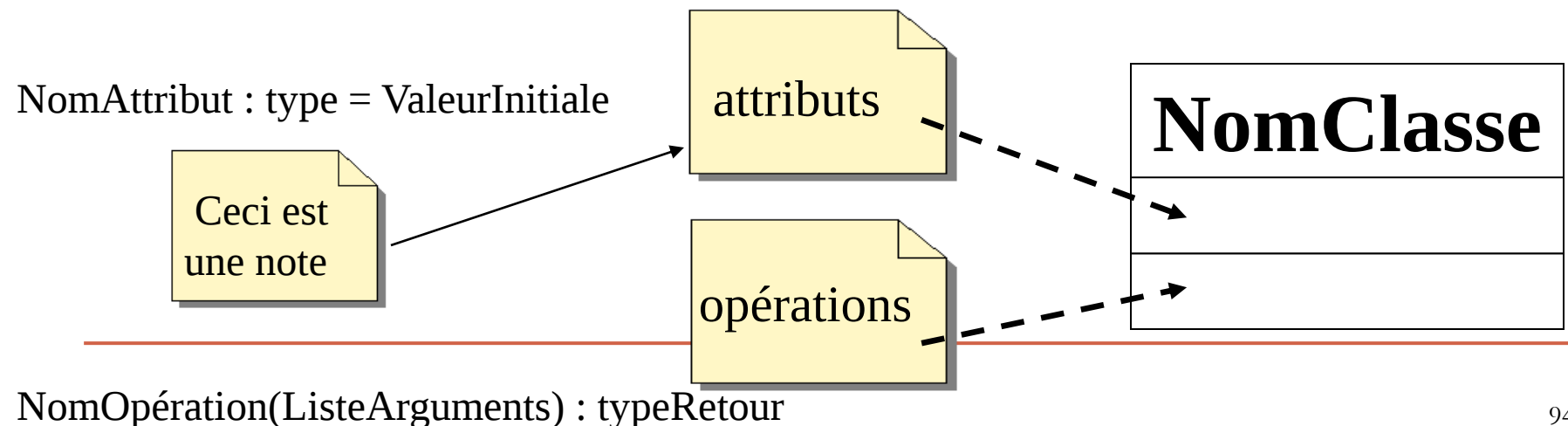
DIAGRAMMES DE CLASSES (1)

(structure statique du système sur lequel on a appliqué un concept de classification)

Classe = description abstraite d'un ensemble d'objets ayant :

- Des propriétés similaires (attributs)
- Un comportement commun (opérations)
- Des relations communes avec d'autres objets
- Des sémantiques communes

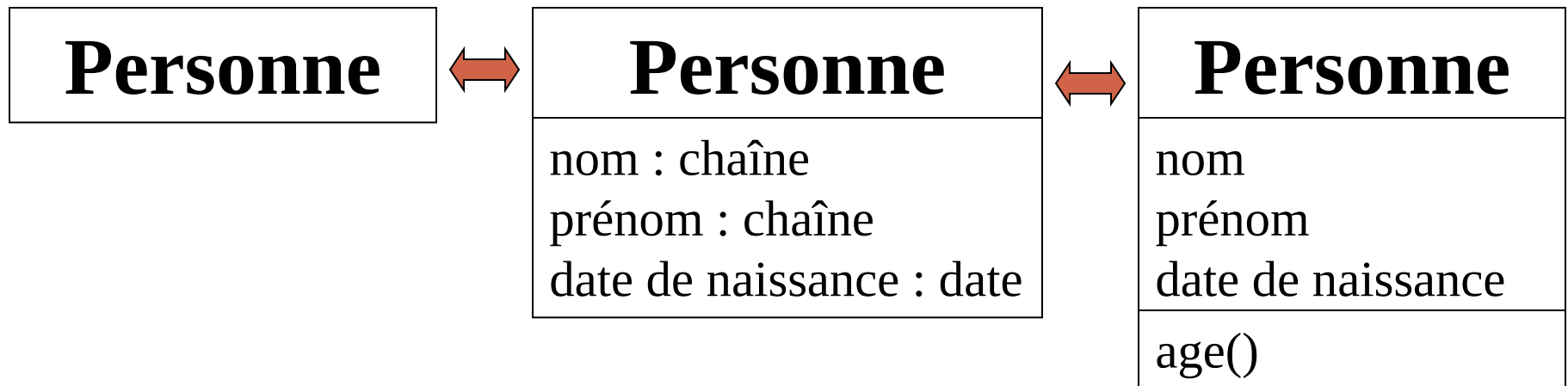
*Tout nom de classe commence par une **majuscule**.*



DIAGRAMMES DE CLASSES (2)

Les compartiments d'une classe peuvent être omis si leur contenu n'est pas pertinent dans le contexte du diagramme.

Exemple :



DIAGRAMMES DE CLASSES (3)

Accessibilité aux attributs et opérations d'une classe :

3 niveaux de protection :

- Public (+) : accès à partir de toute entité interne ou externe à la classe
- Protégé (#) : accès à partir de la classe ou des sous-classes
- Privé (-) : accès à partir des opérations de la classe

Exemple :

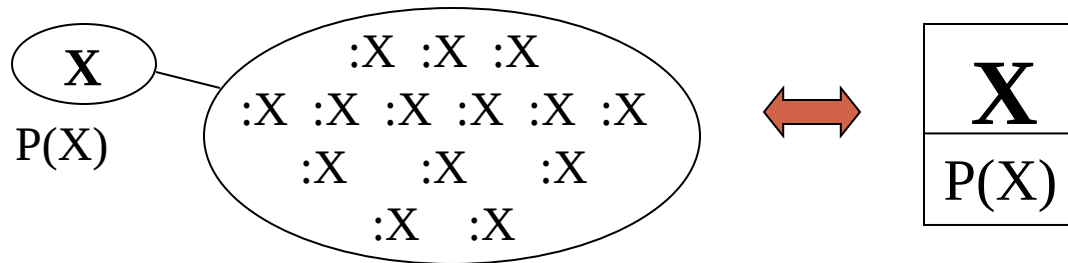
Personne	
- nom : chaîne - prénom : chaîne - date de naissance : date	
+ getNom() getDateNaissance()	+

DIAGRAMMES DE CLASSES (4)

De l'ensemble aux classes (1)

La notion de classe est très proche de la notion d'ensemble

- *Soit un ensemble X , la propriété caractéristique d'un ensemble X est noté $P(X)$.*

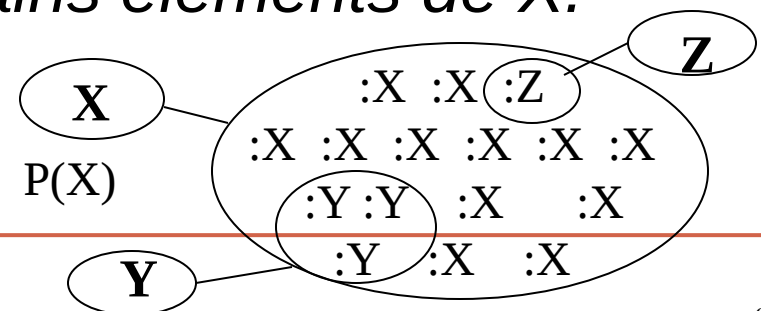


- *L'ensemble X peut être divisé en sous-ensembles (ici Y et Z) pour distinguer des particularités supplémentaires partagées seulement par certains éléments de X .*

(X): Livres

(Y): Livres pour enseignants

(Z): Livres pour enfants



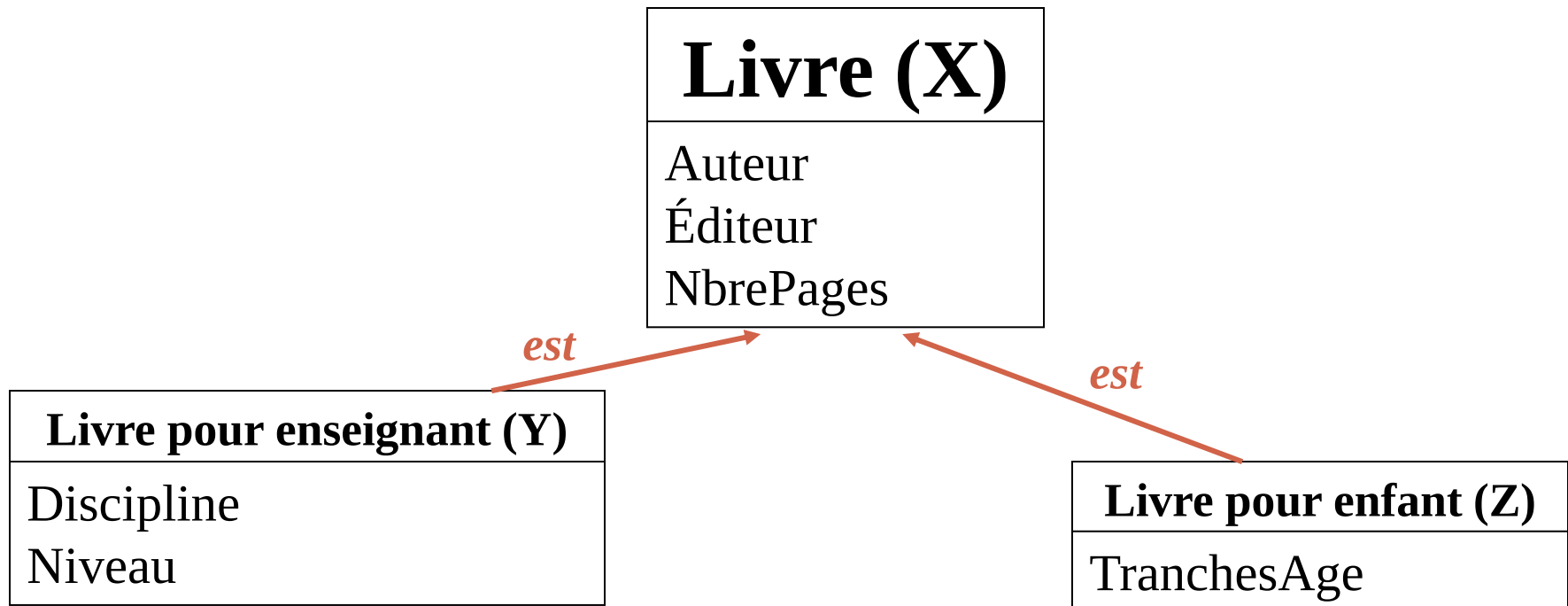
DIAGRAMMES DE CLASSES (5)

De l'ensemble aux classes (2)

Les éléments des ensembles de Y et Z sont d'abord des éléments de l'ensemble X .

Les propriétés caractéristiques de $P(Y)$ et $P(Z)$ englobent la propriété caractéristique de $P(X)$. Les classes et les sous-classes sont des ensembles et des sous-ensembles.

Nous obtenons la classification suivante :

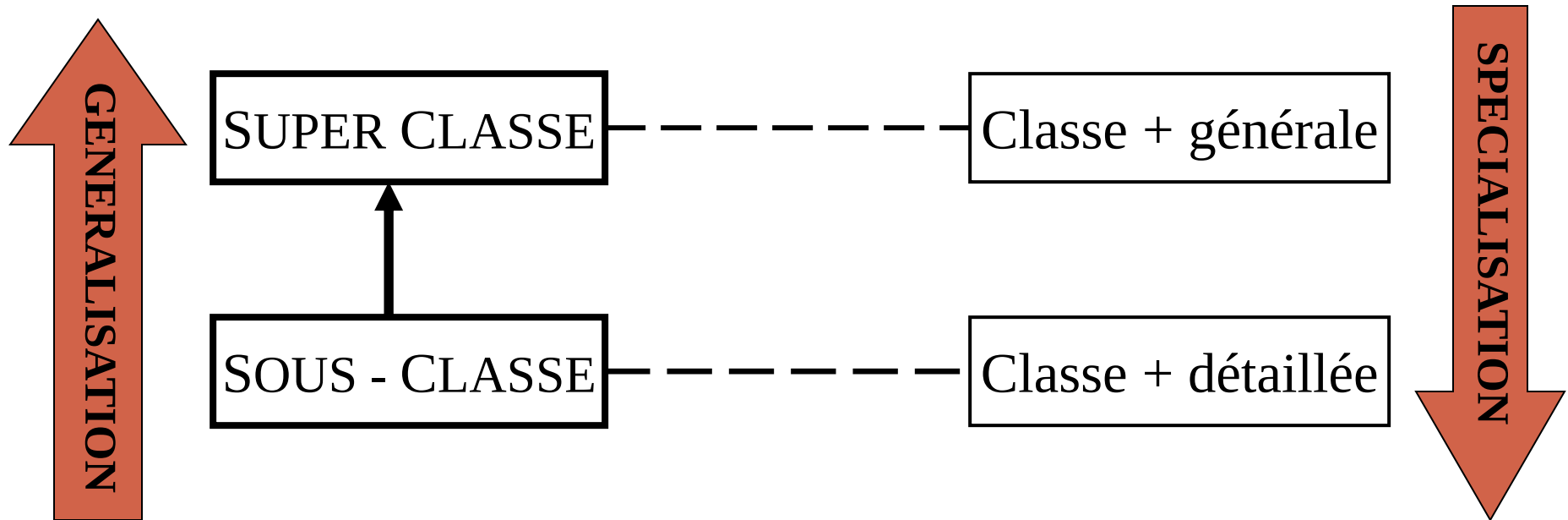


DIAGRAMMES DE CLASSES (6)

La classification

2 approches : * la généralisation } points de vue portés sur les
 * la spécialisation hiérarchies de classes

Symbole de représentation : ↑



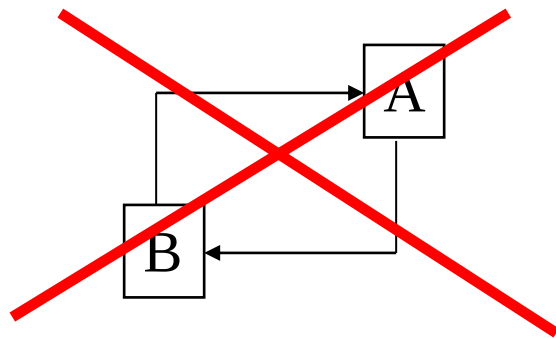
DIAGRAMMES DE CLASSES (7)

La classification : La généralisation

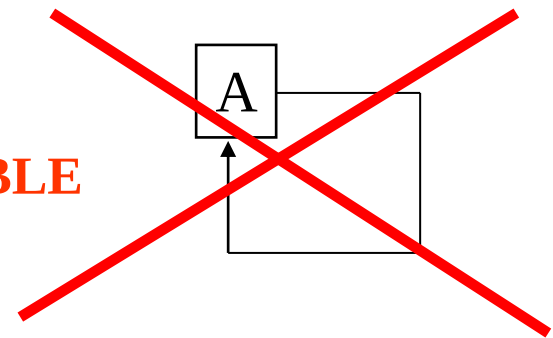
Généralisation

= factoriser les éléments communs (attributs, opérations, contraintes) d'un ensemble de classes dans une classe plus générale appelée Super-Classe (ou classe mère).

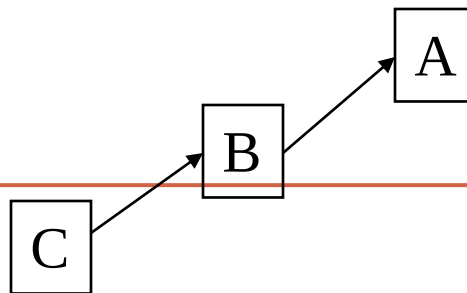
= relation non réflexive :



IMPOSSIBLE

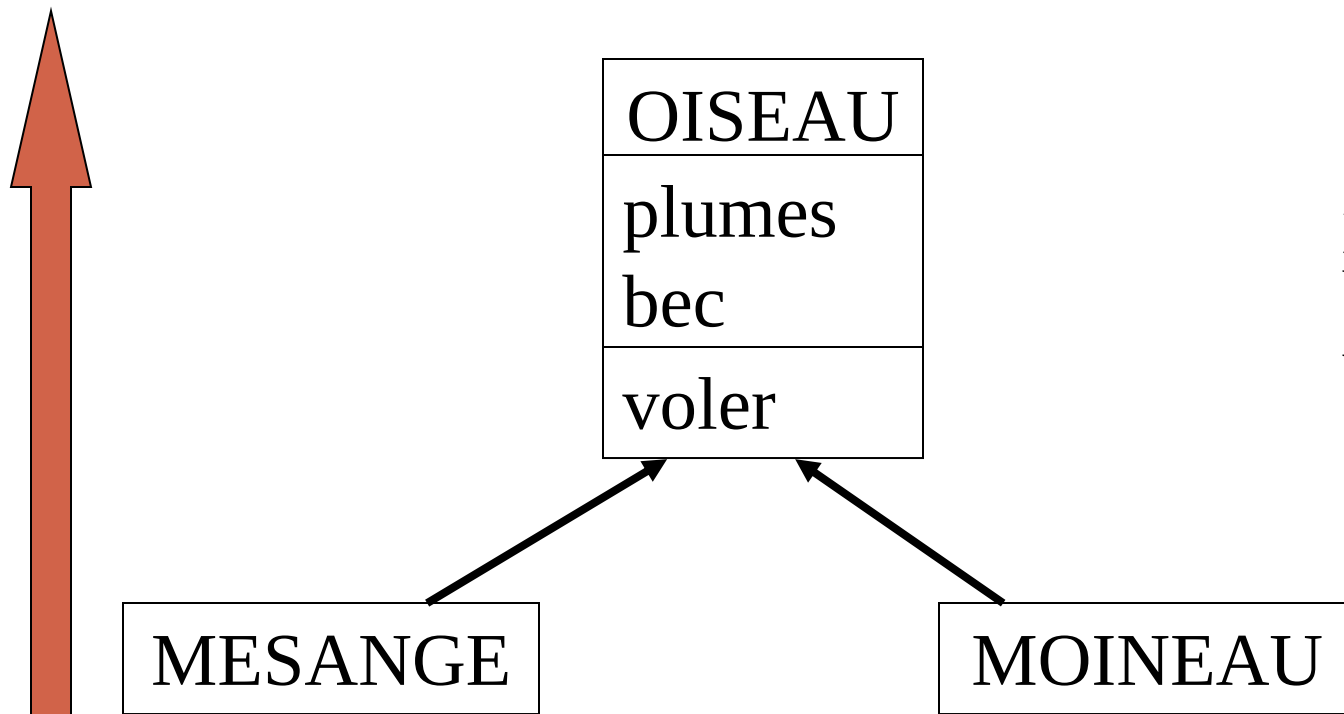


= transitive :



DIAGRAMMES DE CLASSES (8)

La classification : La généralisation - Exemple



Oiseau : classe
Moineau : classe
Mésange : classe
plumes : attribut
bec : attribut
voler : opération

Tous les attributs qui se trouvent dans la super-classe sont vrais dans toutes les sous-classes

La classification : La spécialisation

Spécialisation

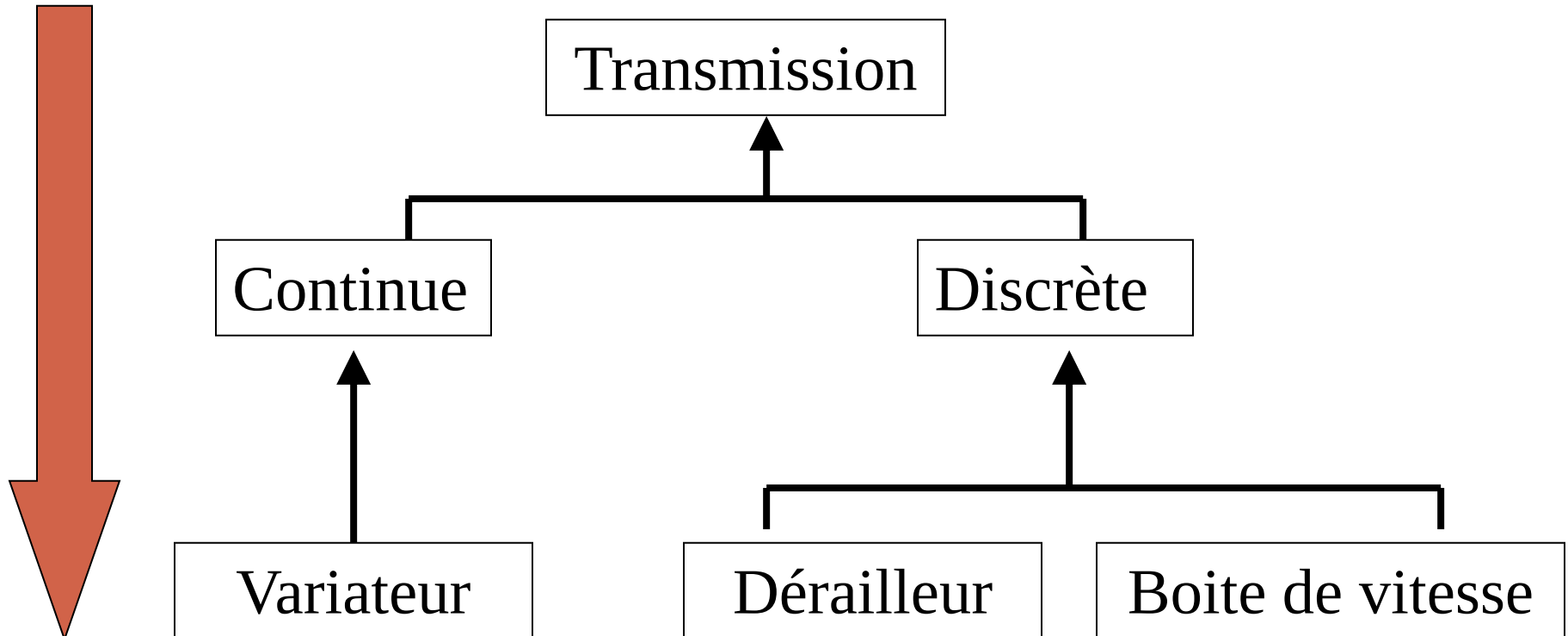
= inverse de la généralisation

= permet de capturer les particularités d'un ensemble d'objets non discriminés par les classes déjà existantes

La spécialisation est une technique très efficace pour l'extension cohérente d'un ensemble de classes

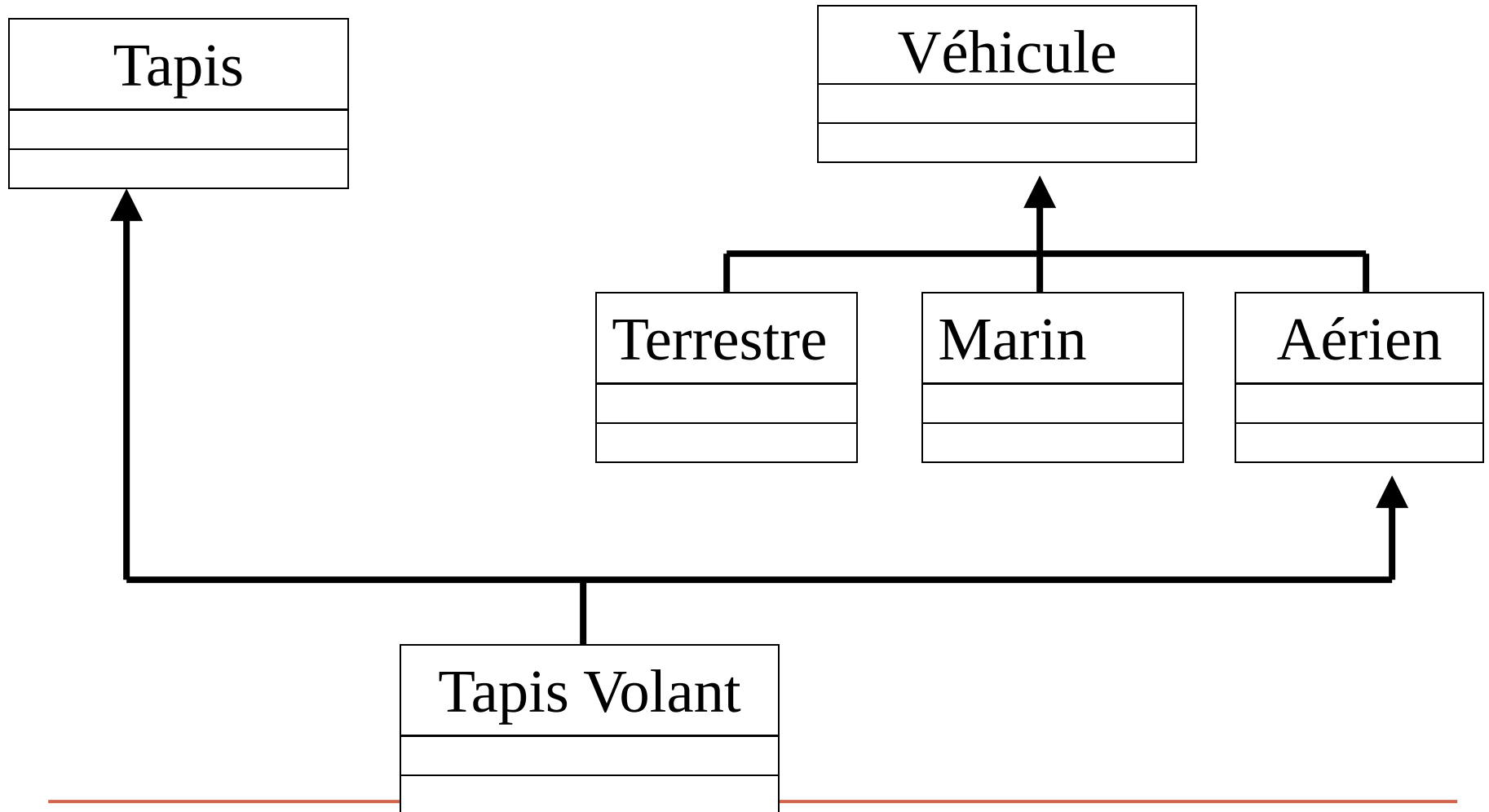
DIAGRAMMES DE CLASSES (10)

