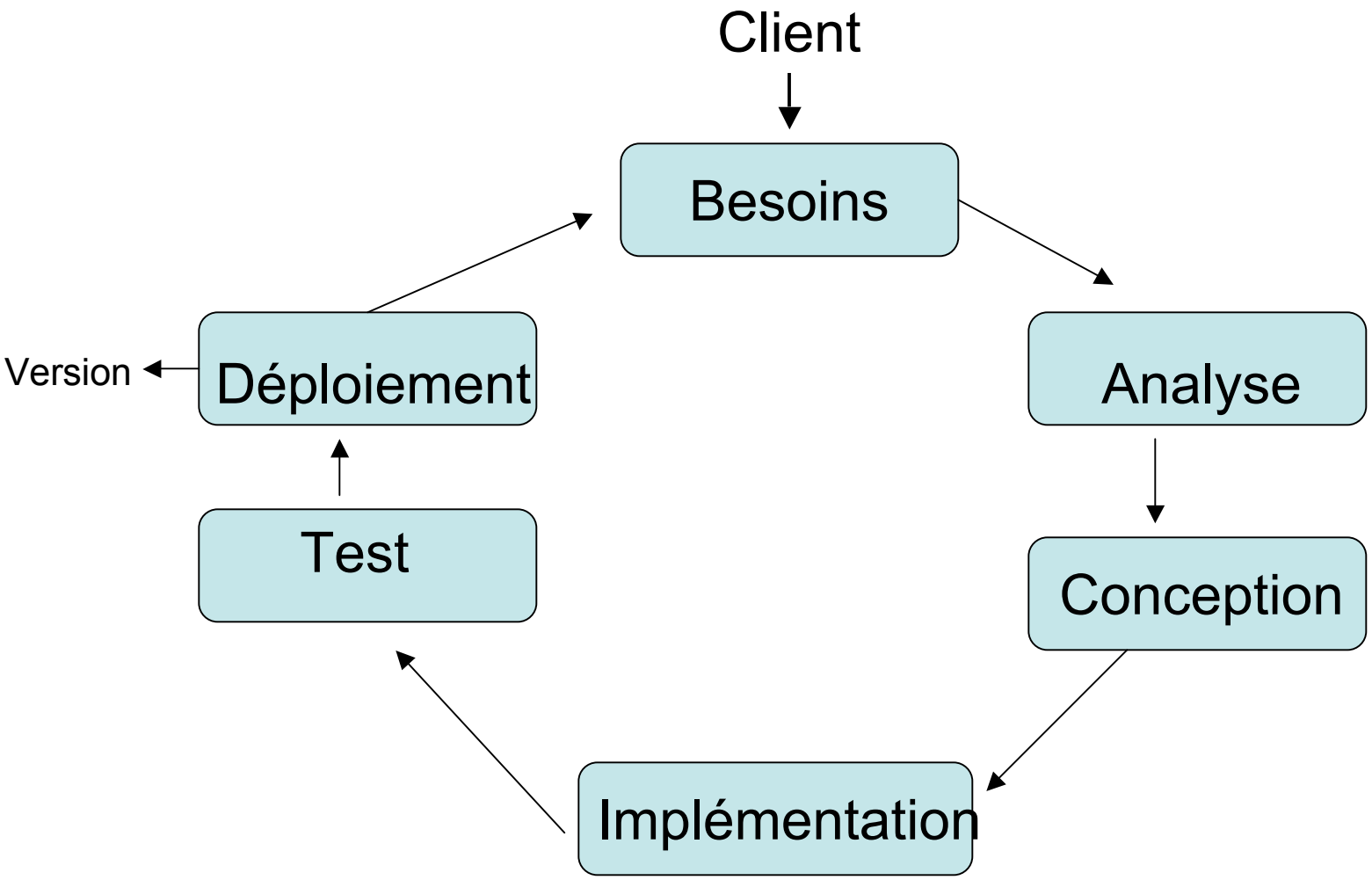
Unified Modeling Language
UML

Salima Hassas

Cours sur la base des transparents de : Gioavanna Di Marzo Serugendo et Frédéric Julliard

Cycle de vie du logiciel



Développement Logiciel

- A faire
  - Comprendre et conceptualiser le problème (besoins, analyse)
  - Résoudre le problème (conception)
  - Donner une solution (implémentation)
  - Documenter

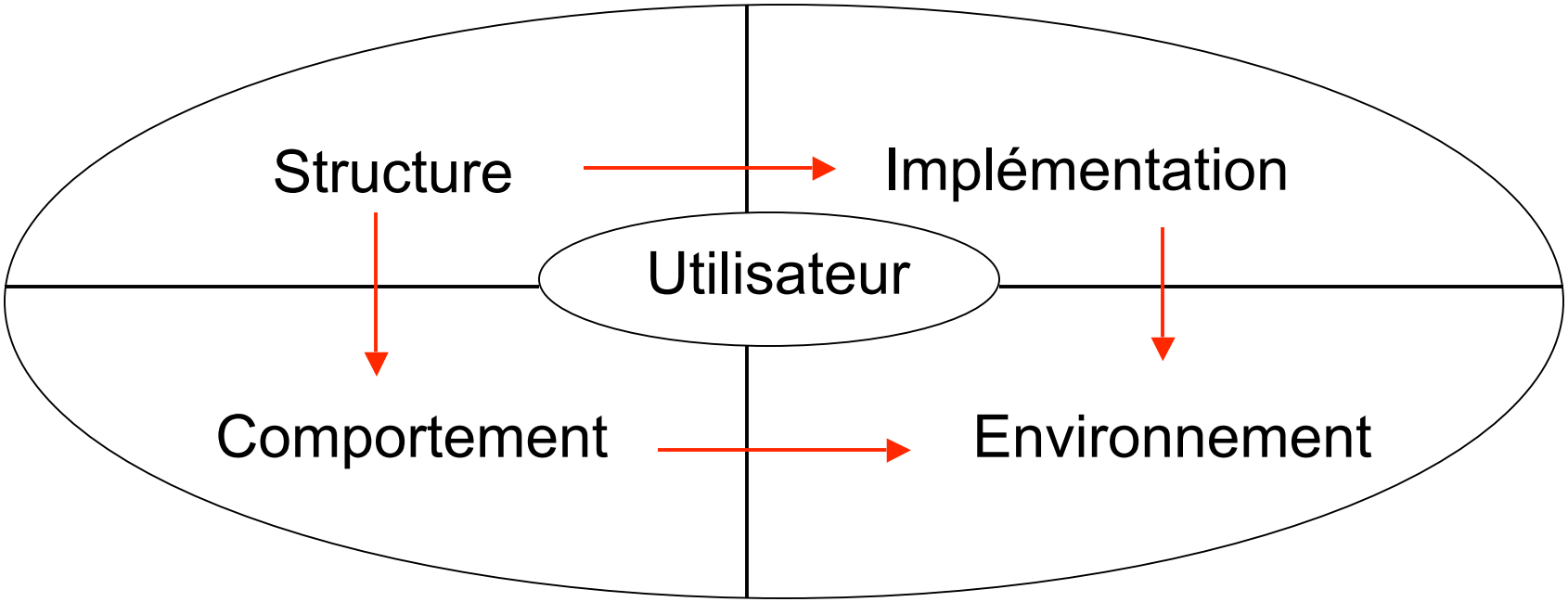
- UML permet de
  - Spécifier, visualiser et comprendre le problème
  - Capturer, communiquer et utiliser des connaissances pour la résolution du problème
  - Spécifier, visualiser et construire la solution
  - Documenter la solution

UML: définition

- Unified Modeling Language
  - Langage servant à décrire des modèles d'un système matériel ou logiciel basé sur des concepts orienté-objets
  - Système de notations pour modéliser les systèmes en utilisant des concepts orienté-objets
  - Langage de modélisation visuelle (graphique) qui permet de:
    - Spécifier le problème et sa solution
    - Visualiser le problème et la solution sous différents angles
    - Construire la solution
    - documenter

Modèles et Vues

- Les modèles du système sont visualisés par des vues
- Chaque vue correspond à une facette différente du système
- Chaque participant au développement du système en a une vue différente



Les vues d'UML (1/3)

- Vue Utilisateur
  - Buts et objectifs des clients du systèmes
  - Besoins requis par la solution
- Vue structurelle
  - Aspects statiques représentatnt la structure du problème
- Vue Comportementale
  - Aspects dynamiques du comportement du problème et de sa solution
  - Interactions et collaborations entre éléments de la solution

Les vues d'UML (2/3)

- Vue implémentation
  - Aspects de la structure et du comportement de la solution
- Vue environnementale
  - Aspects de la structure et du comportement du domaine dans lequel est réalisée la solution

Les vues d'UML (3/3)

- Les vues doivent être consistantes entre elles
- Elles doivent accompagner le cycle de vie du logiciel
- La modification d'une vue implique la modification des autres vues

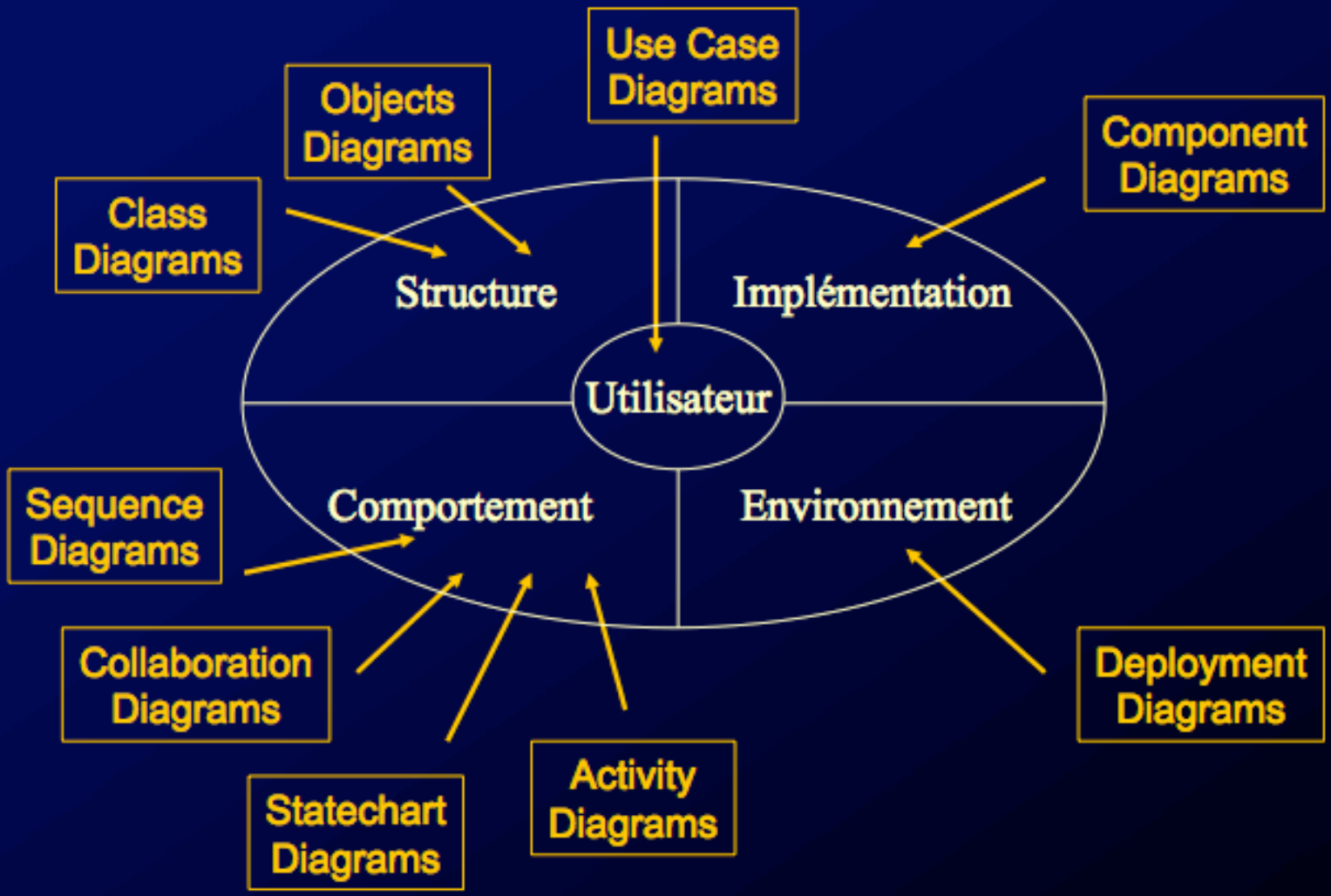
Les diagrammes d'UML

- Les diagrammes sont des graphes (nœud et arcs) qui permettent de représenter le problème et sa solution selon différents points de vue
- Les diagrammes forment ainsi des modèles du système (spécifier et visualiser)
- La combinaison de diagrammes représentent les vues du système
- Il existe 9 diagrammes représentants les deux aspects:
  - Statiques et structurels : relations
  - Dynamiques: comportement, collaborations, responsabilités

Les diagrammes d'UML

- 5 diagrammes structurels (vue statique)
  - Cas d'utilisation (Use case)
  - Classes
  - Objets
  - Composants
  - Déploiement
- 4 diagrammes comportementaux (vue dynamique)
  - Séquence
  - Activités
  - États -Transitions (Stateshart)
  - Collaboration

Diagrammes et Vues



Diagrammes de Cas d'Utilisation

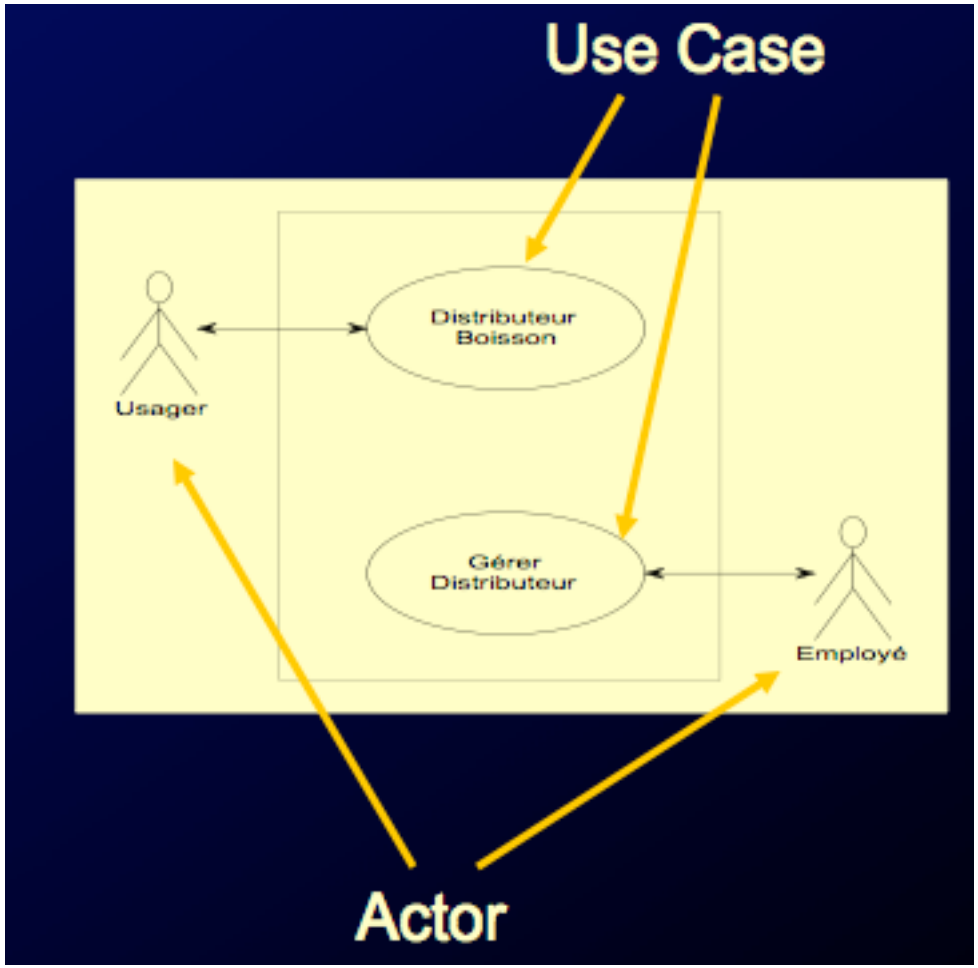
Use Cases

Les Cas d'Utilisation

- Représenter les besoins
  - La phase d'analyse des besoins nécessite de comprendre les besoins, de les exprimer et les formaliser
- Moyens pour représenter les besoins en UML :
  - Diagramme de cas d'utilisation: exprime l'organisation de l'utilisation du système par ses acteurs
  - Diagramme de séquence: pour chaque cas d'utilisation donne une description temporelle de l'interaction d'un acteur avec le système (scénario)
  - Diagramme Objets/classes: informations échangées entre système et acteurs
  - Diagramme de collaboration: interactions entre les objets métiers du système

Les Cas d'Utilisation (Use Case : Jacobson 92)

- Décrit la fonctionnalité que le système délivre à ses utilisateurs (humains ou autre système) et les liens qui peuvent exister entre eux (include, uses ou extends)
- Acteur (Actor) : Rôle joué par un utilisateur du système
- Cas d'utilisation (Use case): séquence d'actions que le logiciel garantit. Il définit les besoins du client



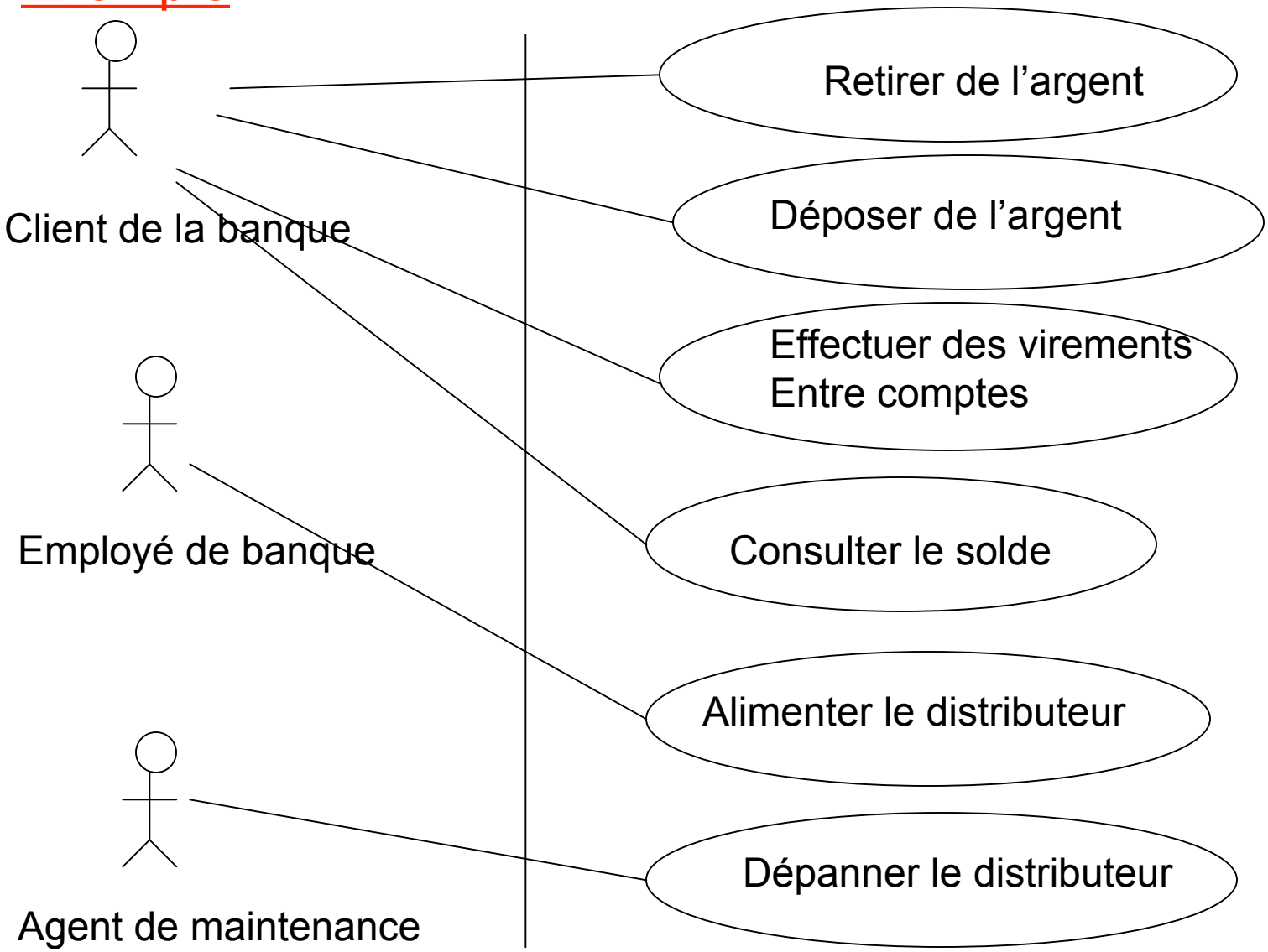
Les Cas d'Utilisation (Use Case : Jacobson 92)

- Acteur: définition
  - Entité externe (humain ou système) qui agit selon un rôle sur le système
  - Identifié par un nom correspondant à son rôle
  - Il peut être accompagné d'une description textuelle du rôle

- Cas d'utilisation: définition
  - ensemble des actions réalisées par le système en réponse à une action d'un acteur
  - Suite d'interactions entre un acteur et le système
  - Correspond à une fonction visible par l'utilisateur
  - Permet d'atteindre un objectif aux yeux de l'utilisateur
  - Doit être utile
  - Les cas d'utilisation ne doivent pas se chevaucher

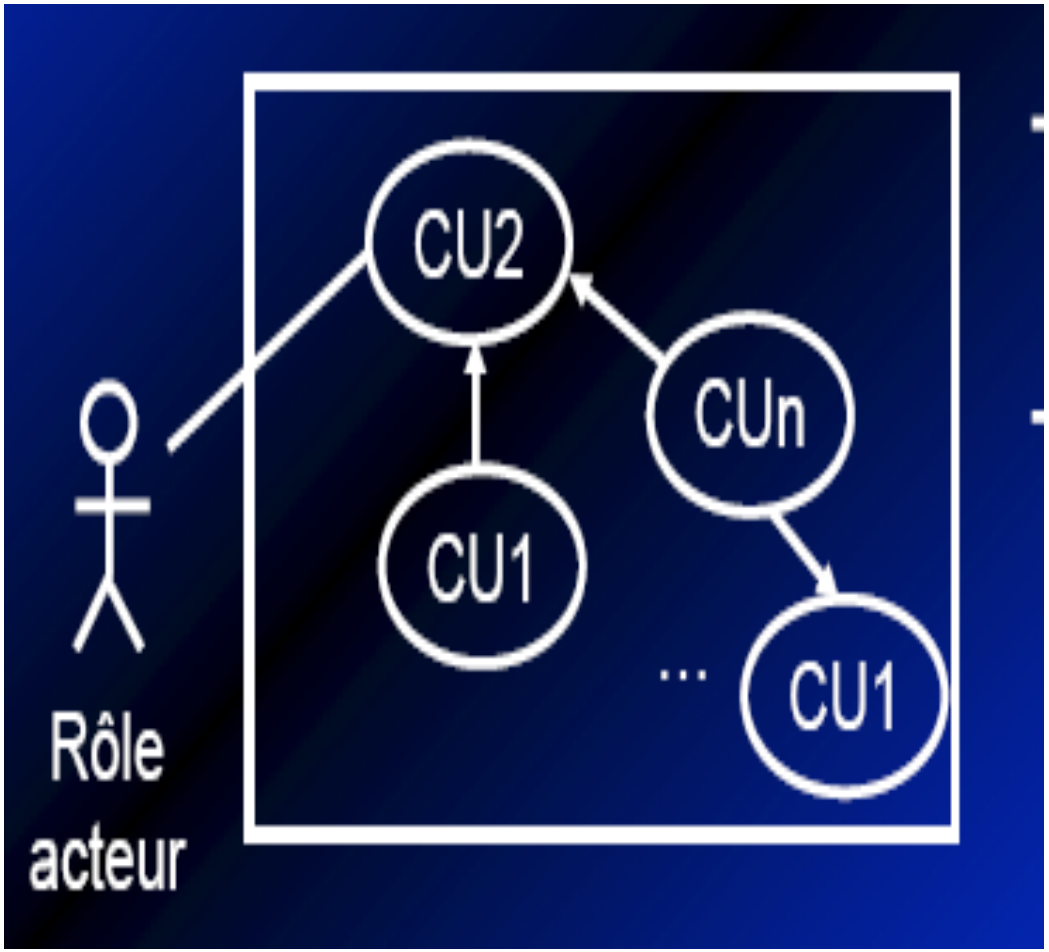
Les Cas d'Utilisation (Use Case : Jacobson 92)

Exemple



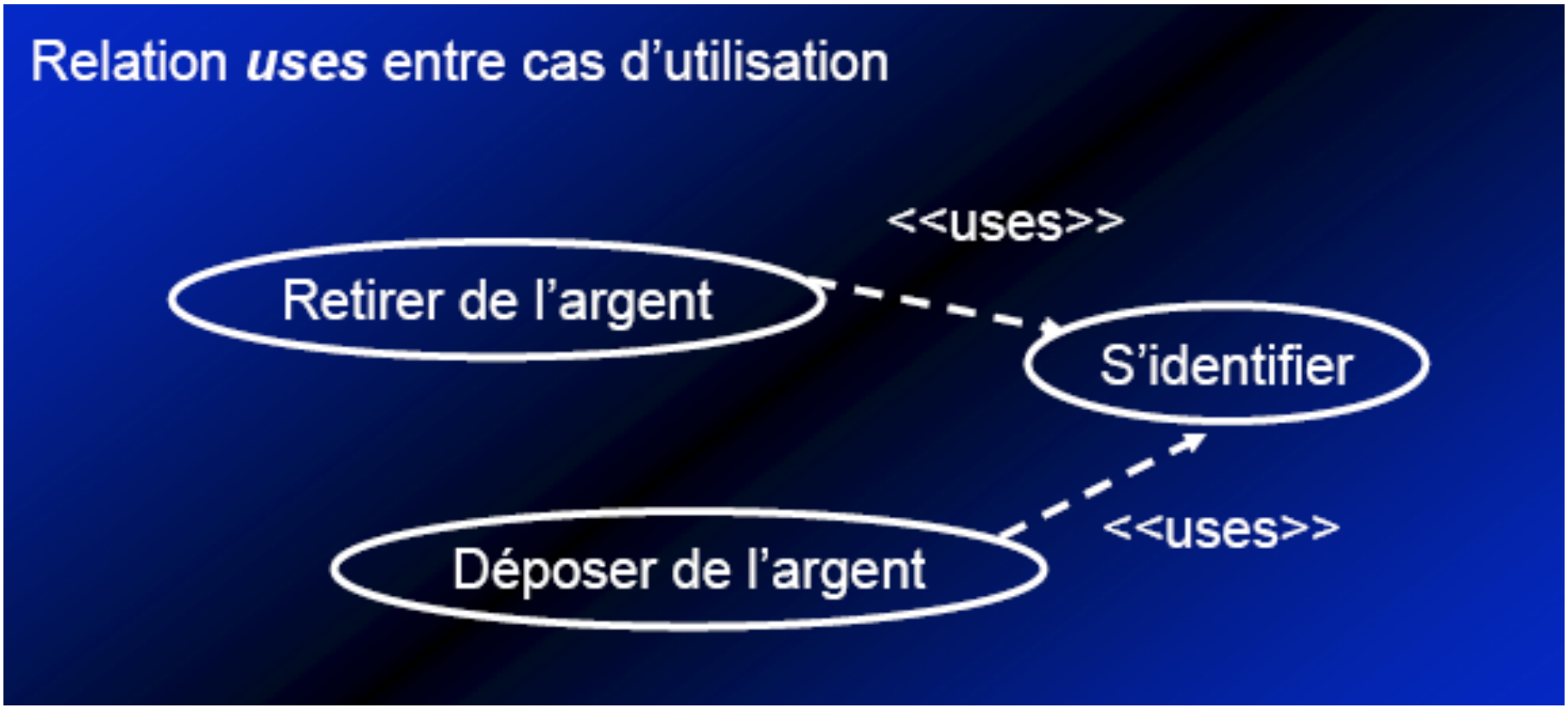
Les Cas d'Utilisation (Use Case : Jacobson 92)

