Rappel

- **Introduction**:

  historique du génie logiciel

- **Modèles de Cycle de vie et les étapes**:

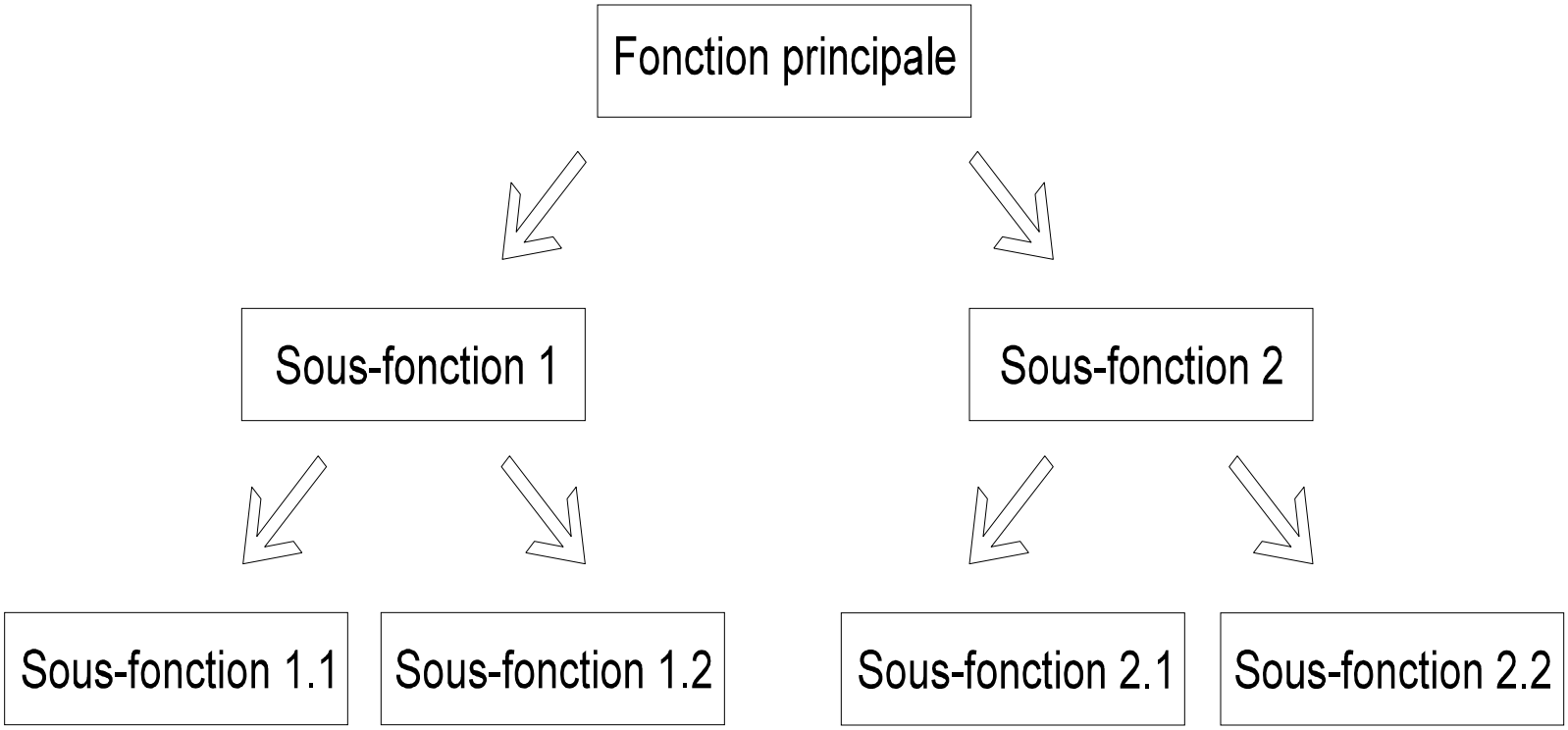
  Cascade, V, Prototypage, Incrémental, Spiral, Scrum