La classification : La spécialisation - Exemple



DIAGRAMMES DE CLASSES (11)

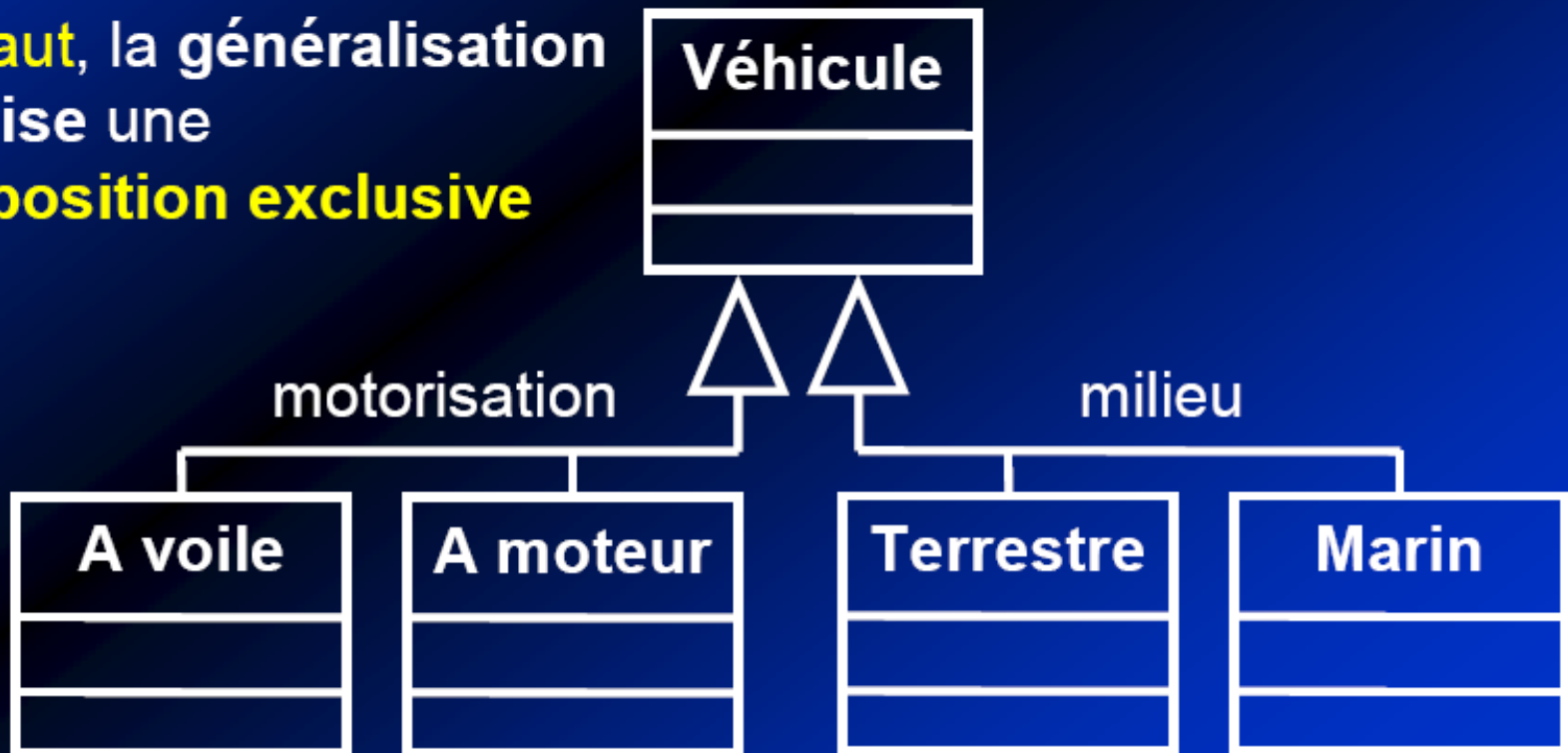
La classification : L'Héritage multiple



DIAGRAMMES DE CLASSES (12)

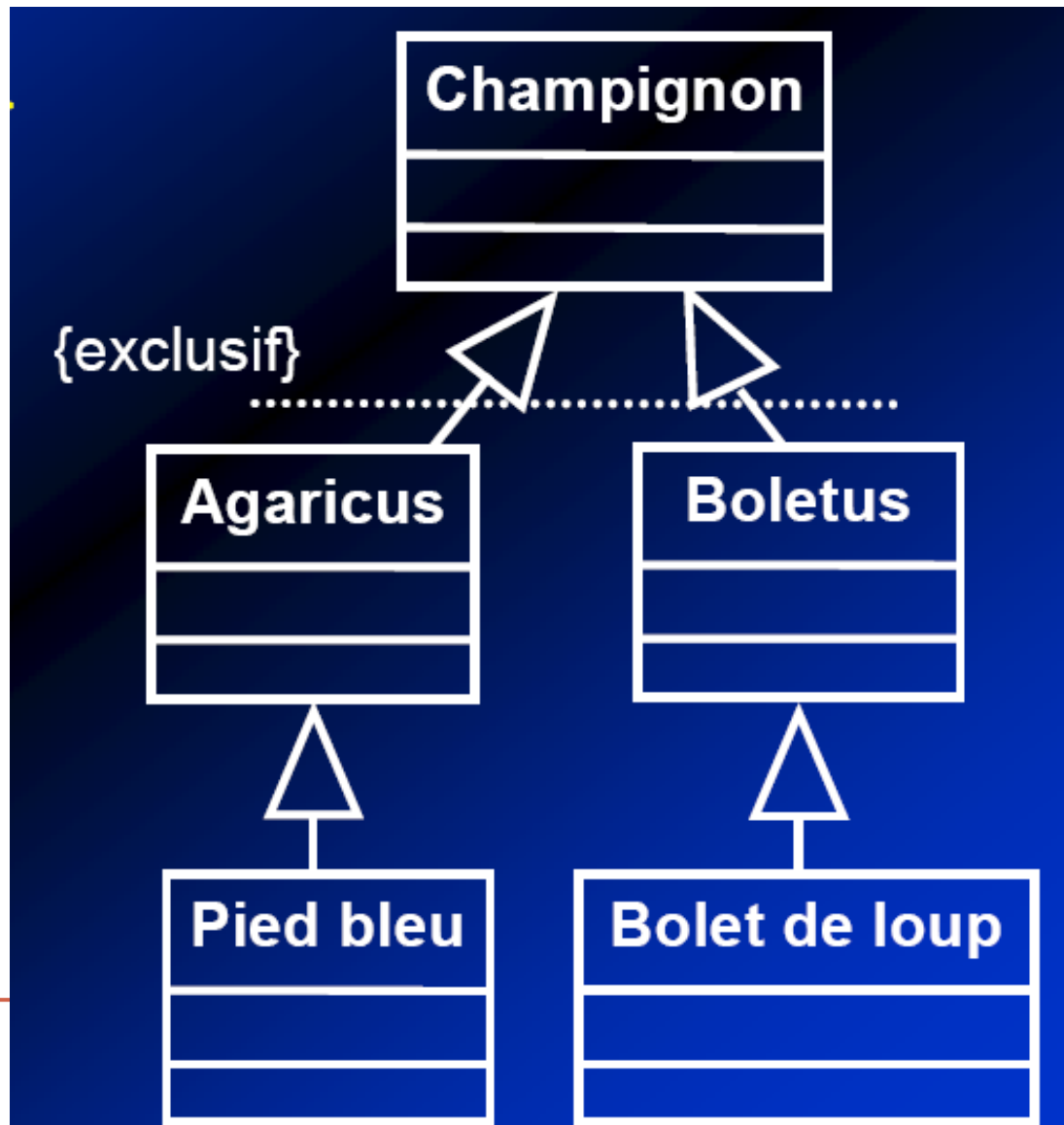
Contraintes de généralisation

- Une classe peut être **spécialisée** selon plusieurs critères
- Certaines **contraintes** peuvent être posées sur les relations de généralisation
- Par **défaut**, la généralisation symbolise une **décomposition exclusive**



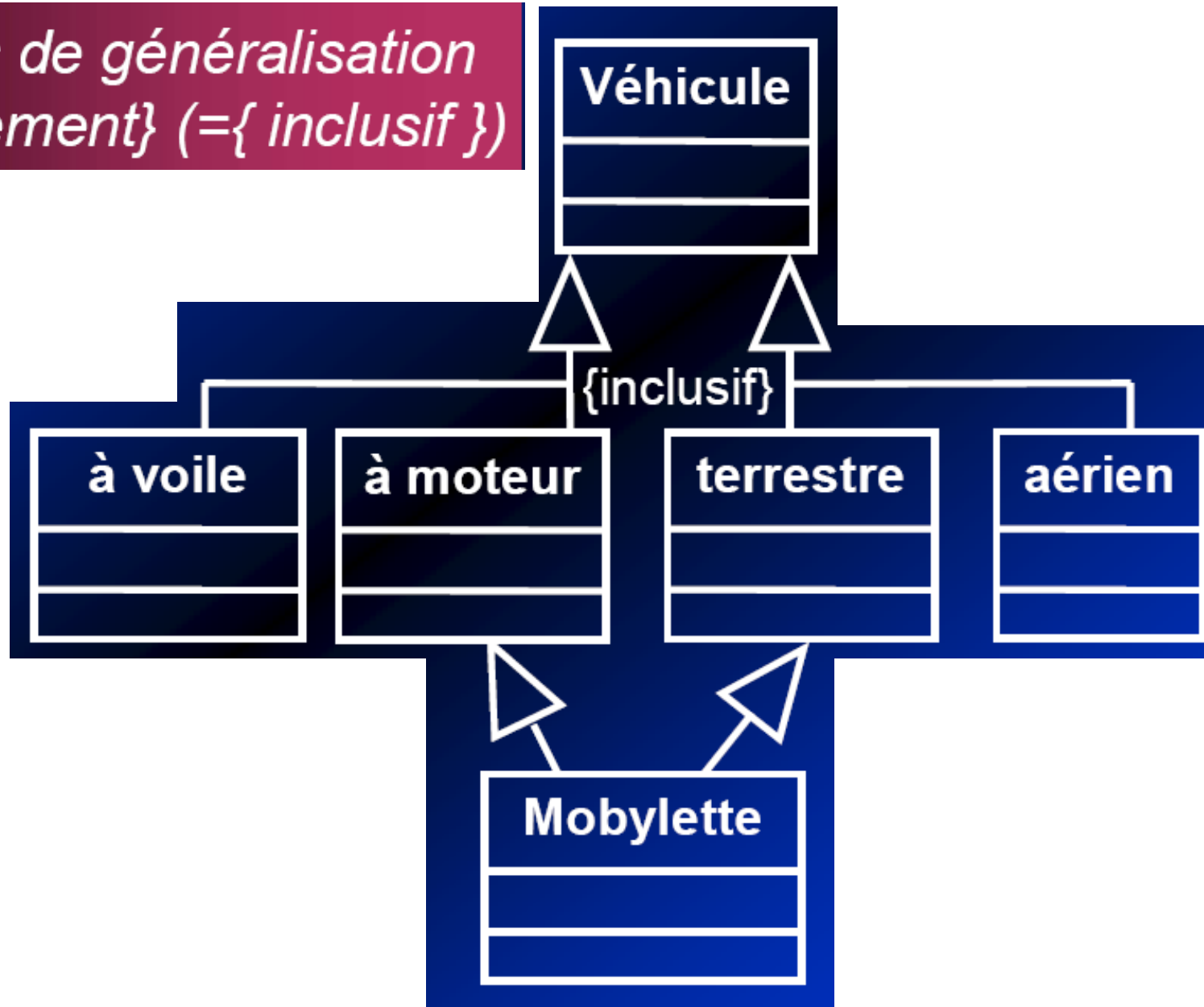
DIAGRAMMES DE CLASSES (13)

Contraintes de généralisation {disjoint} (= {exclusif})



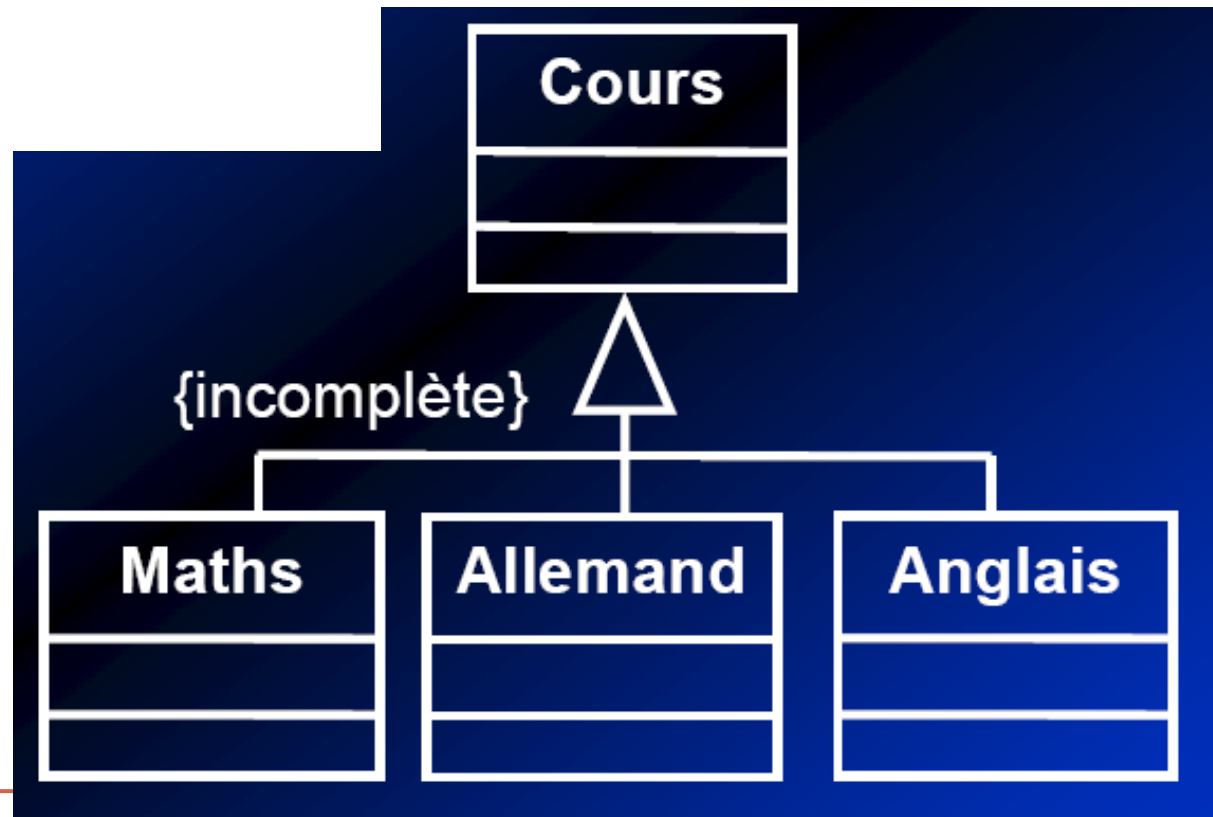
DIAGRAMMES DE CLASSES (14)

*Contraintes de généralisation
{chevauchement} (= {inclusif})*



DIAGRAMMES DE CLASSES (15)

Contraintes de généralisation
{Complète} $\neq$ {Incomplète}



DIAGRAMMES DE CLASSES (16)

Les associations

- Une association est une relation entre 2 classes (client-fournisseur)
- Nom d'association : en italique au milieu

→ forme verbale active ou passive , symbole < et >

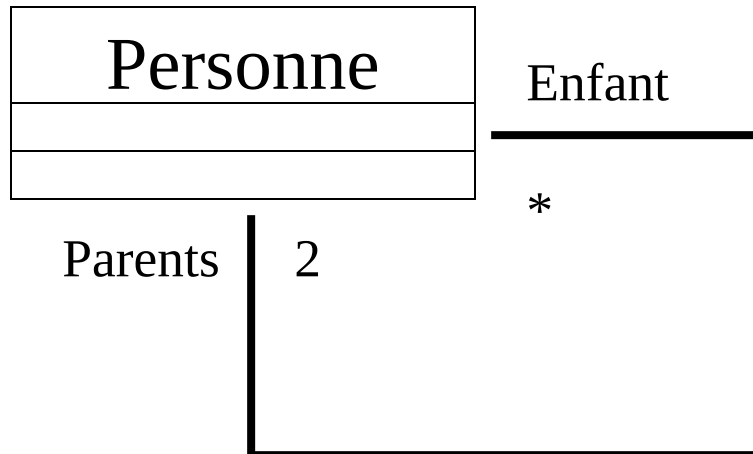


- Nom de rôle : préciser le rôle d'une classe au sein de l'association



Les associations

- Association réflexive :



DIAGRAMMES DE CLASSES (18)

Les associations

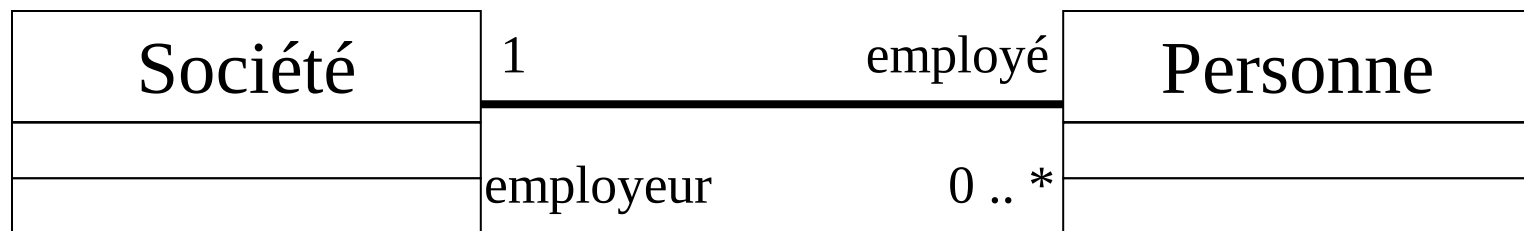
- Multiplicité (cardinalités):

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

DIAGRAMMES DE CLASSES (19)

Les associations

- Multiplicité – Exemple :
- Chaque personne travaille pour une et une seule société
- Une société emploie de zéro à plusieurs employés



Restriction des associations

- La **restriction** (dite **qualification** en UML)

- réalisée au moyen d'une clé, ensemble d'attributs particuliers

DIAGRAMMES DE CLASSES (21)

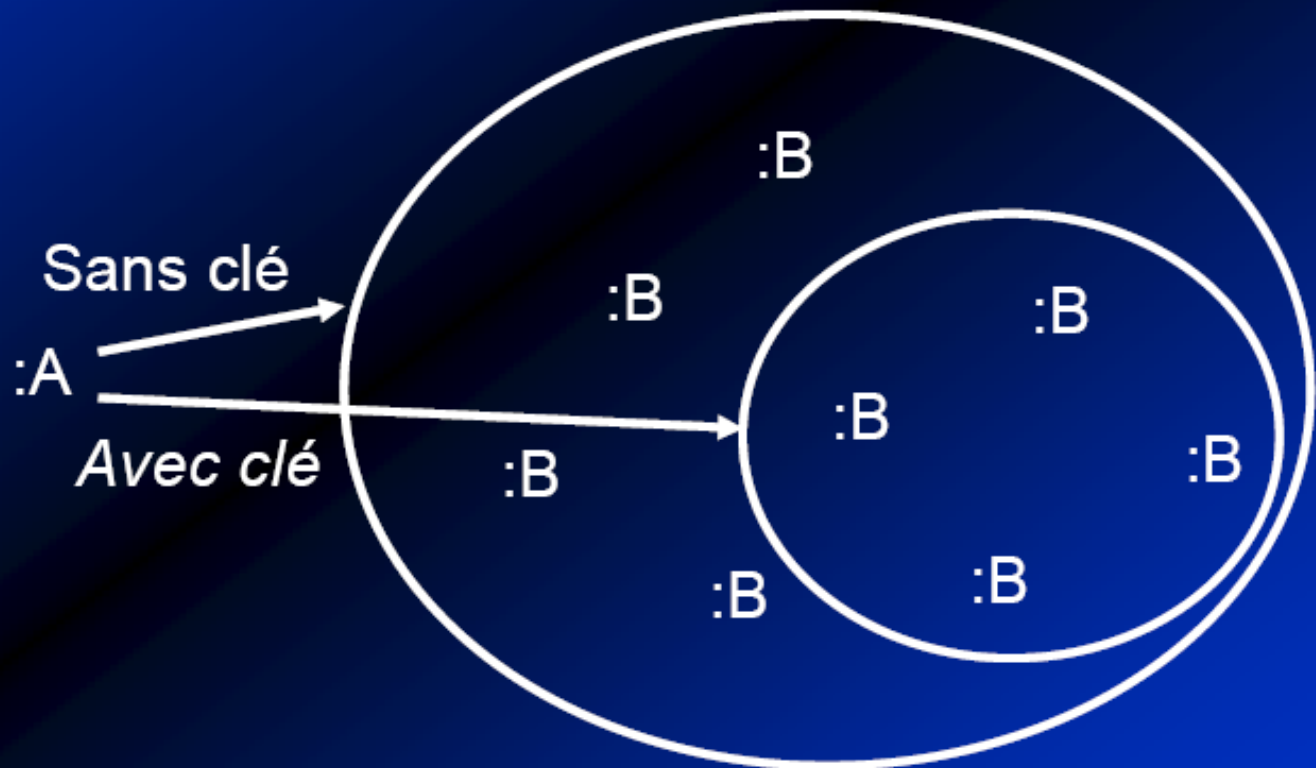
Restriction des associations



DIAGRAMMES DE CLASSES (22)

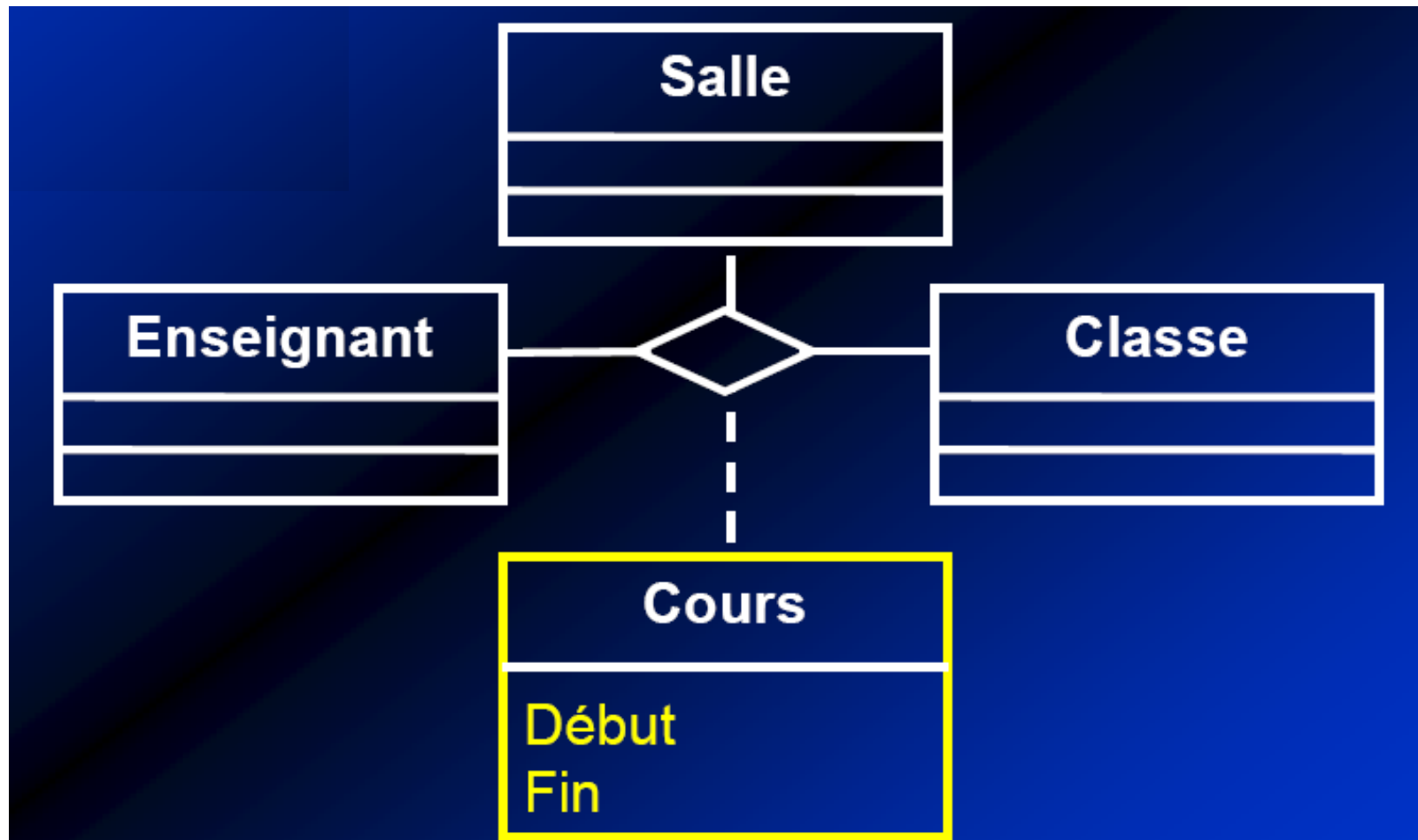
Restriction des associations

- Une **restriction** réduit le nombre d'instances qui participent à une association :