- Relations entre cas d'utilisations
  - Il n'existe pas de communications entre les cas d'utilisation d'un système mais simplement des relations d'utilisation (uses ou include) ou d'extension (extends)
  - Les communications entre acteurs ne sont pas représentées



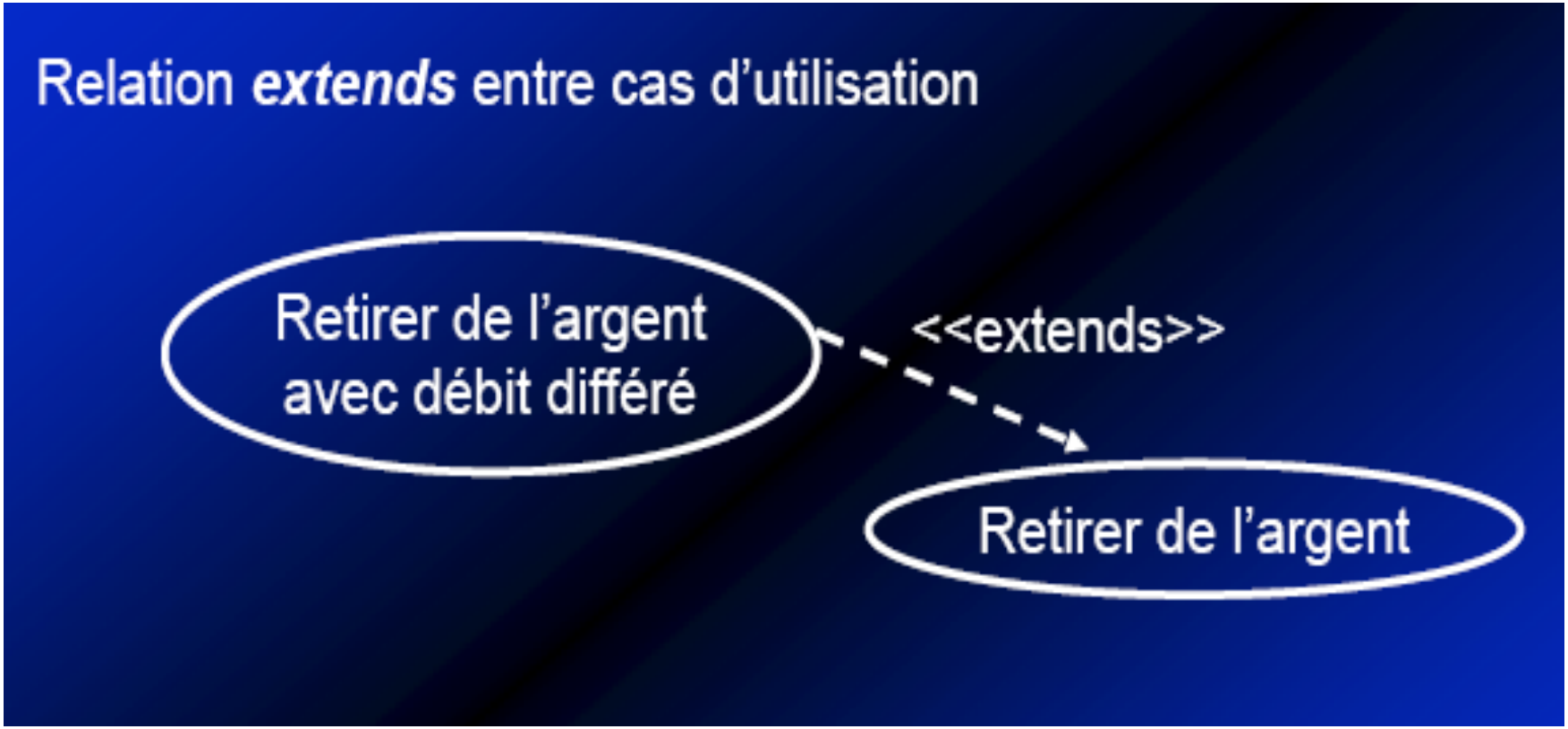
Les Cas d'Utilisation (Use Case : Jacobson 92)

- Relations entre cas d'utilisations



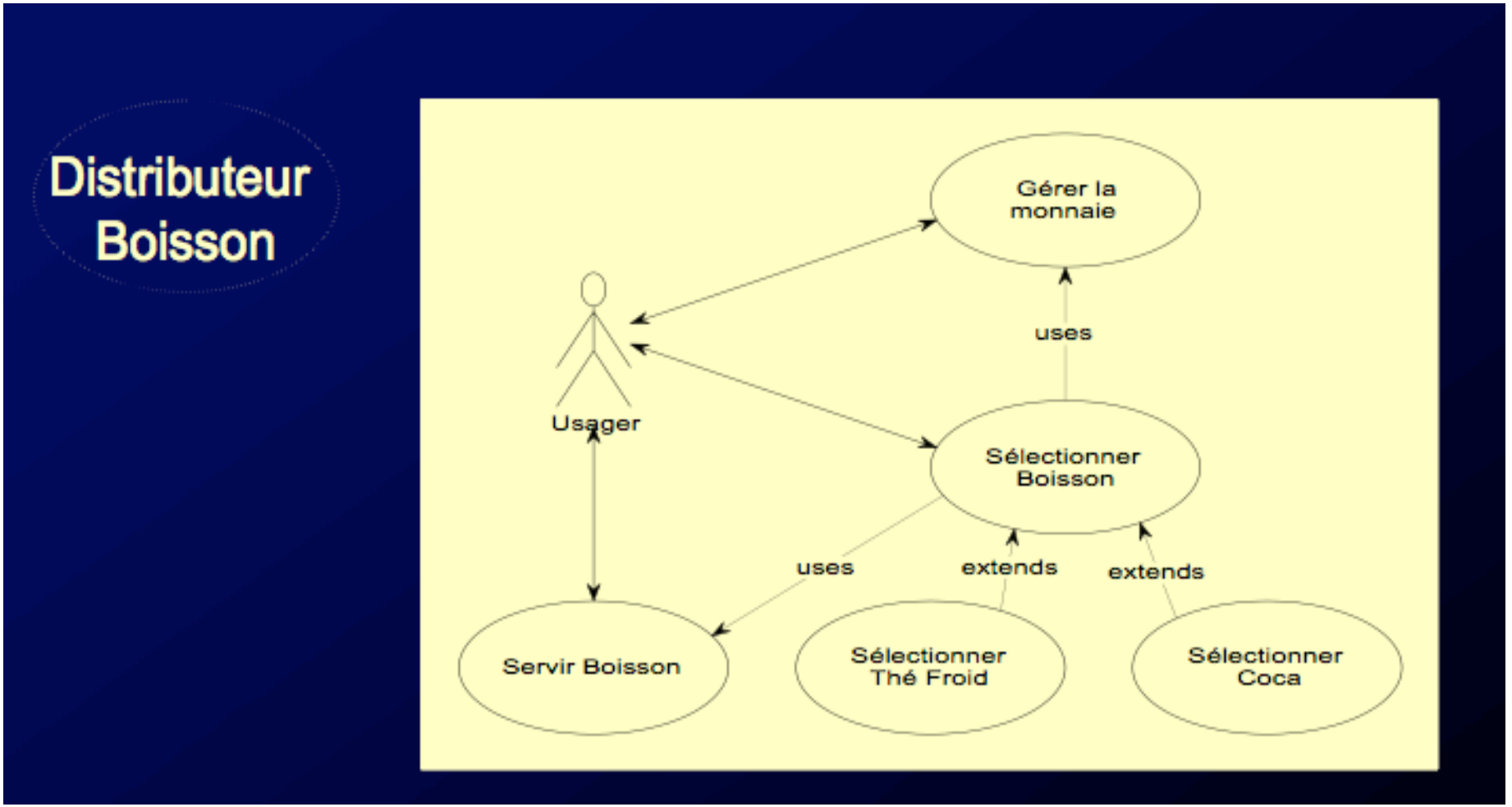
Les Cas d'Utilisation (Use Case : Jacobson 92)

- Relations entre cas d'utilisations



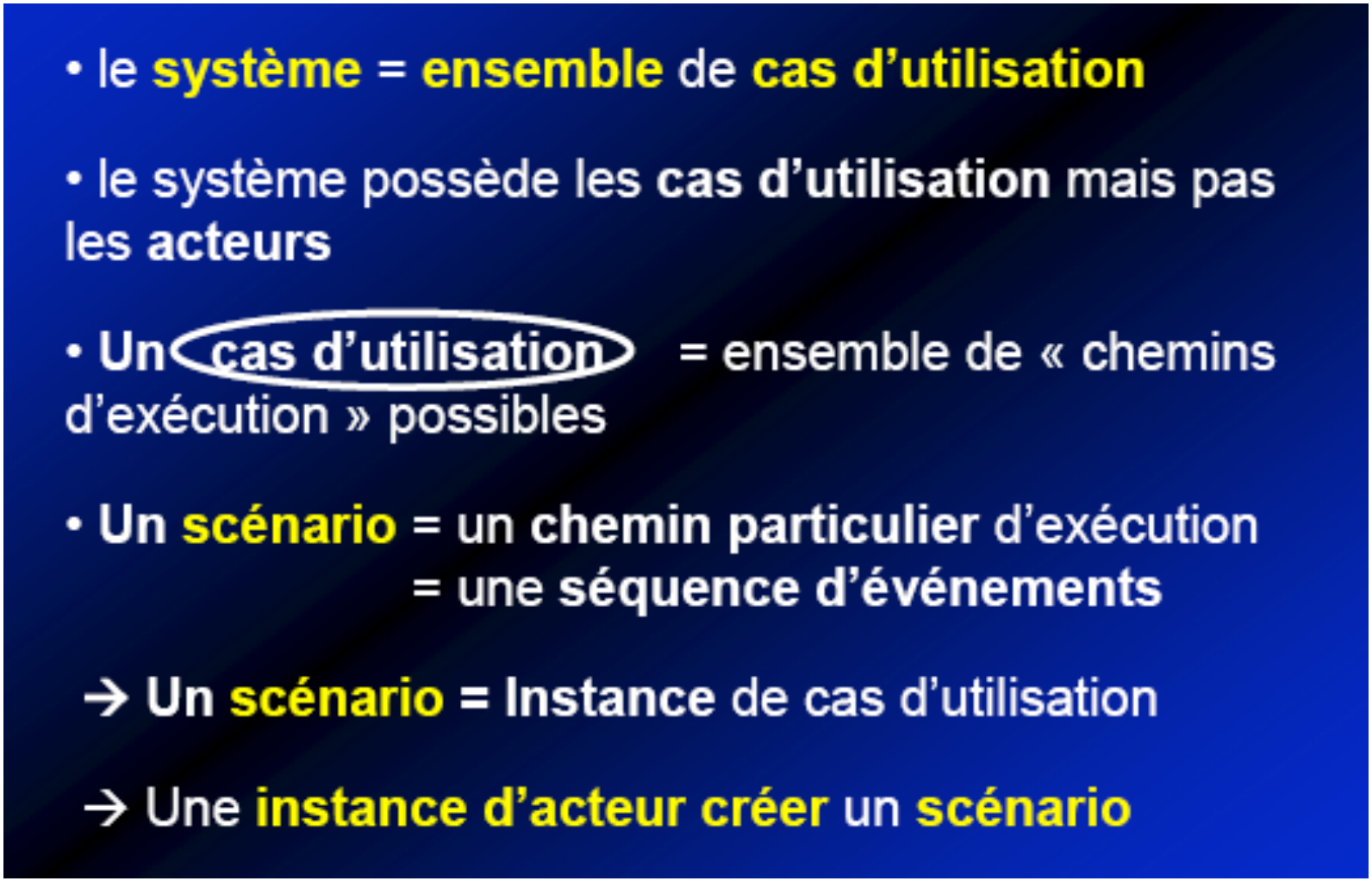
Les Cas d'Utilisation (Use Case : Jacobson 92)

- Relations entre cas d'utilisations



Les Cas d'Utilisation (Use Case : Jacobson 92)

- Cas d'utilisation et scénario



Les Cas d'Utilisation (Use Case : Jacobson 92)

- Cas d'utilisation et scénario

un scénario peut être représenté par diagramme de **séquence** qui décrit un échange particulier entre un ou plusieurs acteurs et le système :

- nature des infos **échangées** entre des **instances** d'acteurs ou d'objets du système
- aspect **temporel** : **flot ordonné d'événements**

un scénario peut également être représenté par un diagramme de **collaboration** (cf. Chapitre IV)

Les Cas d'Utilisation (Use Case : Jacobson 92)

- Cas d'utilisation et scénario

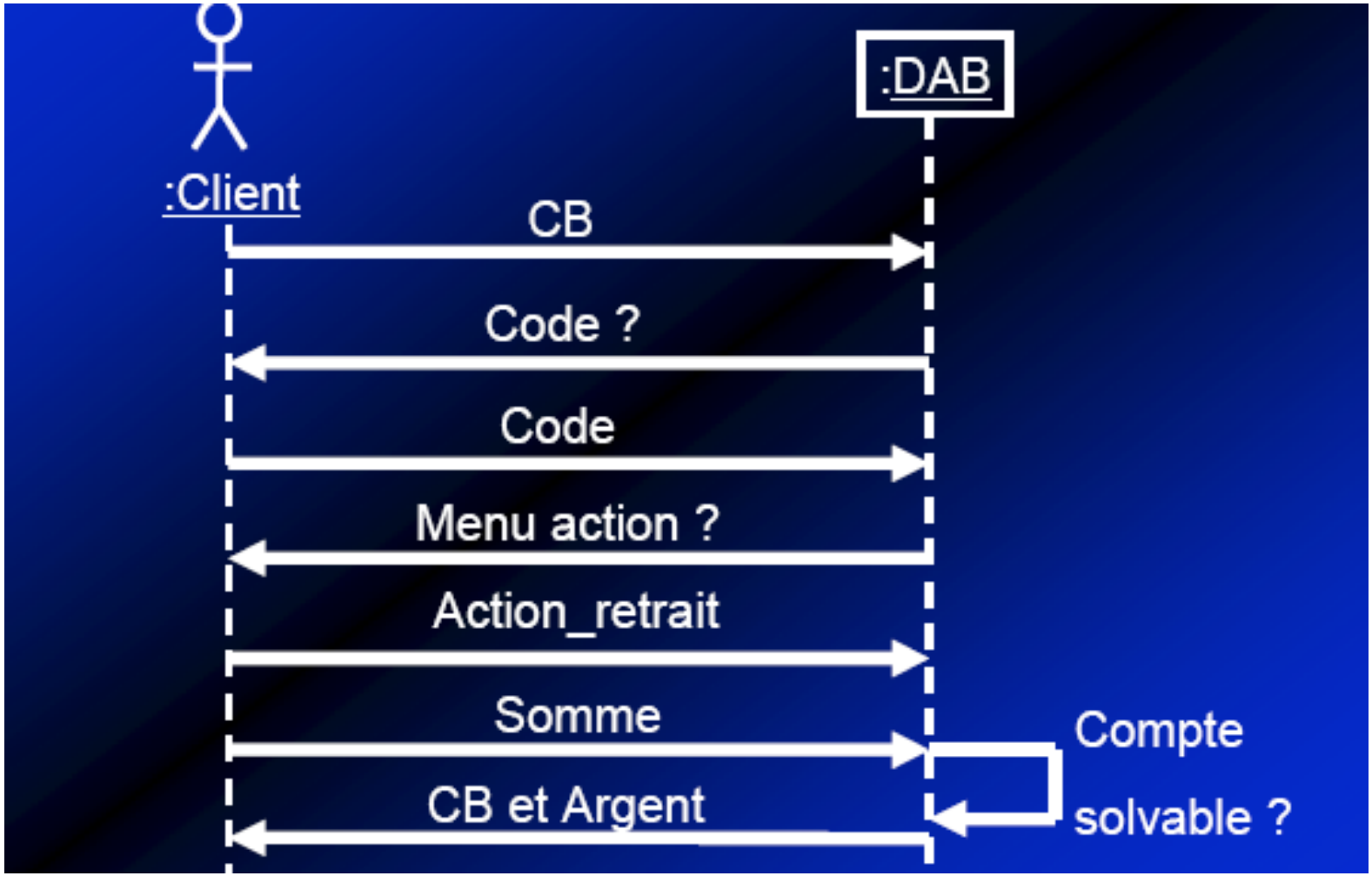
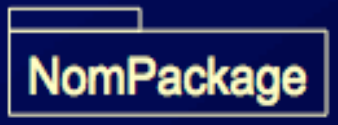


Diagramme de Classes
et
diagramme d’objets

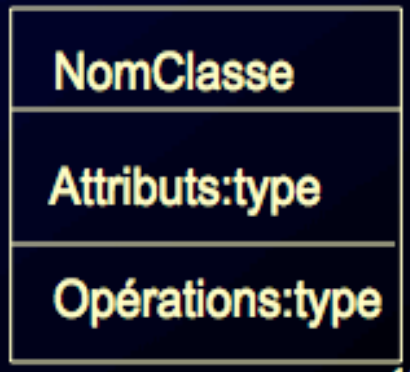
Diagramme de Classes

- Décrivent la structure statique du système à l'aide de classes, packetages, et relations
- Nœuds:
  - Packages



  - Interfaces

  - Classes: Attributs, Opérations (visibilité et paramètres)
    - + public
    - - privé
    - # protégé
    - package

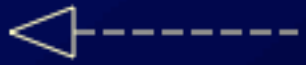


12

Diagramme de Classes

- **Arcs:**
  - dépendance
    - référence à des classes ou objets passés en paramètres ou statiques (« use »)
    - relations entre packetage
  - association
    - navigabilité entre objets, message entre objets
    - flèche est optionnelle
  - agrégation 
    - « has-a »

Diagramme de Classes

- composition
  - « has-a » avec responsabilité sur durée de vie
- généralisation 
  - héritage
- réalisation 
  - implements
  - réalisation d'un use case