- **1$^{\text{ères}}$ étapes du Cycle de vie**:

  Analyse et Spécification des besoins – Cahier des charges, Document de spécification

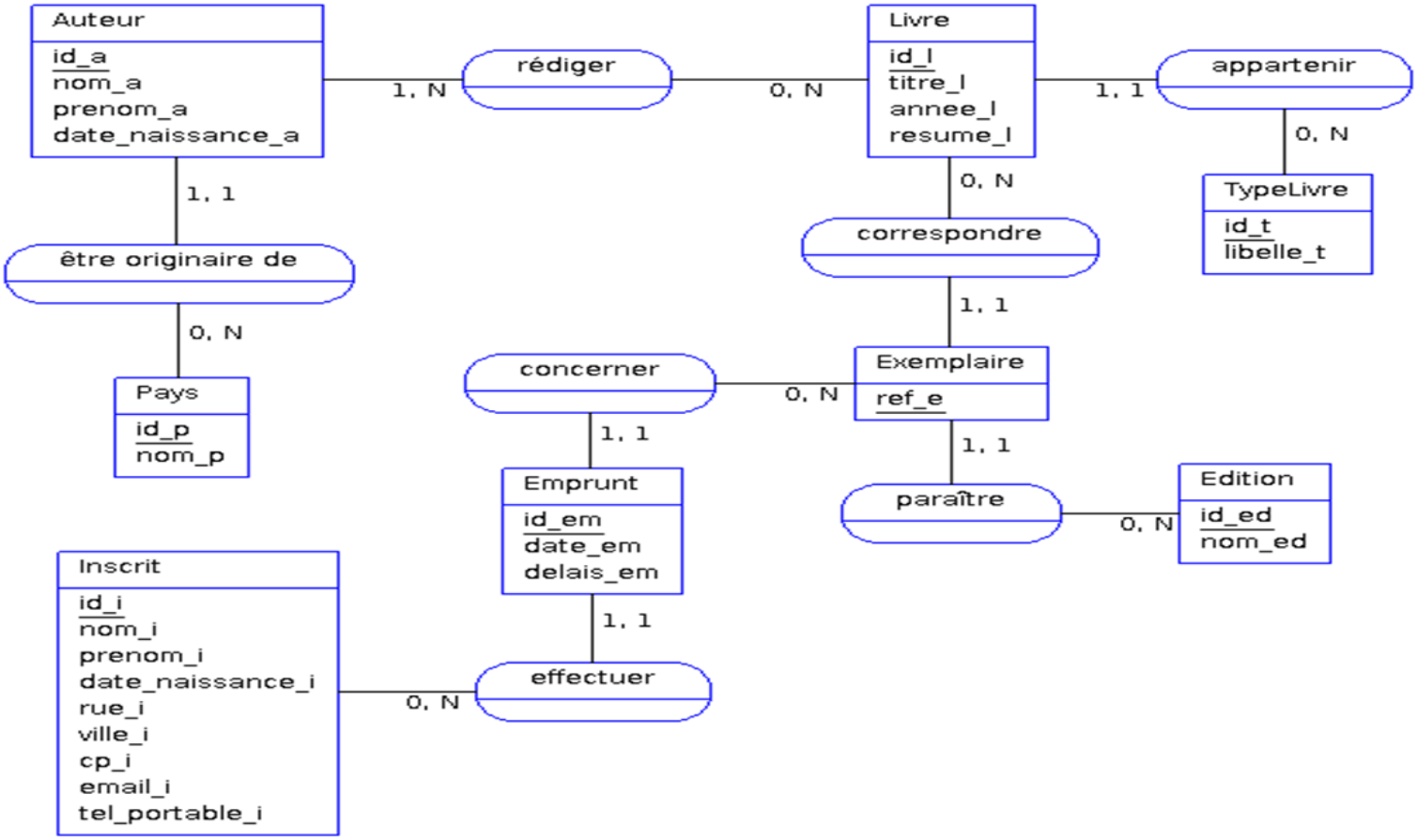
Méthodes de développement de logiciels: Historique