DIAGRAMMES DE CLASSES (23)

Association n-aire



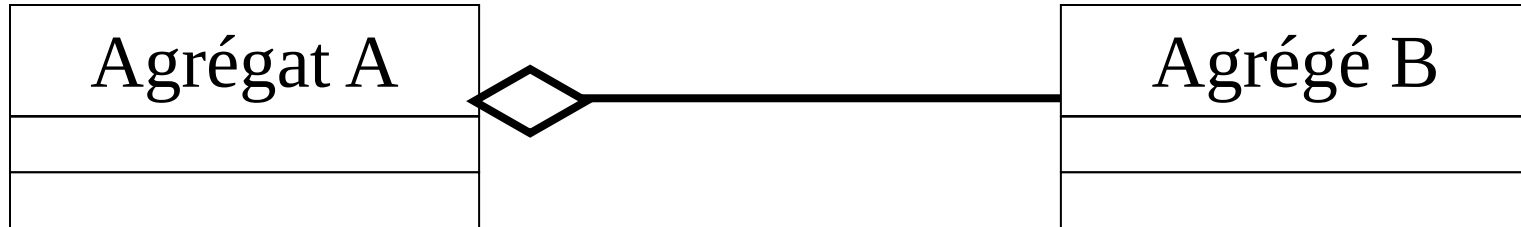
DIAGRAMMES DE CLASSES (24)

Les associations particulières : l'agrégation

Agrégation

= association non symétrique

= symbole : losange du coté de l'agregat



Critères impliquant une agrégation:

- Une classe (B) fait partie d'une autre classe (A)
- Les valeurs d'attributs d'une classe (A) se propagent dans les valeurs d'attributs d'une autre classe (B)
- Une action sur une classe (A) implique une action sur une autre classe (B)

Les associations particulières : l'agrégation - Exemple

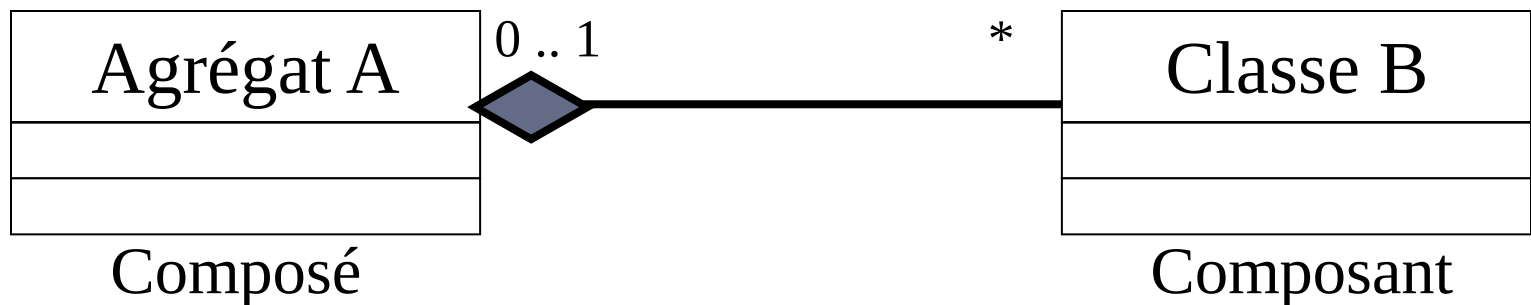


DIAGRAMMES DE CLASSES (26)

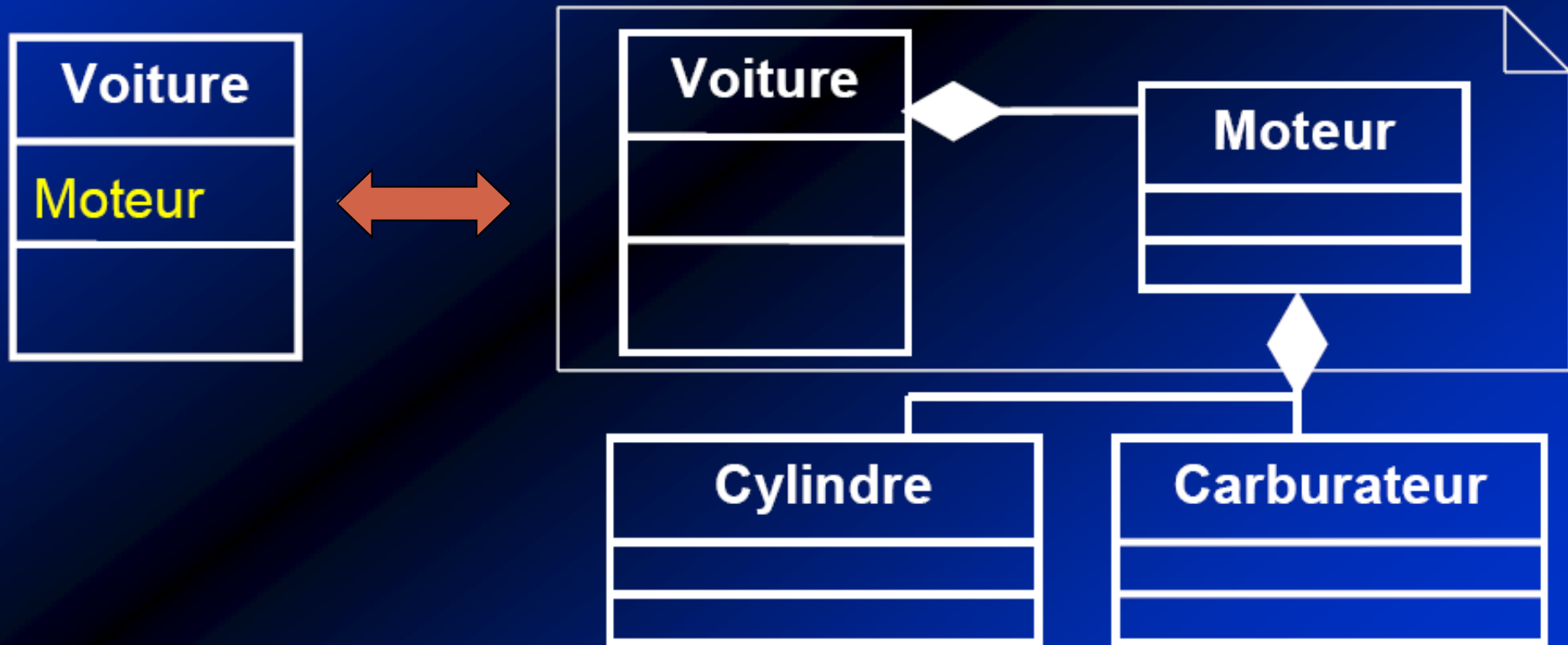
Les associations particulières : la composition

Composition

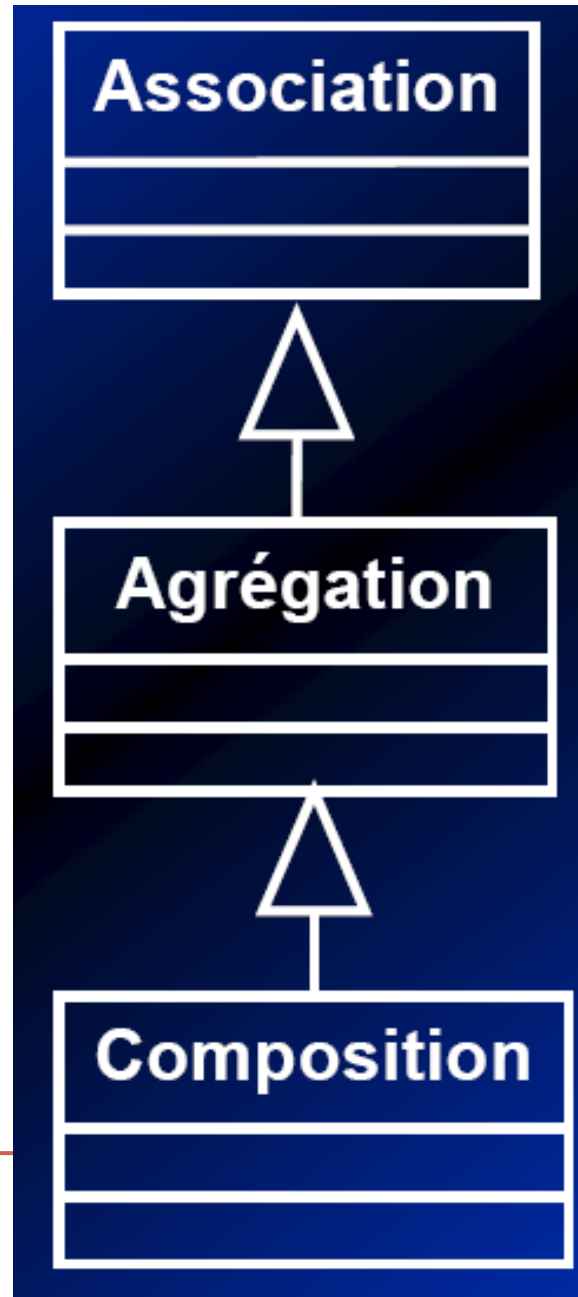
- = forme particulière d'agrégation
- = symbole : losange plein du côté de l'agrégat
- = les classes jouent les rôles de contenant et de contenu
- = implique une contrainte sur la valeur de la multiplicité du côté de l'agrégat (0 ou 1)
- = durée de vie : si ~~composé~~ alors ~~composants~~



Les associations particulières : la composition - Exemple



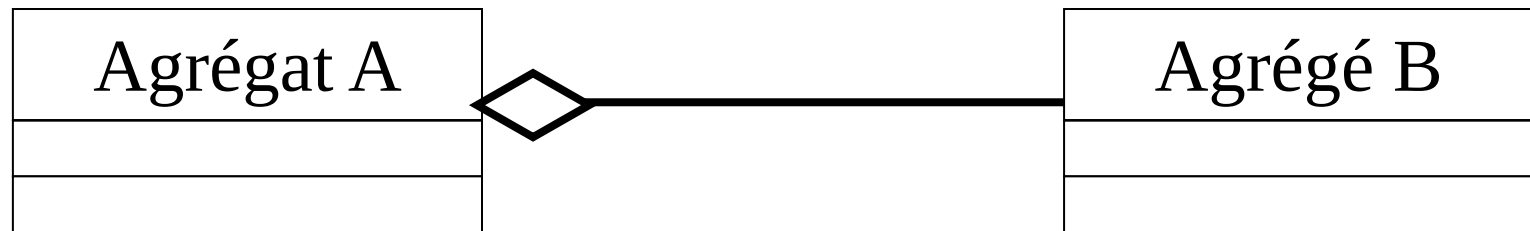
DIAGRAMMES DE CLASSES (28)



DIAGRAMMES DE CLASSES (29)

■ UML et JAVA : Agrégation – Composition

■ Une agrégation va en général s'implémenter sous forme d'attribut. Pour deux classes A et B, avec une agrégation du côté de A et une multiplicité 1 du côté de B.



■ Exemple en JAVA :

```
class A {  
    private B objB;  
    ...  
}
```

DIAGRAMMES DE CLASSES (30)

Exemple de
production en JAVA
:

Personne

- nom : chaîne	
- prénom : chaîne	-
dateNaissance : date	
+ getNom()	
+ getDateNaissance()	
+ setDateNaissance(dn : Date)	

```
public class Personne
{
    private String nom;
    private String prenom;
    private Date dateNaissance;

    public Personne ()
    {
        // constructeur
    }

    public Date getDateNaissance()
    {
        return dateNaissance;
    }

    public void setDateNaissance (Date de)
    {
        dateNaissance = de;
    }
}
```

Conseil pratique

- Bien comprendre le problème à résoudre, la qualité du modèle en dépendra directement
- Avoir une liste précise des informations manipulées avec leur signification sémantique exacte afin de déterminer les classes, les associations et les multiplicités
- Éviter les associations n-aires avec $n > 2$
- Ne pas surcharger le modèle global qui doit rester clair et lisible, si telle ou telle partie doit être plus détaillée (spécialisation,...), faire un paquetage que l'on détaillera (*UML 2.0*)

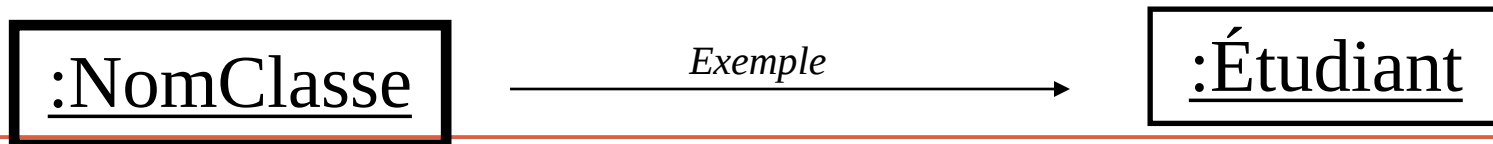
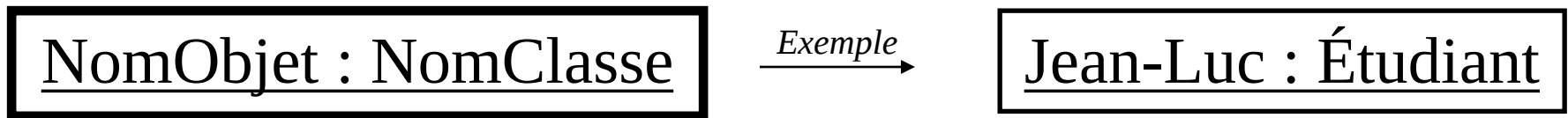
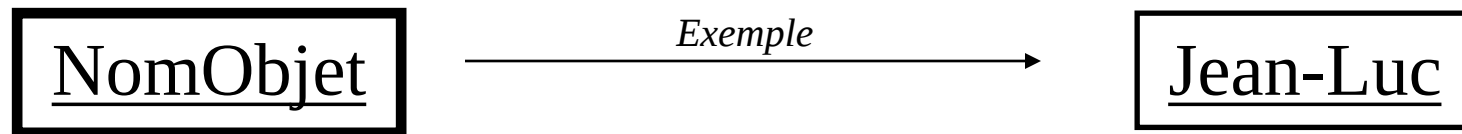
DIAGRAMMES D'OBJETS

DIAGRAMMES D'OBJETS (1)

(structure statique du système en terme d'objets)

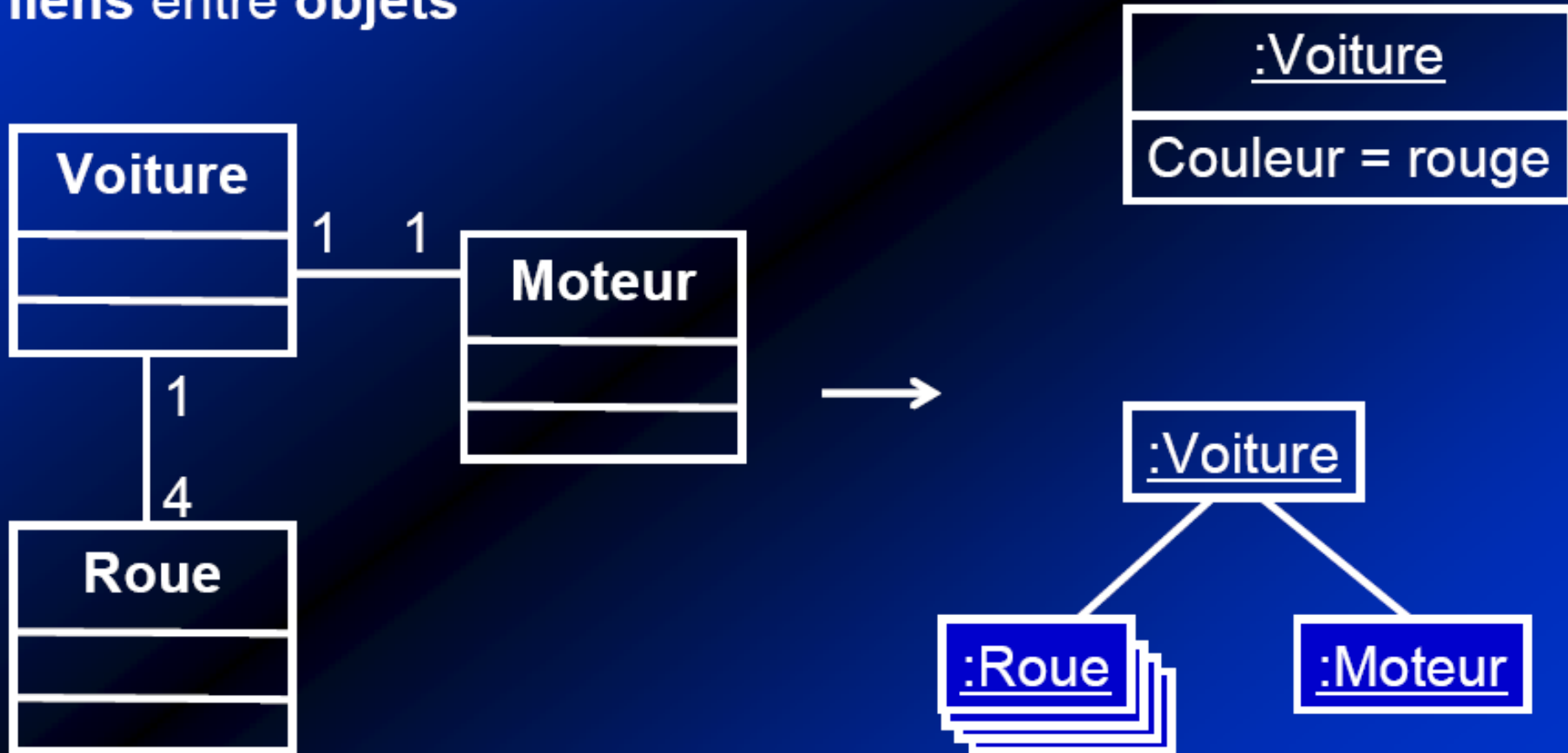
- Objet = sens précis dans le contexte du problème étudié

*Un objet est représenté sous la forme d'un **rectangle**.*



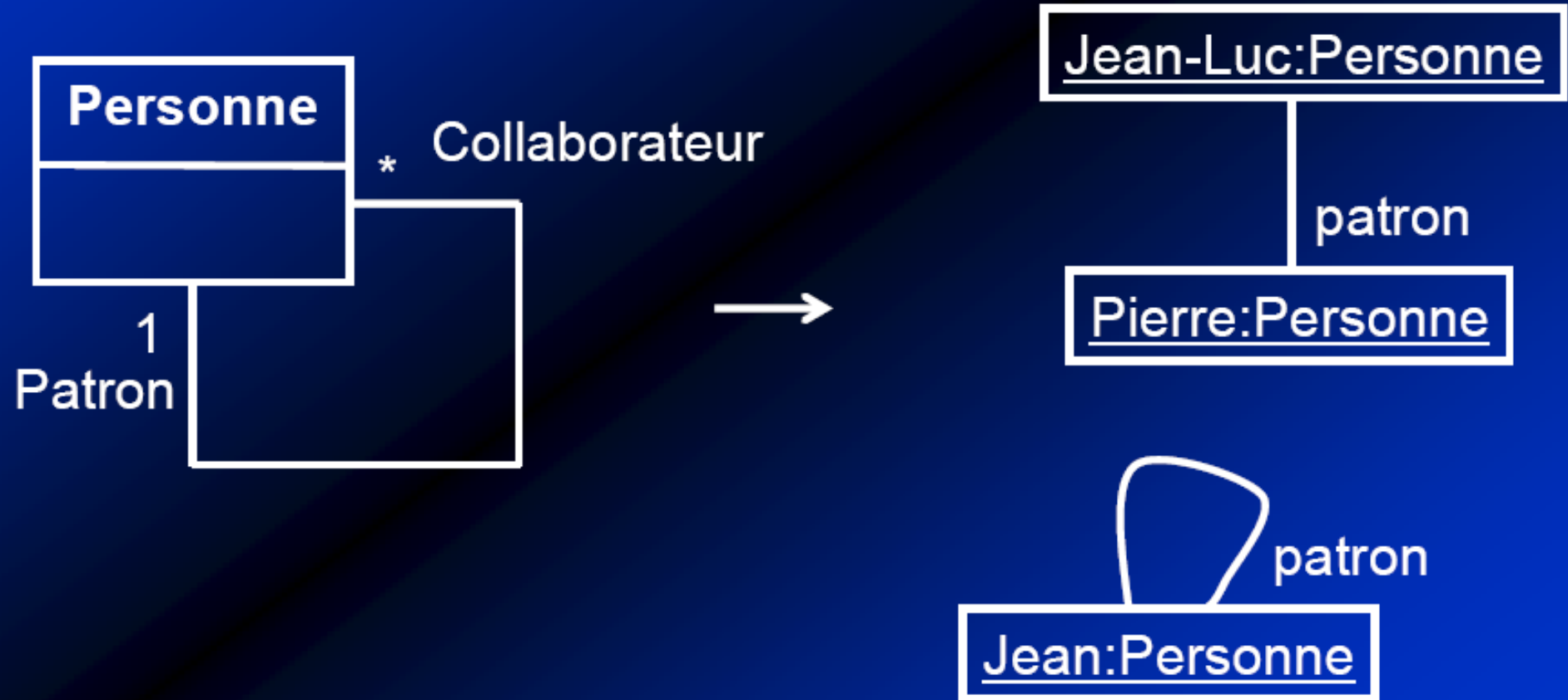
DIAGRAMMES D'OBJETS (2)

- les valeurs des attributs sont optionnelles ainsi que les liens entre objets



DIAGRAMMES D'OBJETS (3)

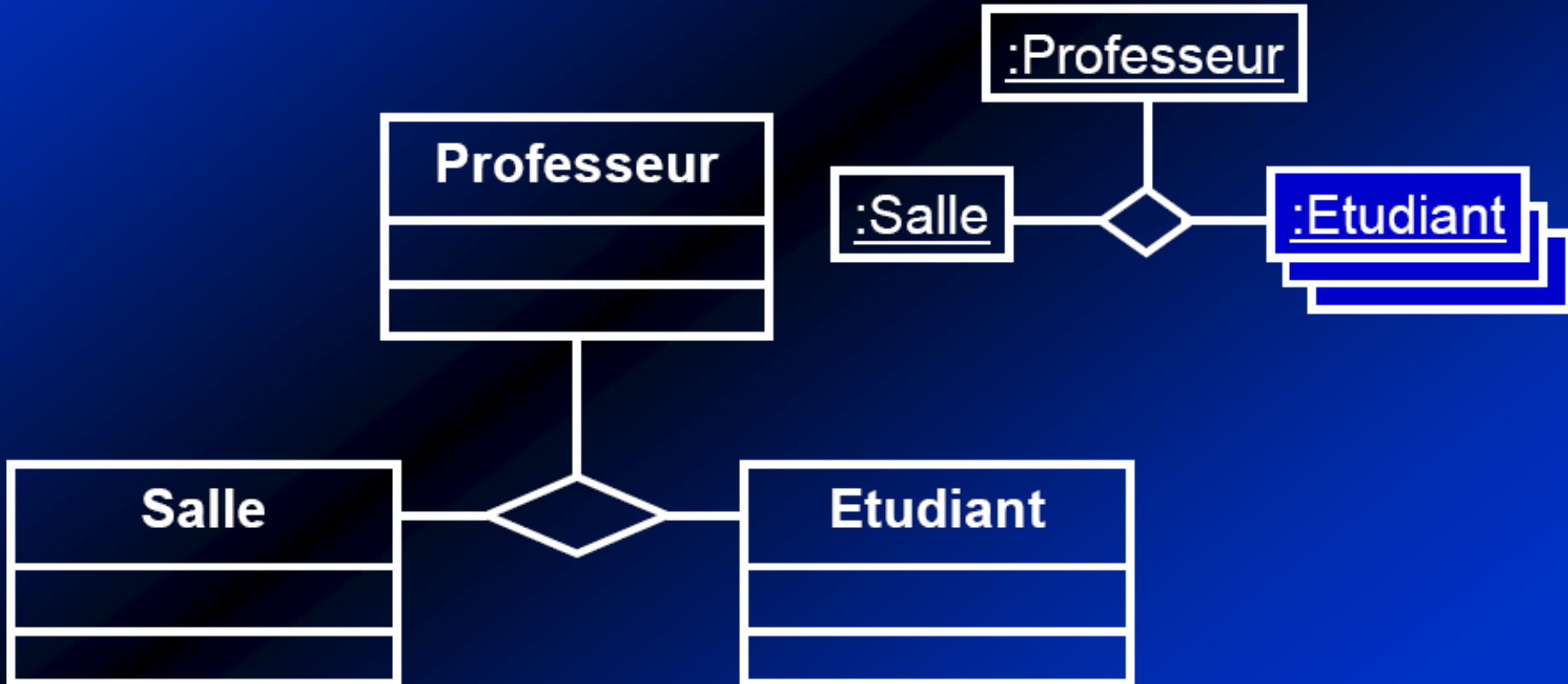
- Les liens instances des **associations réflexives** peuvent **relier** un objet à **lui même**