- Les liens contiennent la multiplicité d'objets associés

Diagramme de Classes

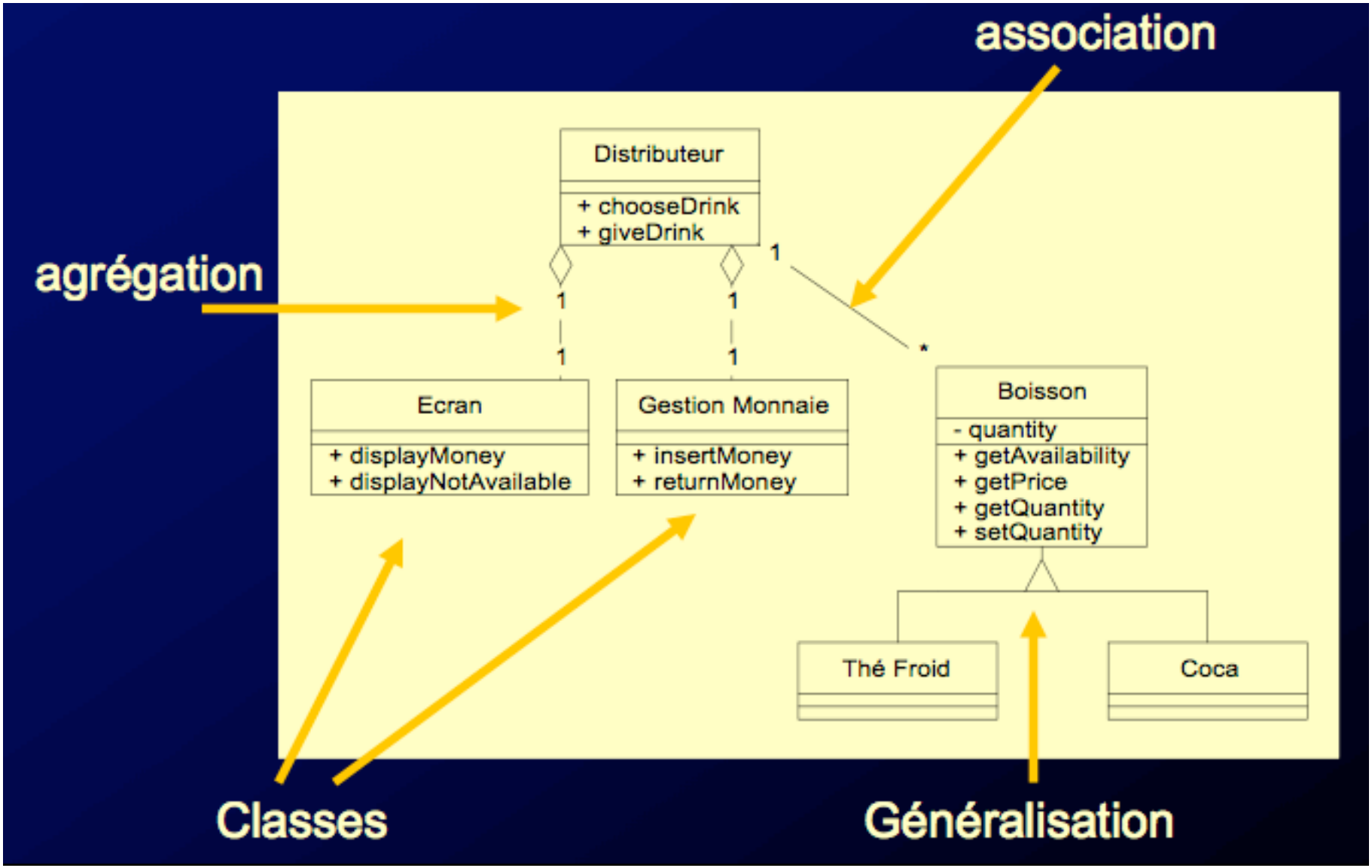


Diagramme de Classes

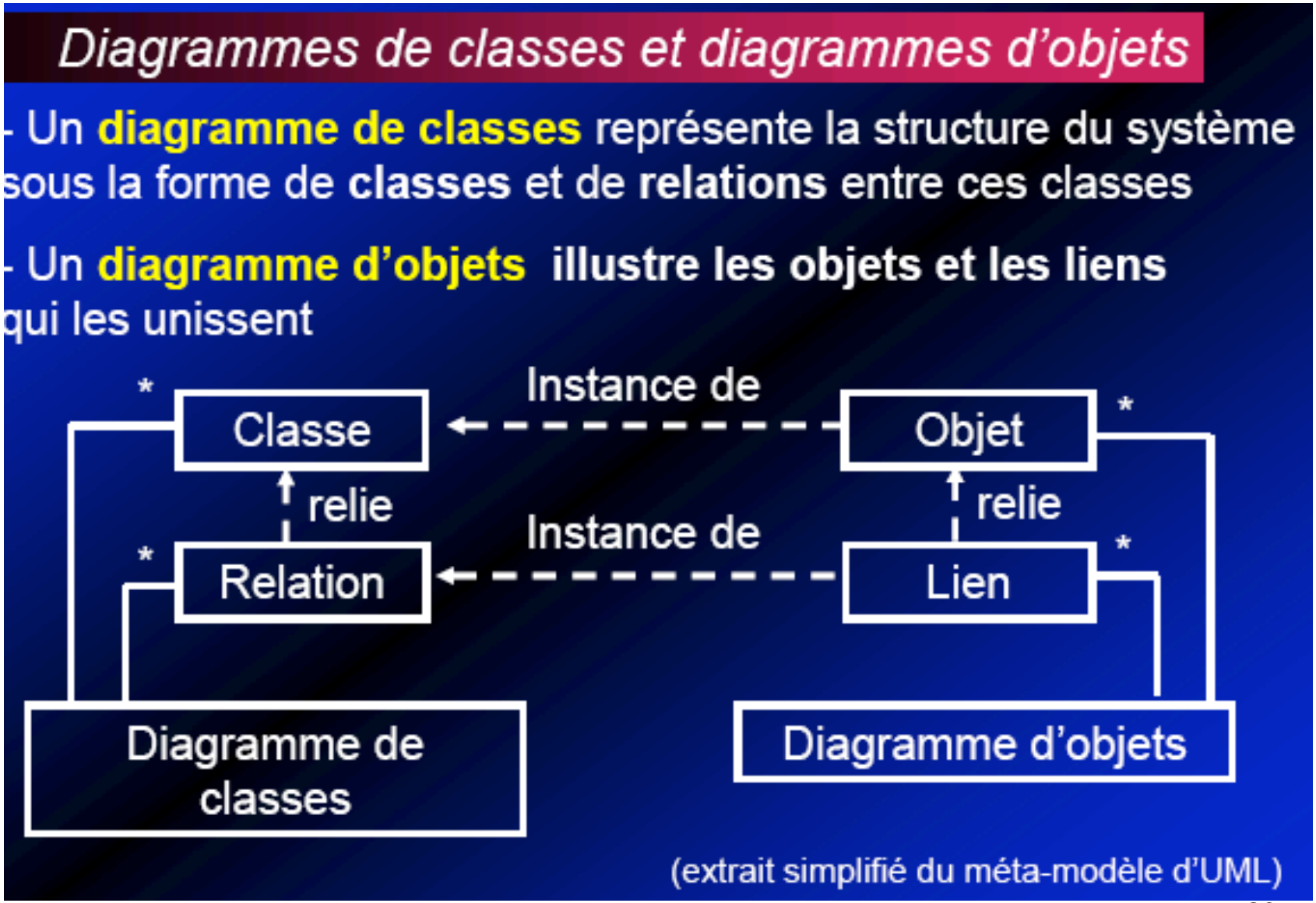


Diagramme de Classes

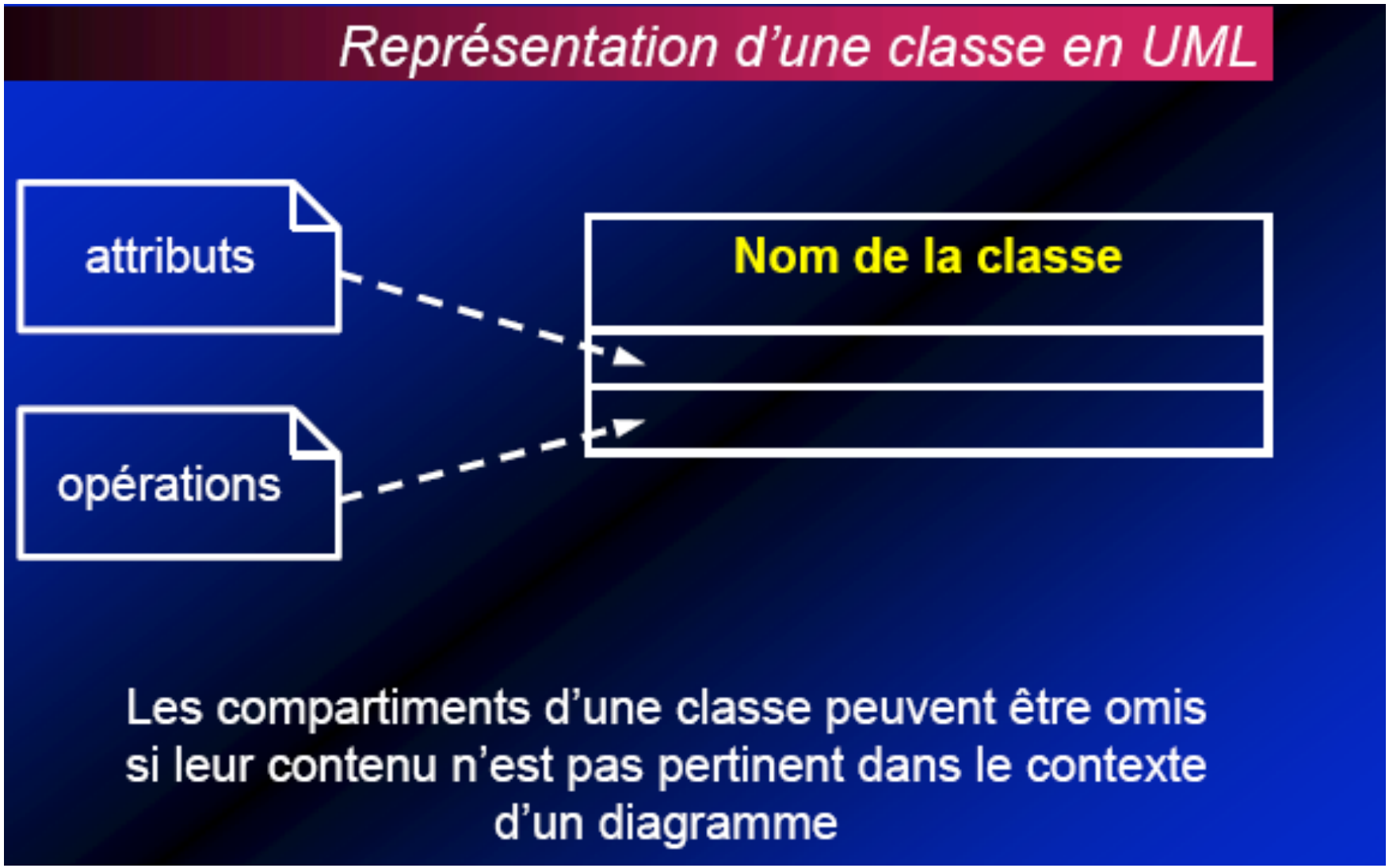
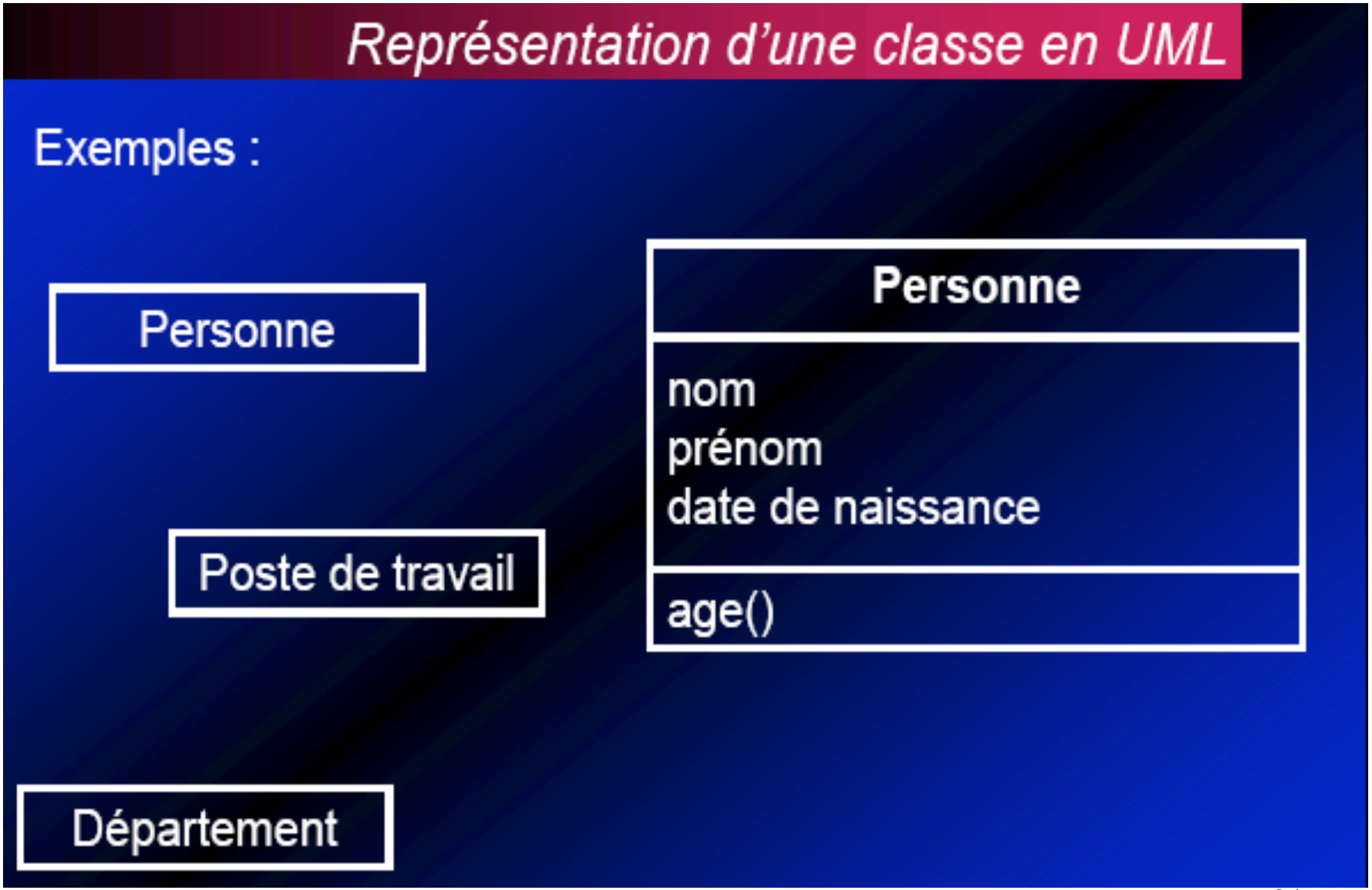
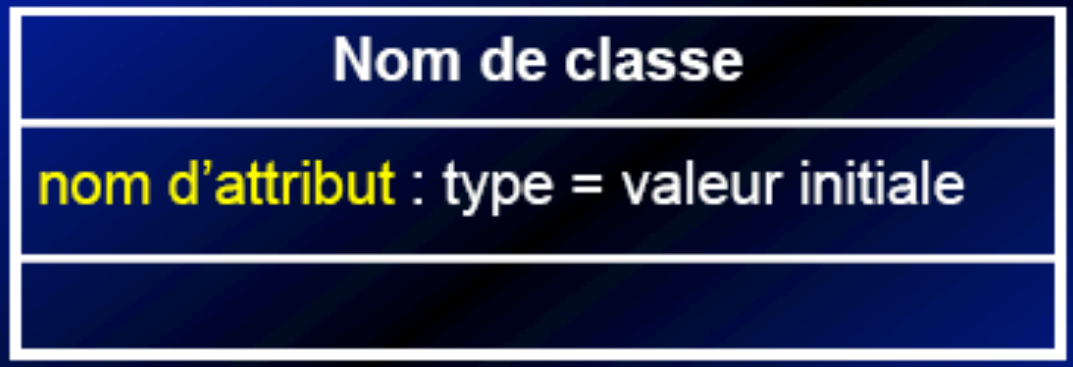


Diagramme de Classes



Attributs de classe

Un attribut de classe définit une propriété commune aux objets d'une classe.



Les noms d'attributs **d'une** classe sont **uniques**.

Chaque objet, instance d'une classe, a sa **propre identité, indépendante** des valeurs de ses attributs.

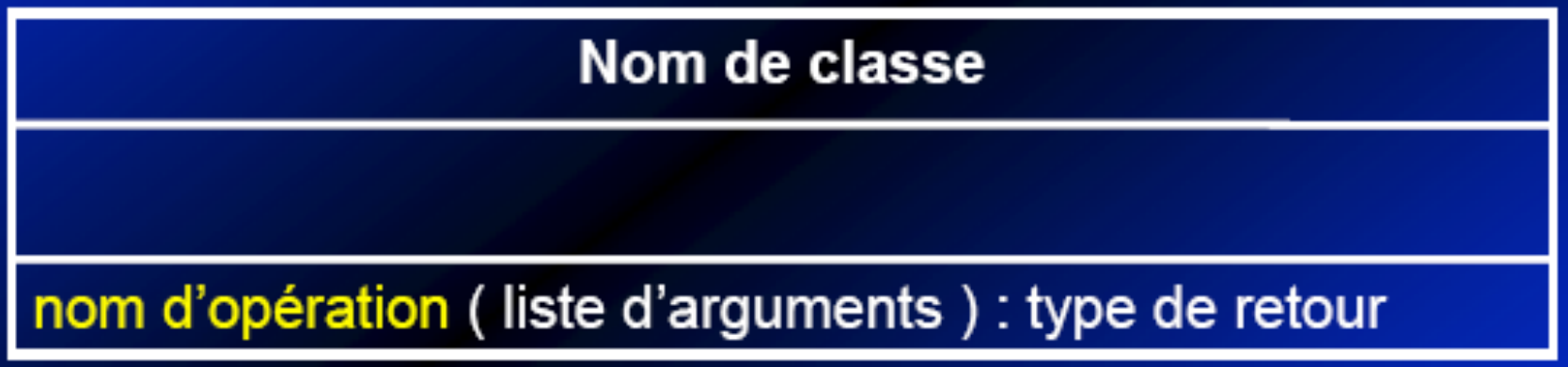
L'identification d'un objet est donc facultative.

Diagramme de Classes

Opération de classe

Exemples :

Une opération définit une fonction appliquée à des objets d'une classe :



Propriétés des attributs et des opérations :

Accessibilité aux attributs et opérations d'une classe

Trois **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

Diagramme de Classes

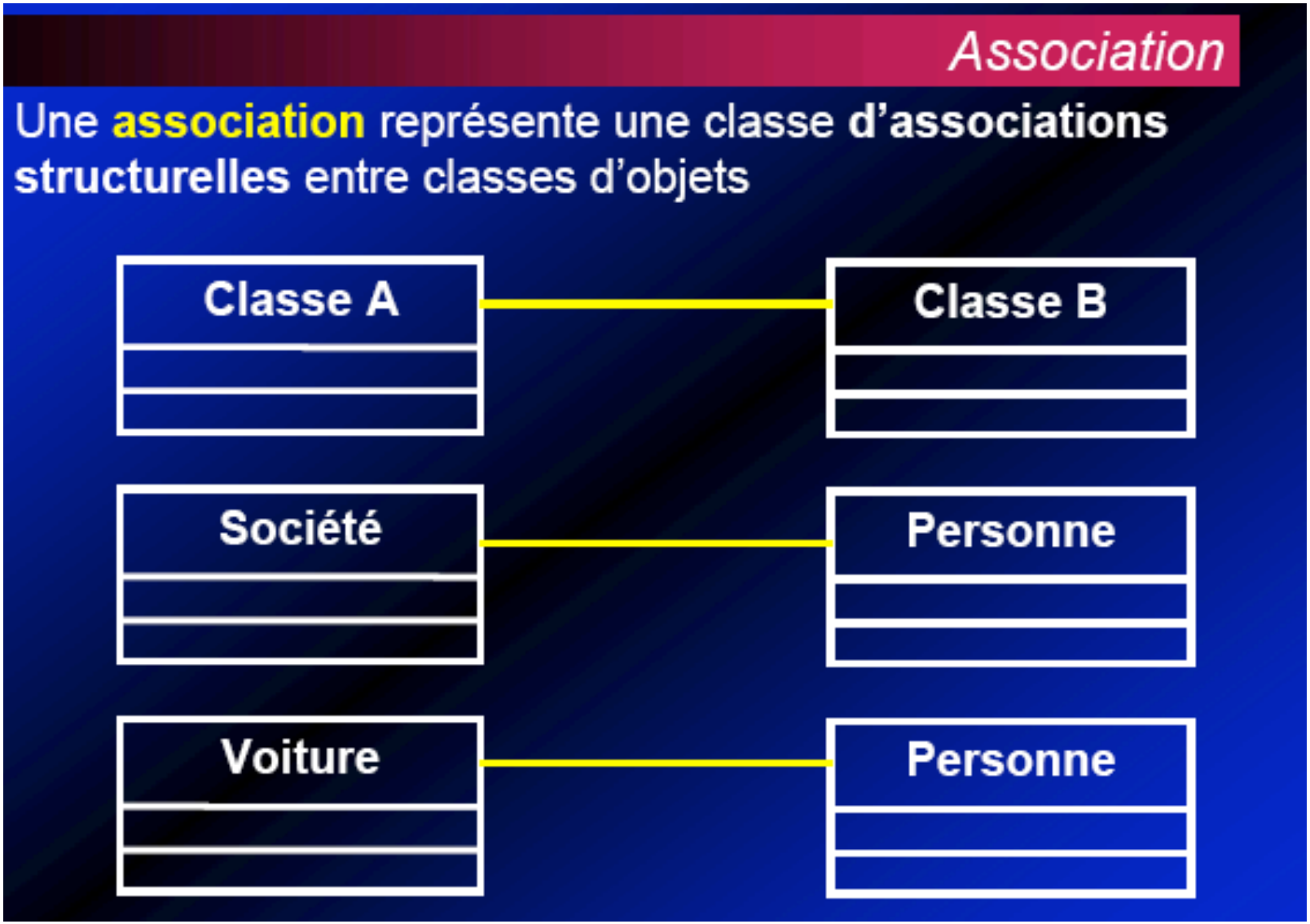
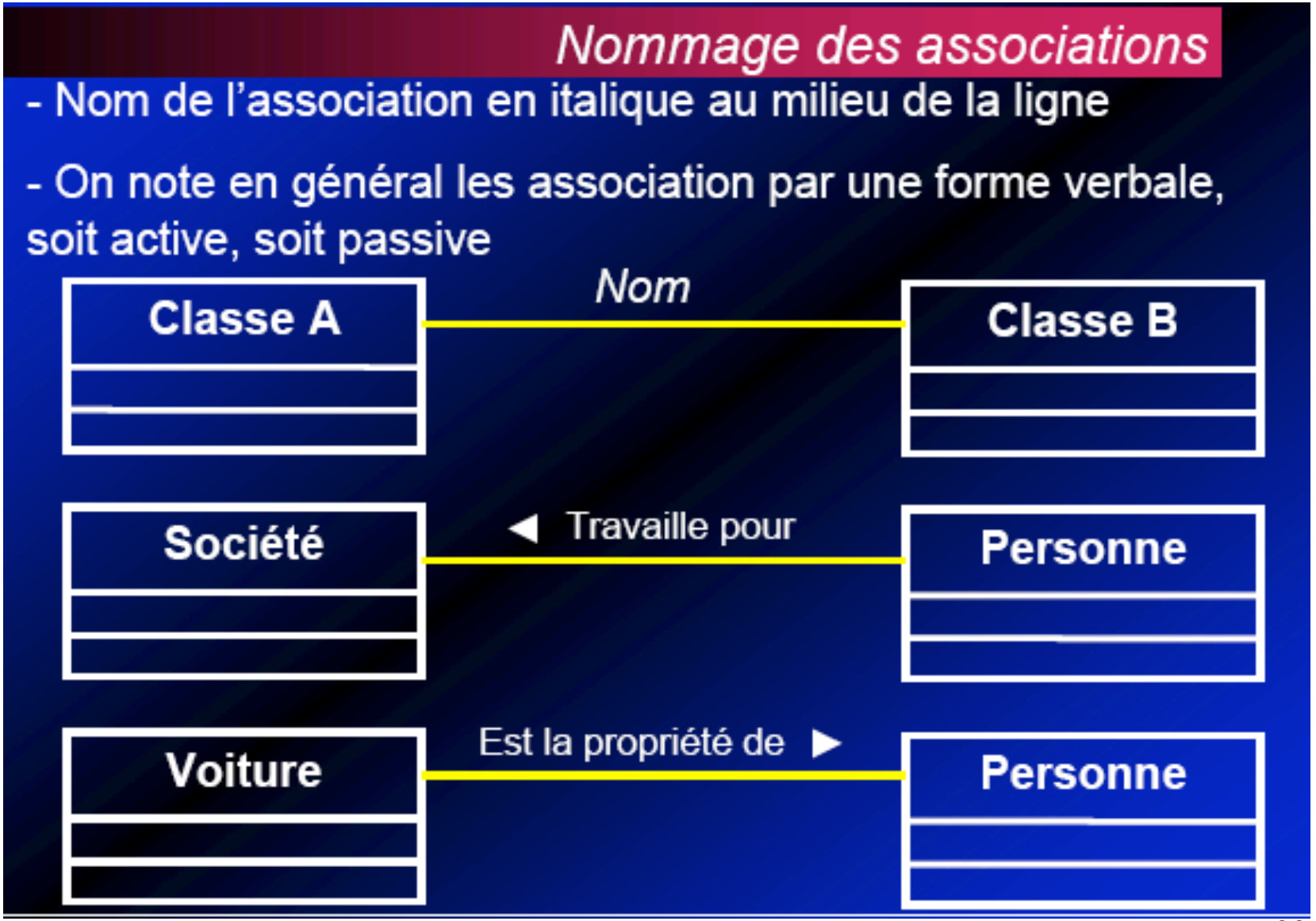


Diagramme de Classes

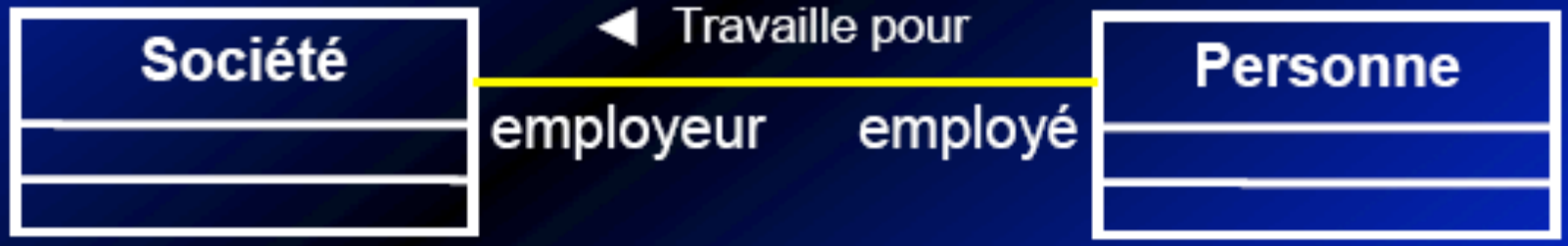


Nommage des rôles

- Toute association binaire possède 2 rôles

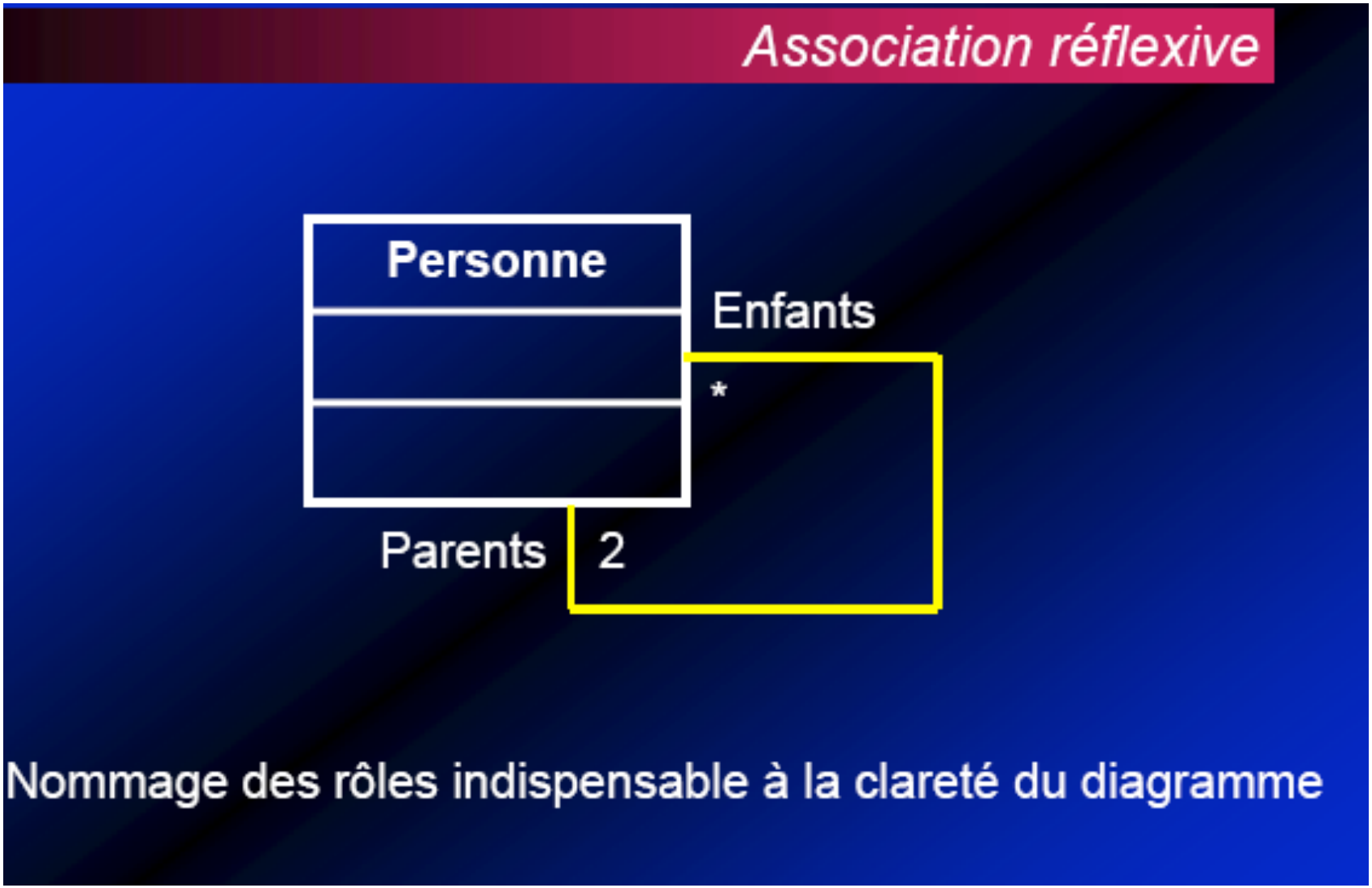
- un rôle définit la manière dont une classe intervient dans une relation

- Le nommage des associations et le nommage des rôles ne sont pas exclusifs l'un de l'autre



- Intérêt des rôles dans le cas où plusieurs associations lient deux classes : distinction des concepts attachés aux associations

Diagramme de Classes

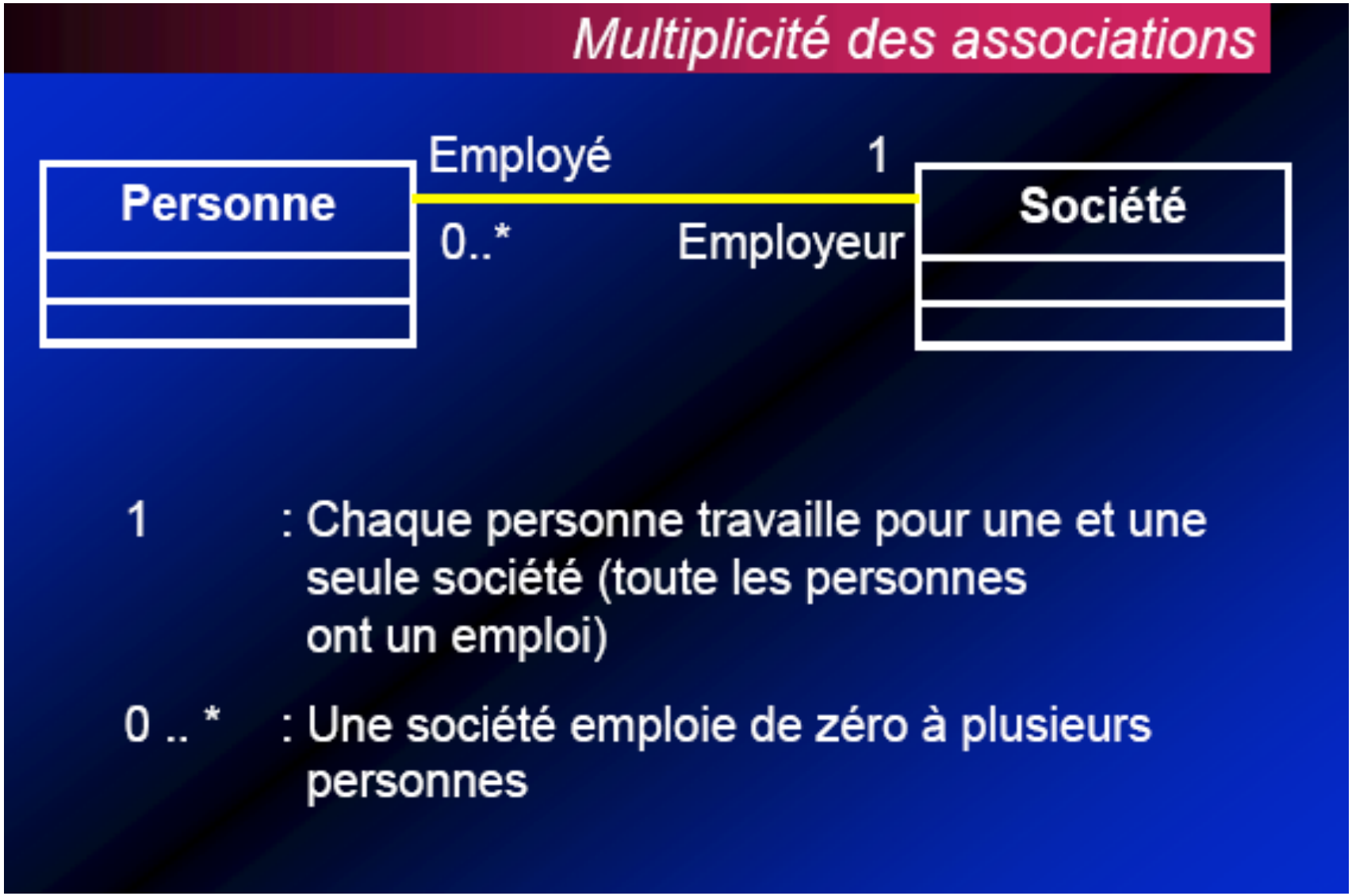


Multiplicité des associations

La **multiplicité** est une **information portée** par le **rôle**, qui **quantifie** le **nombre de fois** où un **objet participe** à une **instance** de **relation**

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 naturel)