- **Années 70 - Méthodes orientées vers les fonctionnalités des systèmes : Verbe**
  - Décomposer le système par les fonctions ou les actions qu'il doit réaliser
  - Se concentrer sur les traitements
  - Un module = unité de traitement ou sous-programme
  - Pas de masquage d'information
  - **Exemple** : SADT, SART



Méthodes de développement de logiciels: Historique

- **Années 80: Méthodes systémiques : Sujet**
  - Modélisation des données du système et de leur relation: Modèle Entité/association
  - Définition des traitements de manière séparée des données
  - **Exemple:** Merise



Méthodes de développement de logiciels: Historique

- **1990-1995: Émergence des méthodes orientées objets : Sujet + Verbe = Objet**
  - Concevoir les systèmes en se basant sur les objets qu'ils manipulent et non sur ce qu'ils fonts ou la fonction qu'ils réalisent
  - Modélisation des systèmes en considérant de manière unifiée les données et les traitements
  - **Exemple :** OMT (Rumbaugh), OOA (Jacobson), OOD (Booch), OOSE, Fusion, …

Aucune méthode ne s'est réellement imposée

- **1995-1997: Unification et normalisation des méthodes orientées objets**

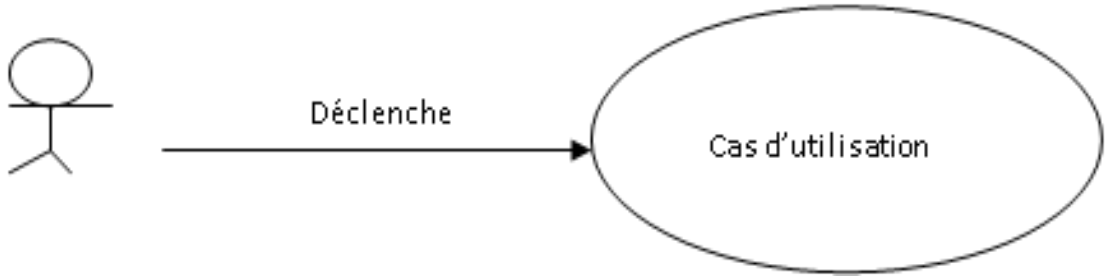
  Un travail de synthèse et de fusion des méthodes dominantes a conduit à l'élaboration d'un langage de modélisation: Unified Modeling Language (UML): un standard incontournable

UML: Généralités

- Un langage de modélisation graphique pour décrire les aspects statiques et dynamiques des systèmes: plusieurs diagrammes ou modèles graphiques
- Le concept d'objet est le noyau central du langage
- Peut être utilisé dans toutes les étapes du cycle de vie
- Langage simple et facilement « accessible » aux non informaticiens
- Attention: UML n'est pas une méthode. Donc on peut l'utiliser en s'appuyant sur les étapes et les conseils proposés par les méthodes orientées objets existantes

UML : une vue synthétique des différents diagrammes

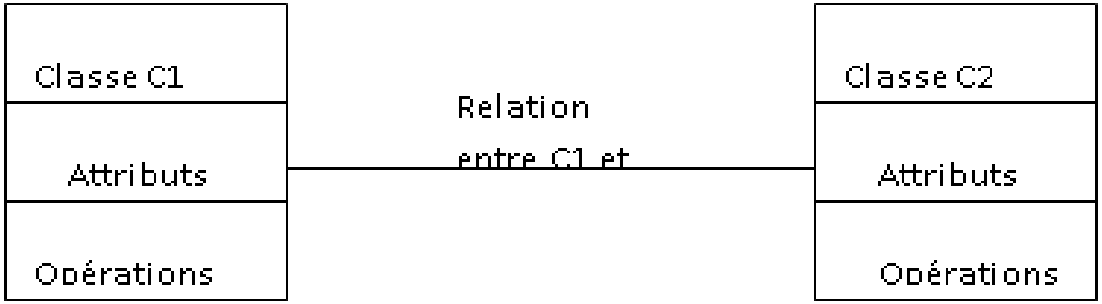
- **Diagramme des cas d'utilisation :** représentation et structuration des besoins fonctionnels des utilisateurs