DIAGRAMMES D'OBJETS (4)

Diagramme d'objets : liens entre objets

- Les liens d'arité supérieure à 2 ou la multiplicité peuvent être représentés :



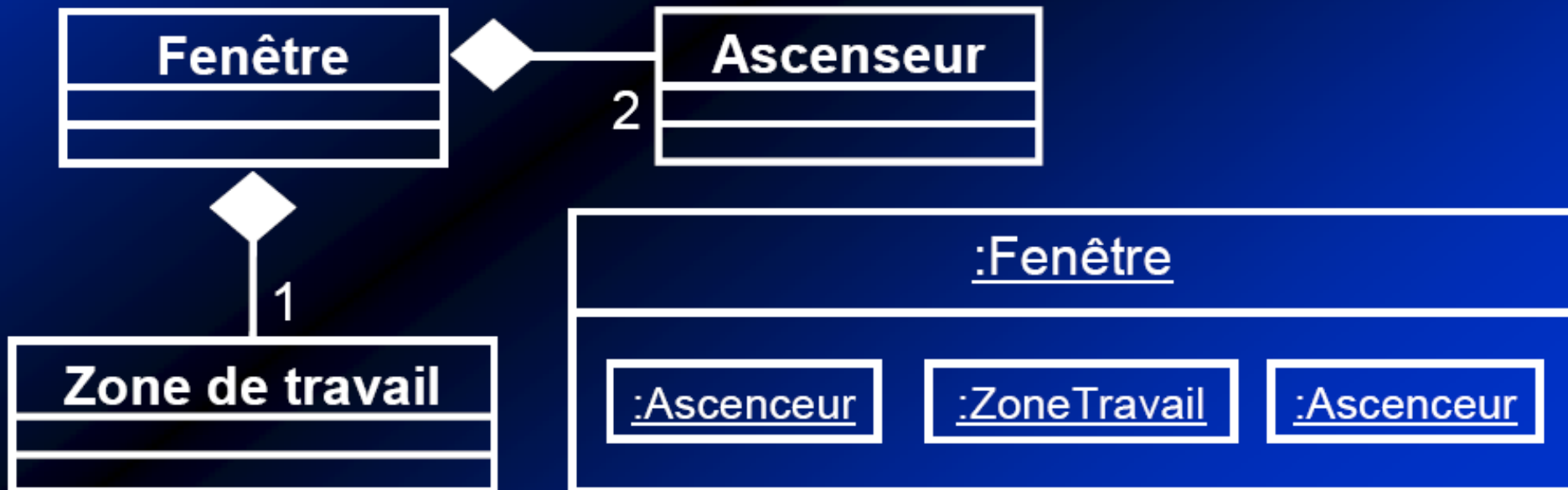
DIAGRAMMES D'OBJETS (5)

Diagramme d'objets : liens entre objets

- Les objets composés de sous-objets peuvent être visualisés :



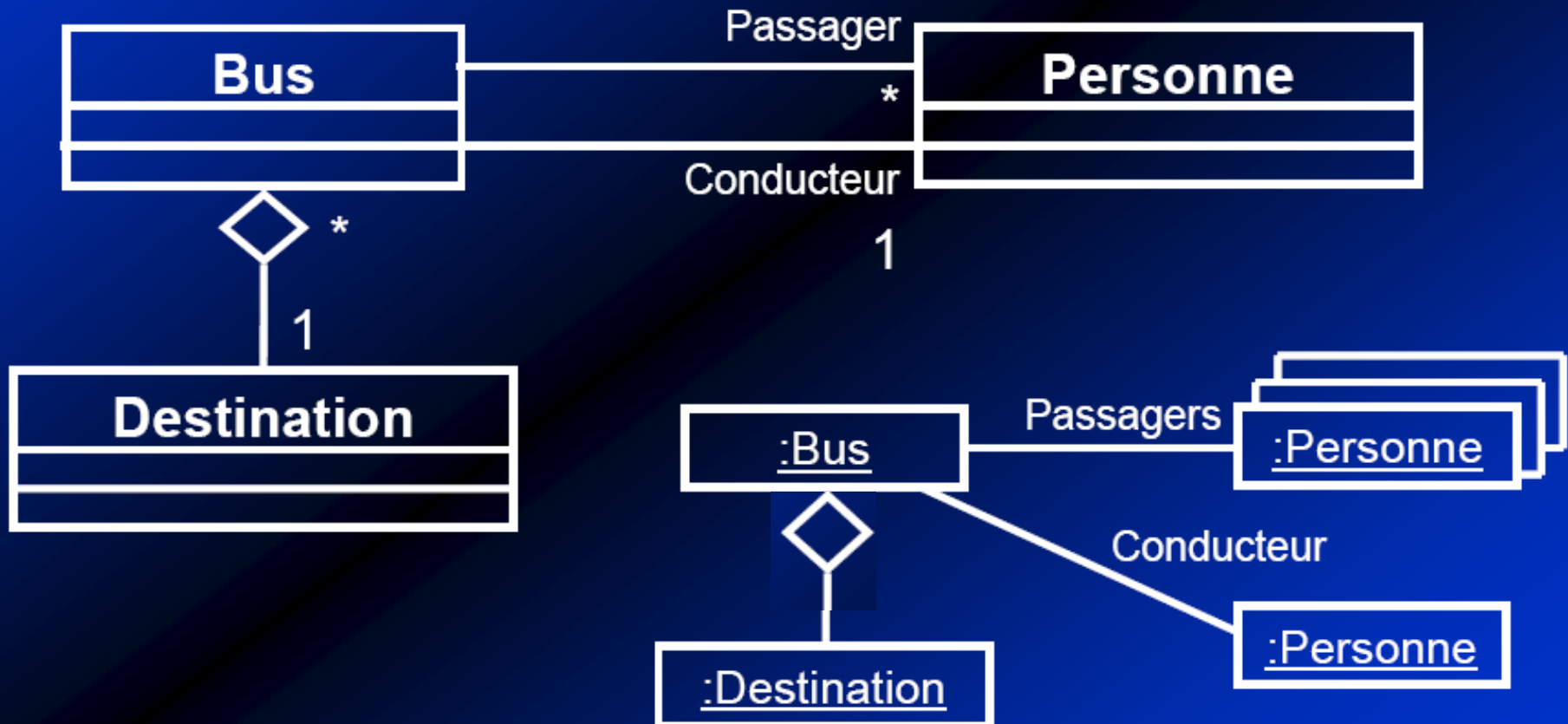
- Les objets **composites** sont instances de classes **composites** :



DIAGRAMMES D'OBJETS (6)

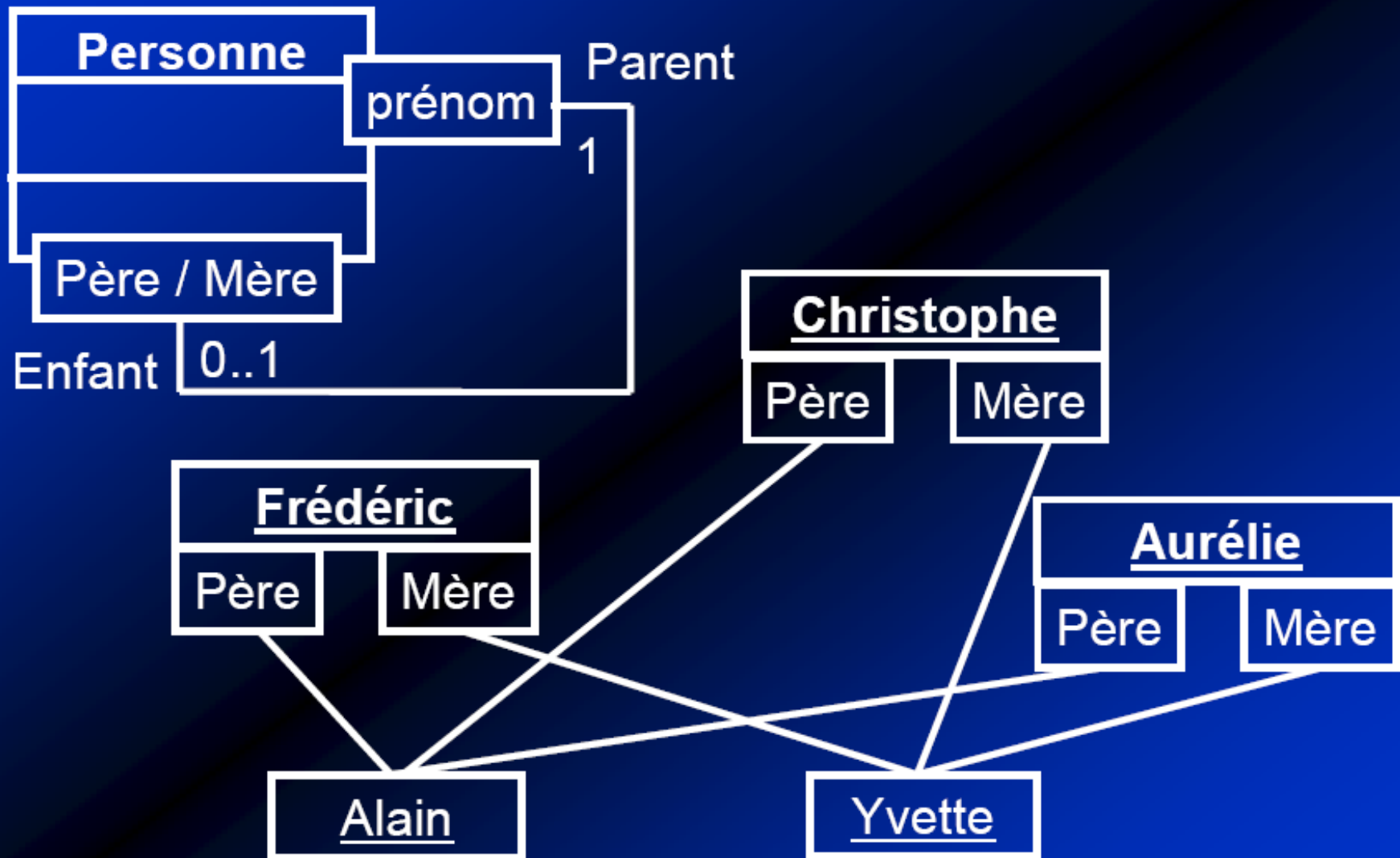
Diagramme d'objets : structures complexes

- Les diagrammes d'objets facilitent la compréhension et l'élaboration d'un diagramme de classes :



DIAGRAMMES D'OBJETS (7)

Diagramme d'objets : structures complexes



DIAGRAMMES DE COLLABORATION ou DE COMMUNICATION (UML 2.0)

DIAGRAMMES DE COLLABORATION (1)

(collaboration entre les objets d'un système pour réaliser une action)

Introduction

Diagramme de collaboration (d'objets) : extension des diagrammes d'objets : vue *dynamique*

- Décrit le **comportement collectif** d'un **ensemble d'objets**,
- en vue de **réaliser** une **opération**
- en décrivant leurs **interactions** modélisées par des **envois** (éventuellement **numérotés**) de **messages**

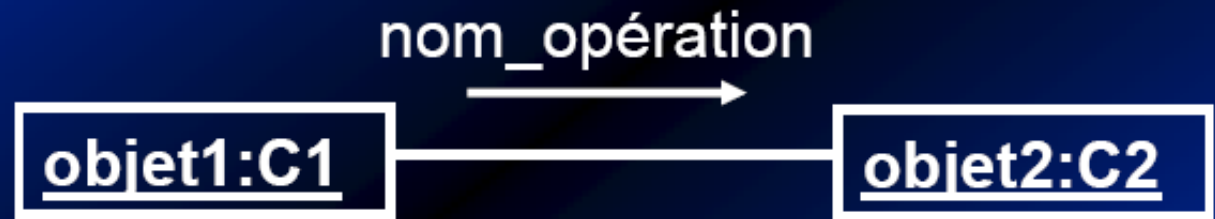
DIAGRAMMES DE COLLABORATION (2)

Envois de messages entre objets

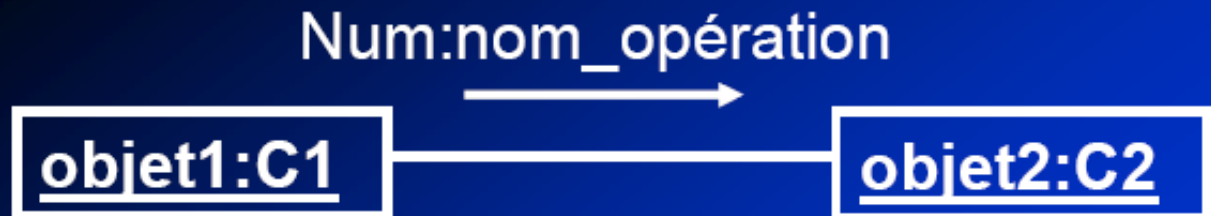
Message = *nom opération*



A white arrow pointing to the right, representing a message.



Envois éventuellement numérotés :
ordre des envois de message au cours d'une opération

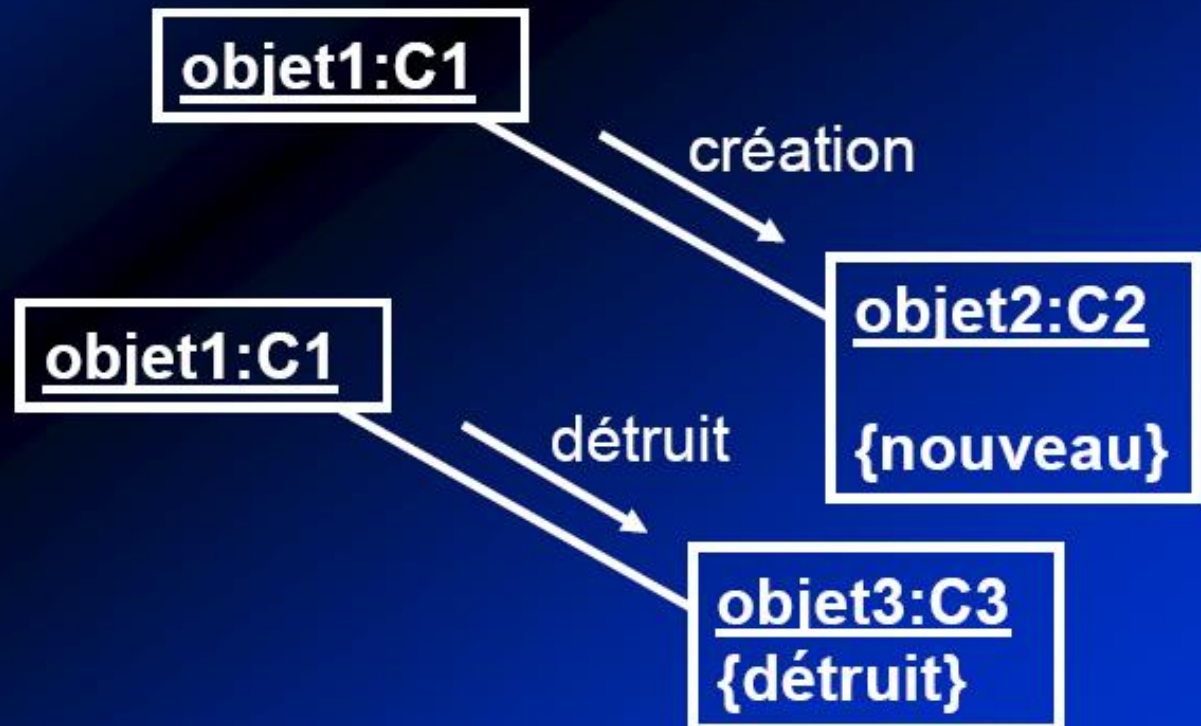


DIAGRAMMES DE COLLABORATION (3)

Contraintes associées aux envois de message

Les objets (et les liens) **créés** ou **détruits** au cours d'une interaction peuvent **respectivement** porter les contraintes :

- {Nouveau}
- {Détruit}

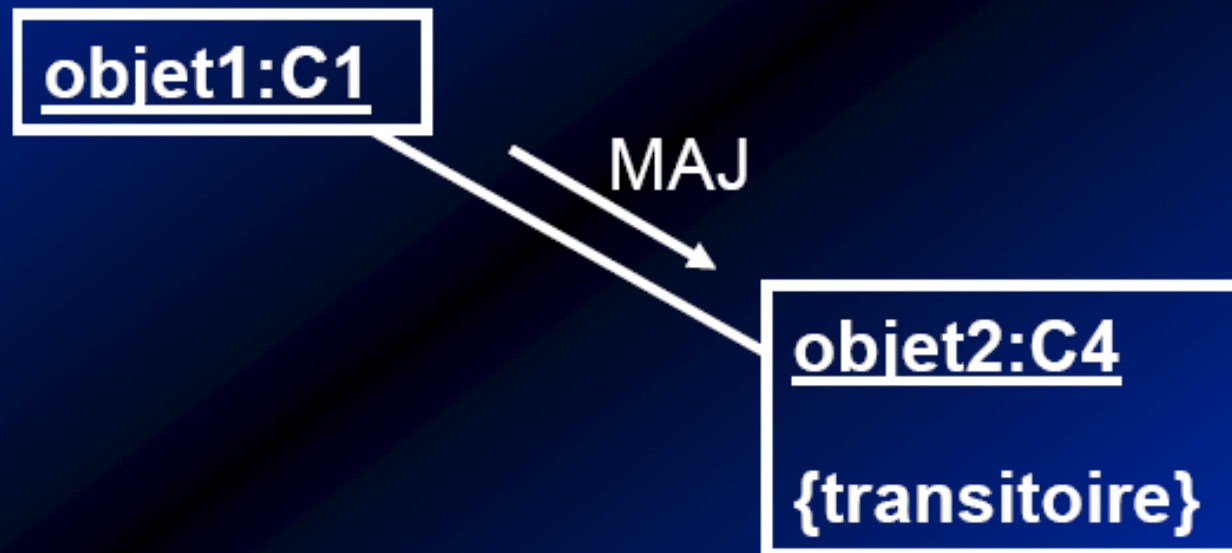


DIAGRAMMES DE COLLABORATION (4)

Contraintes associés aux envois de message

Les objets (et les liens) **créés** et **détruit** au cours de la même interaction porte la contraintes :

- {transitoire}



DIAGRAMMES DE COLLABORATION (5)

Itérations dans un diagramme de collaboration

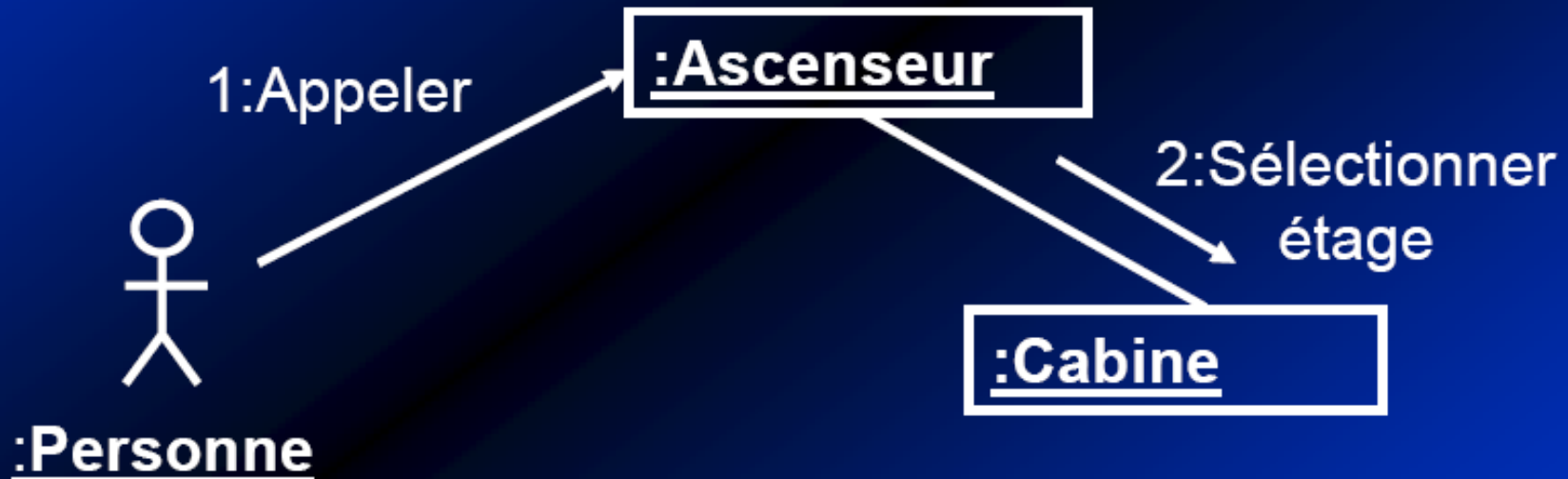
Envois répétitifs :



DIAGRAMMES DE COLLABORATION (6)

Itérations dans un diagramme de collaboration

Intervention d'acteurs :



DIAGRAMMES DE COLLABORATION (7)

Itérations dans un diagramme de collaboration

Objet actif :

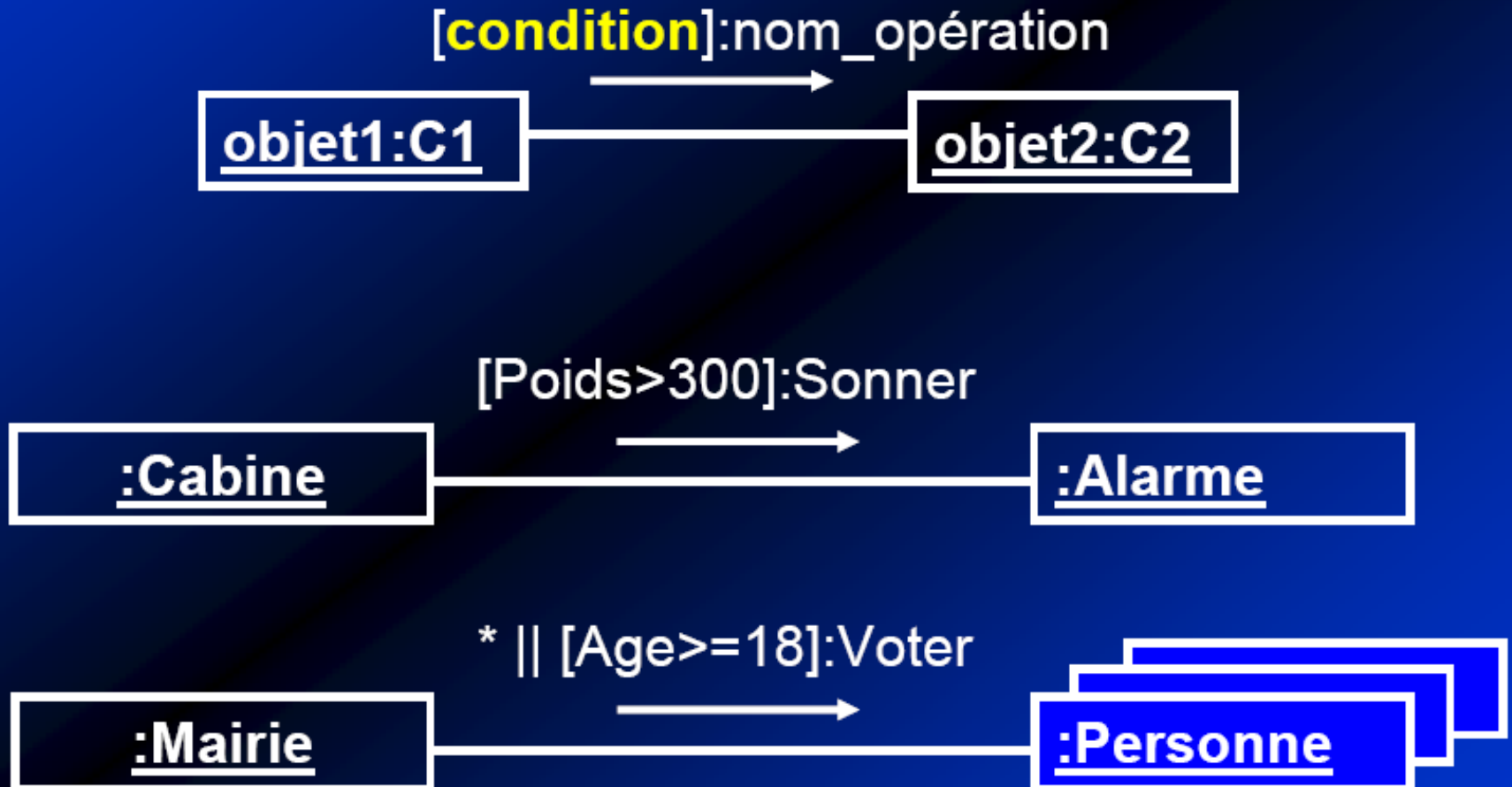
Ex : photocopieuse



DIAGRAMMES DE COLLABORATION (8)

Conditions sur les envoi de message

L'envoi d'un message peut être assorti d'une condition



DIAGRAMMES DE COLLABORATION (9)