Diagramme de Classes

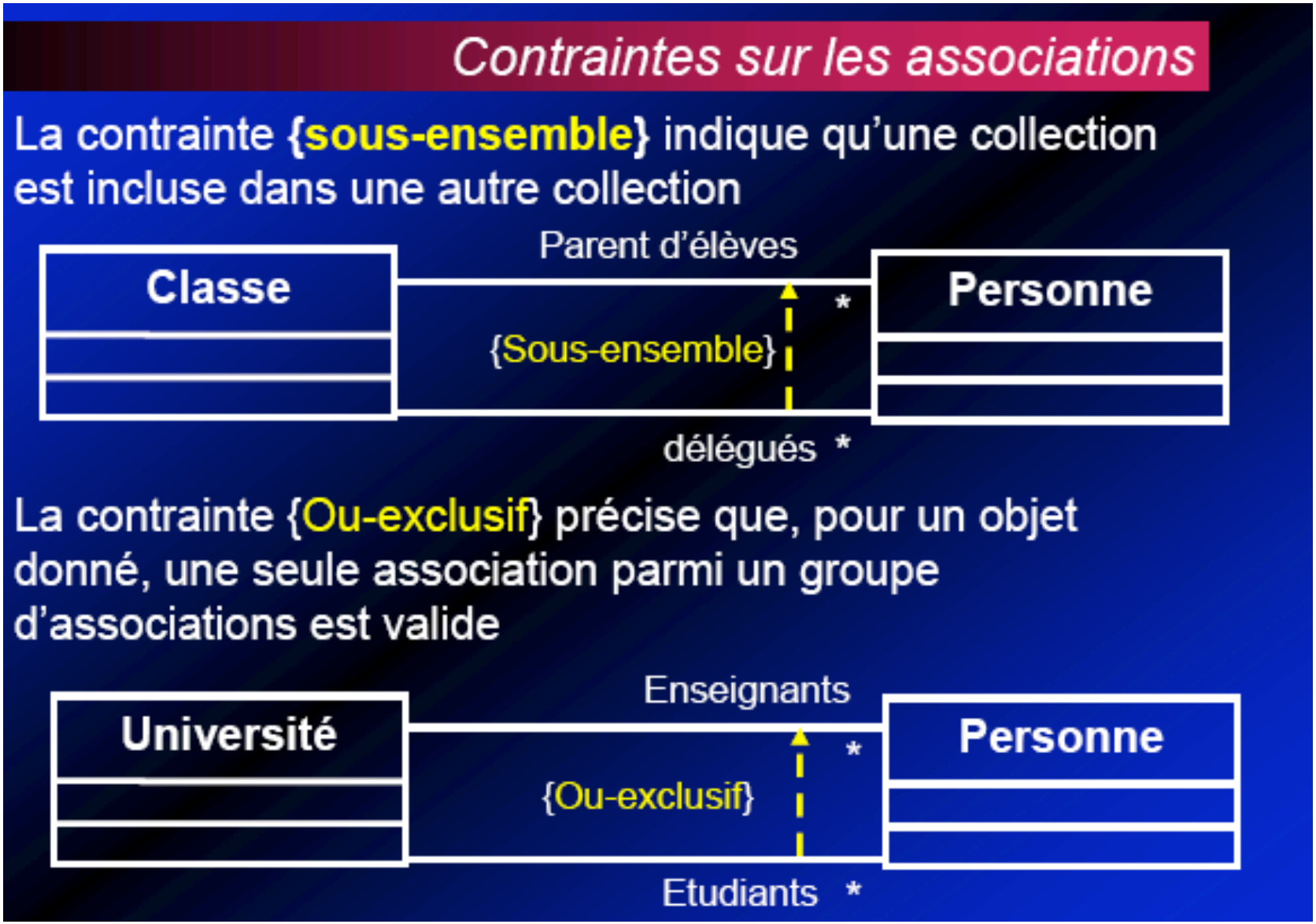


Contraintes sur les associations

Contrainte d'association : porte sur une relation ou sur un groupe de relations (notée {contrainte })

Par exemple, placée sur un rôle, la contrainte **{ordonnée}** définit une relation d'ordre entre les objets de la collection (les comptes) qui sont liés à une personne

Diagramme de Classes



Restriction des associations

- La **restriction** (dite **qualification** en UML) d'une association consiste à sélectionner un sous-ensemble d'objets parmi l'ensemble des objets qui participent à une association

- réalisée au moyen d'une clé, ensemble d'attributs particuliers. La clé appartient à l'association et non aux classes associées

Restriction des associations

• Chaque instance de la classe A accompagnée de la valeur de la clé, identifie un sous ensemble des instances de B qui participent à l'association

Association particulière : Agrégation

Une **agrégation** est une **association *non symétrique*** : **l'une** des **extrémités** joue un rôle **prédominant** par rapport **à l'autre**

Elle se justifie dans les cas suivants :

- Une classe B « **fait partie** » **intégrante** d'une classe A
- Les **valeurs d'attributs** de la classe B se **propagent dans les valeurs d'attributs** de la classe B
- Une **action** sur la classa A implique une **action** sur la classe B
- Les **objets** de la classe B sont **subordonnés** aux **objets** de la **classe** A

(la présence d'une agrégation n'implique obligatoirement tous ces critères)

Association particulière : Agrégation

L'agrégation peut être **multiple**
comme une **association classique :**

En tant que « **propriétaire** », une **personne** est un **agrégat d'immeubles** …

Les **immeubles** dont elle est **propriétaire** font partie de la **description d'une personne**

Agrégation particulière : Composition

La **composition** est une forme particulière **d'agrégation**

Le **composant** est « **physiquement** » **contenu** dans **l'agrégat**

La **composition** implique une **contrainte** sur la valeur de la **multiplicité** du coté de l'agrégat : (0 ou 1)

La **valeur 0** du coté de **l'agrégat** implique un **attribut non renseigné**

Diagramme de Classes

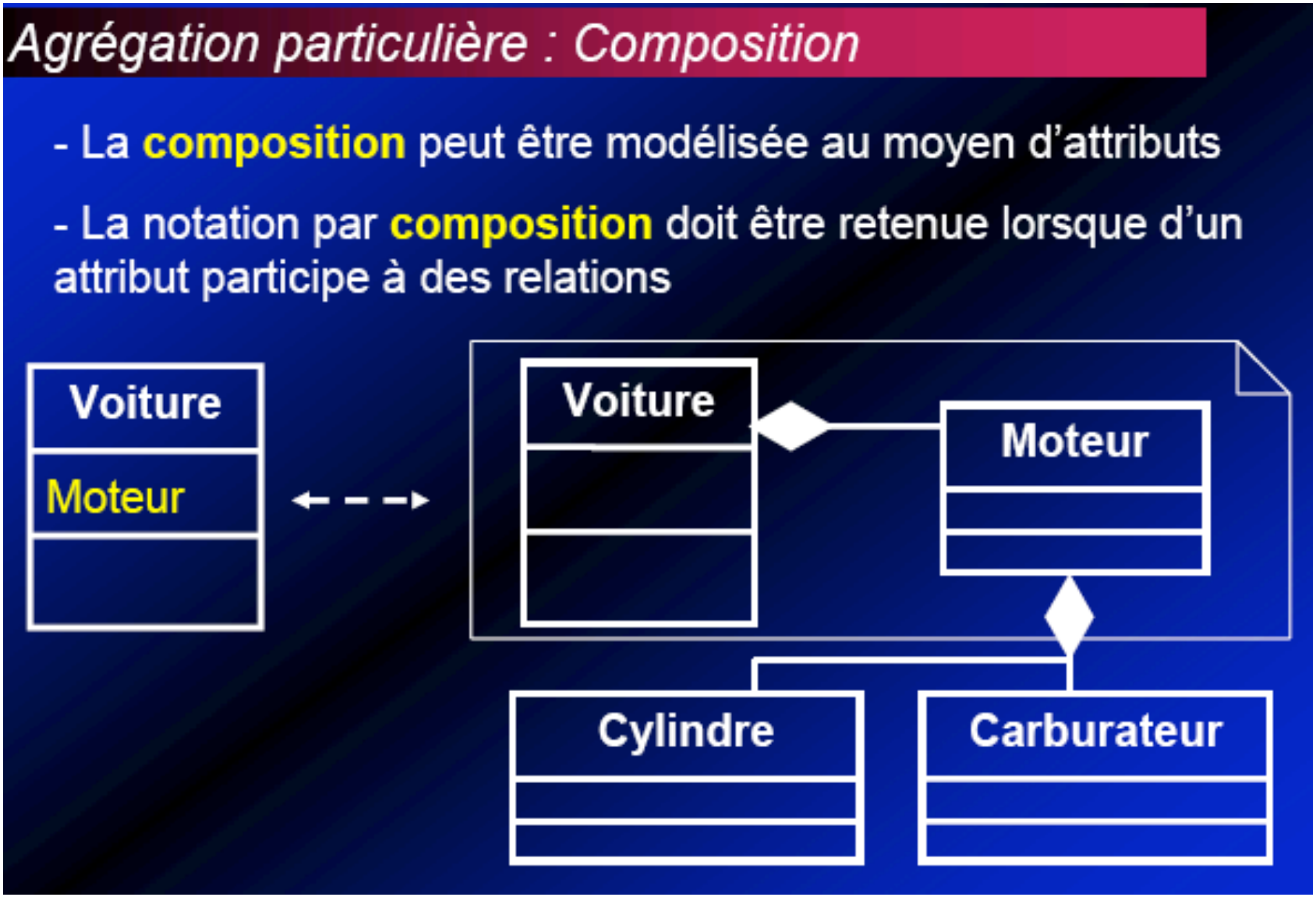


Diagramme de Classes

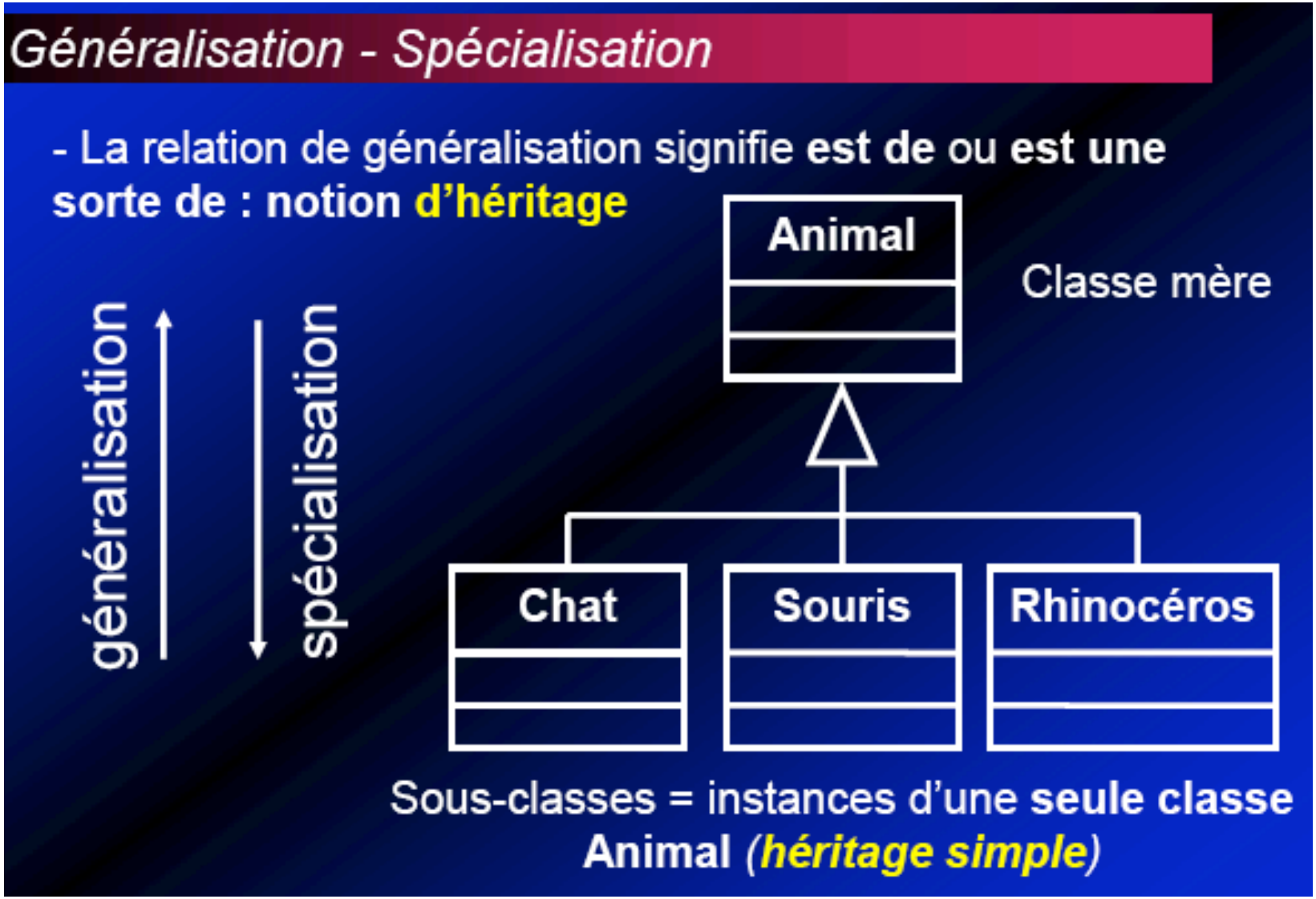


Diagramme de Classes

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**

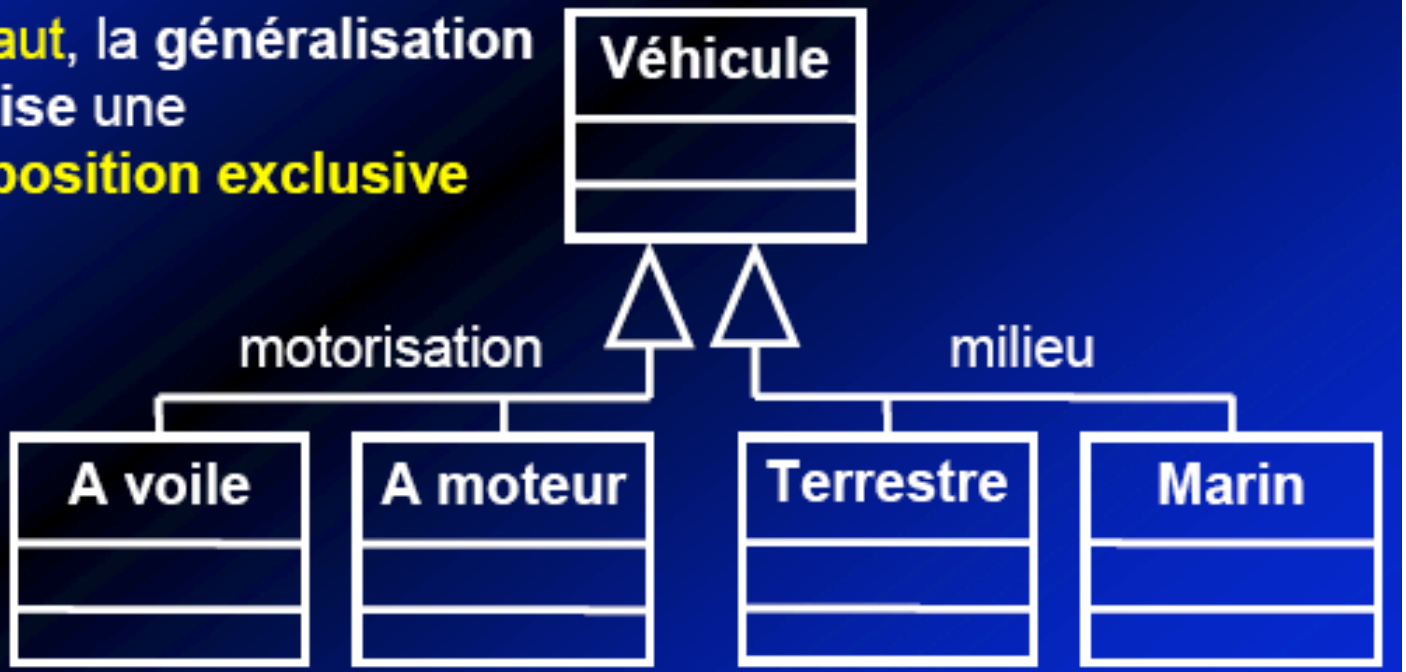


Diagramme d'objets

- Exemples de class diagrams avec des instances d'objets
- Nœuds: objets

- Arcs: relations entre objets, instances d'associations
- Respecter les multiplicités d'objets associés aux arcs (dans le class diagram)

Diagramme d'objets

- Représente les **liens structurel entre instances** de classes
- Facilite la **compréhension** de **structures complexes**
- Trois **représentations** possibles des **instances** :

Diagramme d'objets

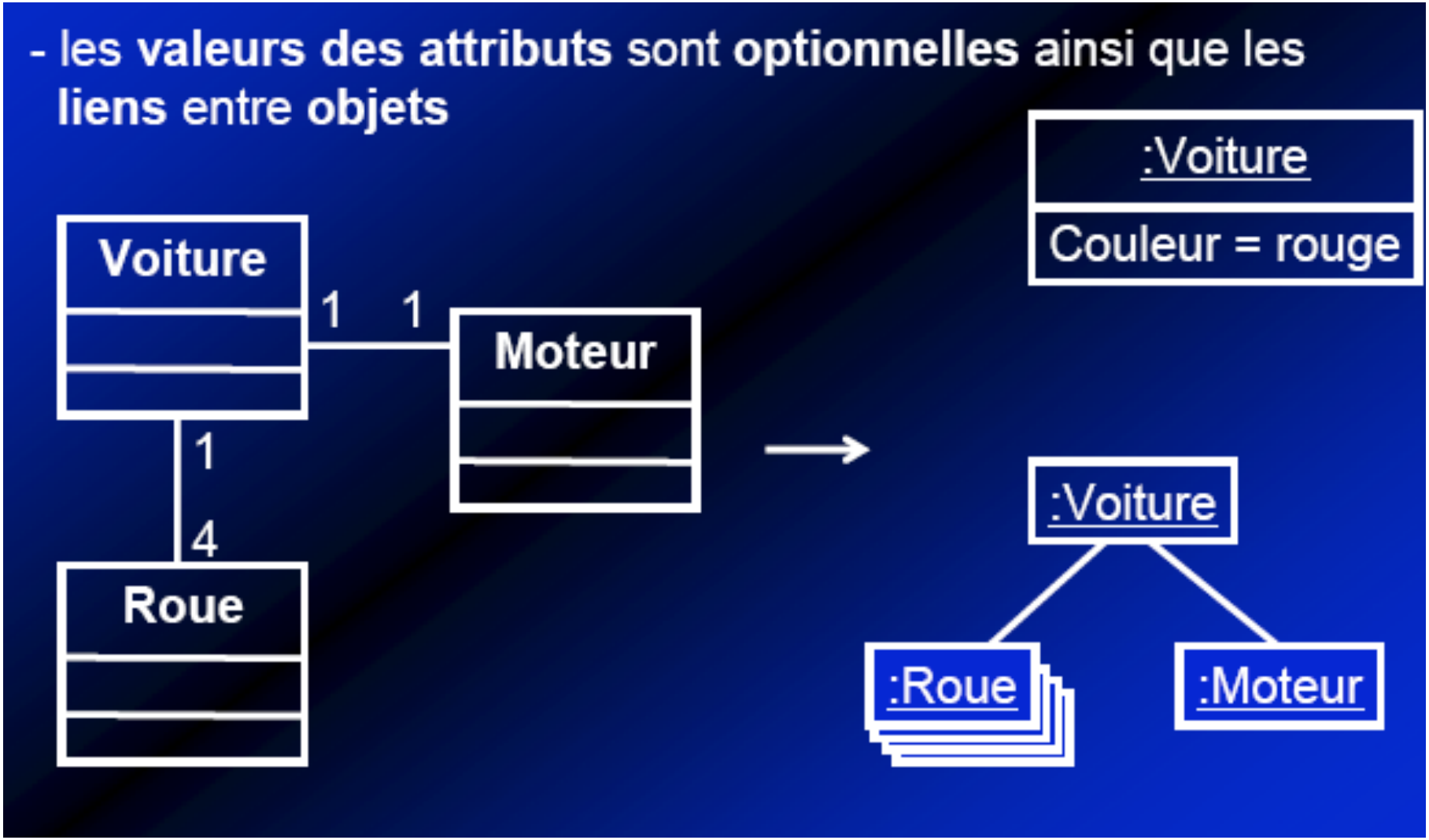


Diagramme d'objets

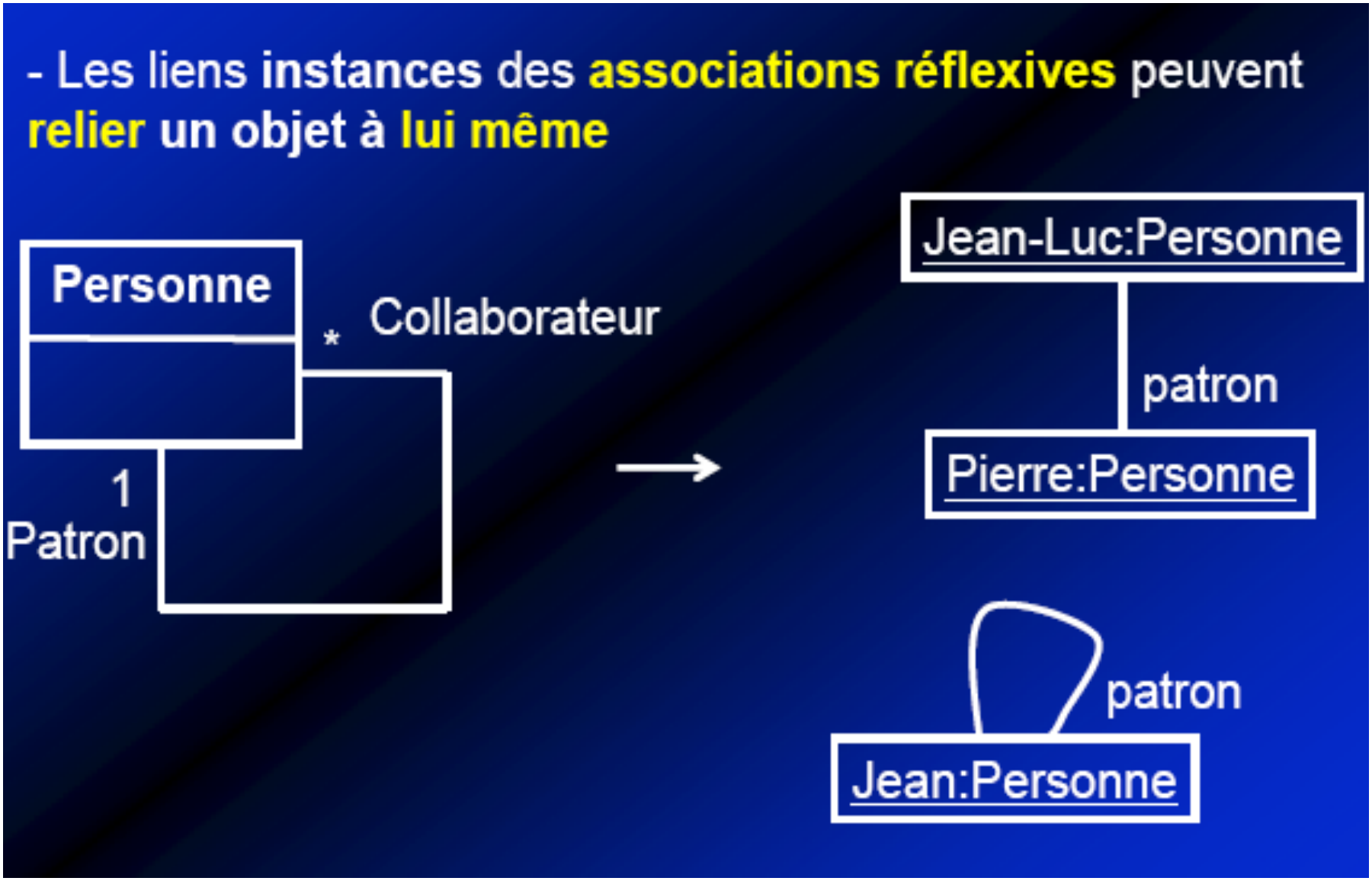


Diagramme d'objets

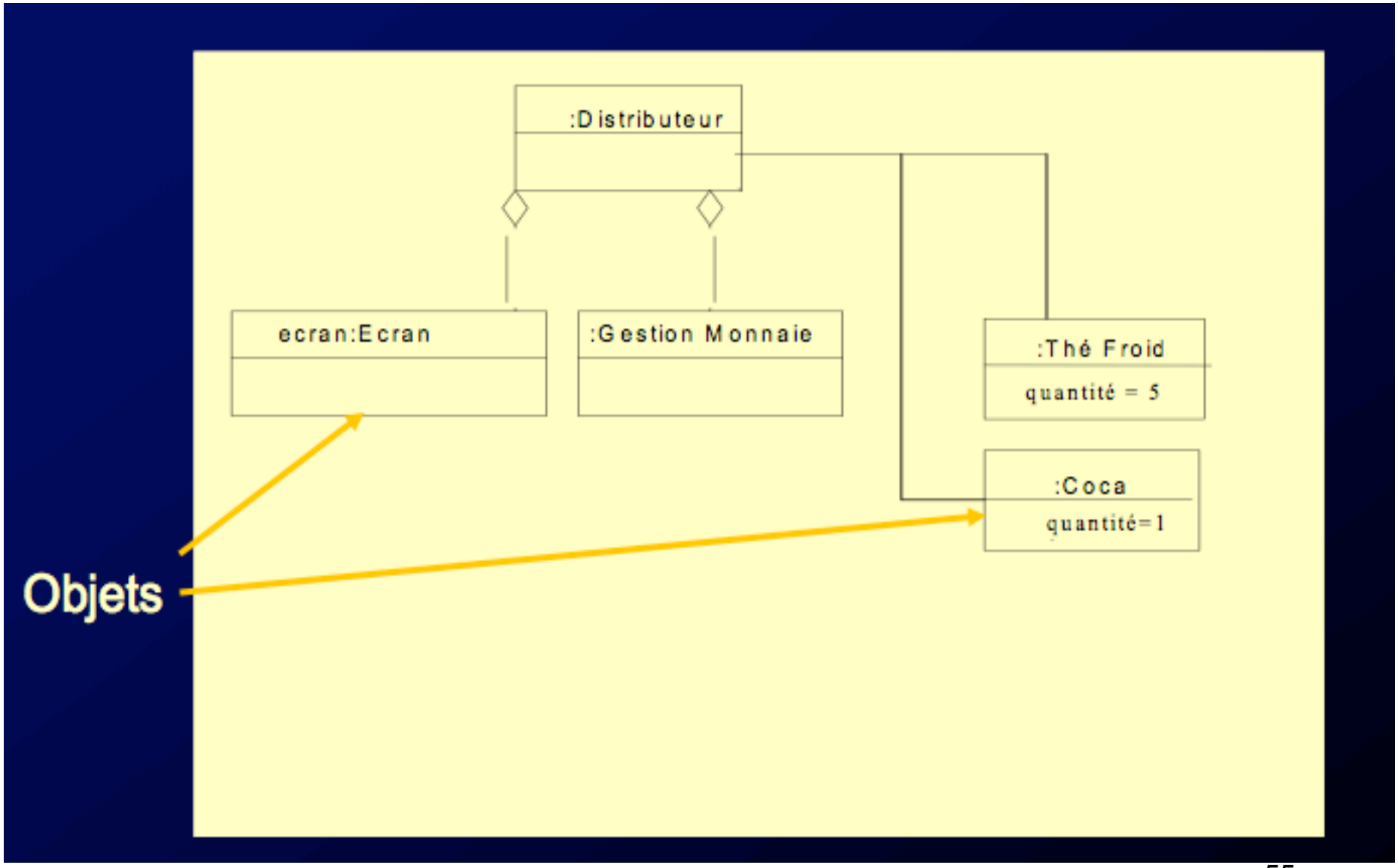


Diagramme de séquence

Diagramme de séquence

- Etablissent le lien entre *Use Case* et *Class Diagrams*
- Décrivent l'échange de messages entre classes
- Class rôles
  - objets participant à l'interaction

- Lignes de temps
- Messages activant des opérations chez l'objet receveur
  - représente la communication entre objets