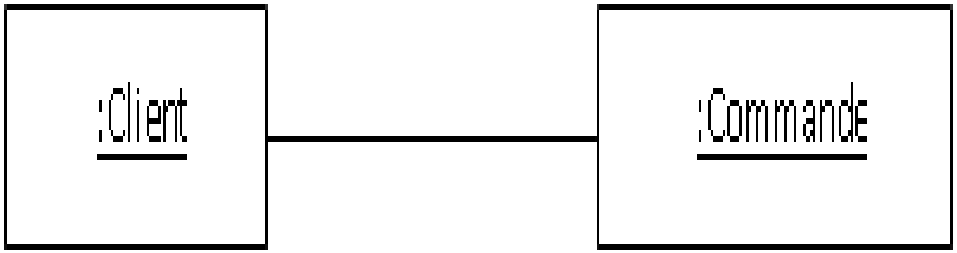
Acteur **Service ou fonctionnalité**

UML : une vue synthétique des différents diagrammes

- **Diagramme de classes :** représentation de la structure du système en terme de classes et de relations entre classes

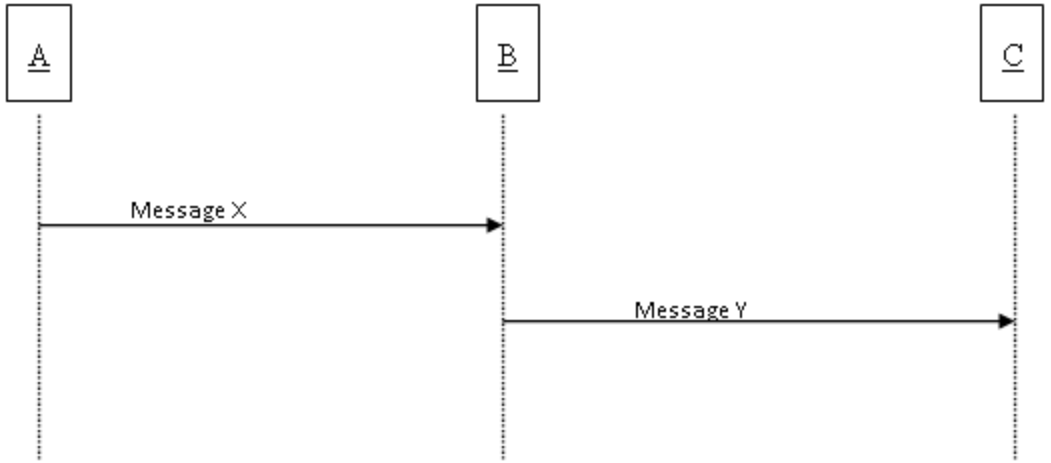


- **Diagramme d'objets:** représentation des objets (instances des classes du modèle précédent) et de leurs liens

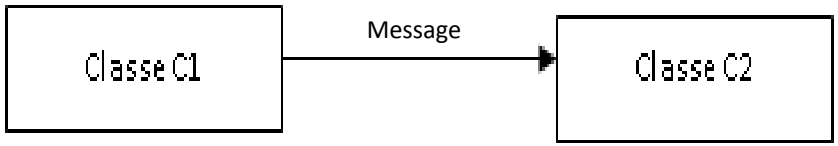


UML : une vue synthétique des différents diagrammes

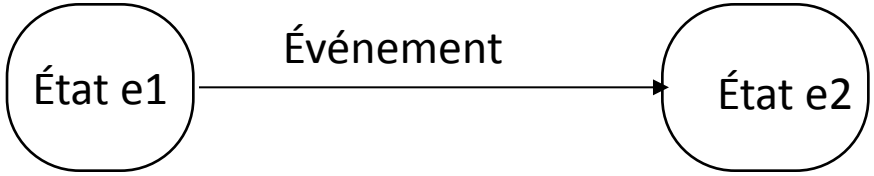
- **Diagramme de séquence:** représentation temporelle des objets et de leurs interactions