Retour d'une liste de valeurs à l'issue d'un envoi de message

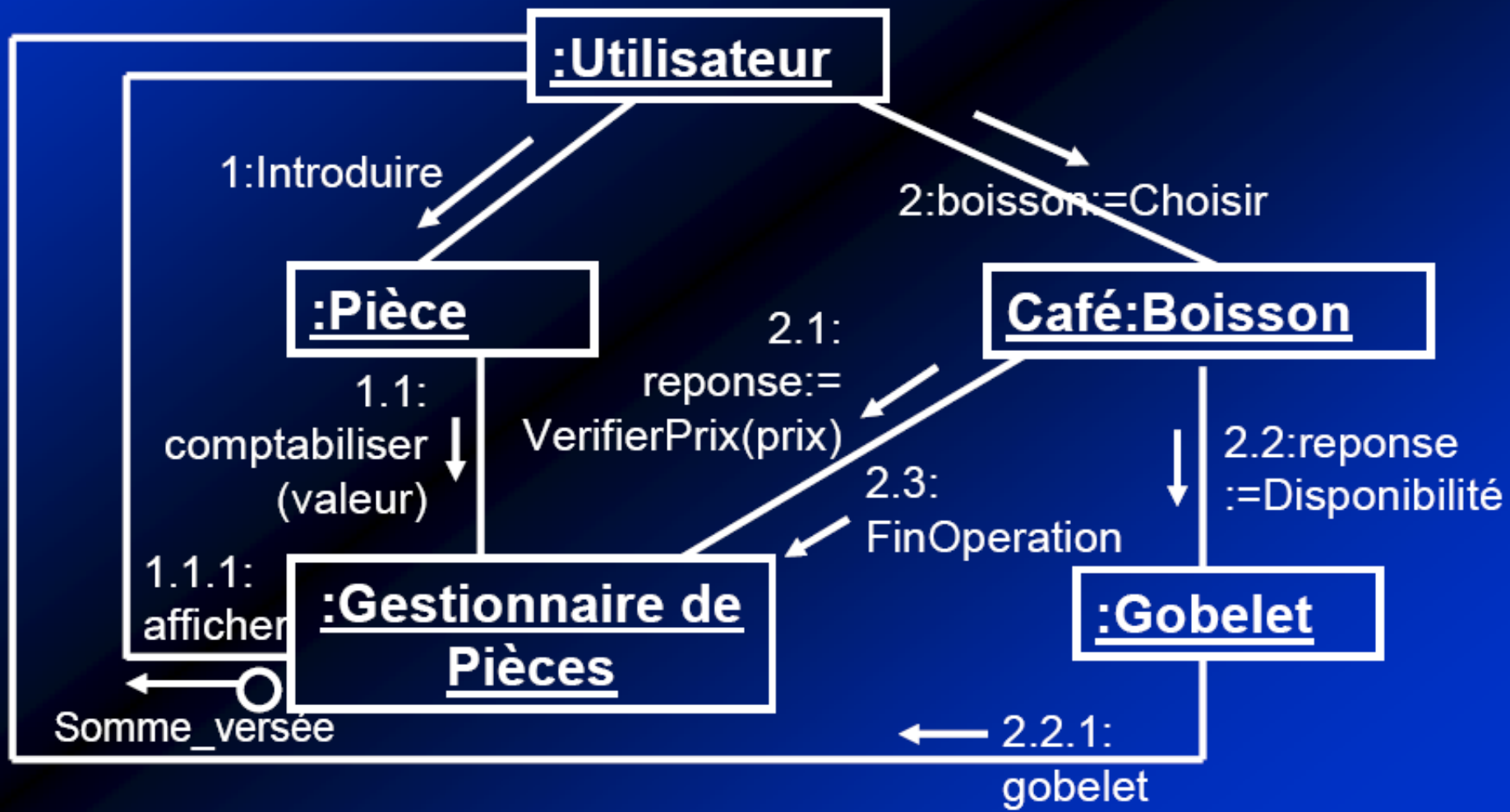
Une liste de valeurs peut être retournée suite à l'envoi d'un message



DIAGRAMMES DE COLLABORATION (10)

Exemple :

Diagramme de collaboration « **demande d'une boisson disponible (café) avec introduction de la somme exacte** »



DIAGRAMMES D'ÉTATS-TRANSITIONS

DIAGRAMMES D'ÉTATS-TRANSITIONS (1)

(les $\neq$ états d'un objet d'une classe)

Diagramme d'états-transitions

- décrit le **comportement** des objets d'une classe au moyen d'un automate d'états associé à la classe
- Le comportement est modélisé par un graphe :
 - **Nœuds** = états possibles des objets
 - **Arcs** = transitions d'état à état.
- Une transition :
 - = **exécution d'une action**
 - = **réaction** de l'objet sous l'effet d'une occurrence d'evt

DIAGRAMMES D'ÉTATS-TRANSITIONS (2)

Notion d'état

- un état = étape dans le cycle de vie d'un objet

- chaque objets possède à un instant donné un état particulier
- chaque état est identifié par un nom
- un état est stable et durable

DIAGRAMMES D'ÉTATS-TRANSITIONS (3)

Notion d'état

- Chaque diagramme d'états-transitions comprend **un état initial**.
- Il est possible de n'avoir **aucun état final** : ex : un système qui ne s'arrête jamais.



Etat initial



Etat
intermédiaire



Etat final

DIAGRAMMES D'ÉTATS-TRANSITIONS (4)

Notion de transition

- Les états sont reliés par des connexions unidirectionnelles appelées transitions



Ex : classe « Place de parking »



DIAGRAMMES D'ÉTATS-TRANSITIONS (5)

Notion d'événement

- un événement correspond à l'**occurrence** d'une **situation** donnée dans le domaine étudié
- un événement est une **information instantanée** qui doit être traitée dans l'instant où il se produit



DIAGRAMMES D'ÉTATS-TRANSITIONS (6)

Notion d'événement

- syntaxe d'un événement :

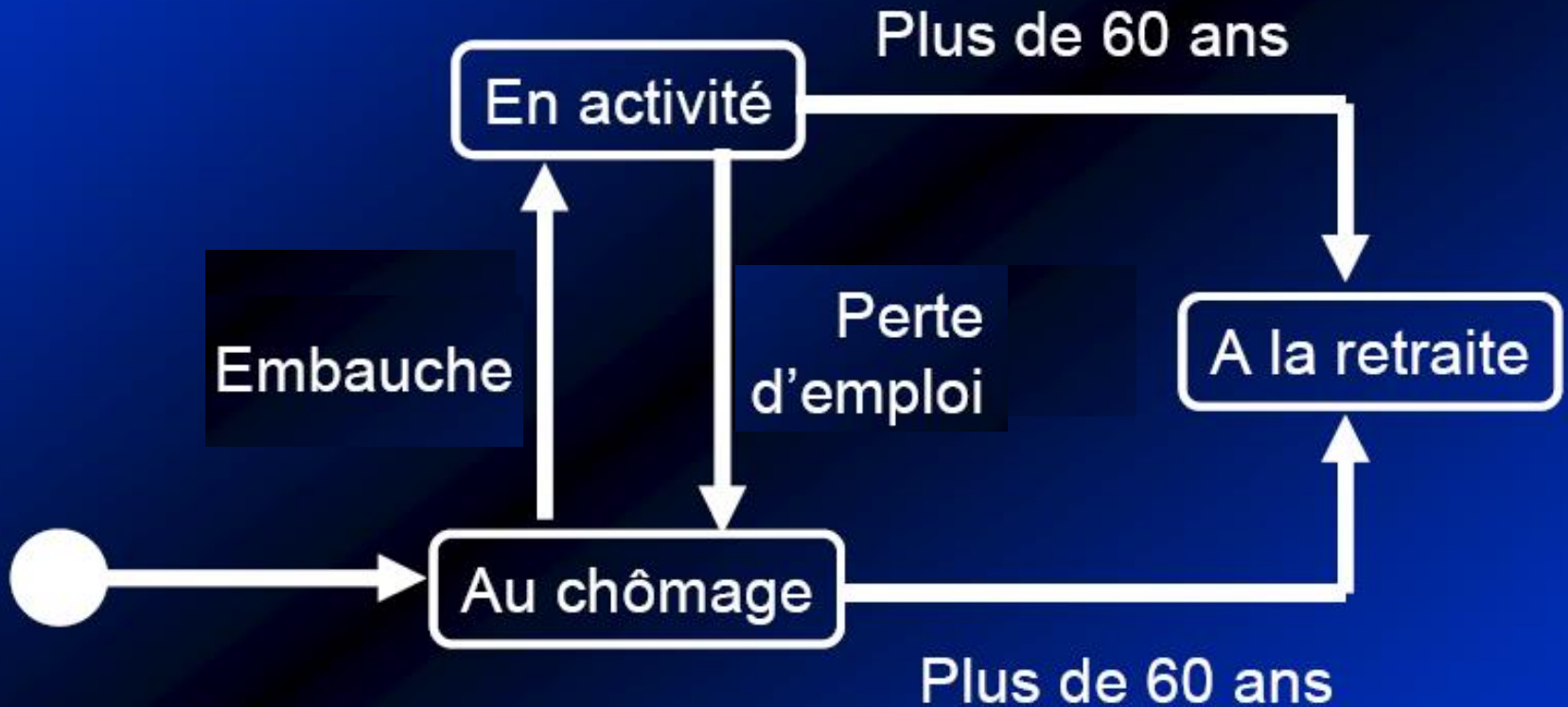
Nom de l'événement (Nom de paramètre : Type, ...)

La description complète d'un évt est donnée par :

- nom de l'événement
- liste des paramètres
- objet expéditeur
- objet destinataire
- sa description textuelle

DIAGRAMMES D'ÉTATS-TRANSITIONS (7)

Notion d'événement



DIAGRAMMES D'ÉTATS-TRANSITIONS (8)

Communication entre objets par événements

- La communication est de type **asynchrone, atomique et unidirectionnelle**.



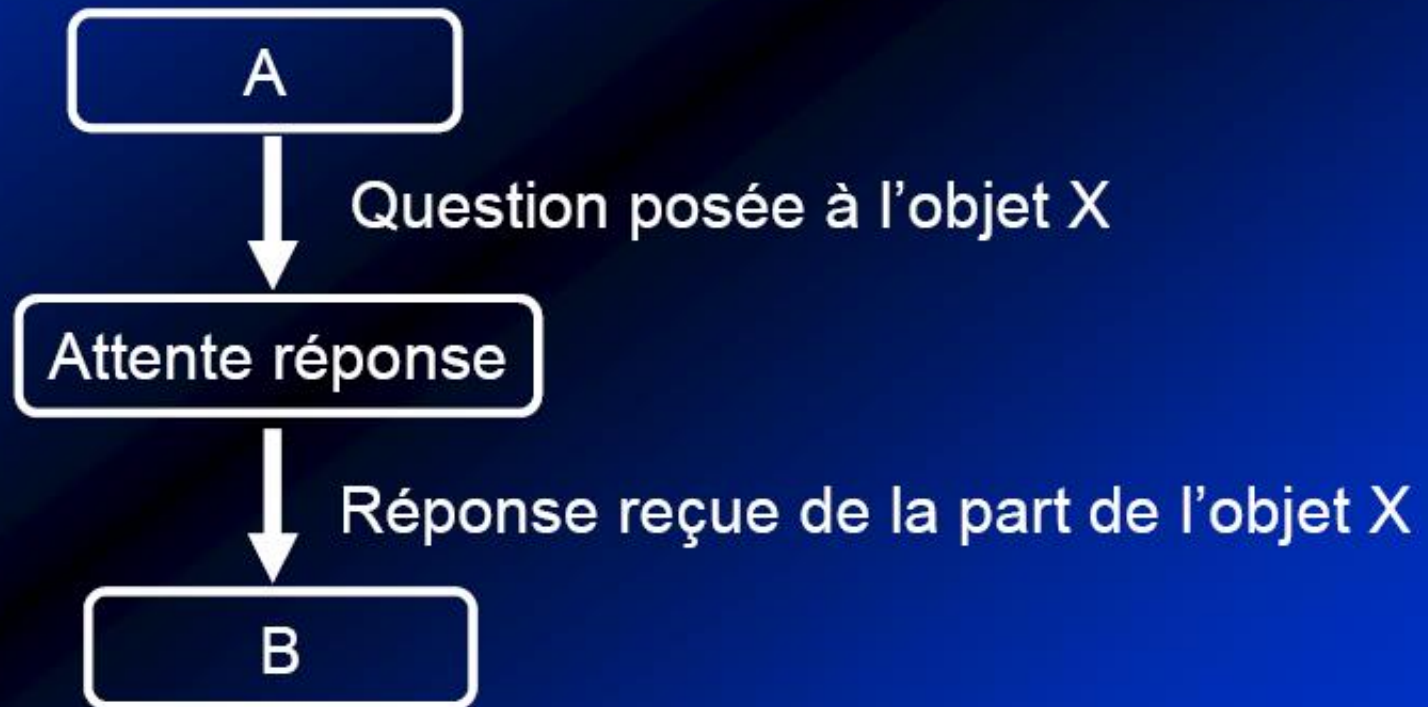
- Les **besoins** de communication par événements **synchrones**



DIAGRAMMES D'ÉTATS-TRANSITIONS (9)

Communication entre objets par événements

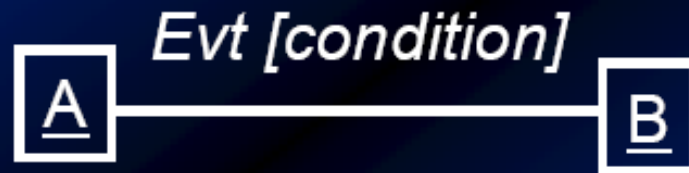
- L'objet émetteur de la requête se met en attente de la réponse de l'objet récepteur de la requête



DIAGRAMMES D'ÉTATS-TRANSITIONS (10)

Notion de garde

- Une garde est une condition booléenne qui permet ou non le déclenchement d'une transition lors de l'occurrence d'un événement



DIAGRAMMES D'ÉTATS-TRANSITIONS (11)

Communication entre objets par événements



DIAGRAMMES D'ÉTATS-TRANSITIONS (12)

Notion d'opération et d'action

- **Action et activités** = le lien entre les opérations définies dans la **spécification** d'une classe et les événements apparaissant dans le diagramme d'états-transitions
- Chaque transition peut avoir une **action** à **exécuter** lorsqu'elle est déclenchée
- L'action est considérée comme **instantanée** et **atomique**
- Une action correspond à l'**exécution** d'une des opérations déclarées dans la classe de l'objet destinataire de l'événement.



- L'action a accès aux paramètres de l'événement ainsi qu'aux attributs de l'objet sur lequel elle s'applique

DIAGRAMMES D'ÉTATS-TRANSITIONS (13)

Actions dans un état

- Les états peuvent également contenir des actions :
 - elles sont exécutées
 - à l'entrée ou à la sortie de l'état
 - l'action d'entrée (**entry**) est exécutée de manière instantanée et atomique
 - l'action de sortie (**exit**) est exécutée à la sortie de l'état
 - lorsqu'une occurrence d'événement interne survient
 - l'action sur un événement interne (**on**) est exécutée lors de l'occurrence d'un événement qui ne conduit pas à un autre état

DIAGRAMMES D'ÉTATS-TRANSITIONS (14)

Actions dans un état

Nom d'un état

entry : action d'entrée

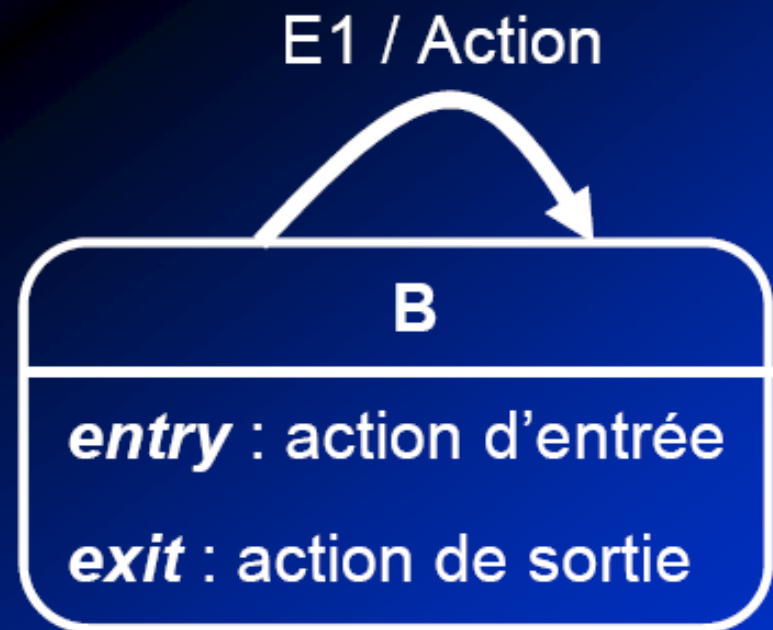
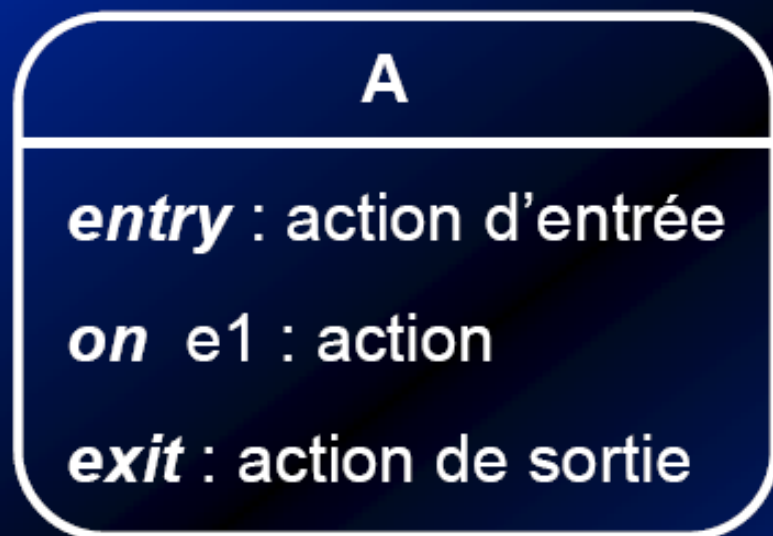
on nom_événement : action

exit : action de sortie

DIAGRAMMES D'ÉTATS-TRANSITIONS (15)

Opérations, actions et activités

- un événement interne n'entraîne pas l'exécution des actions de sortie et d'entrée, contrairement au déclenchement d'une transition réflexive



DIAGRAMMES D'ÉTATS-TRANSITIONS (16)

Opérations, actions et activités

- Les **actions** correspondent à des **opérations** dont la **durée d'exécution** est **négligeable**.

- Le mot clé **do:** indique une activité

DIAGRAMMES D'ÉTATS-TRANSITIONS (17)

Opérations, actions et activités

- Il existe deux formes d'activités :

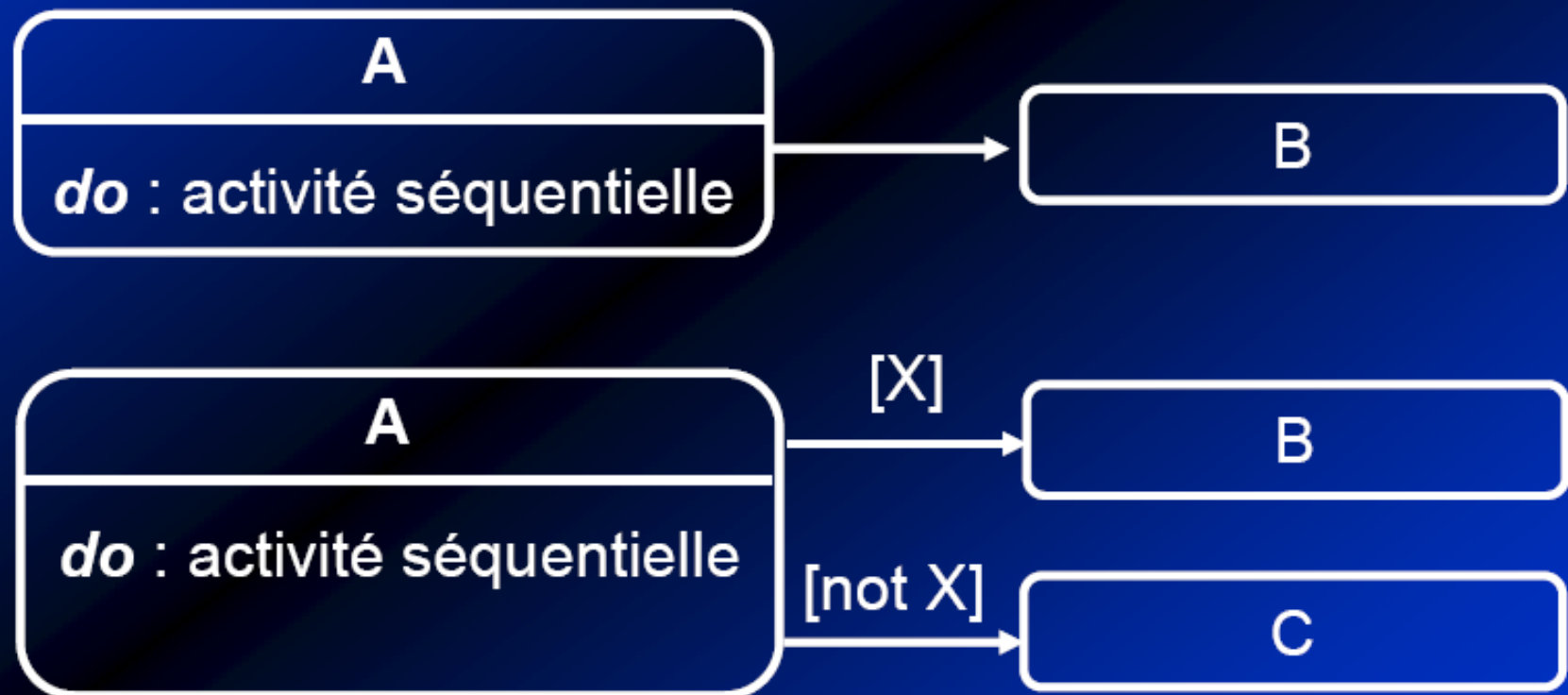
- activité cyclique

- activité séquentielle

DIAGRAMMES D'ÉTATS-TRANSITIONS (18)

Opérations, actions et activités

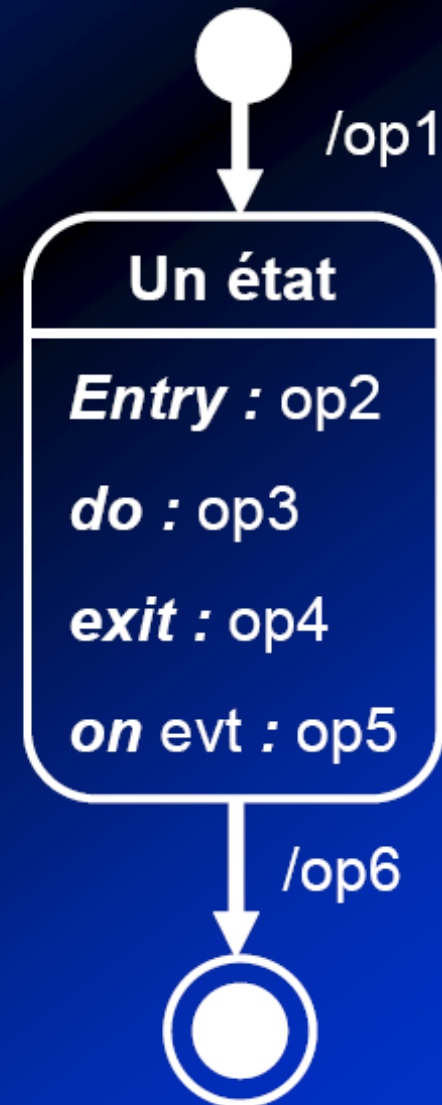
Lorsqu'une **activité se termine**, les **transitions automatiques** (sans événement), mais éventuellement protégées par des **gardes**, sont déclenchées



DIAGRAMMES D'ÉTATS-TRANSITIONS (19)