- Scénario: cas particulier de séquence

Diagramme de séquence

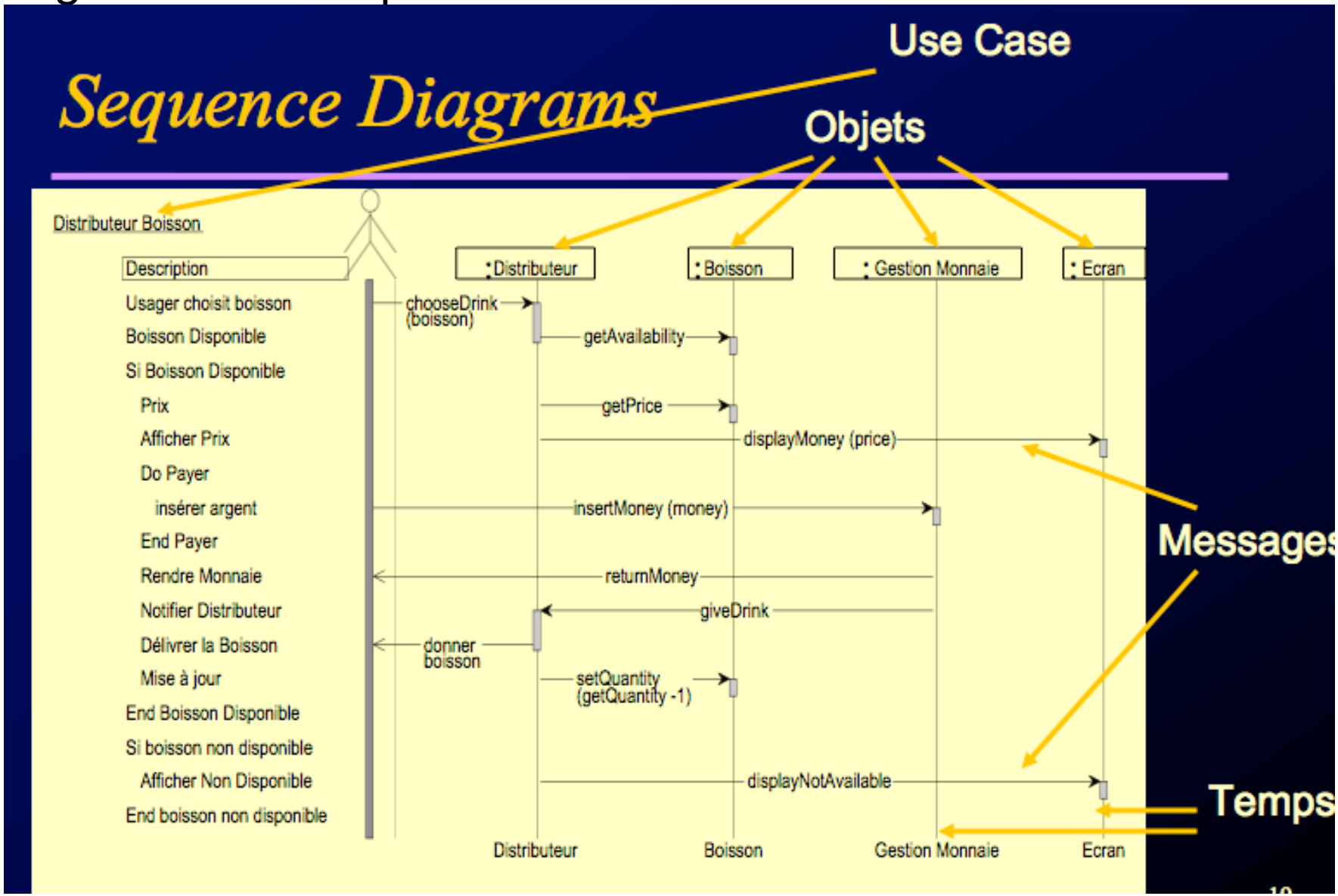
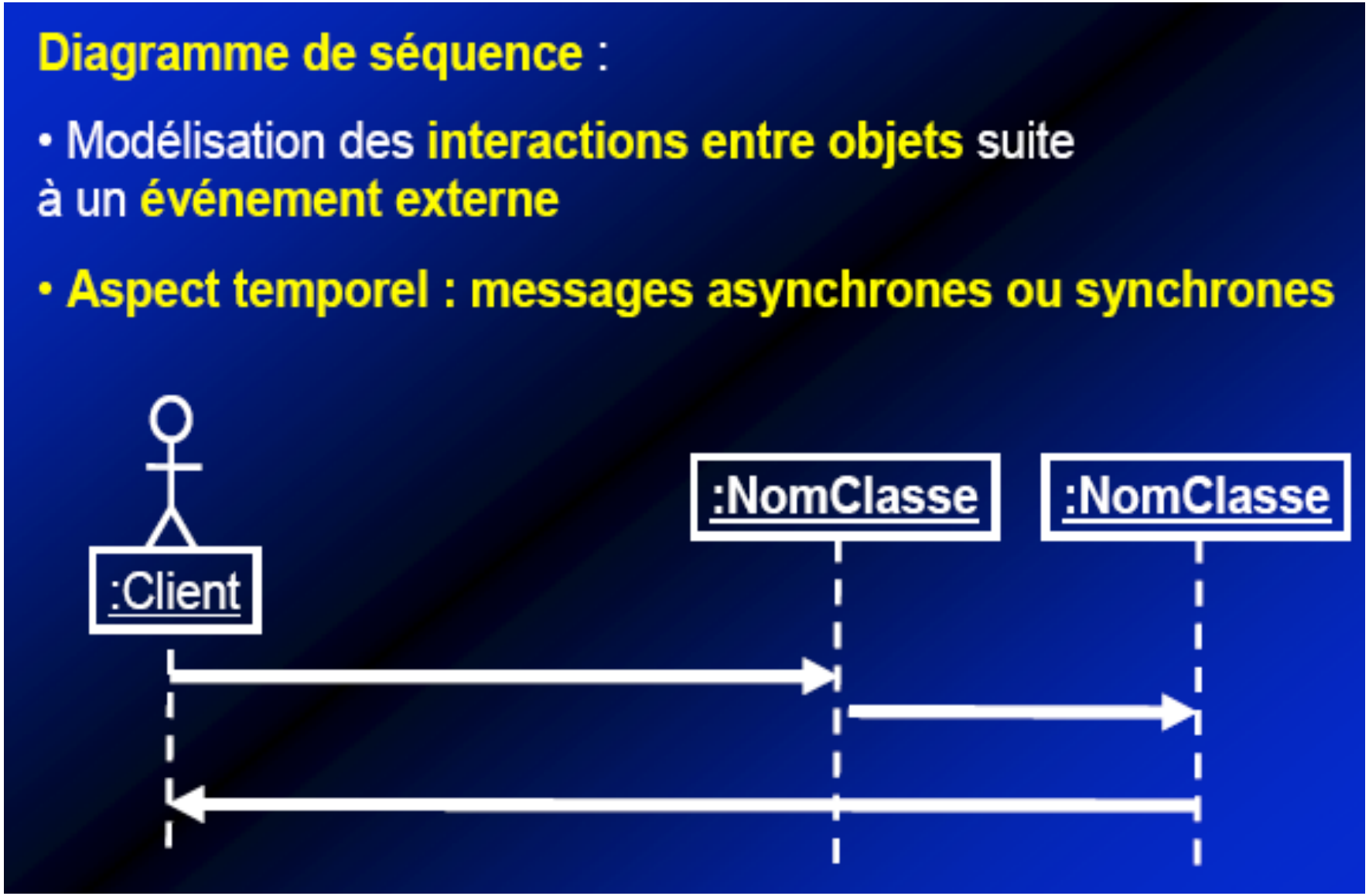


Diagramme de séquence

Diagramme de séquence



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

Diagramme de séquence

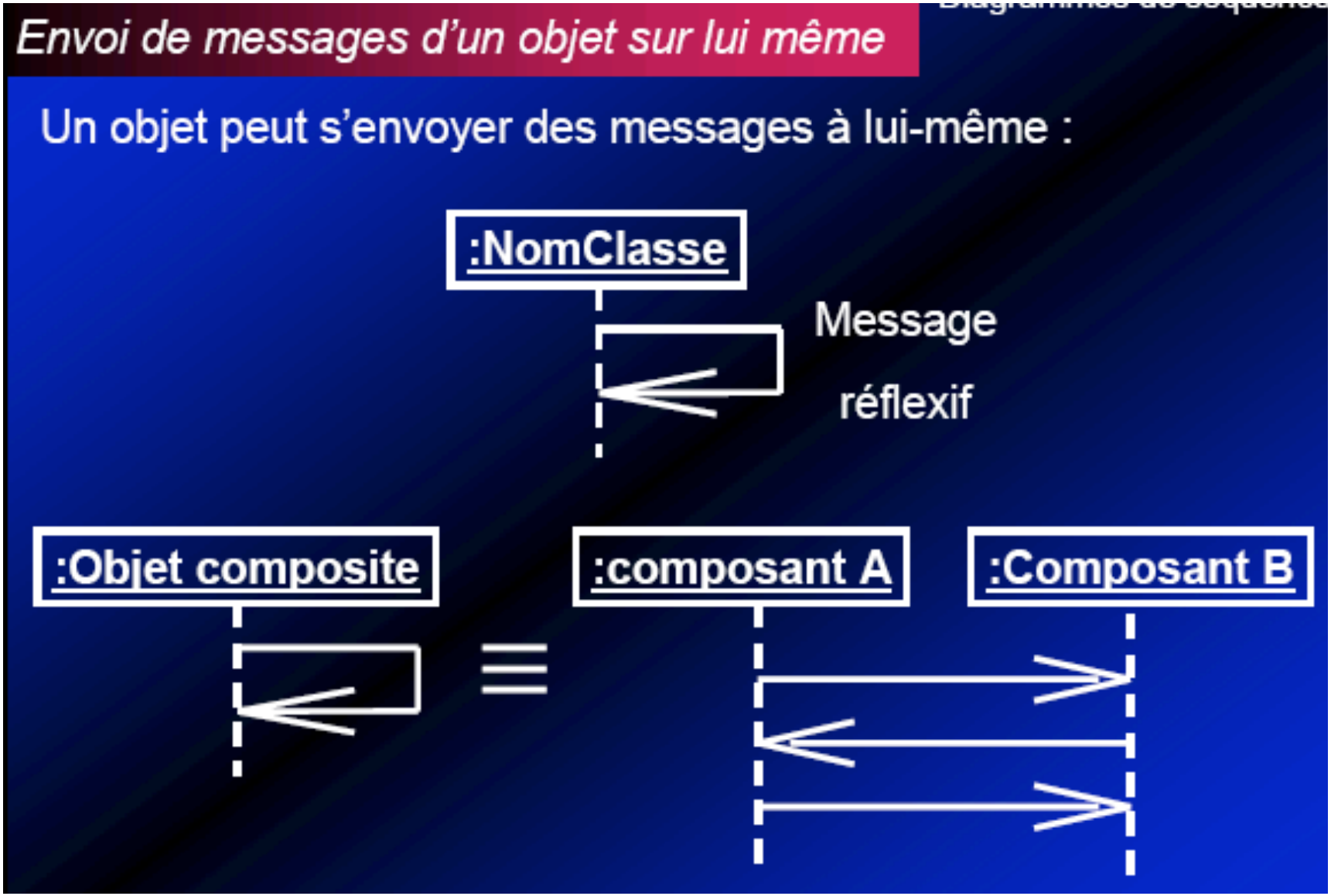


Diagramme de séquence

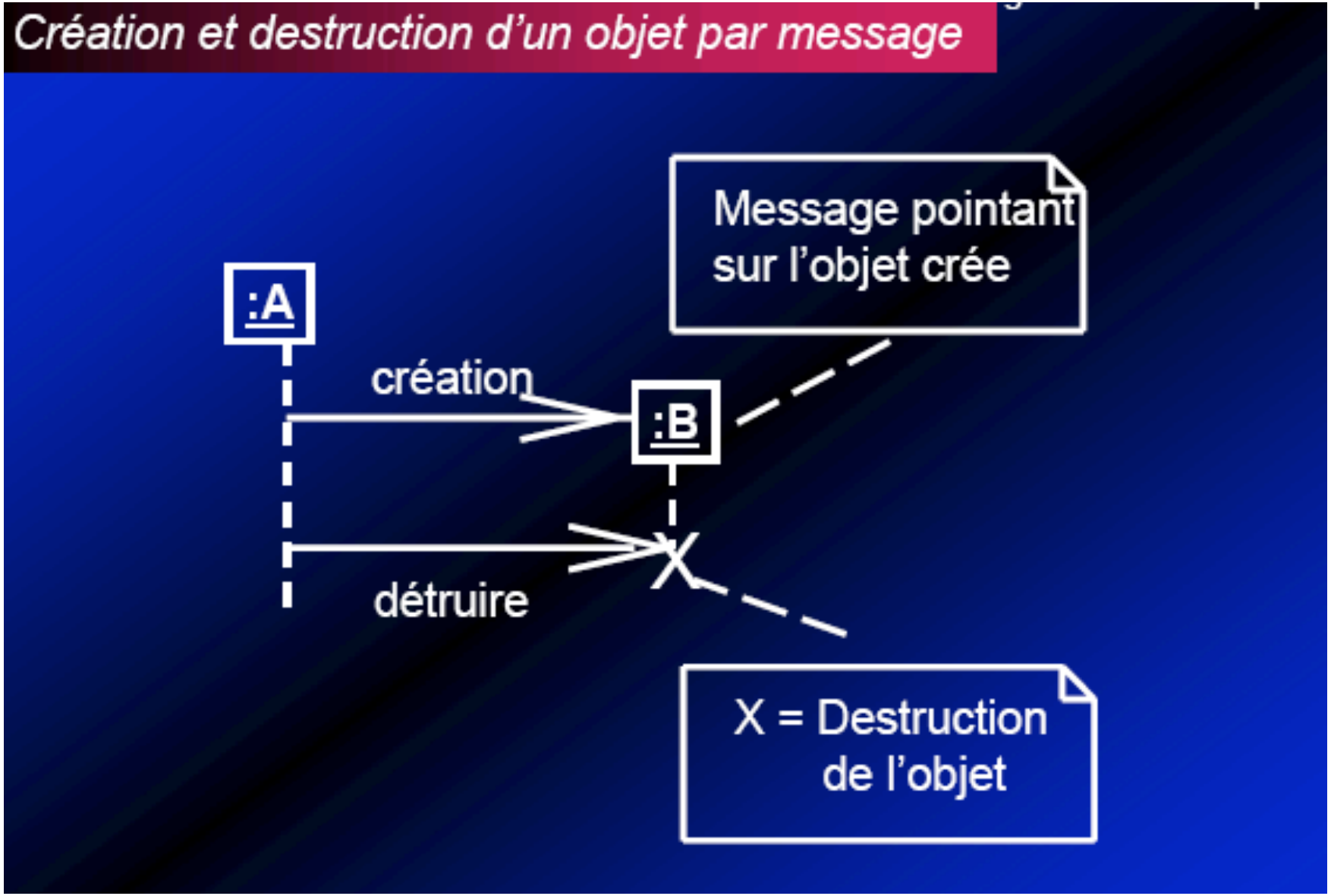


Diagramme de séquence

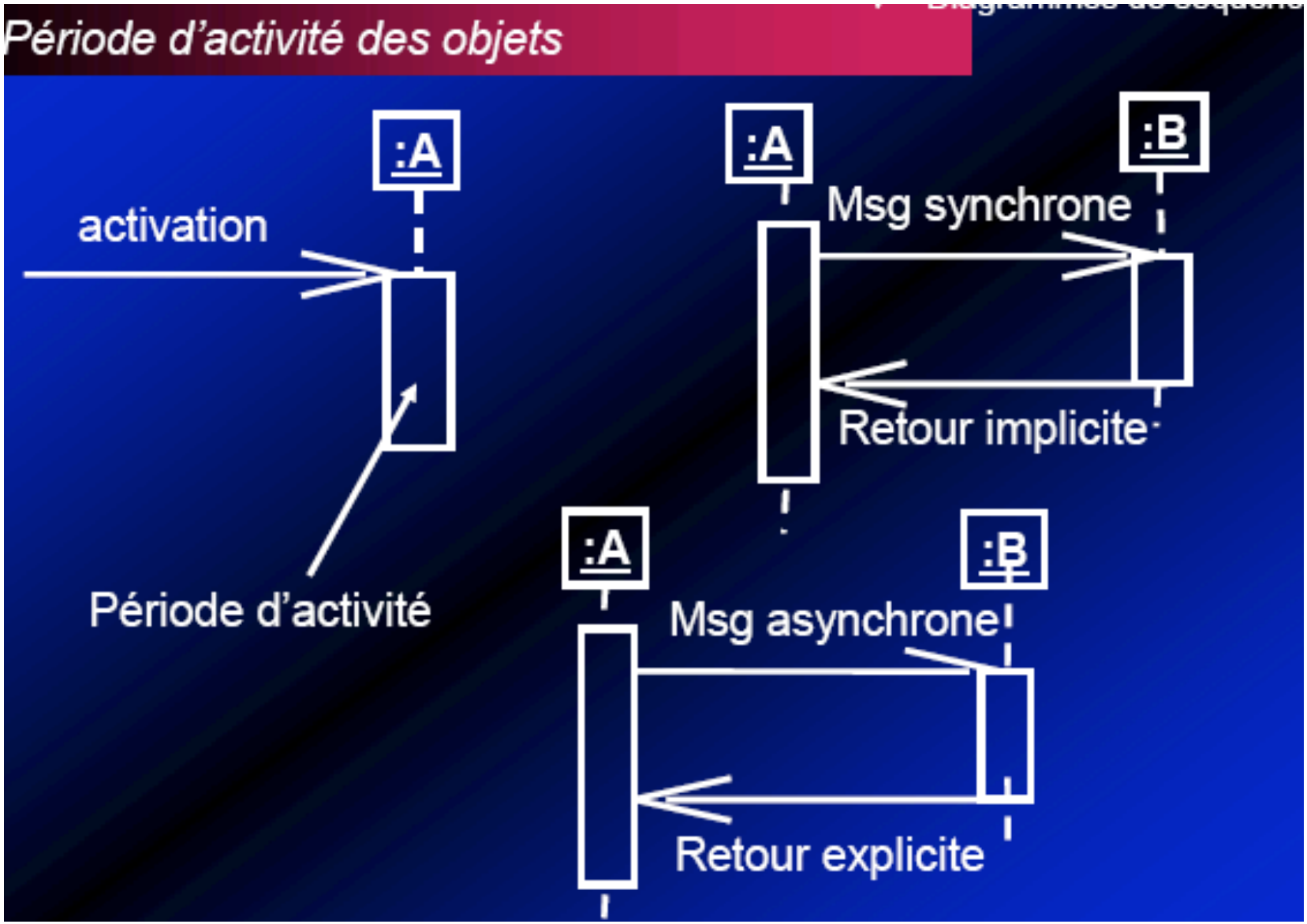


Diagramme de séquence

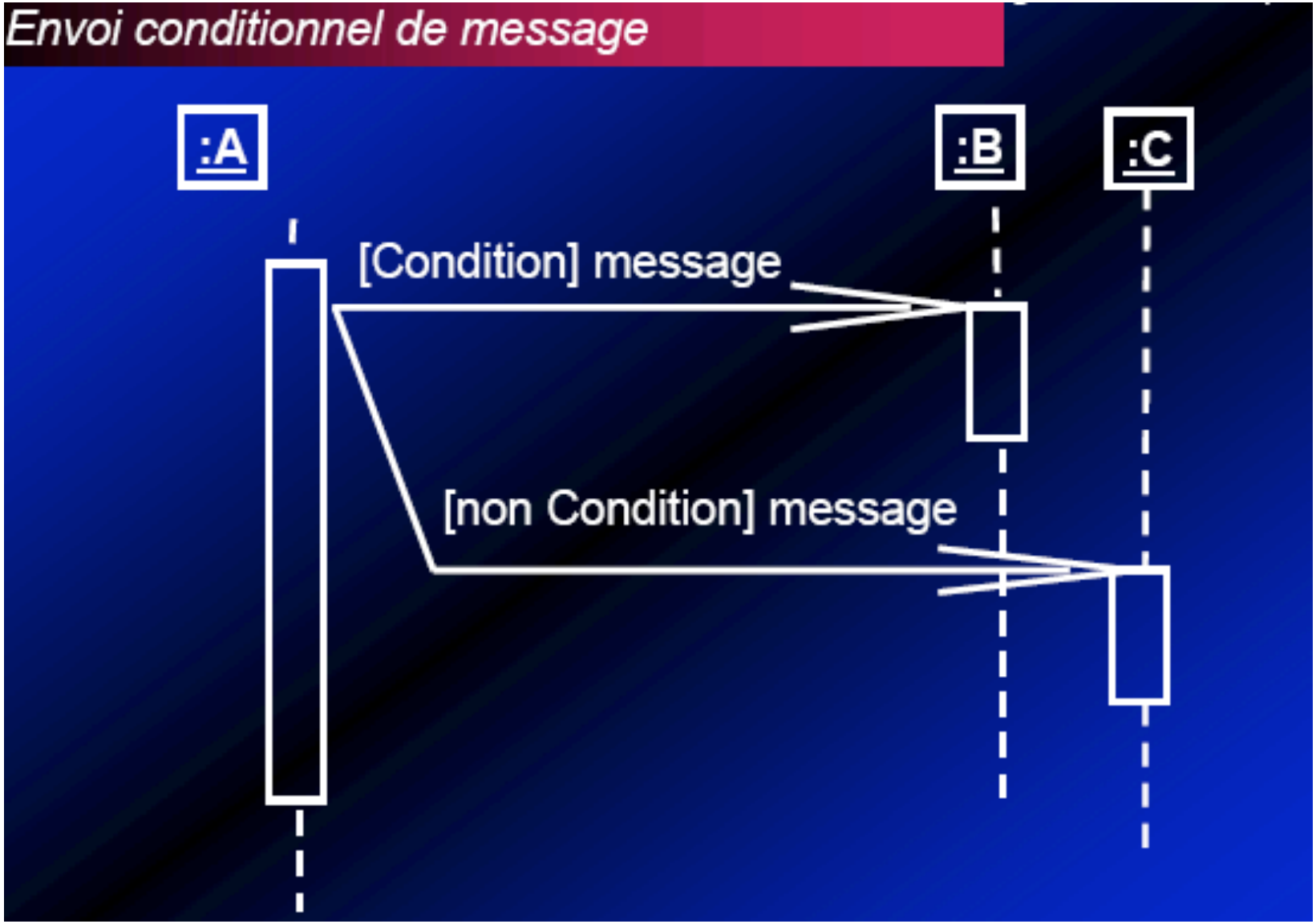


Diagramme de séquence

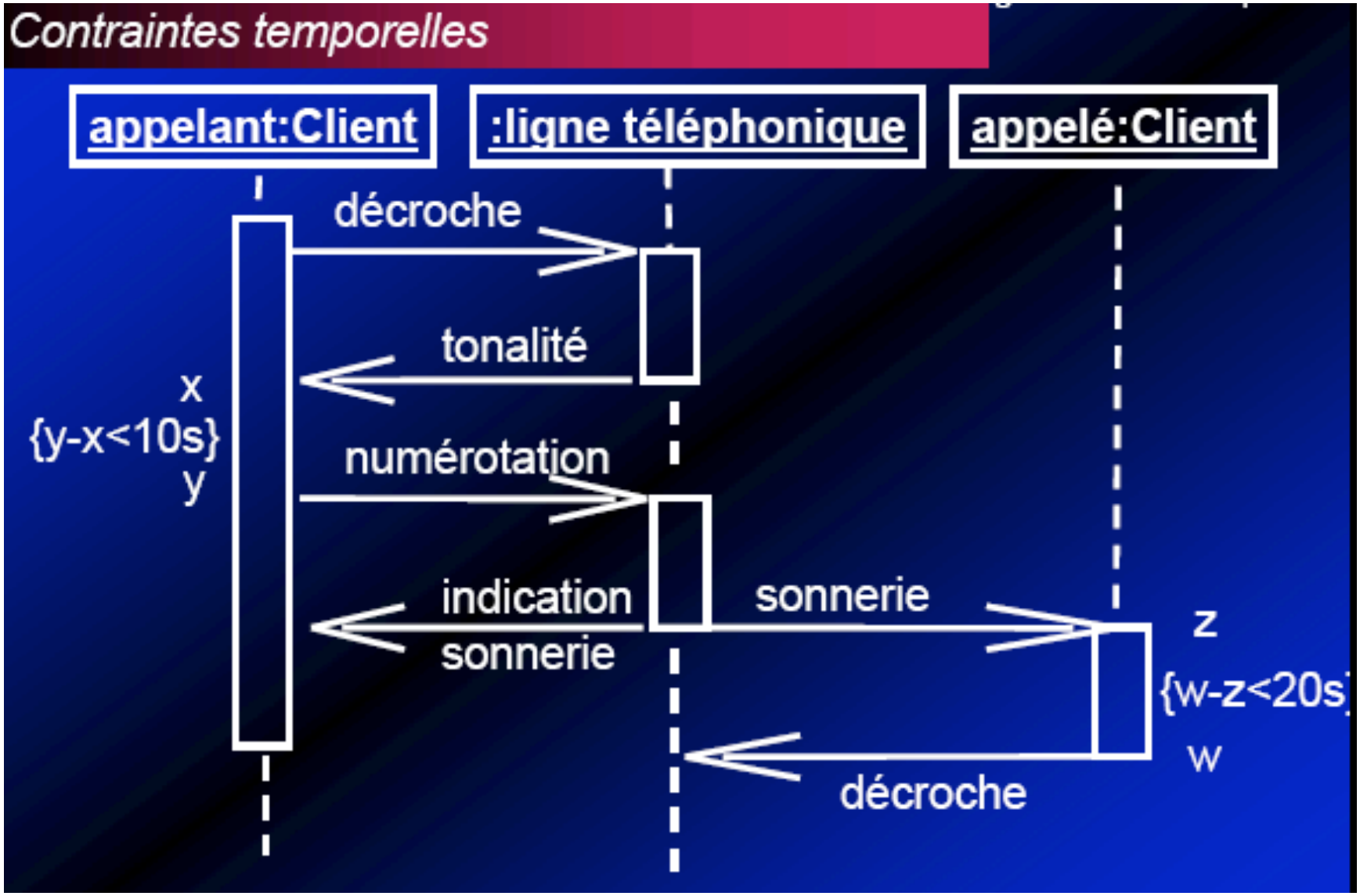


Diagramme de séquence

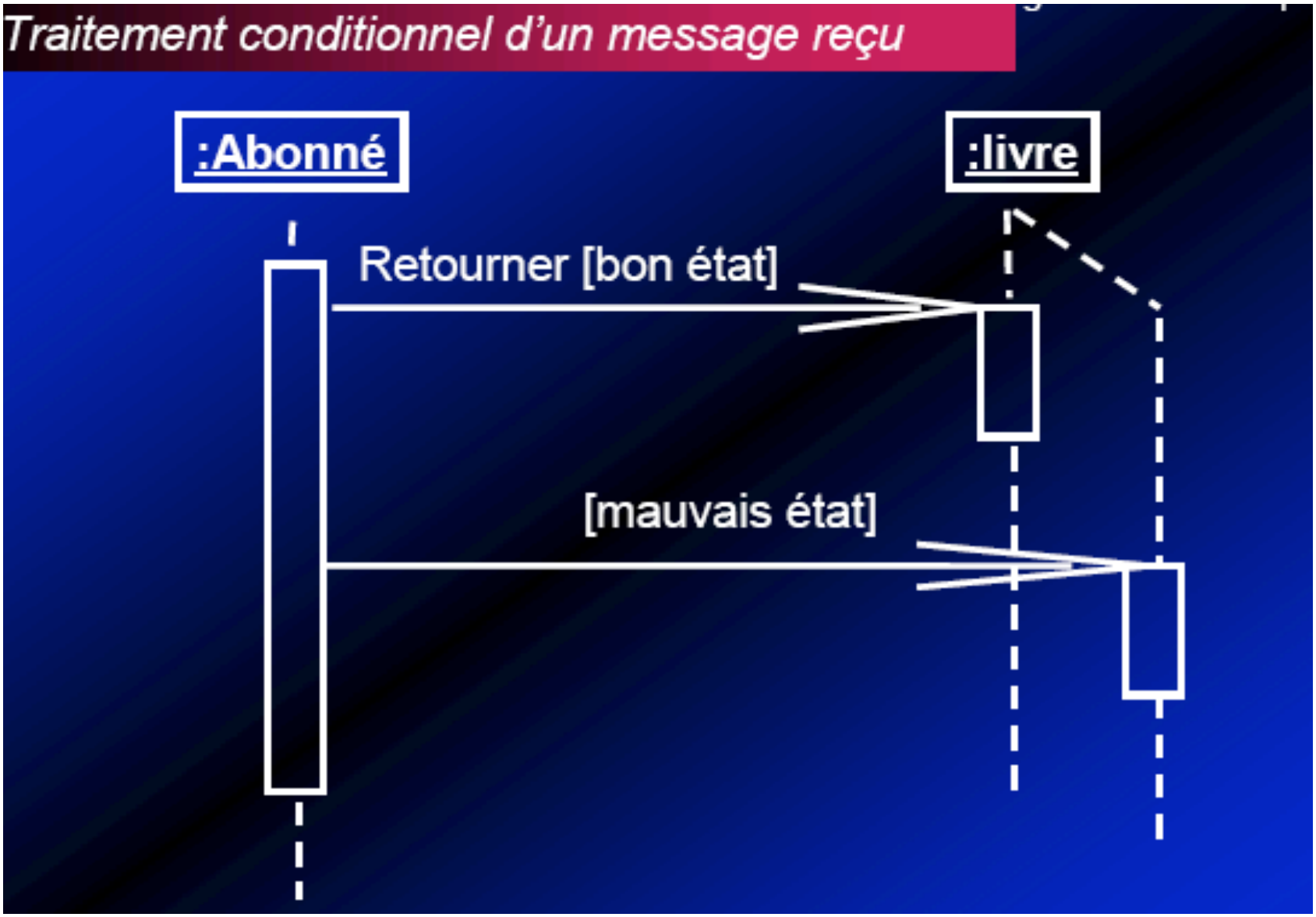


Diagramme de Collaboration

Diagramme de Collaboration

- Décrivent les échanges de *messages* entre classes, et définissent les associations
  - sémantiquement équivalents aux sequence diagrams, mais …
    - sequence diagrams illustrent l'ordre des événements
    - collaboration diagrams représentent les interconnections entre objets et sont visuellement différents
- Class roles: objets participant à l'interaction
- Liens: instances d'associations
- Messages: envoyés le long des liens
- Scénarios: cas particulier