- **Diagramme de collaboration:** représentation spatiale des objets, des liens et des interactions

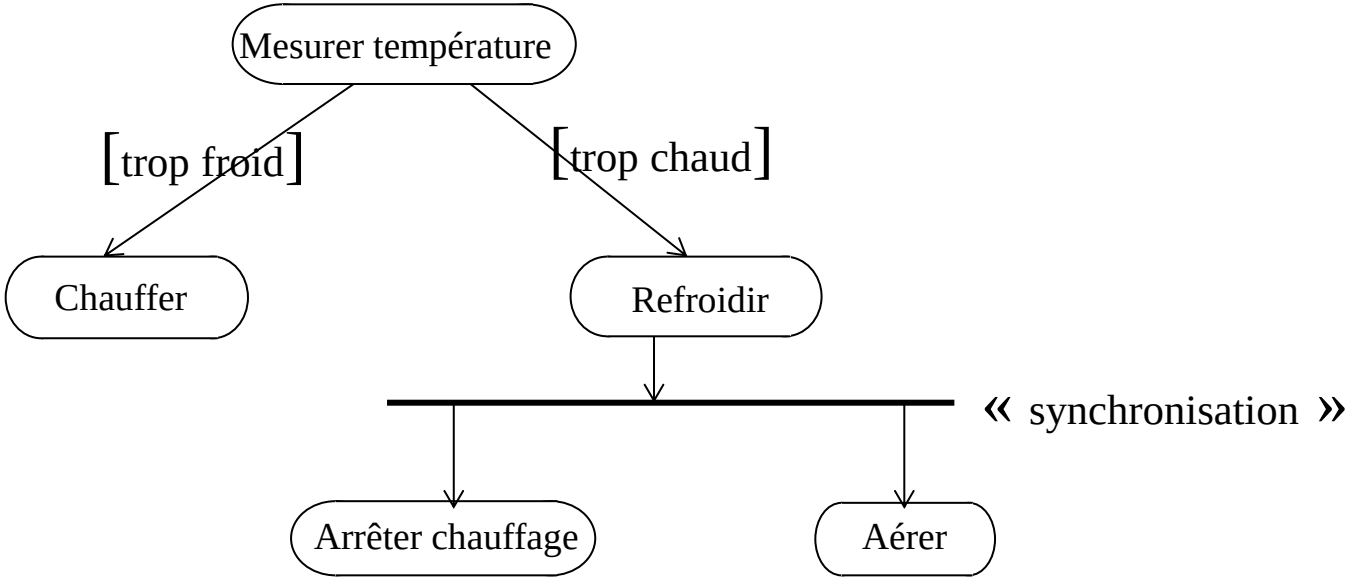


- **Diagramme d'états-transitions:** représentation du comportement d'une classe en terme d'états et de transitions entre états



UML : une vue synthétique des différents diagrammes

- **Diagramme d'activités:** représentation du comportement d'une opération en terme d'actions



- **Diagramme de composants:** représentation de l'architecture logicielle du système en termes de modules