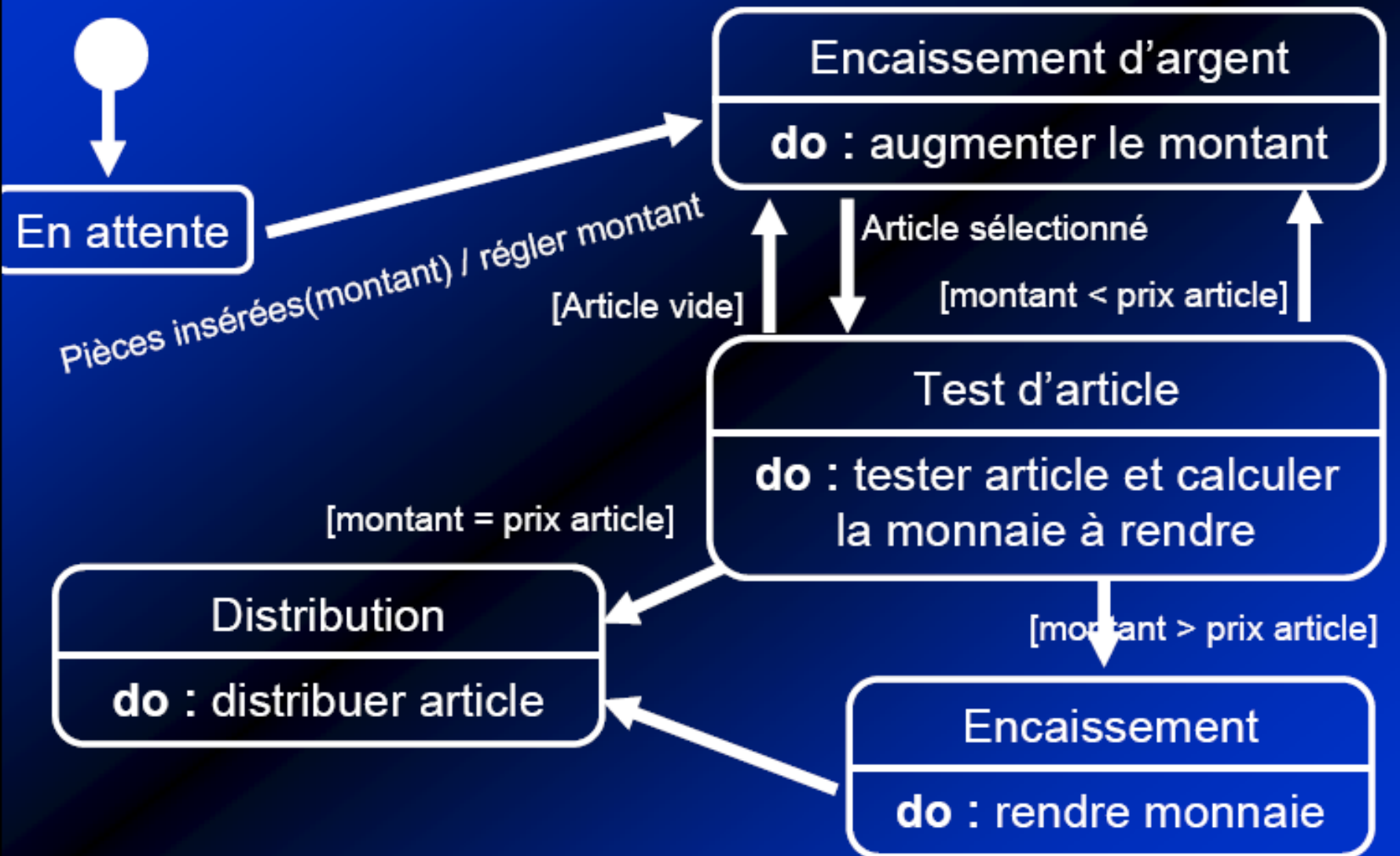
Point d'exécution des opérations

- 6 manières d'associer une opération à une transition :
 - l'action associée à la **transition d'entrée** (op1)
 - l'action d'entrée de l'état (op2)
 - l'**activité** dans l'état (op3)
 - l'action de **sortie** de l'état (op4)
 - l'action associée aux **événements internes** (op5)
 - l'action associée à la **transition de la sortie** de l'état (op6)



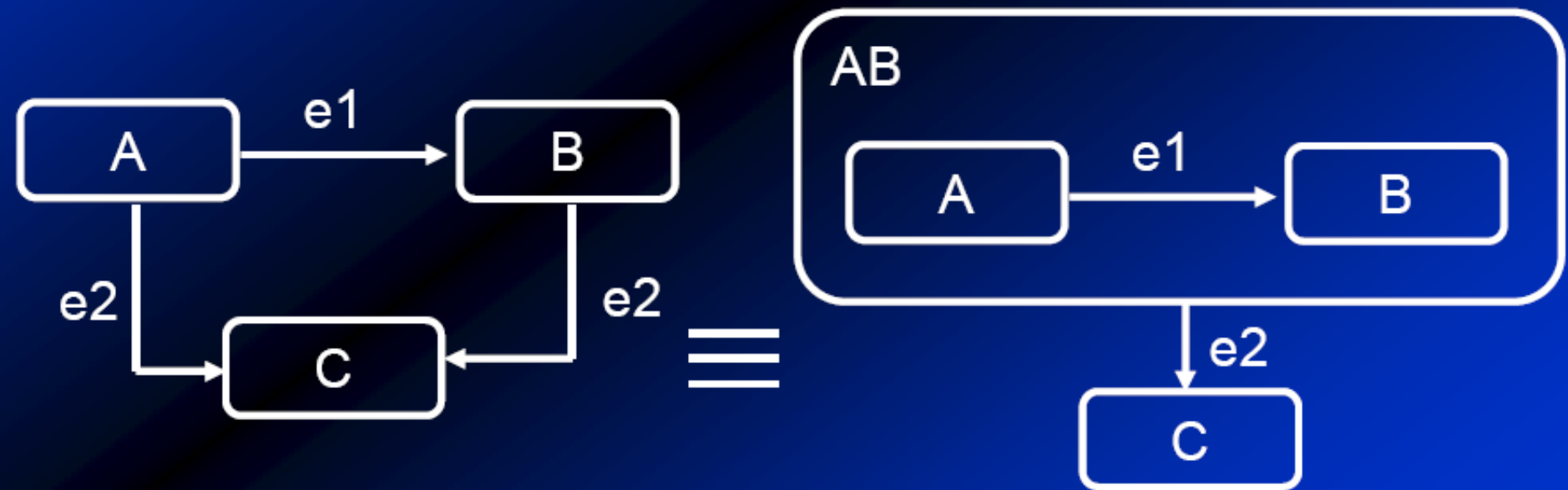
DIAGRAMMES D'ÉTATS-TRANSITIONS (20)

Diagramme d'états-transitions du Distributeur Automatique de Boissons



DIAGRAMMES D'ÉTATS-TRANSITIONS (21)

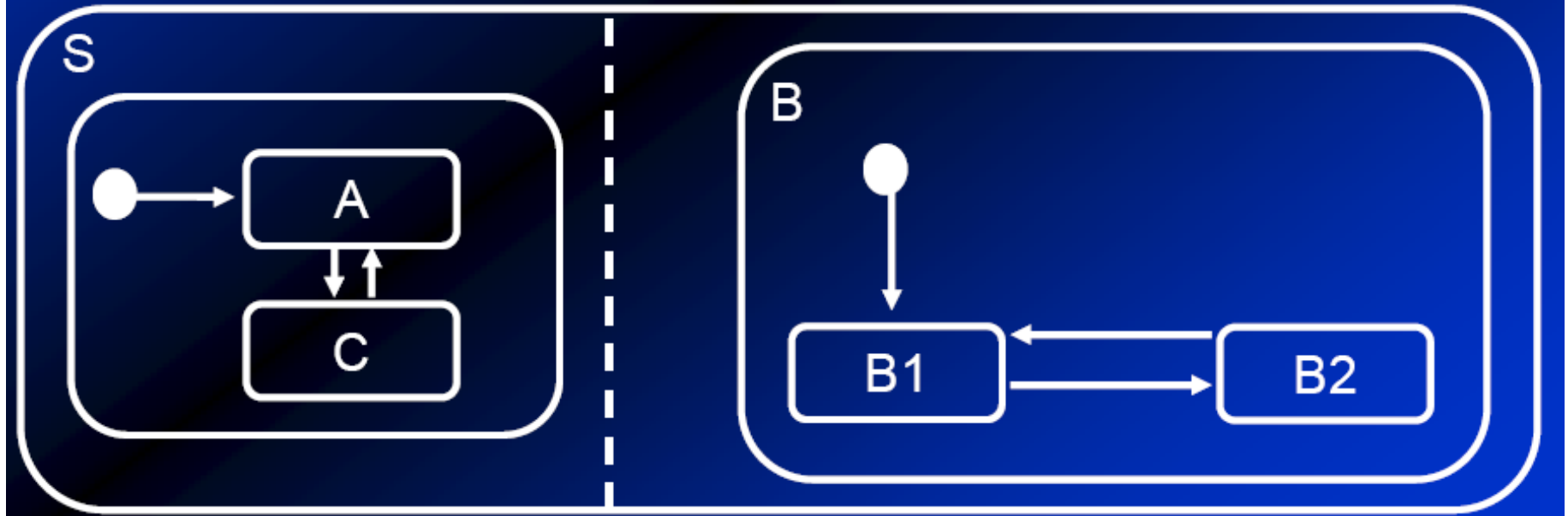
Généralisation d'états



DIAGRAMMES D'ÉTATS-TRANSITIONS (22)

Agrégation d'états

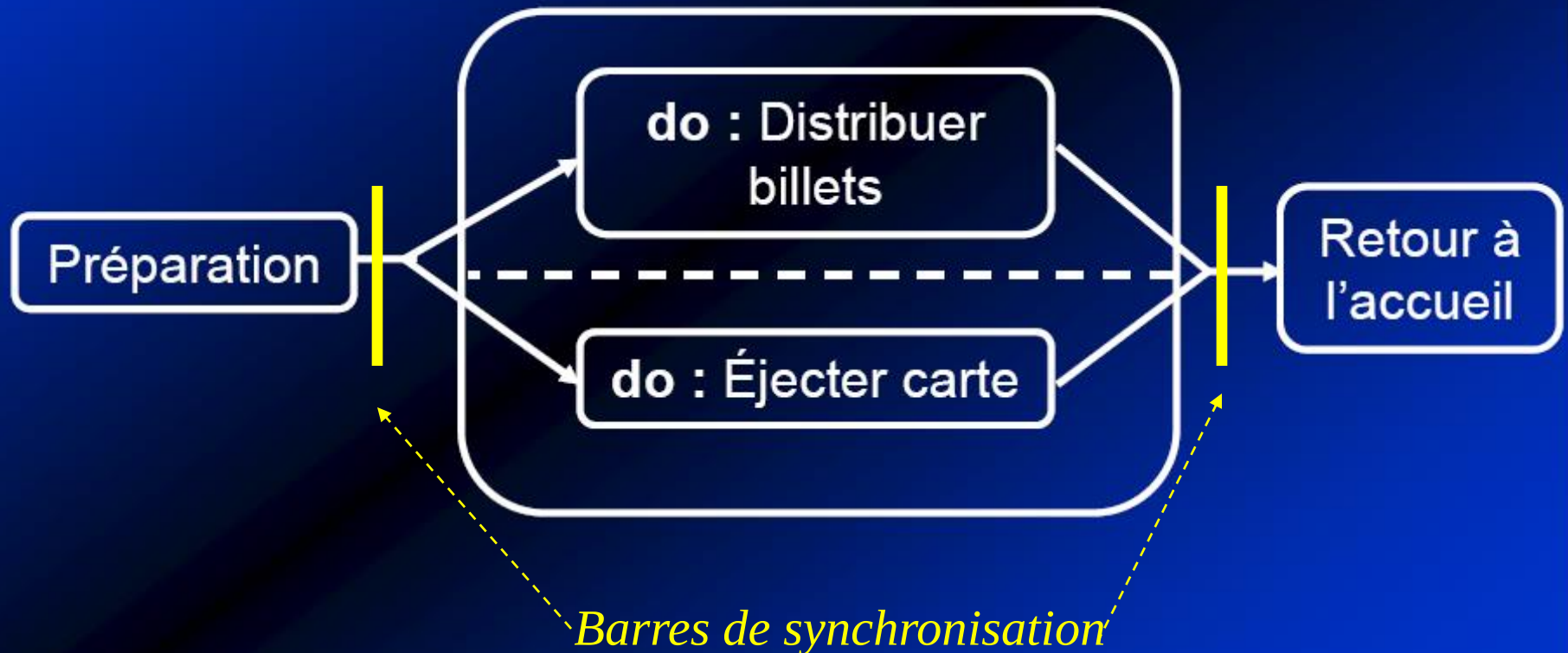
- Forme de parallélisme entre automates



DIAGRAMMES D'ÉTATS-TRANSITIONS (23)

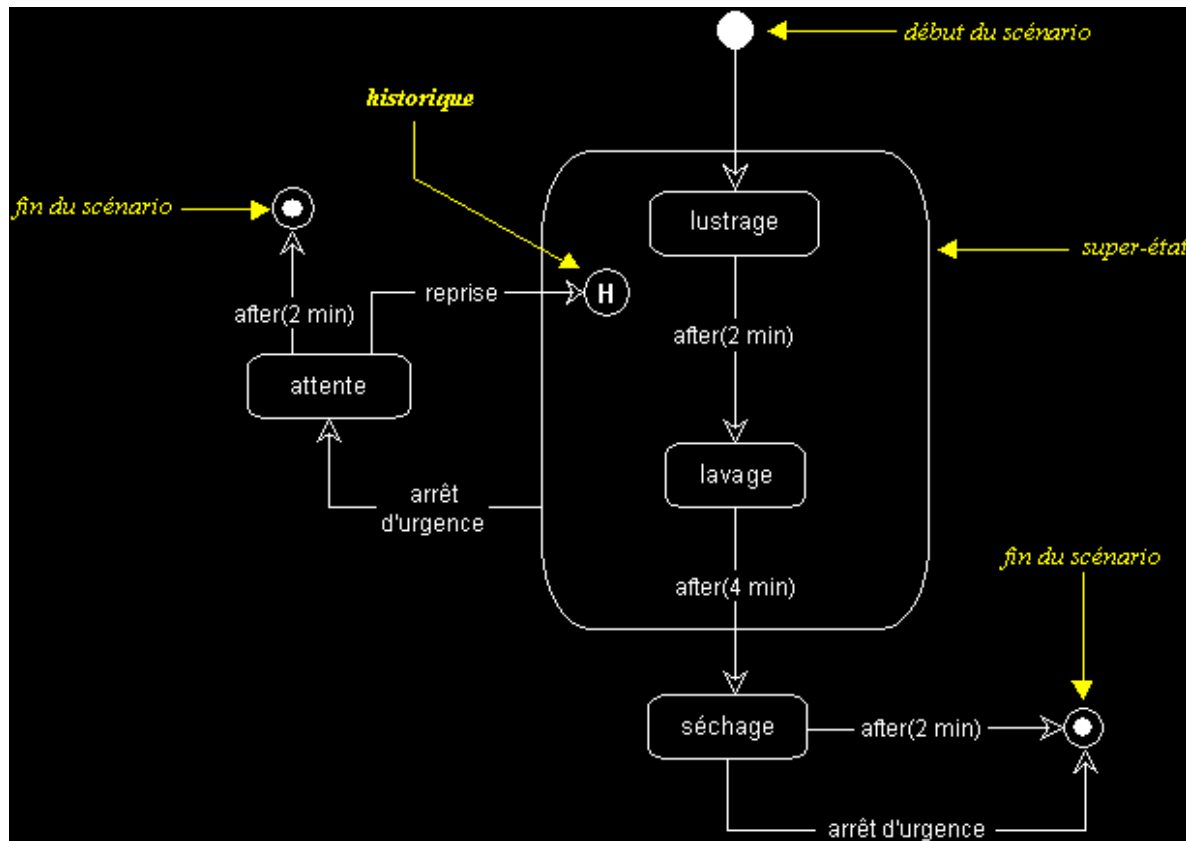
Agrégation d'états

- Exemple : activité d'émission de billets



DIAGRAMMES D'ÉTATS-TRANSITIONS (24)

Etat transition : machine à laver



Pseudo-état History :
Il permet à un super-état de se souvenir du dernier sous-état séquentiel qui était actif avant une transition sortante.
Une transition vers l'état *History* rend à nouveau actif le dernier état actif au lieu de ramener vers le sous-état initial.

DIAGRAMMES D'ACTIVITÉS

DIAGRAMMES D'ACTIVITÉS (1)

(les $\neq$ activités (fonctions) du système)

Introduction

- Variante des diagrammes d'états-transitions :



DIAGRAMMES D'ACTIVITÉS (2)

Gardes

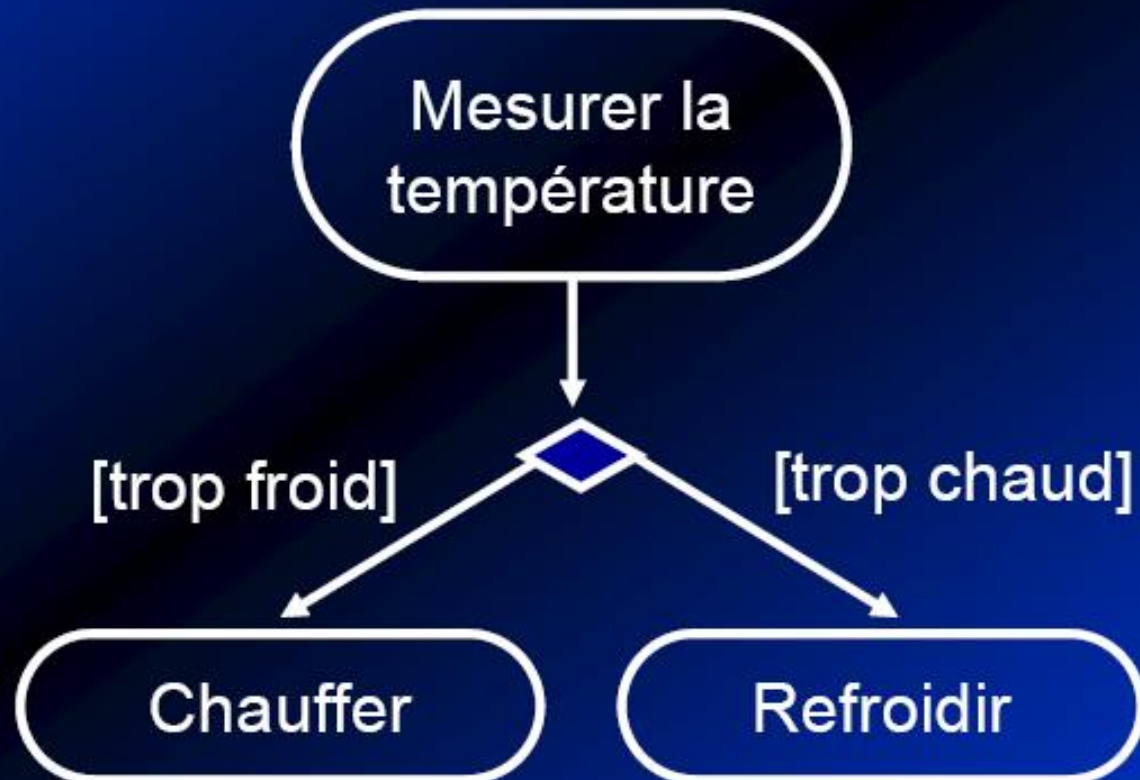
- Les **gardes** sont les **labels des transitions** dont elles valident le **déclenchement**



DIAGRAMMES D'ACTIVITÉS (3)

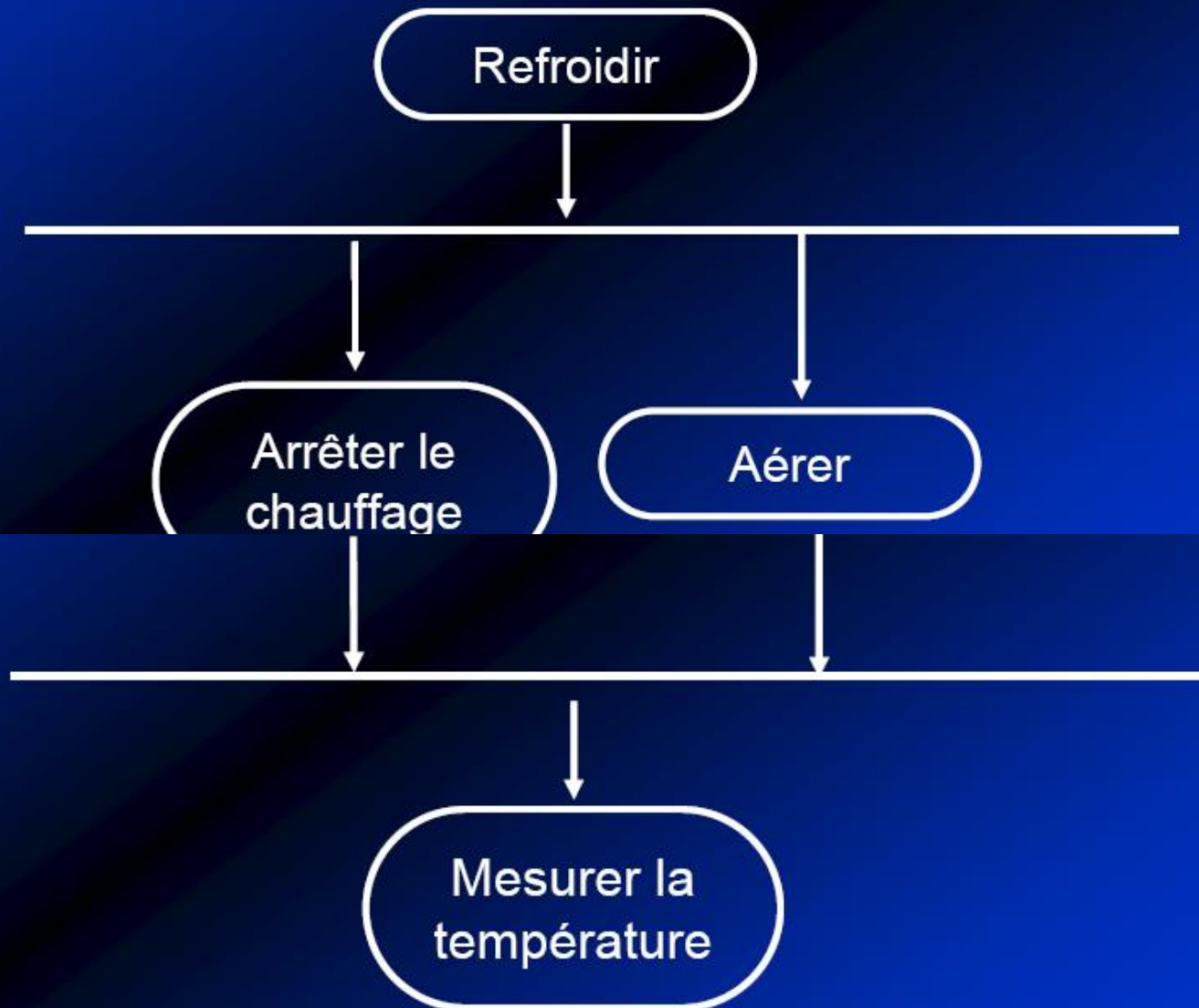
Gardes

- Une **condition** peut être matérialisée par un **losange** dont sortent plusieurs transitions :



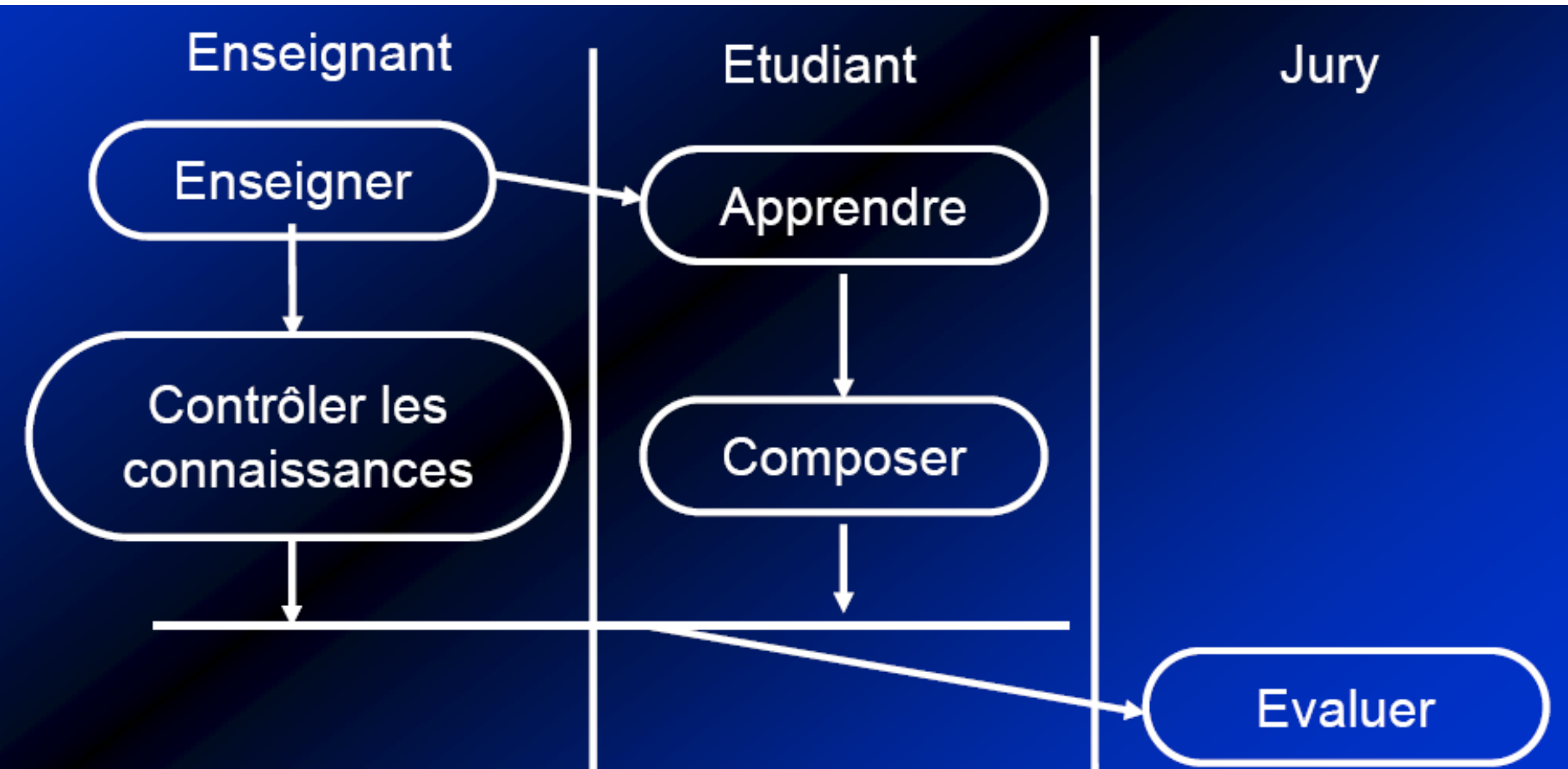
DIAGRAMMES D'ACTIVITÉS (4)