Diagramme de Collaboration

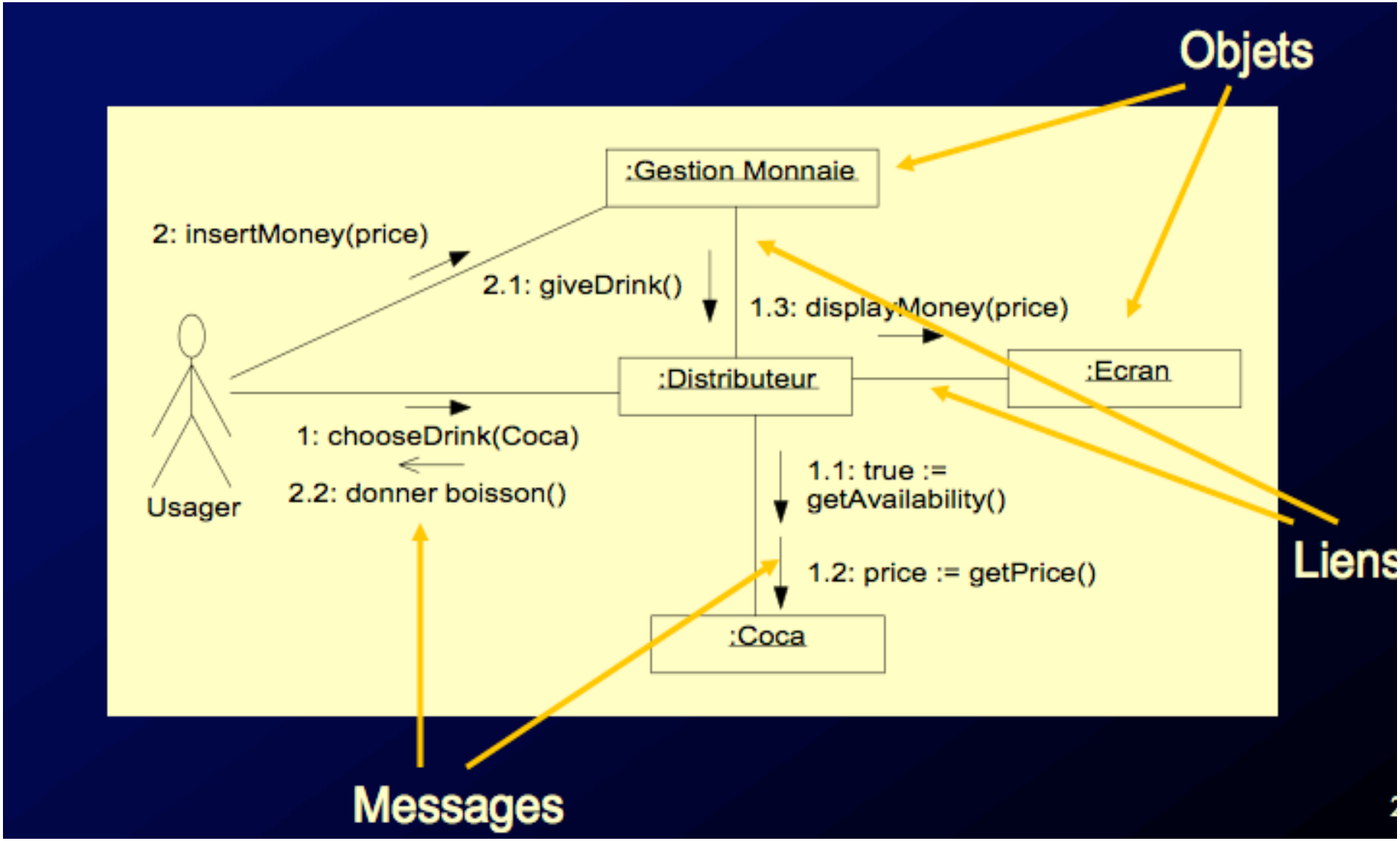


Diagramme de Collaboration

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**

Diagramme de Collaboration

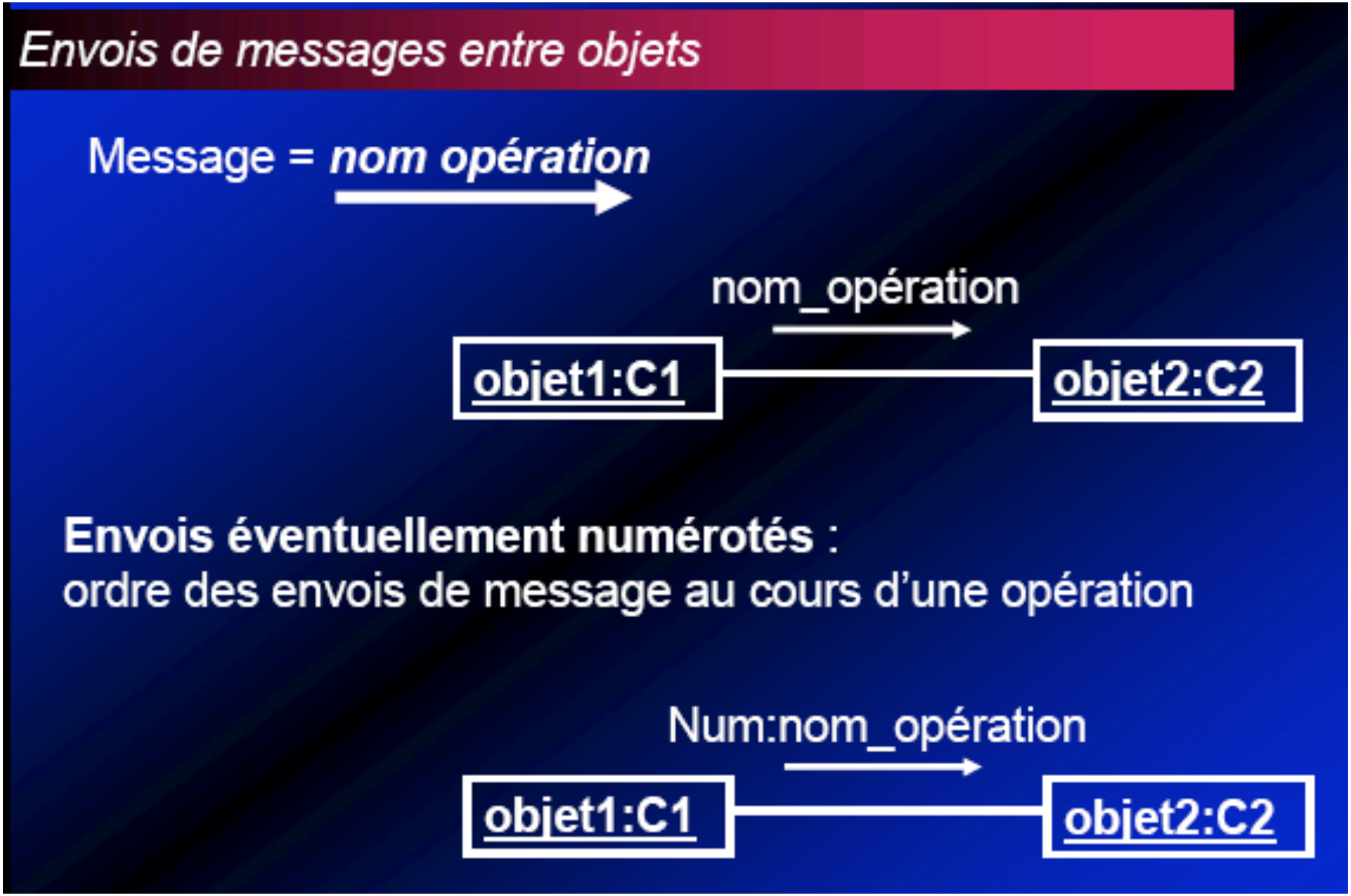


Diagramme de Collaboration

Contraintes associés aux envois de message

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

- **{Nouveau}**
- **{Détruit}**

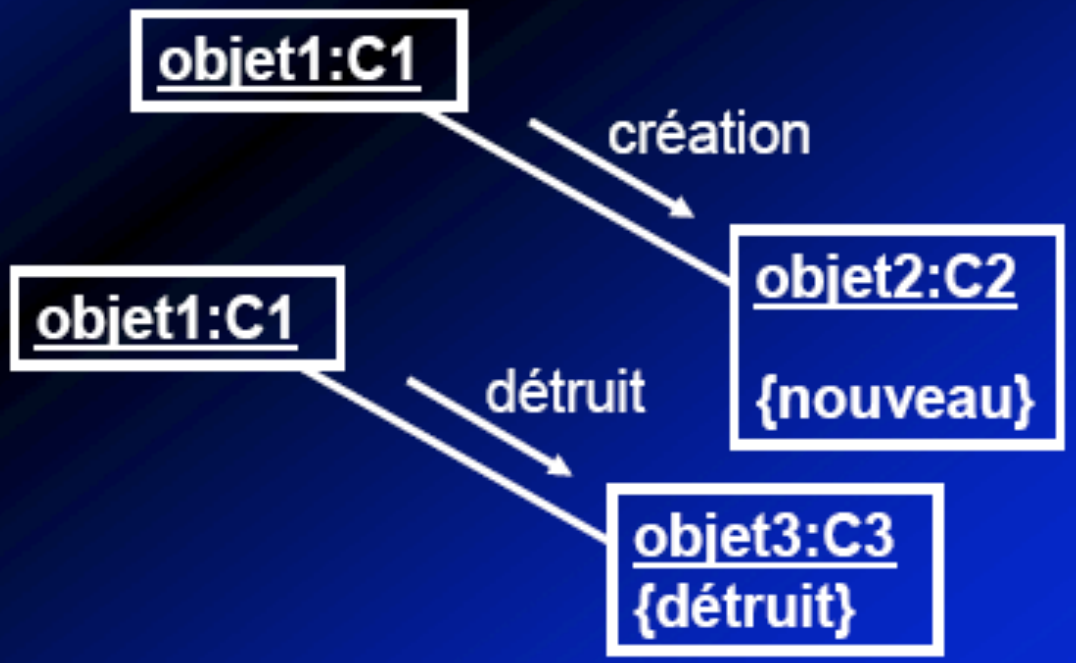


Diagramme de Collaboration

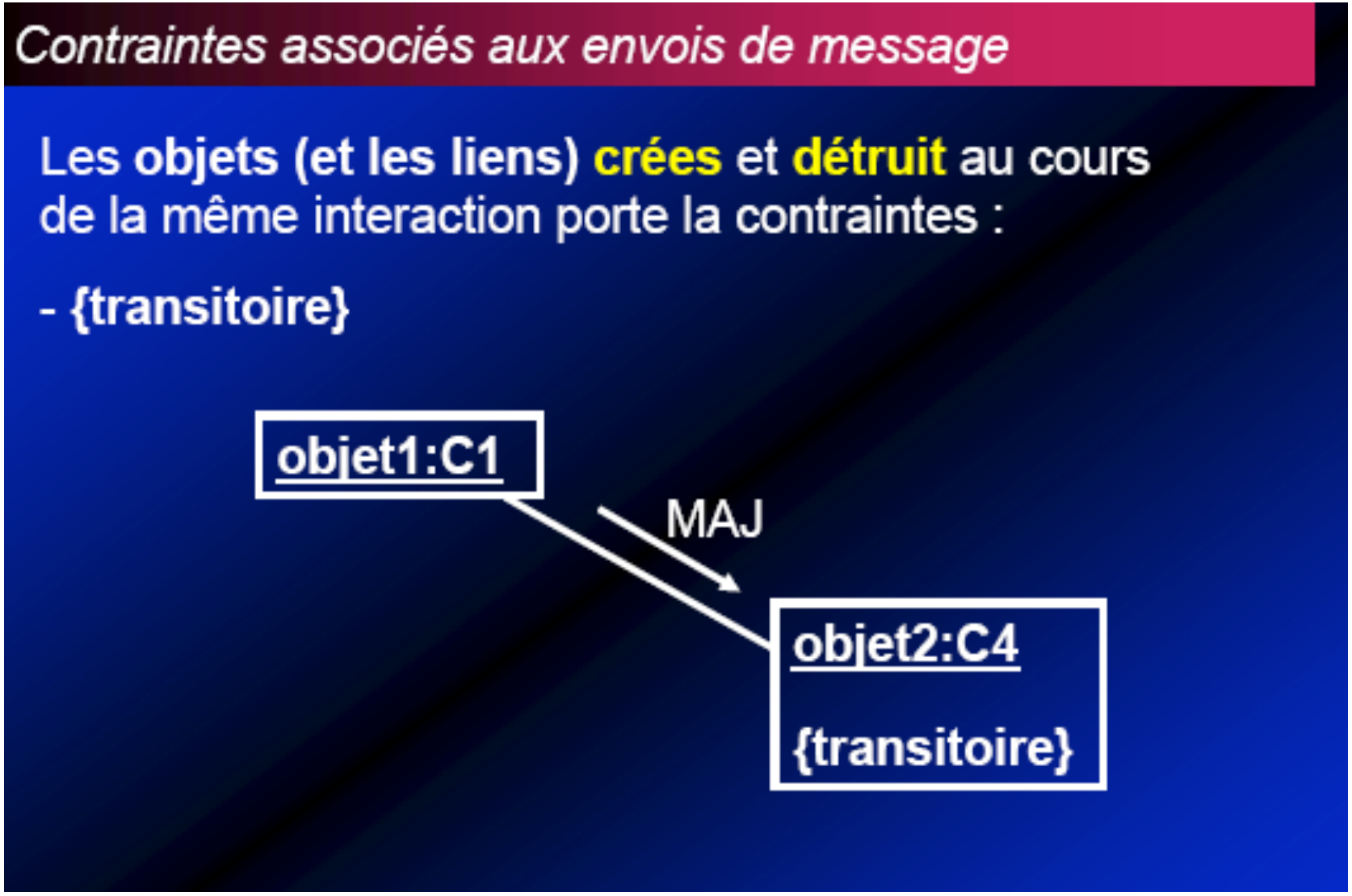


Diagramme de Collaboration

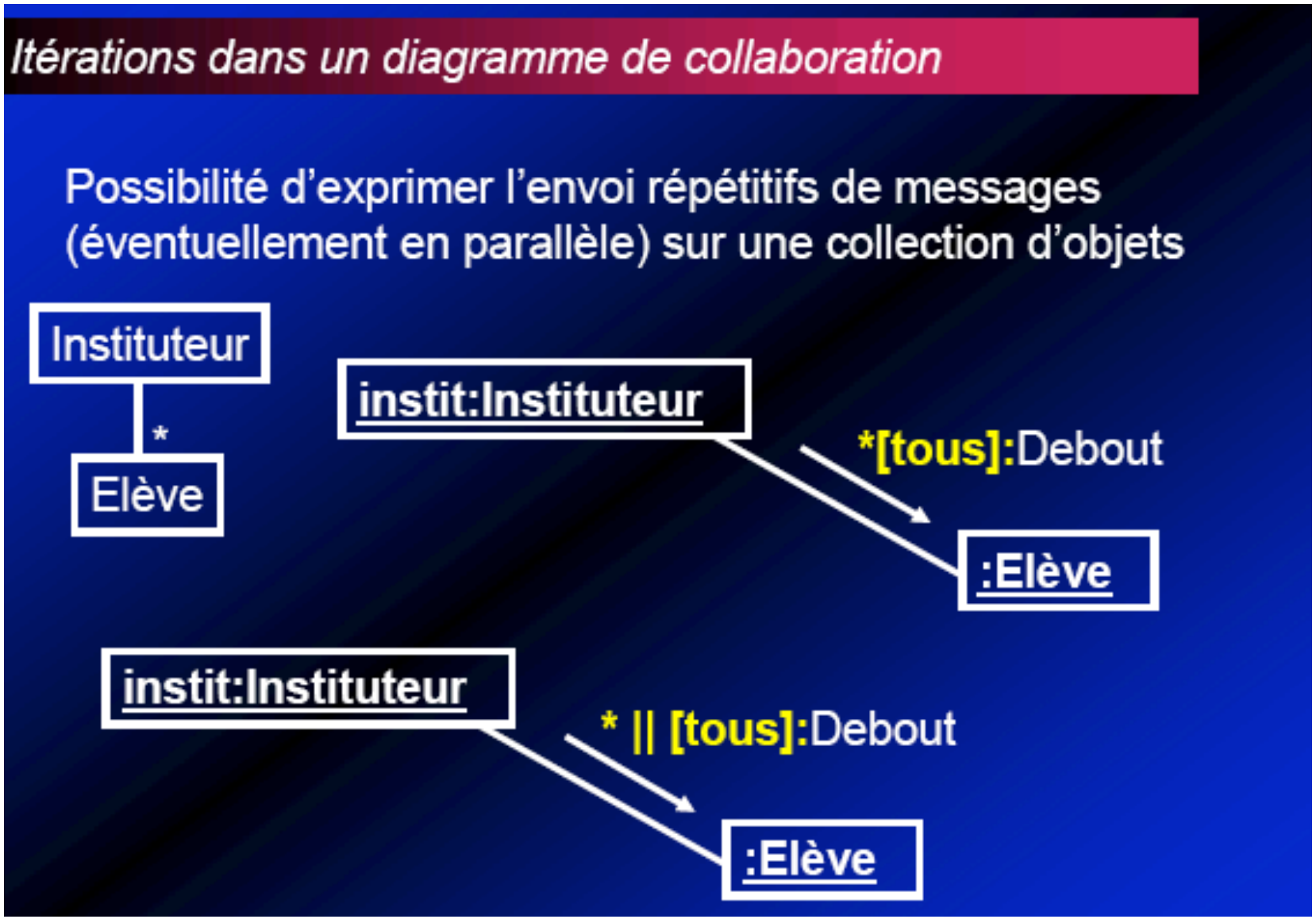


Diagramme de Collaboration

Itérations dans un diagramme de collaboration

Il est possible de faire intervenir un **acteur** (cf. chapitre III) dans un diagramme de collaboration : afin de représenter le **comportement** du système sous l'effet d'un **stimuli externe**

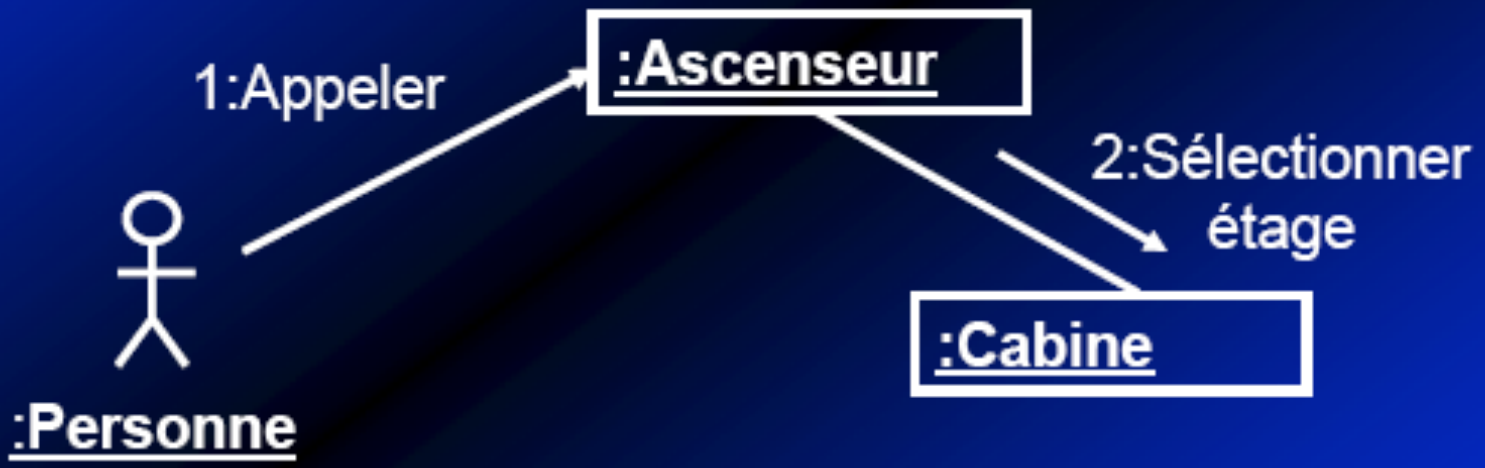


Diagramme de Collaboration

Itérations dans un diagramme de collaboration

Les **objets** qui contrôlent le flot sont dits **actifs**

Un **objet actif** peut **activer** un **objet passif** en lui envoyant un **message**. Une fois le message **traité**, le flot de **contrôle** est **restitué** à **l'objet appelant**