Synchronisations



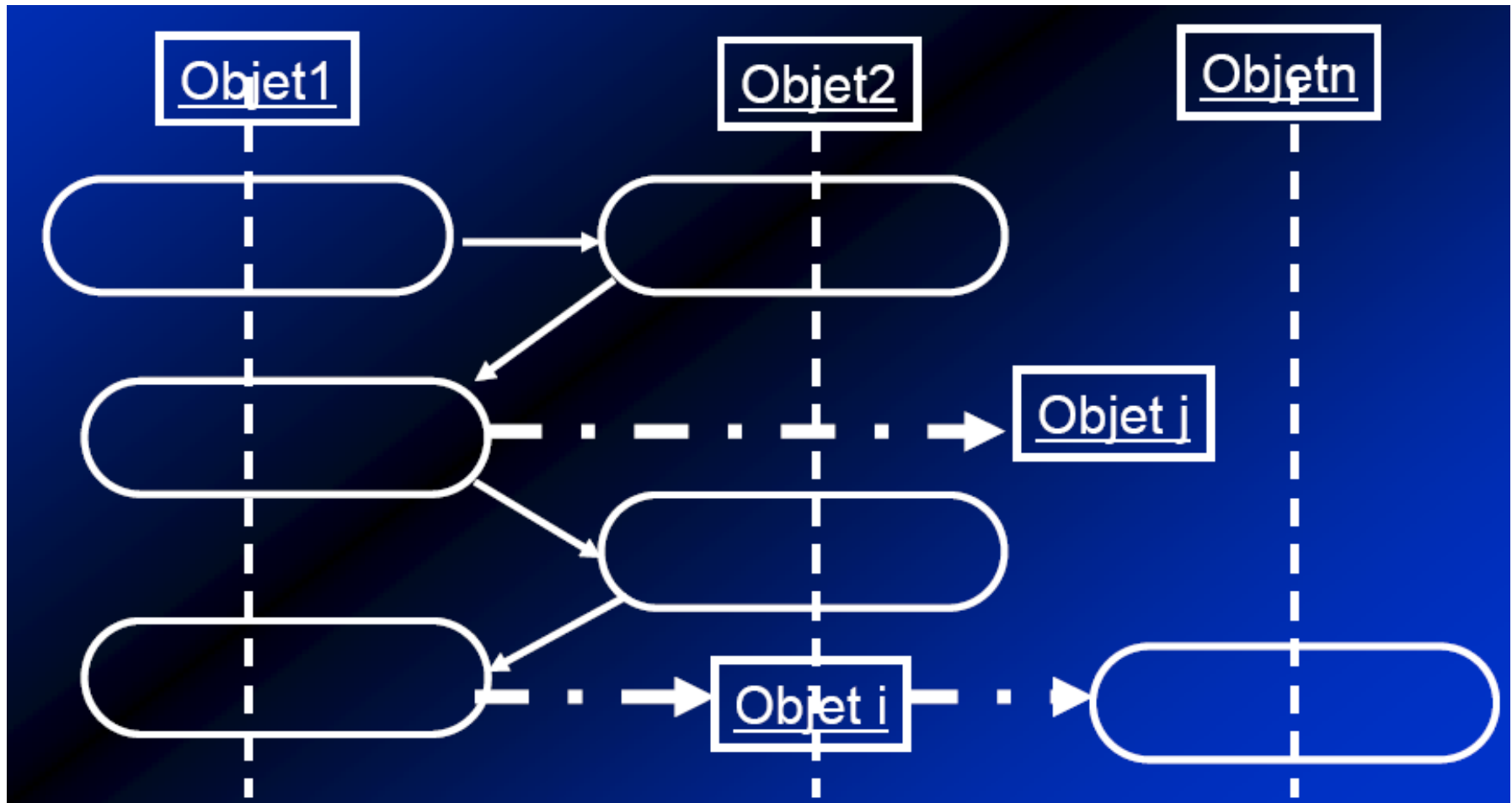
DIAGRAMMES D'ACTIVITÉS (5)

Découpage :



DIAGRAMMES D'ACTIVITÉS (6)

Objets dans un diagramme d'activités



DIAGRAMMES DE COMPOSANTS

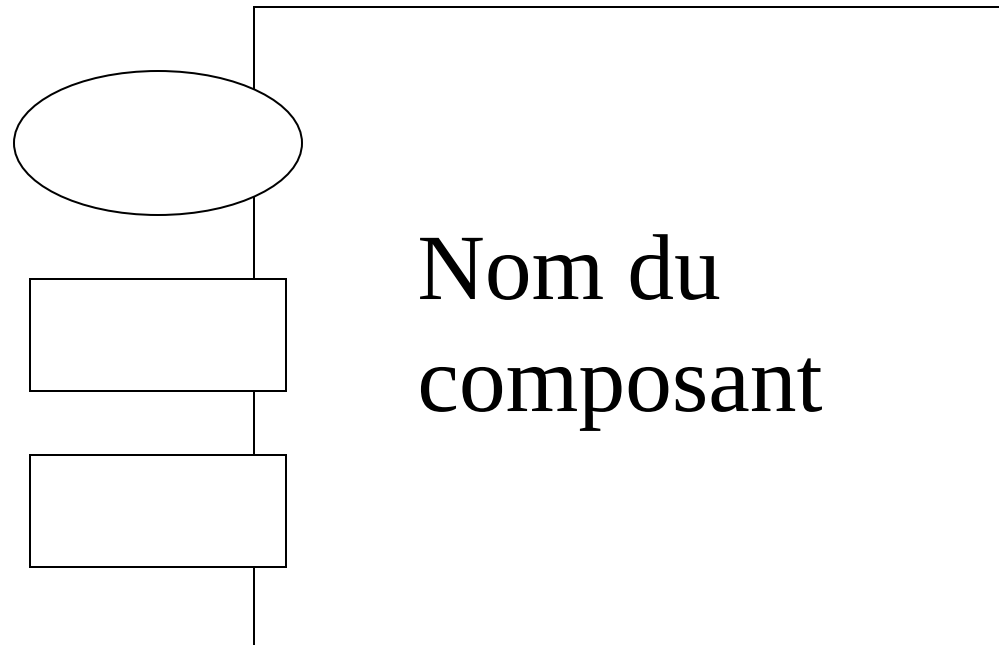
DIAGRAMMES DE COMPOSANTS (1)

(liens de dépendance entre les composants logiciels)

- Décrivent les éléments physiques et leurs relations dans l'environnement de réalisation:
 - montre les dépendances entre les composants logiciels
 - permet de spécifier l'architecture logicielle dans un environnement de développement donné

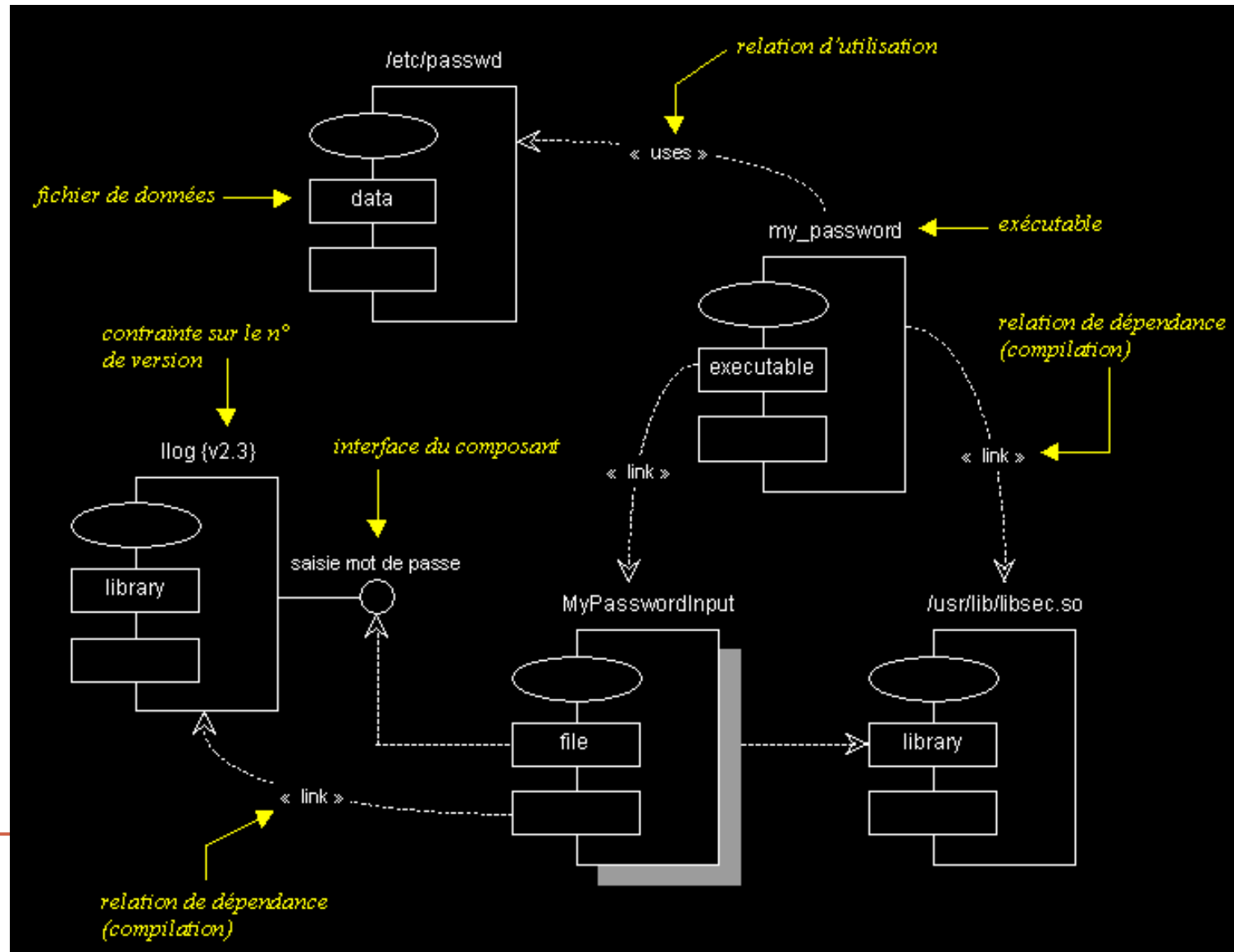
DIAGRAMMES DE COMPOSANTS (2)

Notation:



DIAGRAMMES DE COMPOSANTS (3)

Exemple :



DIAGRAMMES DE DÉPLOIEMENT

DIAGRAMMES DE DÉPLOIEMENT (1)

(représentation de l'architecture matérielle et logicielle du système)

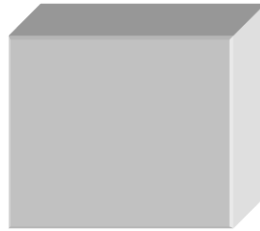
- Disposition physique des différents matériels (les nœuds) qui entrent dans la composition d'un système
- Répartition des programmes exécutables sur ces matériels

- Un lien de dépendance représente un lien de communication
- Et permet
 - de donner la structure d'une plate-forme physique
 - de préciser où se trouvent les processus
 - de montrer comment les objets se créent et se déplacent dans une architecture distribuée

DIAGRAMMES DE DÉPLOIEMENT (2)

Notation :

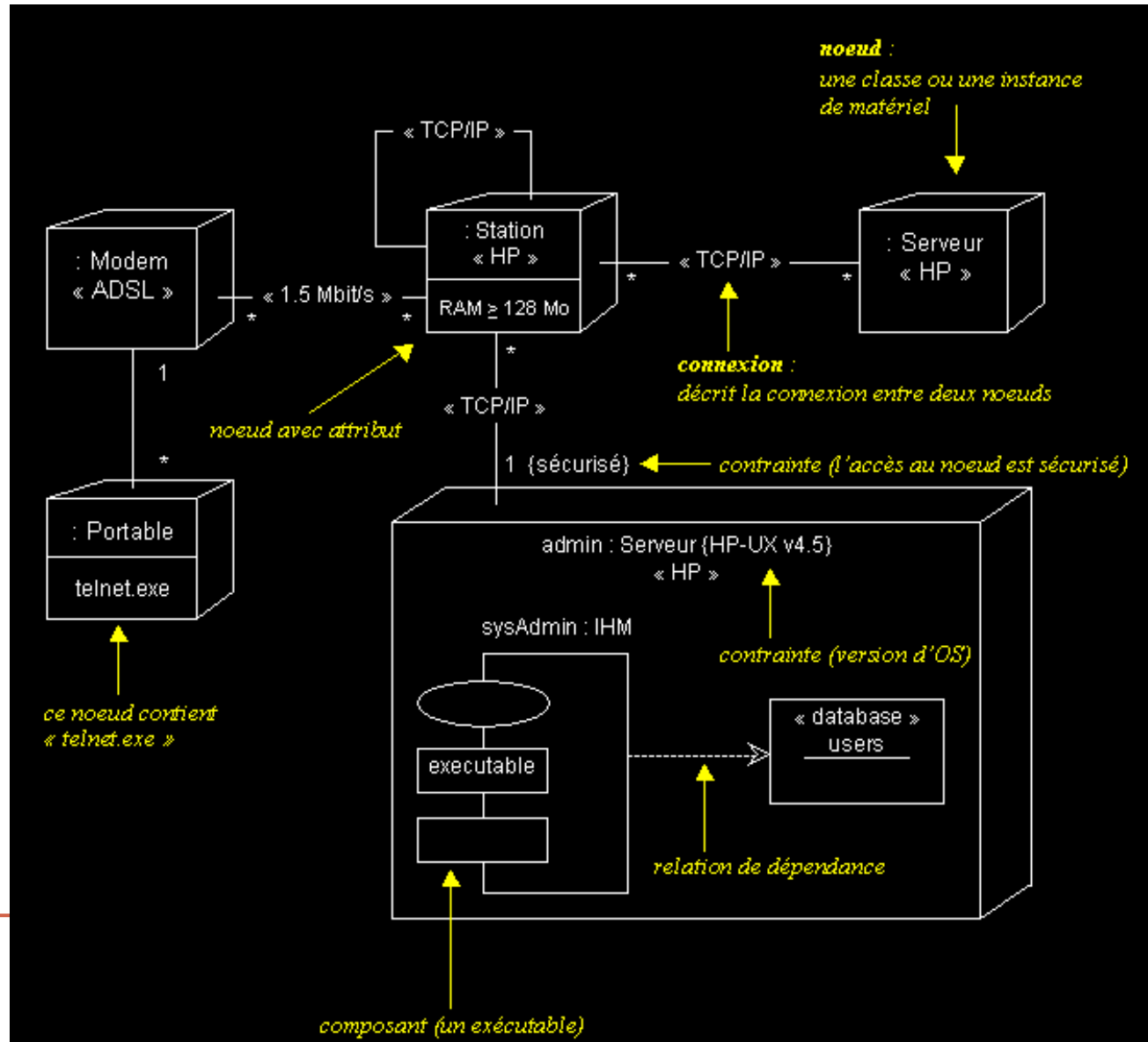
Nœuds :



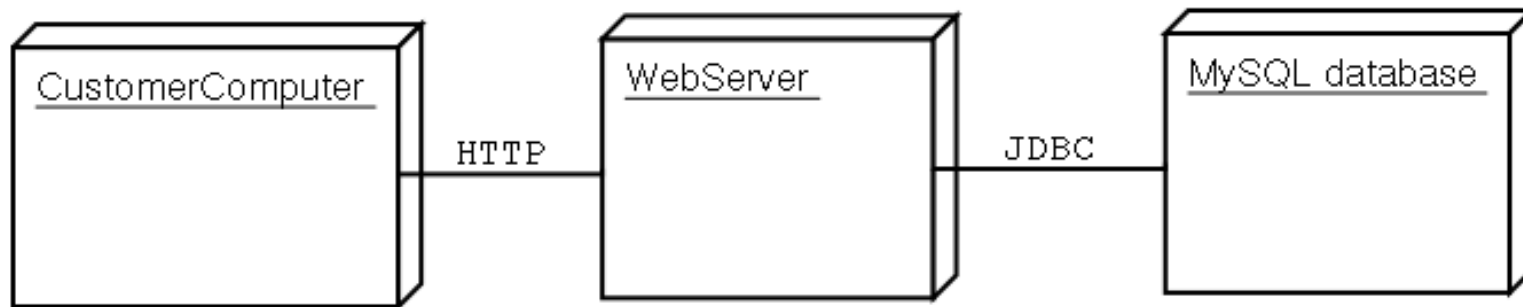
Modules : principaux programmes

DIAGRAMMES DE DÉPLOIEMENT (3)

Exemple :



DIAGRAMMES DE DÉPLOIEMENT (4)



Auteur du diagramme : *Ralph L. Vinciguerra*

UML 2.0

BIBLIOGRAPHIE

- « UML 2 : Initiation, exemples et exercices corrigés », Laurent DEBRAUWER et Fien VAN DER HEYDE, ED. eni
- « UML 2 en action : De l'analyse des besoins à la conception J2EE », Pascal ROQUES et Franck VALLEE, ED. Eyrolles
- www.uml.org
- www.omg.com

4 principales motivations :

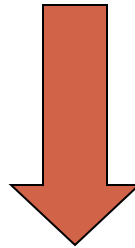
- MDA / MDD
- Automatisation
- Abstraction
- Modularisation

LES MOTIVATIONS (2)

1) MDA / MDD :

MDA : Model Driven Architecture

MDD : Model Driven Development



**Construction par modélisation
plutôt que par codage**

LES MOTIVATIONS (3)

2) Automatisation

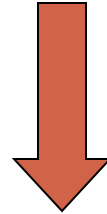
**Langage descriptif
prescriptif**



Langage

3) Abstraction

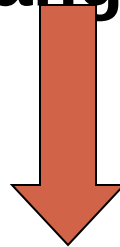
Élever le niveau d'abstraction



**Développement plus naturel
(détaché du code pur)**

4) Modularisation

Modulariser ce langage en différents sous-langages



Pas nécessaire d'apprendre toute la syntaxe UML

Quatre nouveaux diagrammes :

- Diagramme de modules ou paquetages
(package diagram)
- Diagramme de structure composite
(composite structure diagram)
- Diagramme global d'interaction
(interaction overview)
- Diagramme de temps
(timing diagram)

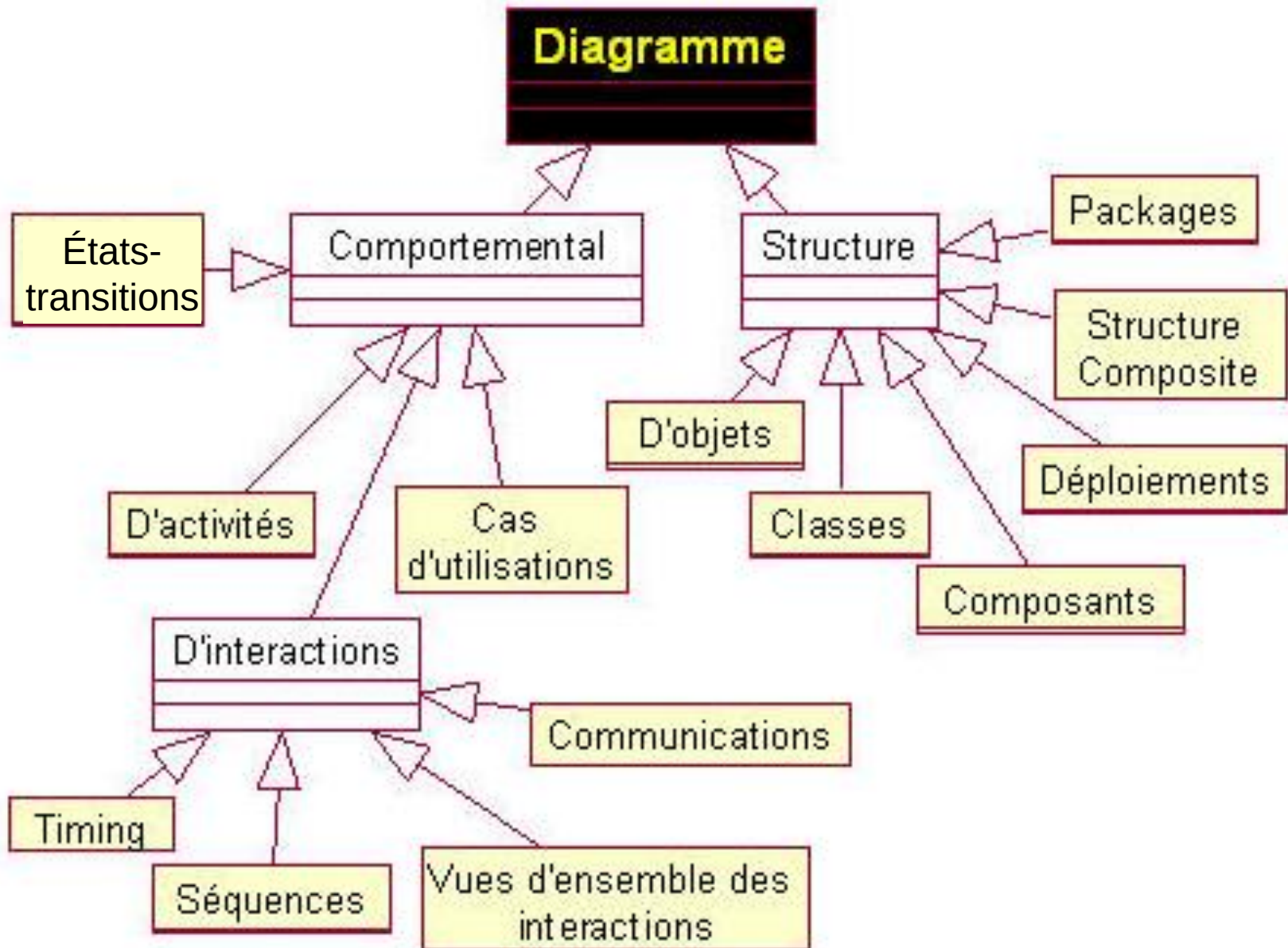
LES DIAGRAMMES (1)

UML 1.4 = 9 diagrammes

UML 2.0 = 13 diagrammes

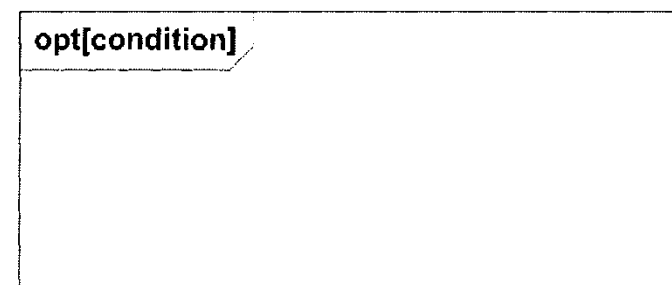
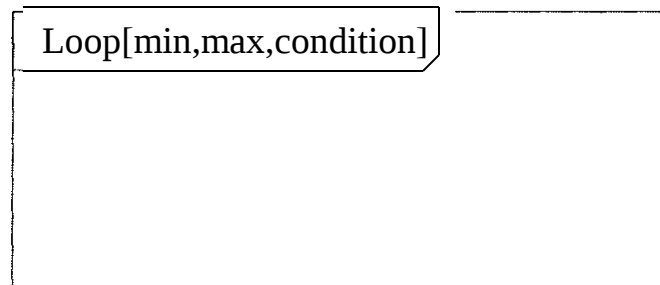
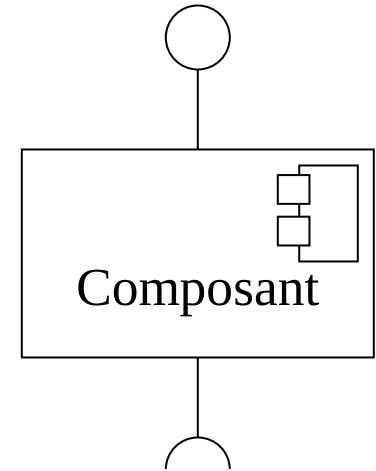
- Classification (3 catégories):
 - Les diagrammes de structure
 - Les diagrammes de comportement
 - Les diagrammes d'interactions (sous-catégories des diagrammes de comportement)

LES DIAGRAMMES (2)