Ex : photocopieuse

Diagramme de Collaboration

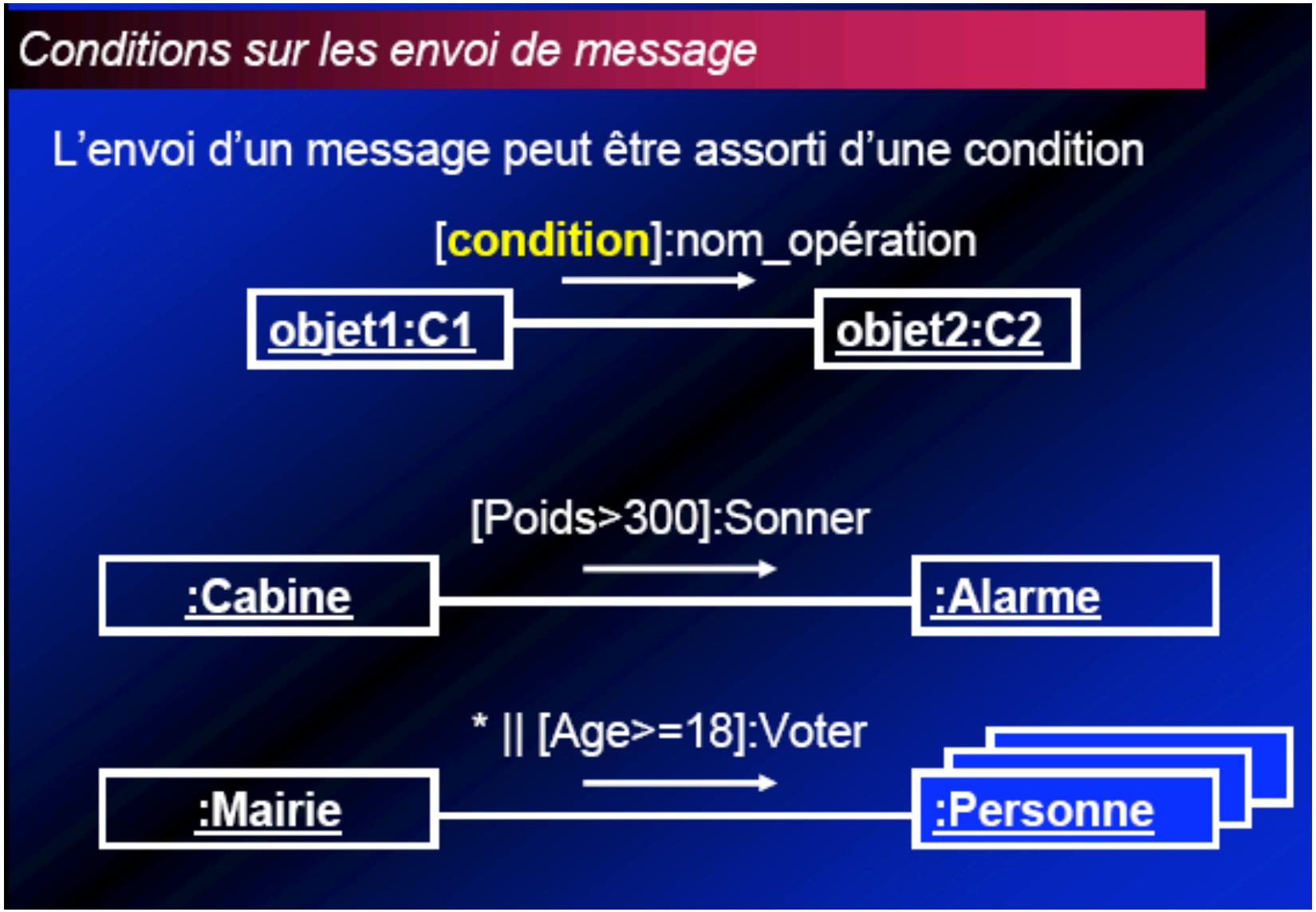


Diagramme de Collaboration

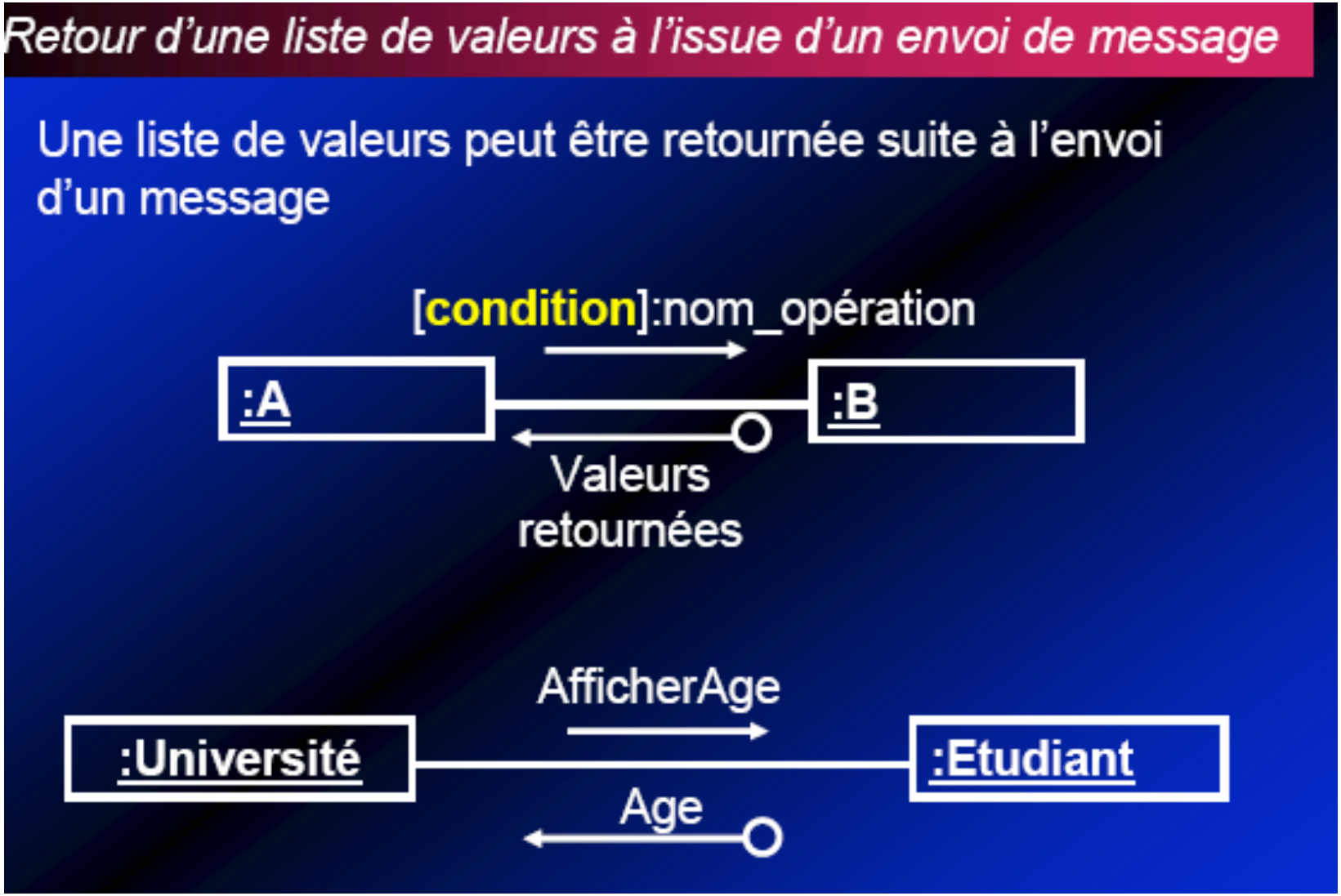


Diagramme de Collaboration

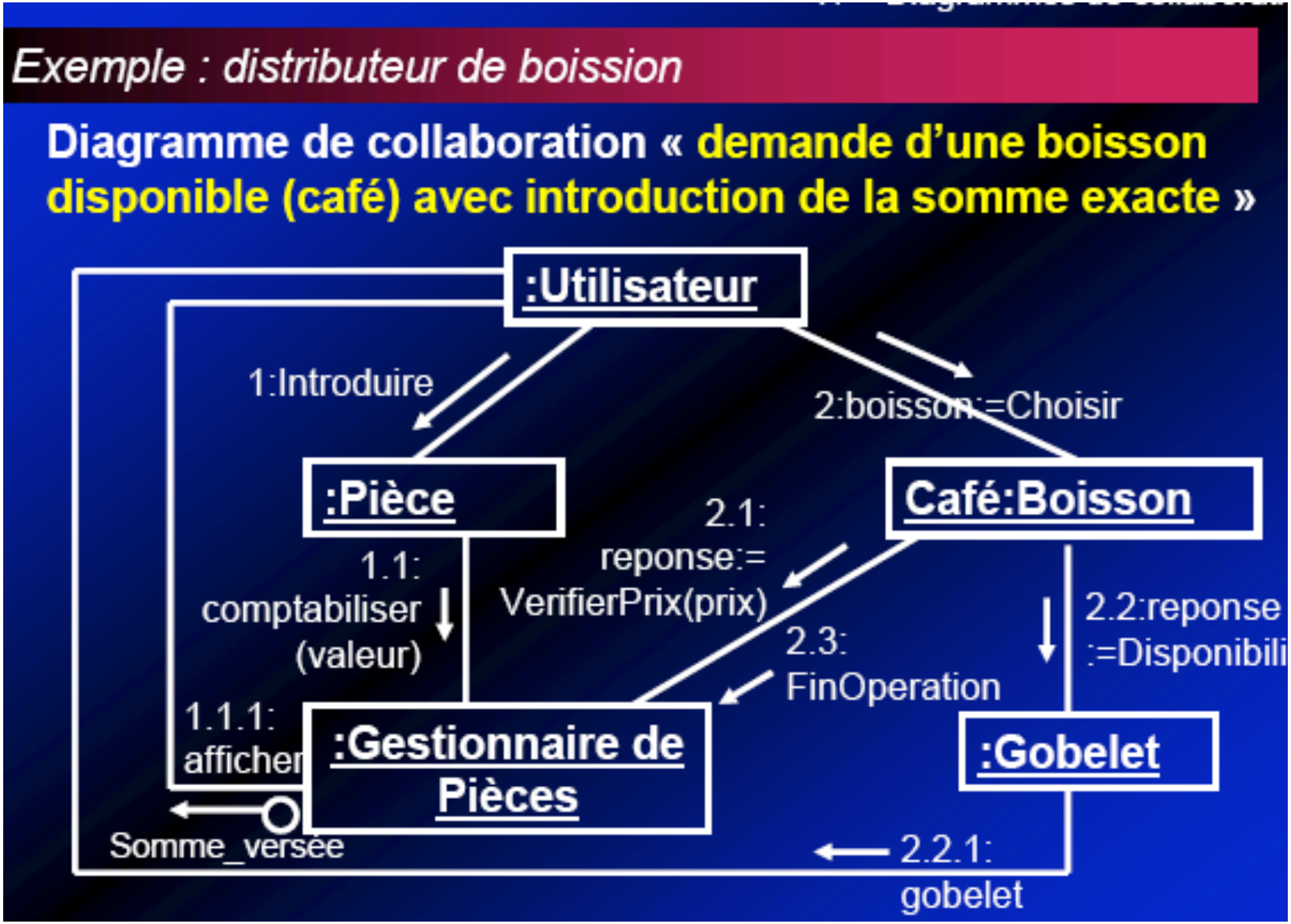


Diagramme d'Etats-Transitions
Statecharts

Diagramme Etats-Transitions

- Décrivent les états et le comportement d'une *classe* en réponse à des événements *extérieurs*
- Etats
  - initial, final, intermédiaire
  - sous-états
- Transitions
  - nom d'événement / action

Diagramme Etats-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

Diagramme Etats-Transitions

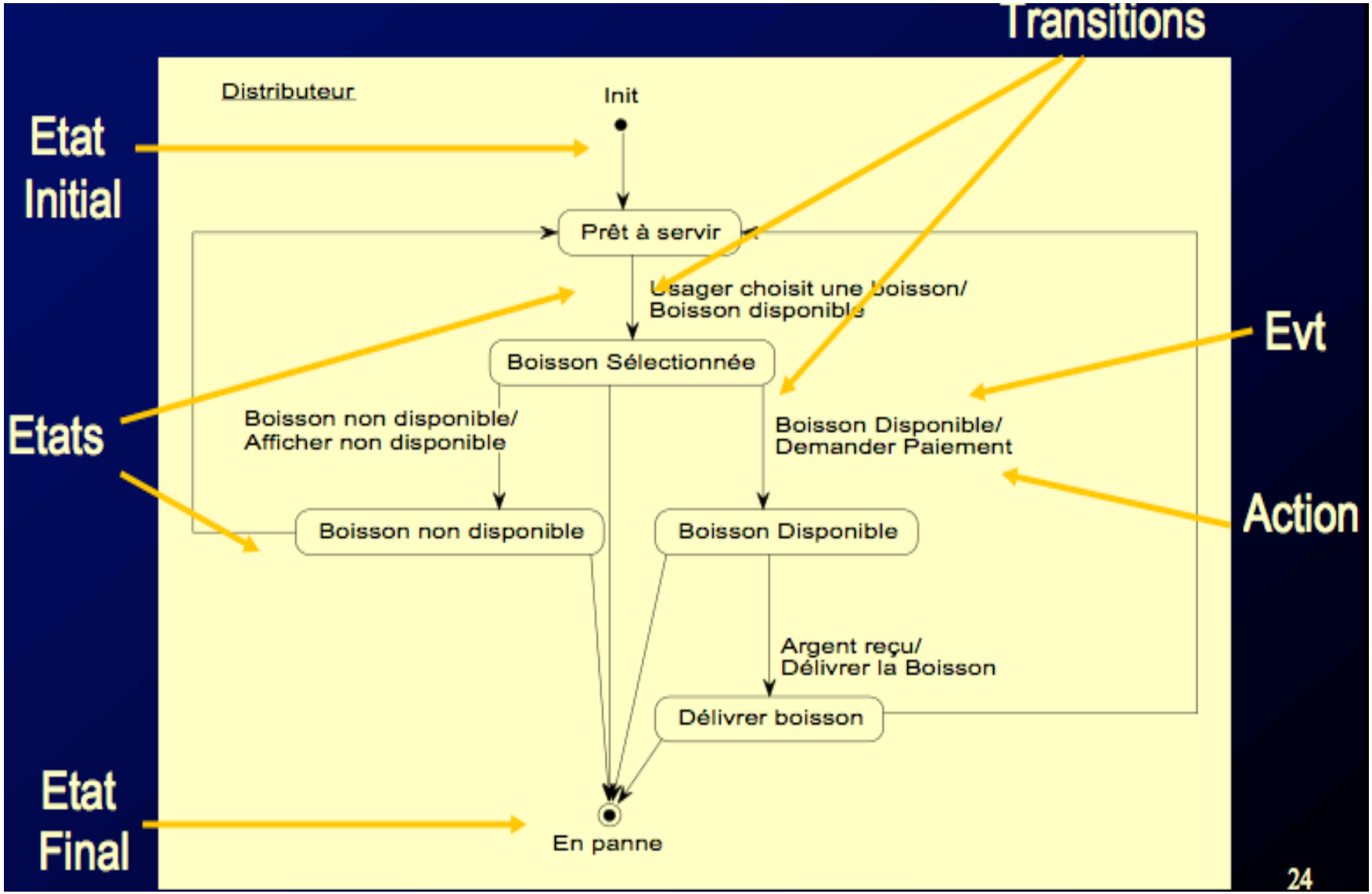


Diagramme Etats-Transitions

Notion d'état

- un **état** = **étape** dans le cycle de vie d'un objet durant lequel
  - il **satisfait** à **certaines conditions**
  - il **réalise certaines actions**
  - ou **attend certains événements**
- chaque **objets** possède à un **instant donné** un **état particulier**
- chaque **état** est **identifié** par un **nom**
- un **état** est **stable** et **durable**

Diagramme Etats-Transitions

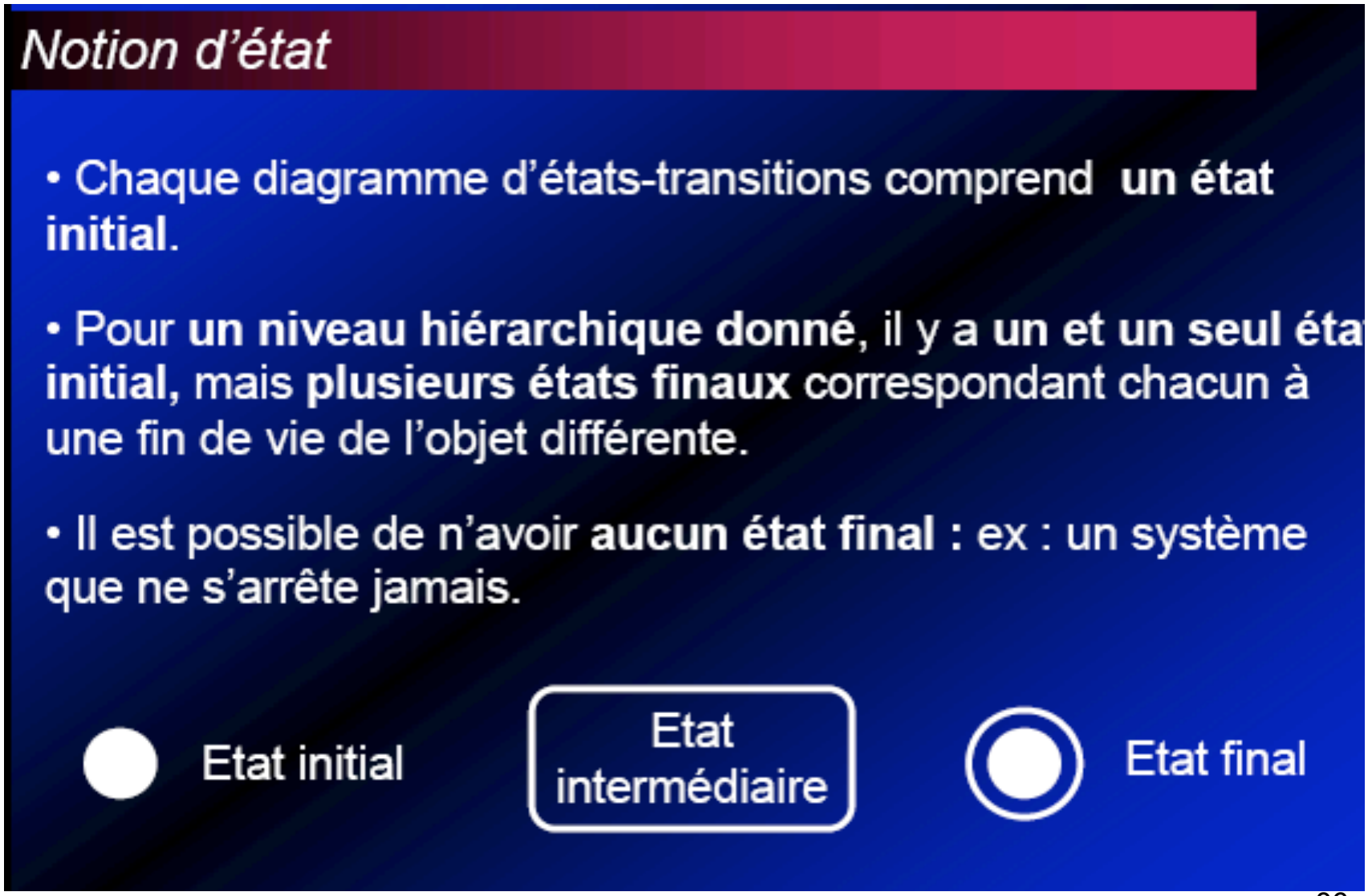


Diagramme Etats-Transitions

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**
- **l'événement** est **déclencheur** de la **transition d'état à état**. Un **objet**, placé **dans un état donné**, **attend l'occurrence** d'un **événement** pour **passer dans un autre état**

Diagramme Etats-Transitions

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

Diagramme Etats-Transitions

Communication entre objets par évenements

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

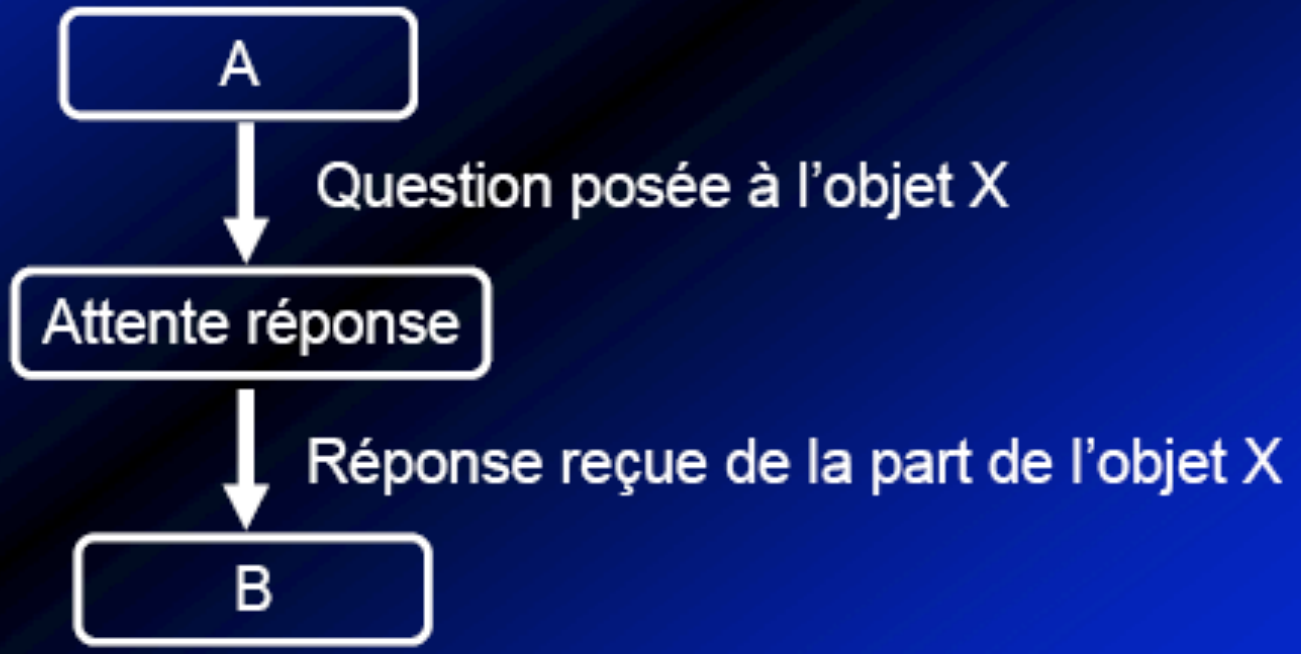


Diagramme Etats-Transitions

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**

Diagramme Etats-Transitions

Communication entre objets par évenements

- Les **gardes** permettent de **conserver** la **propriété** de **déterminisme** d'un automate d'états finis.
- Lorsqu'un occurrence d'**événement survient**, les **gardes,** qui doivent être **mutuellement exclusives**, sont **évaluées**.
- Le **résultat** de cette **évaluation permet** de **valider** puis de **déclencher** une **transition** possible

Diagramme Etats-Transitions

Actions dans un état

- **Les états** peuvent également contenir des actions :
  - elles sont exécutée
    - à **l'entrée** ou à la **sortie** de l'état ou
      - 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**

Diagramme Etats-Transitions

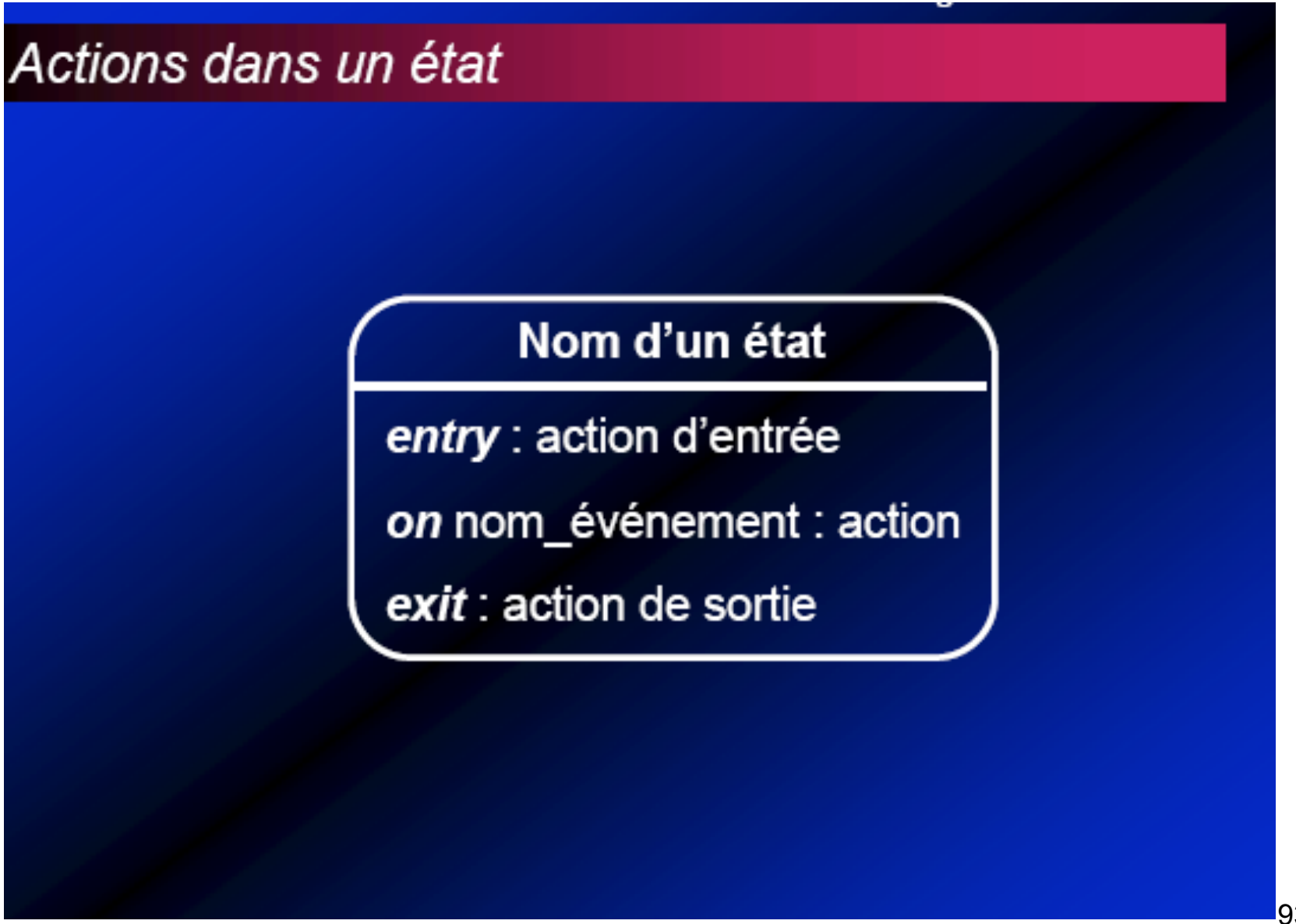


Diagramme Etats-Transitions

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**)

Diagramme Etats-Transitions

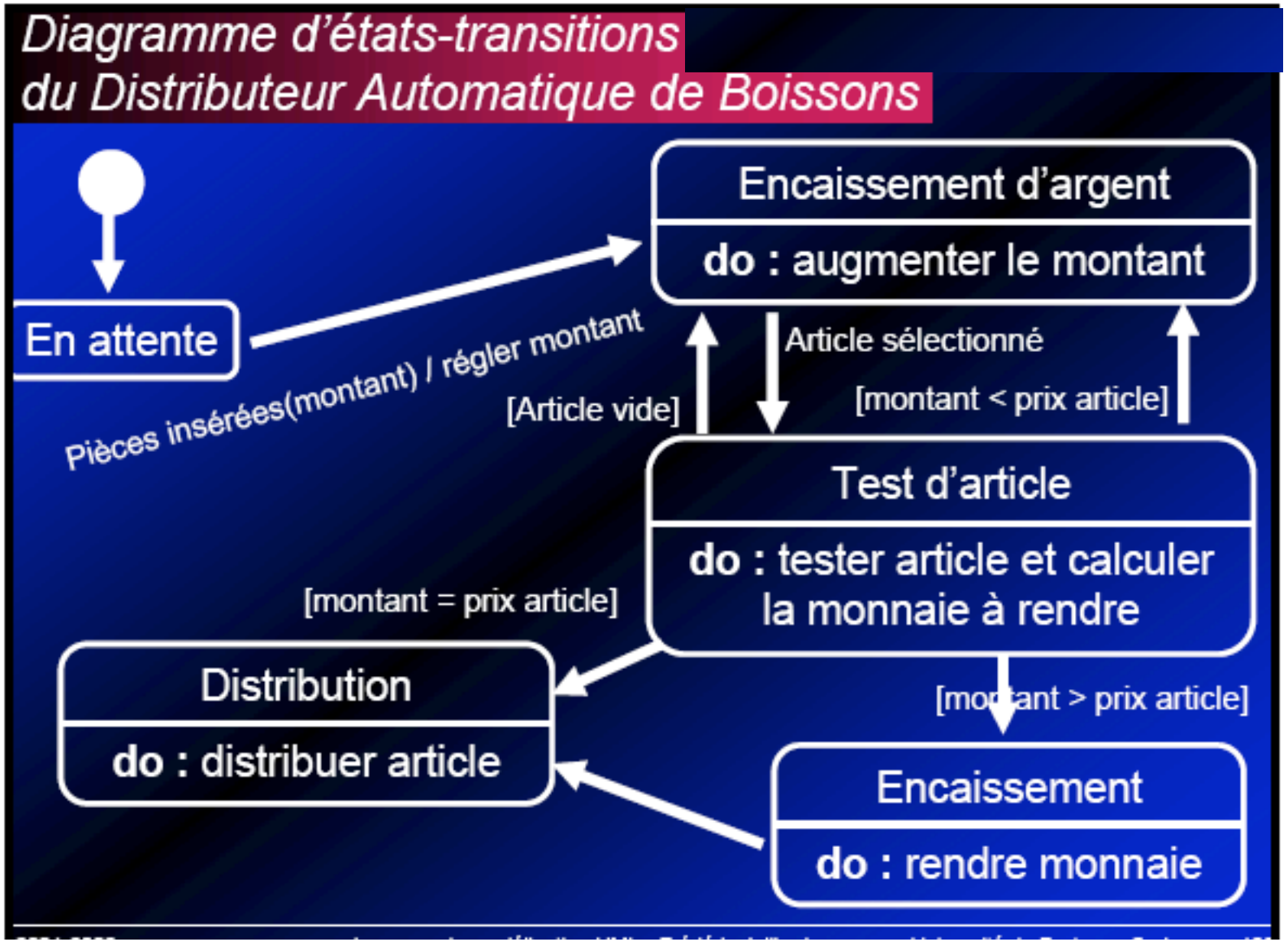


Diagramme Etats-Transitions

Agrégation d'états

- L'agrégation d'états est la composition d'un état à partir de plusieurs autres états indépendants
- La composition est de type conjonctive ce qui implique que l'objet doit être simultanément dans tous les états composant l'agrégation d'états.
- Forme de parallélisme entre automates

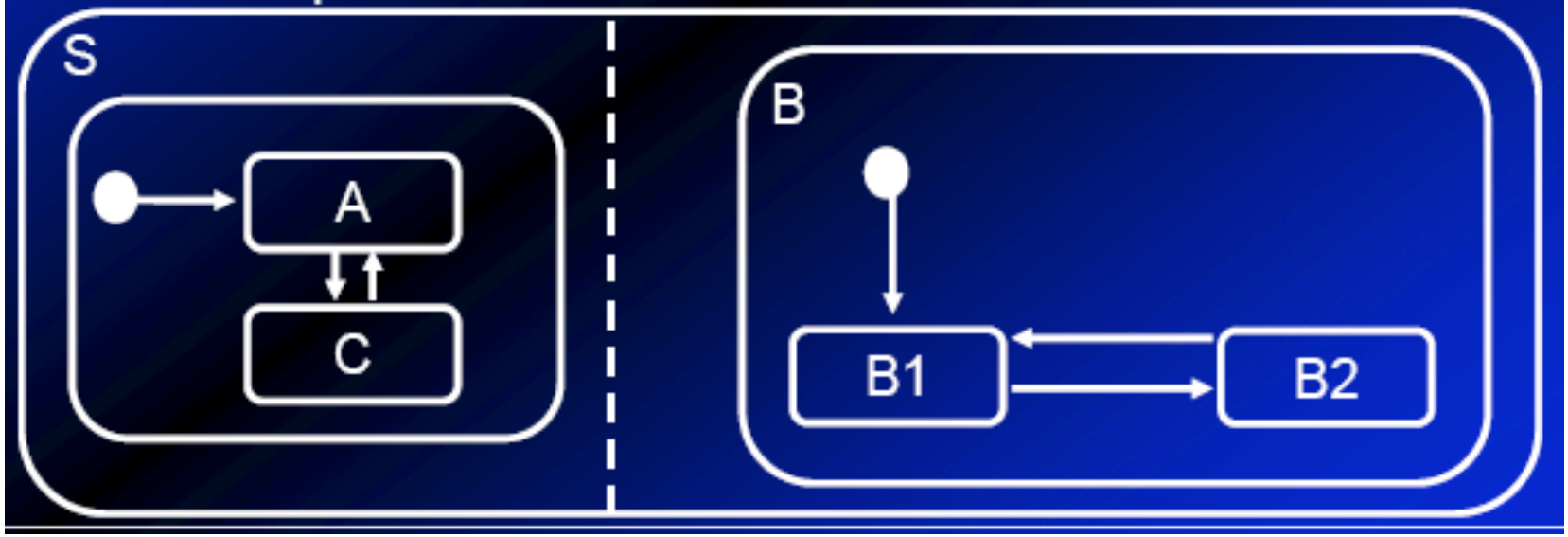


Diagramme Etats-Transitions

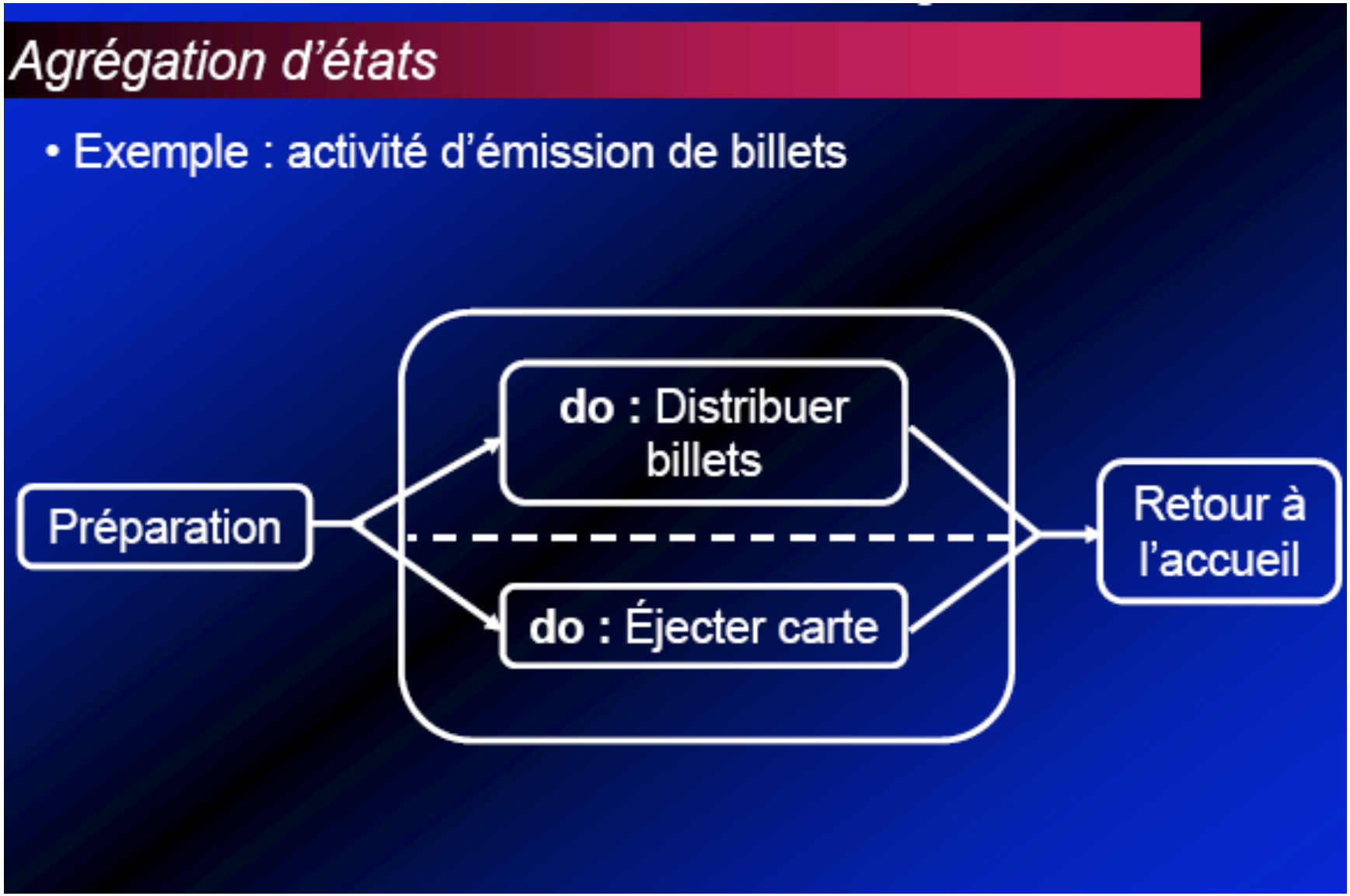


Diagramme d’activités

Diagramme d'activités

- Décrivent le *comportement* d'une *classe* en réponse à des calculs *internes*
- Similaire aux statecharts, mais pour les événements internes (et non extérieurs)

Diagramme d'activités

- Variante des diagrammes d'états-transitions : ce diagramme met l'accent sur les activités, leurs relations et leurs impacts sur les objets

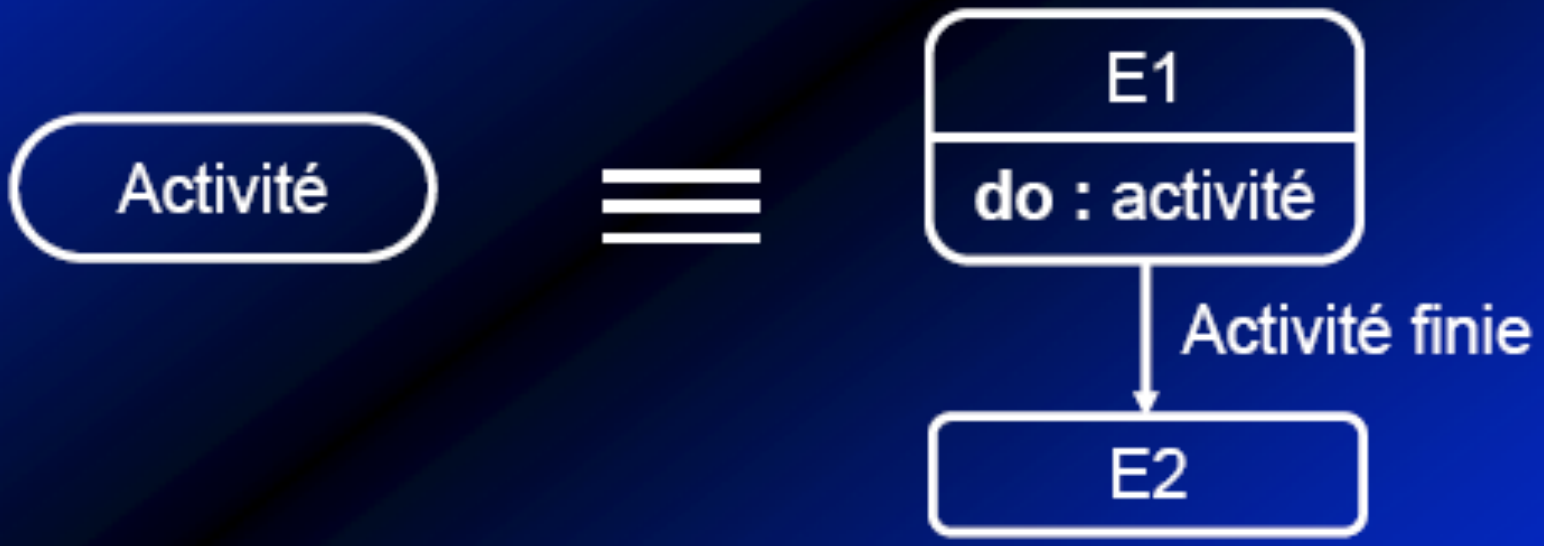


Diagramme d'activités

Gardes

- **Les transitions** entre **activités** peuvent être **gardées** par des **conditions booléennes**, **mutuellement exclusives**.
- Les **gardes** sont les **labels** des **transitions** dont elles valident le **déclenchement**

Diagramme d’activités

Gardes

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

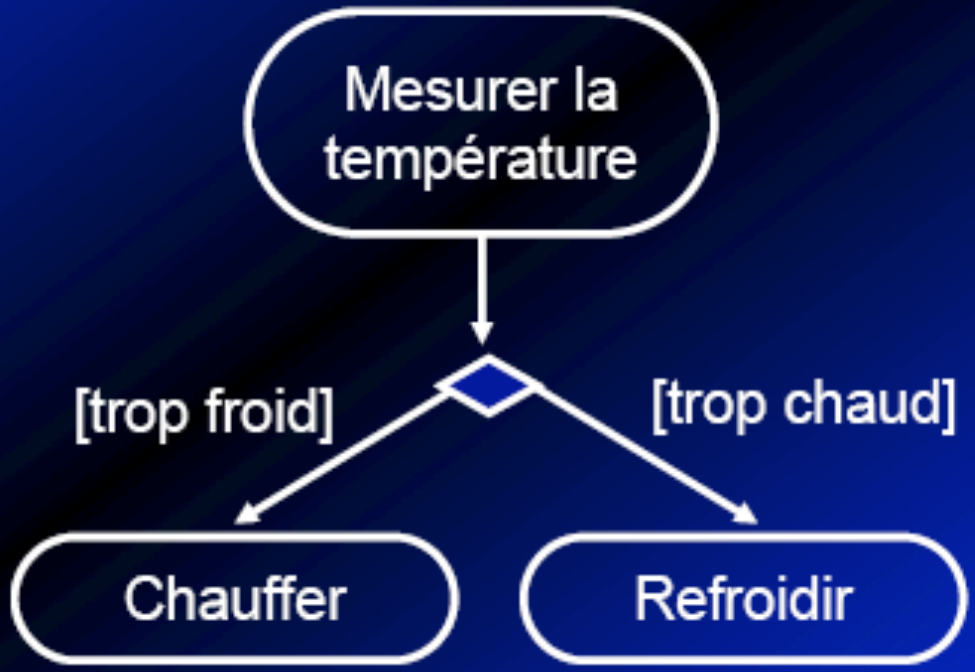


Diagramme d’activités

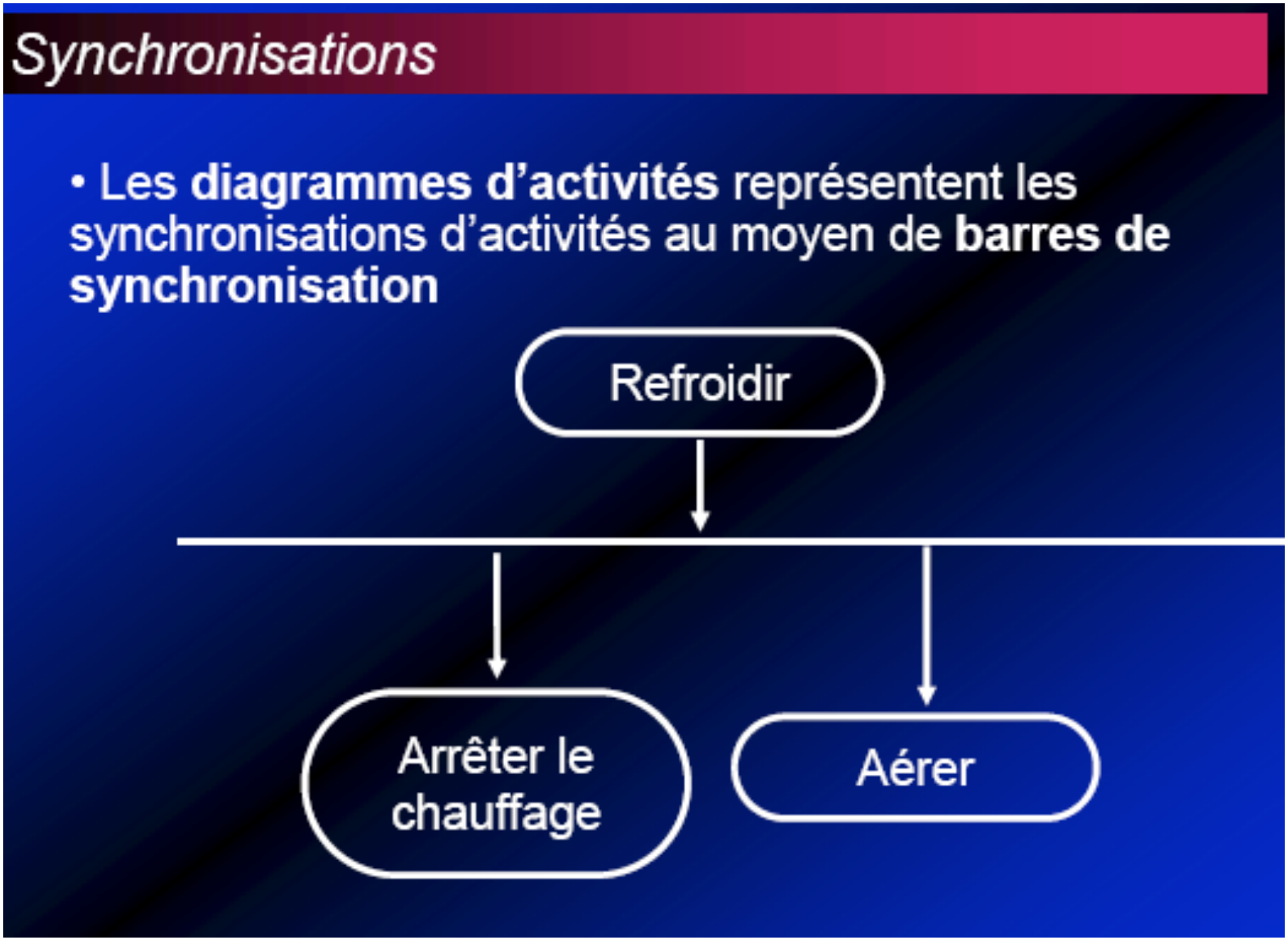


Diagramme d’activités

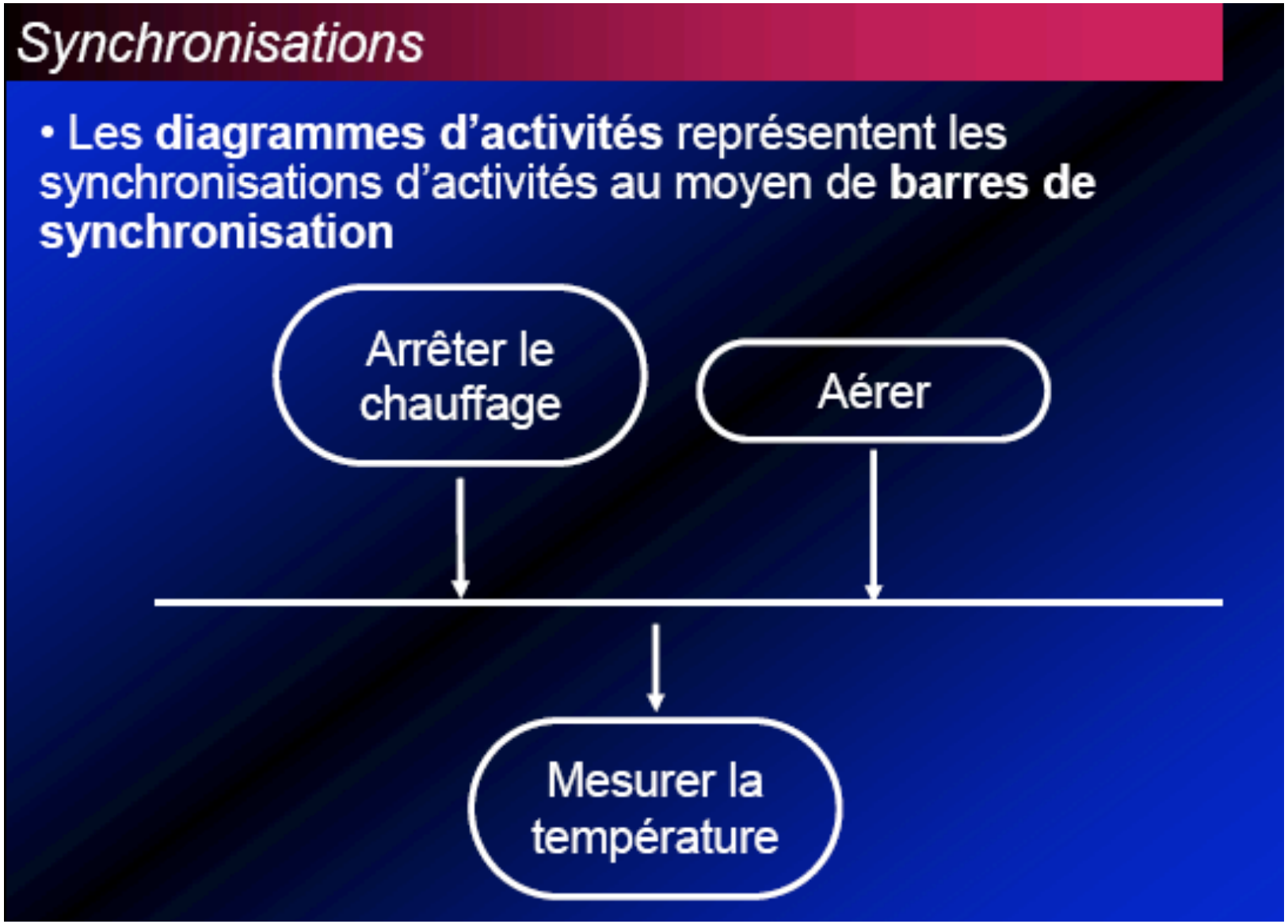


Diagramme d'activités

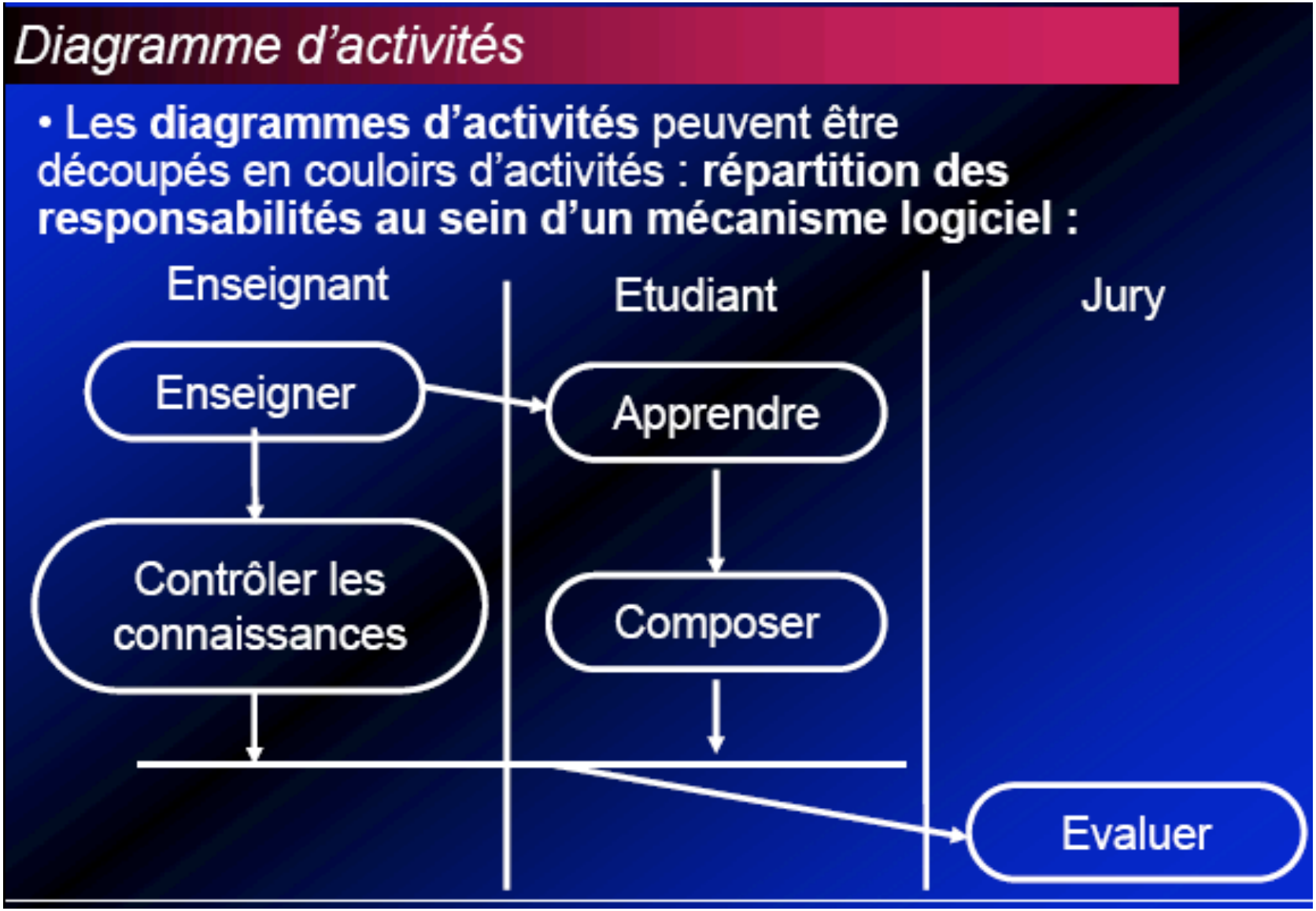
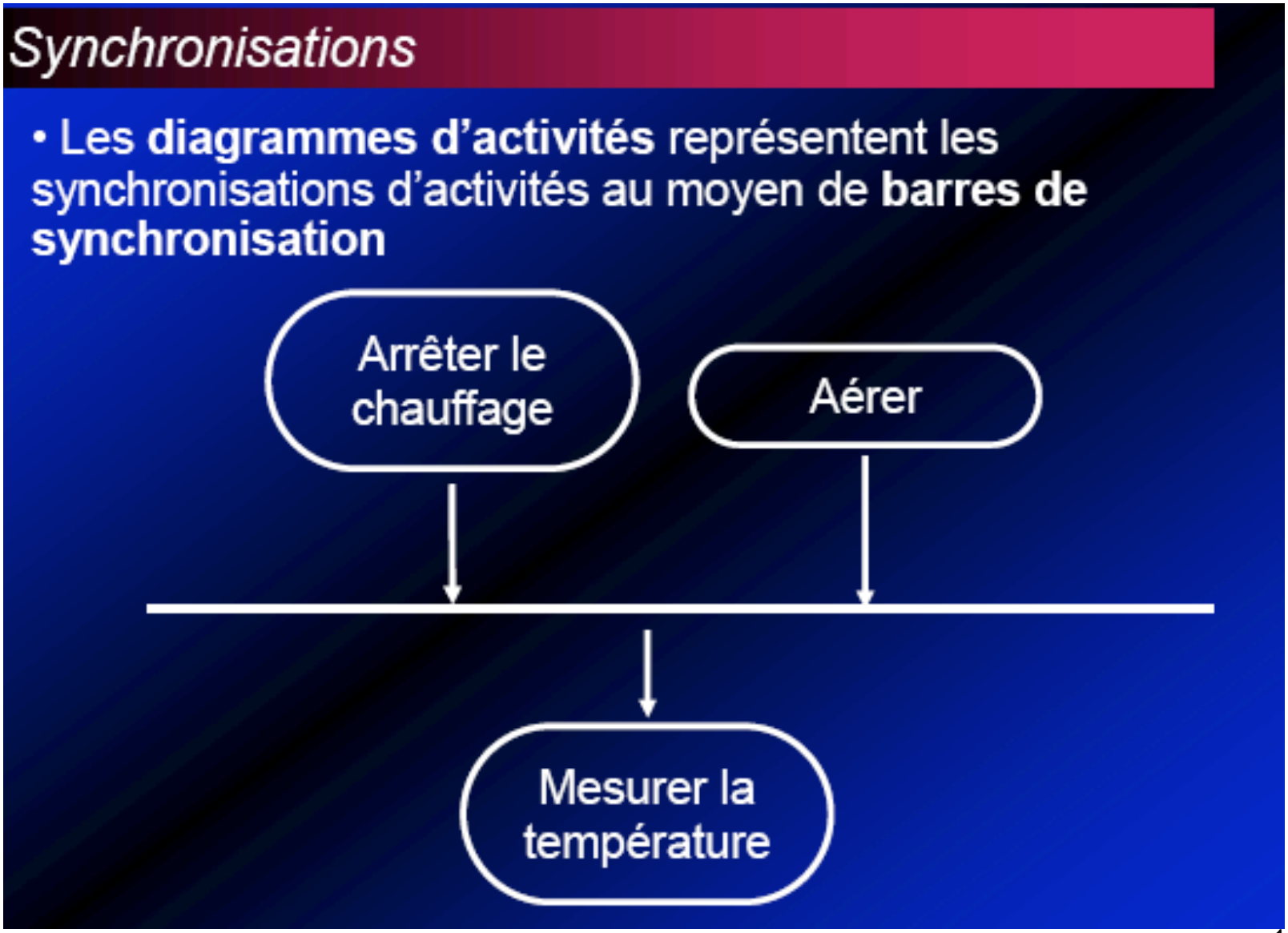


Diagramme d’activités



Autres diagrammes

- **Component Diagrams**
  - Décrivent *l'organisation* des *composants* et les *dépendances* qui les lient
- **Deployment Diagrams**
  - Décrivent les *ressources de calcul*, leurs *configurations* et le lien entre les *exécutables* et les ressources de calcul
- **OCL: Object Constraint Language**
  - Langage permettant d'exprimer des conditions attachées à des éléments d 'un modèle
- **Stéréotypes**
  - nouveaux éléments peuvent être définis et introduits dans un modèle

UML et méthodologies

- UML est une notation
- UML n'est pas une méthodologie
- Les méthodologies peuvent utiliser UML comme notation
  - RUP (Rational Unified Process)
  - OOSE (Object-Oriented Software Engineering, Jacobson)
  - OMT (Object Modeling Technique, Rumbaugh)