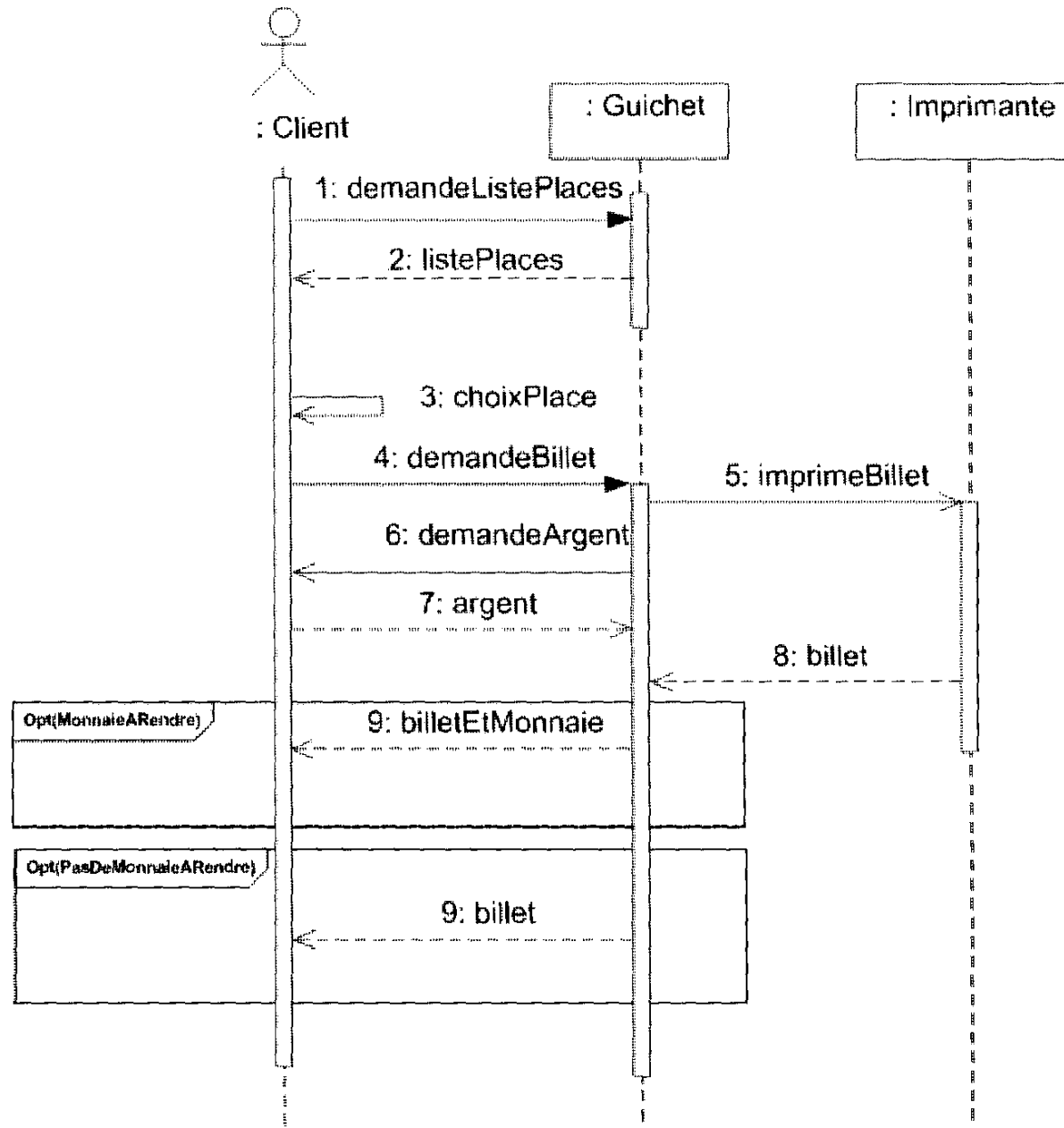


MODIFICATIONS DES DIAGRAMMES EXISTANTS

- Diagramme de collaboration devient diagramme de communication
- Lien entre les composants possibles dans les diagrammes de composants (notion d'interfaces requises et d'interfaces fournies)
- Ajout de cadres d'interaction sur les diagrammes de séquence



EXEMPLE DE CADRES

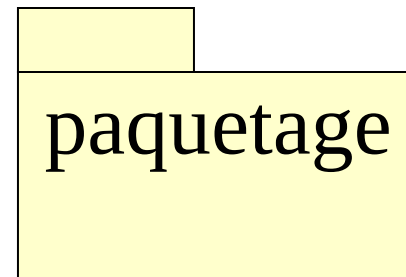


LE DIAGRAMME DE PAQUETAGE

[structure hiérarchique]

LES DIAGRAMMES DE PAQUETAGES (1)

- Utiliser un diagramme de classes pour y représenter la hiérarchie des modules (catégories) d'un projet.
- Représenter la structure hiérarchique au moment de la compilation



LES DIAGRAMMES DE PAQUETAGES (2)

Élevage de chevaux



Achat des chevaux

- Prospection

- Achat

Vente des chevaux

+ Publicité

+ Vente

« import »

Gestion des chevaux

+ Cheval

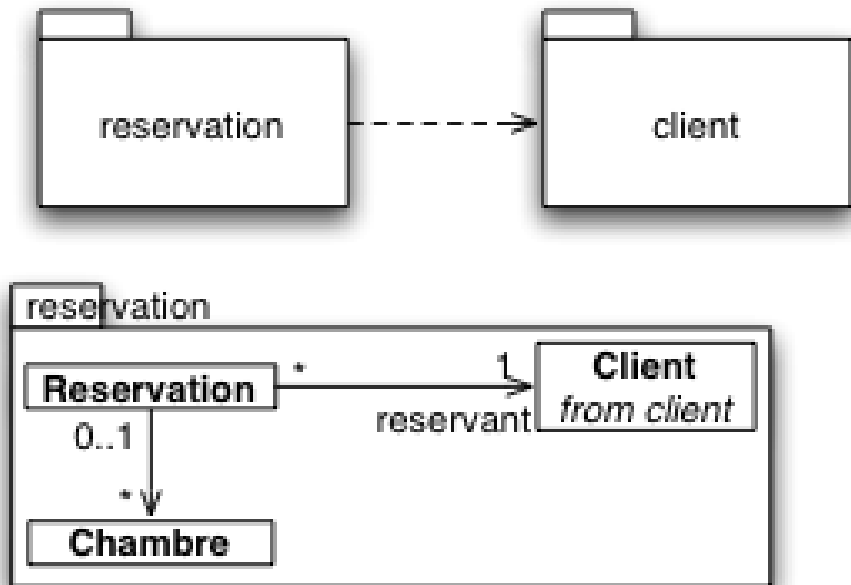
- Papiers Cheval

- Gestion
Vaccinations

« import »

LES DIAGRAMMES DE PAQUETAGES (3)

(Emmanuel Renaux)



Le paquetage `reservation` dépend du paquetage `client`, car au moins une classe du premier est associée à au moins une classe du deuxième. La classe `client` est indiquée comme provenant d'un autre paquetage `client`.

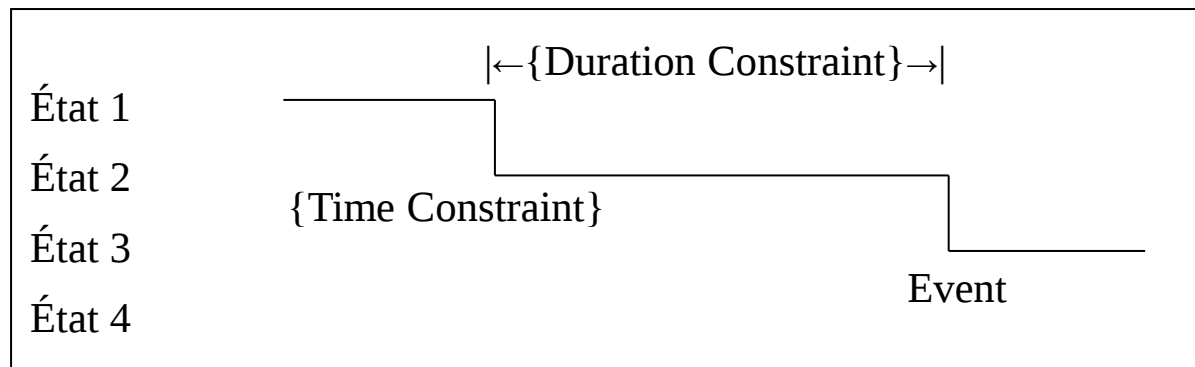
LE DIAGRAMME DE TEMPS

[interactions entre objets]

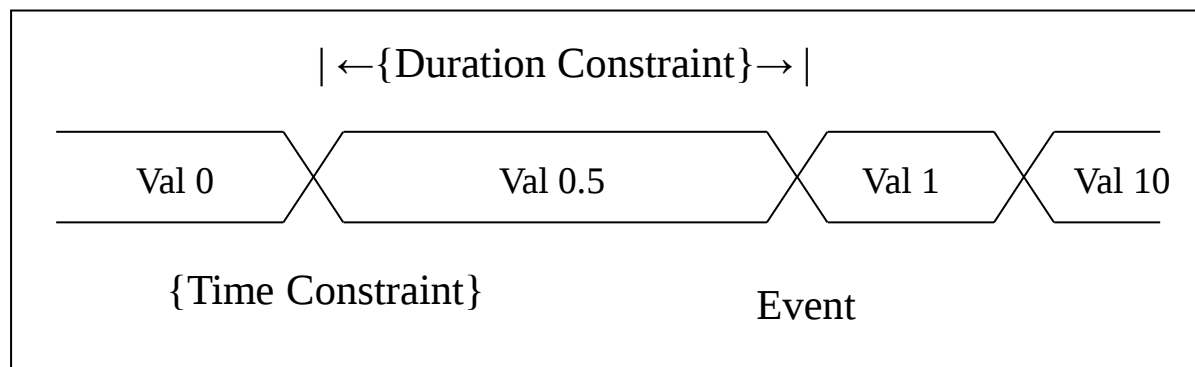
LES DIAGRAMMES DE TEMPS (1)

- Représenter l'interaction entre objets : accent mis sur le temps
- 2 graphiques :

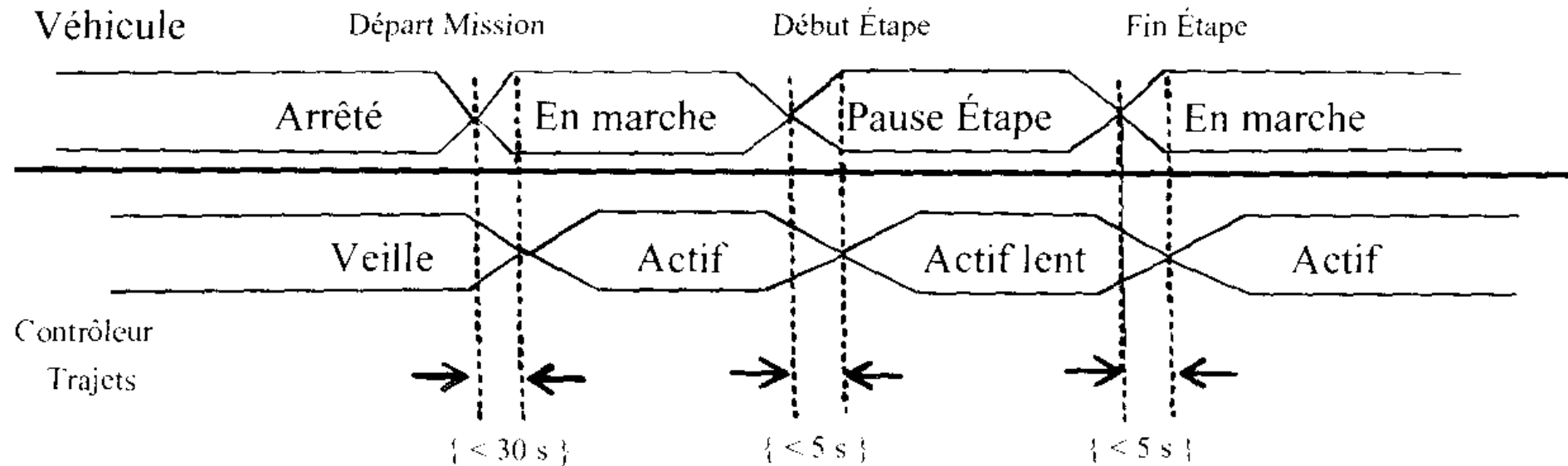
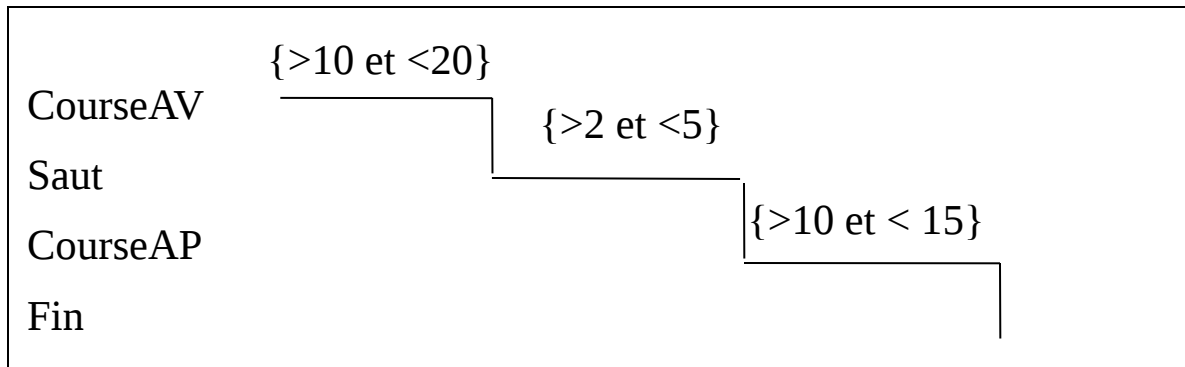
- Évolution de l'état de l'élément



- Évolution de la valeur de l'élément



LES DIAGRAMMES DE TEMPS (2)

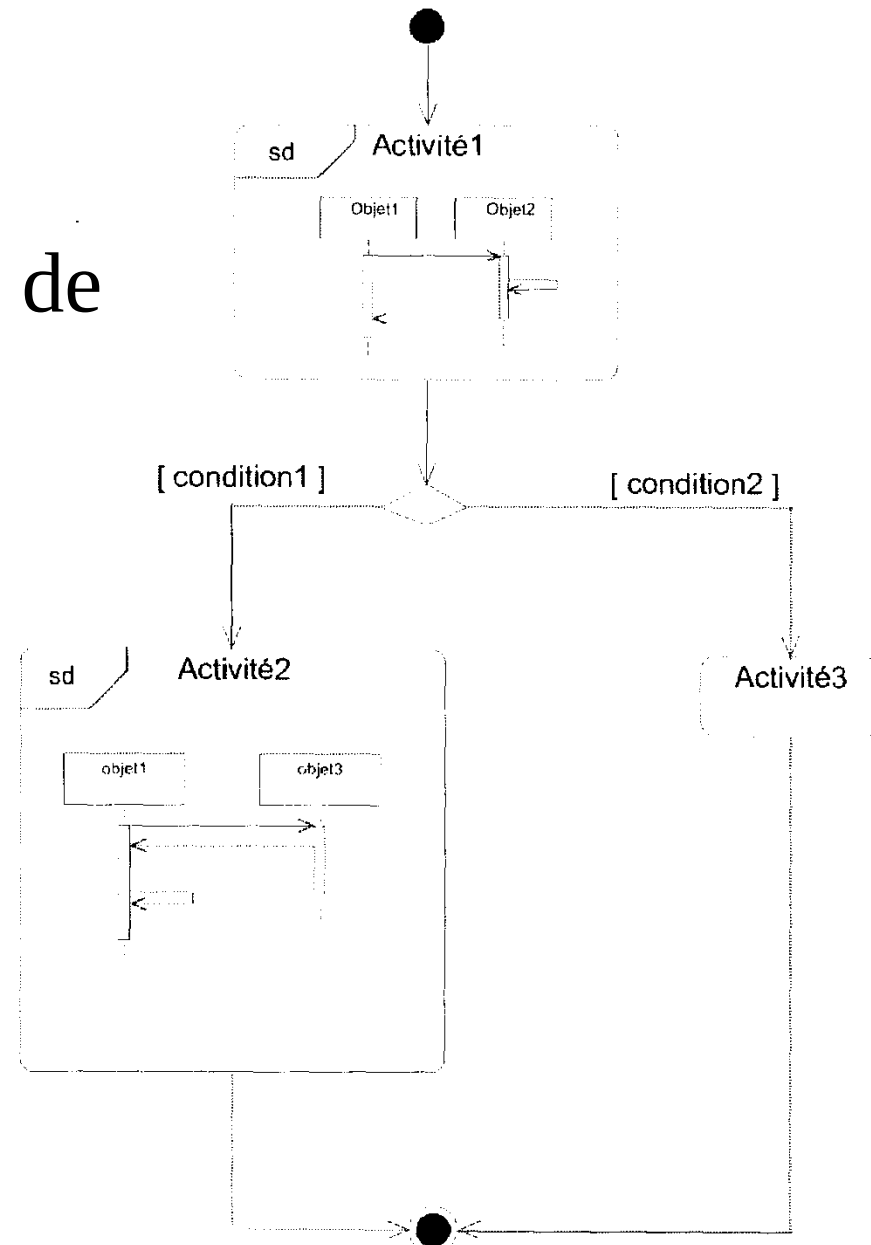


LE DIAGRAMME DE VUE D'ENSEMBLE DES INTERACTIONS

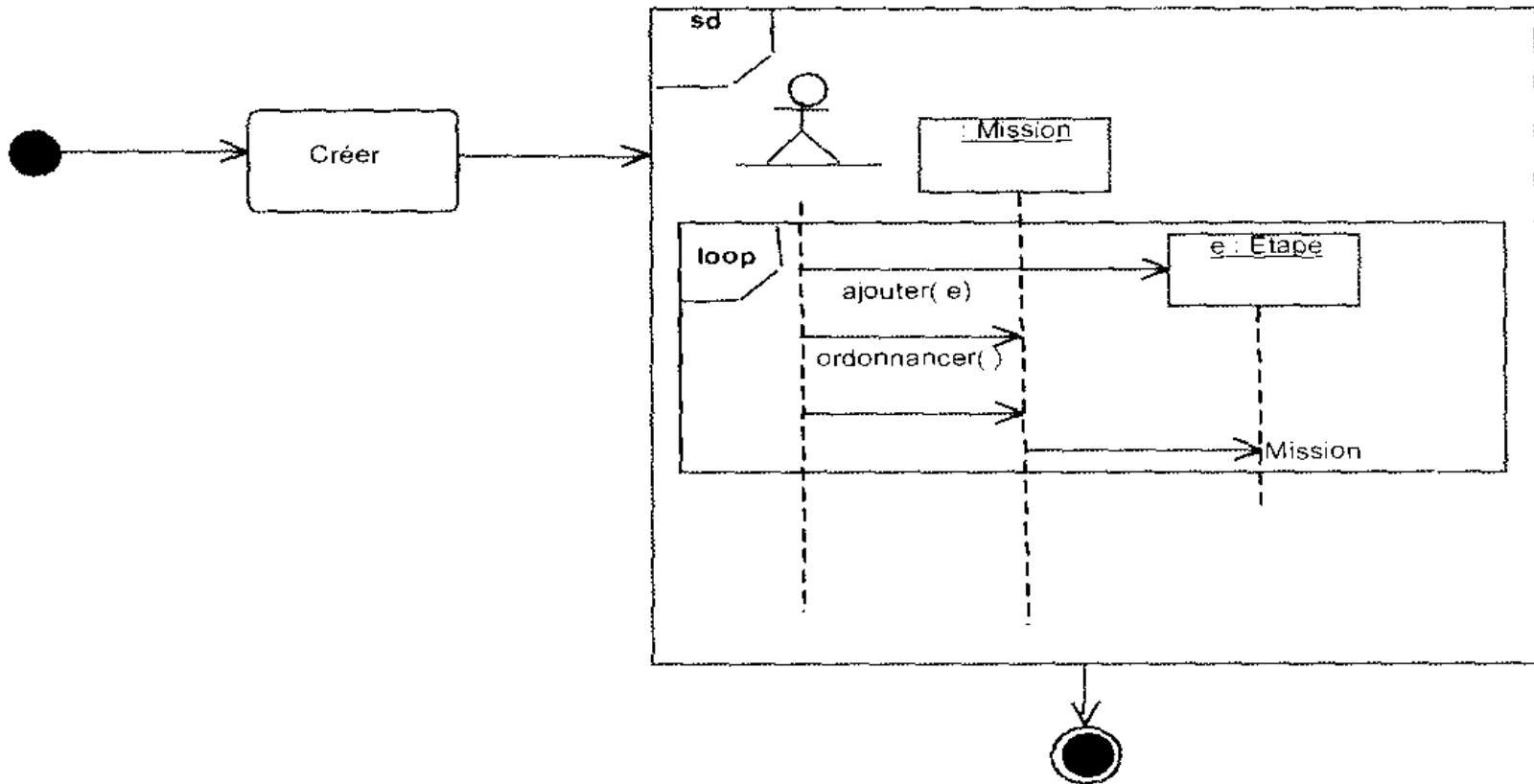
[mélange de diagramme de séquence et
d'activités]

LES DIAGRAMMES GLOBAUX D'INTERACTIONS (1)

= mélange de diagramme de séquence et d'activités



LES DIAGRAMMES GLOBAUX D'INTERACTIONS (2)



LE DIAGRAMME DE STRUCTURE COMPOSITE

[décomposition d'une classe]

LES DIAGRAMMES DE STRUCTURE COMPOSITE (1)

= décomposition d'une classe au moment de l'exécution.

- Décrire la composition d'un objet complexe lors de son exécution

LES DIAGRAMMES DE STRUCTURE COMPOSITE (2)

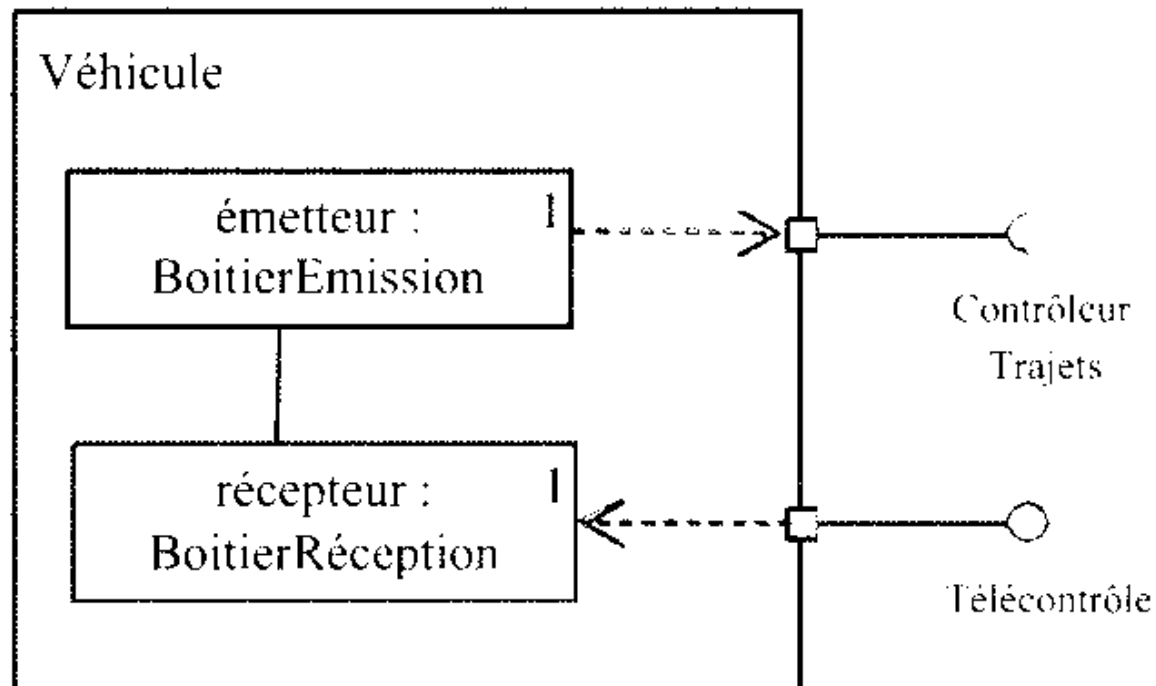


TABLEAU RÉCAPITULATIF

TABLEAU RÉCAPITULATIF : DIAGRAMMES / CYCLE DE VIE

Cycle de vie / Diagrammes	Découverte des besoins	Analyse	Conception
Classes		X	
Packages		X	
Objets		X	
Structure Composite		X	X
Cas d'utilisation	X		
Séquence	X		X
Collaboration	X	X	X

TABLEAU RÉCAPITULATIF : DIAGRAMMES / CYCLE DE VIE

Cycle de vie / Diagrammes	Découverte des besoins	Analyse	Conception
États		X	
Temps			X
Activité	X	X	
Global d'interactions	X	X	X
Composants			X
Déploiement			X

GESTION DE PROJET (de préférence)

1) Vue Utilisateurs :

- 1) Diagramme des cas d'utilisation
- 2) Diagramme de séquence
- 3) Diagramme global d'interactions

2) Vue Logique :

- 1) Diagramme de classes / Diagramme d'objets /
Diagrammes de paquetages / Diagramme de temps
- 2) Diagramme de collaboration
- 3) Diagramme d'états-transitions / Diagramme d'activités

3) Vue Conceptuelle :

- 1) Diagramme de structure composite
- 2) Diagramme de composants
- 3) Diagramme de déploiement