- **Diagramme de déploiement:** représente l'architecture matérielle

UML: place des diagrammes dans le cycle de vie

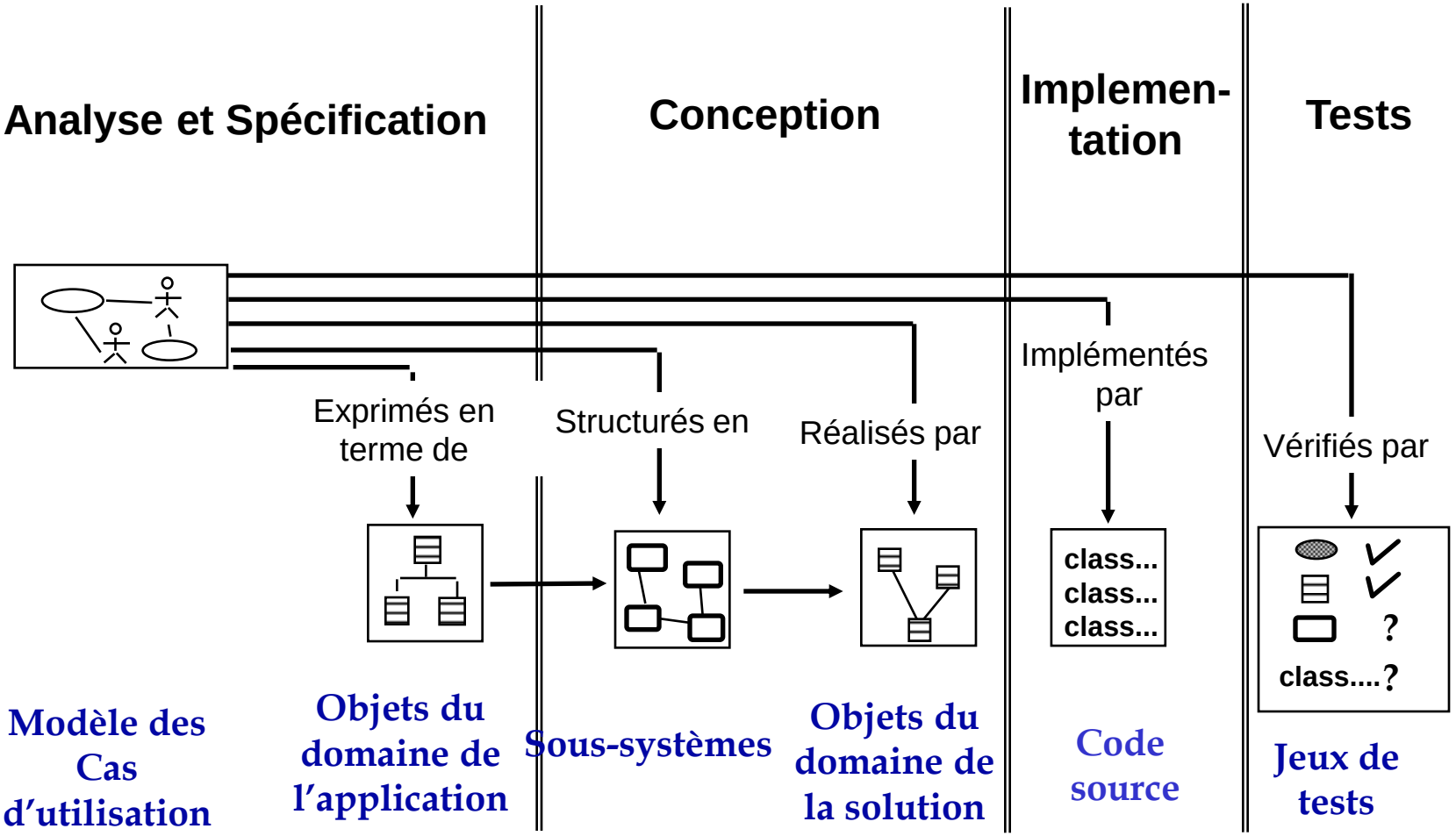


Diagramme des cas d'utilisation

- **Cas d 'utilisation :**
  - représente un ensemble d 'actions réalisées par le système et produisant un résultat observable intéressant pour un acteur externe
  - Peut être vu comme correspondant à un ensemble de scénarios d'utilisation du système

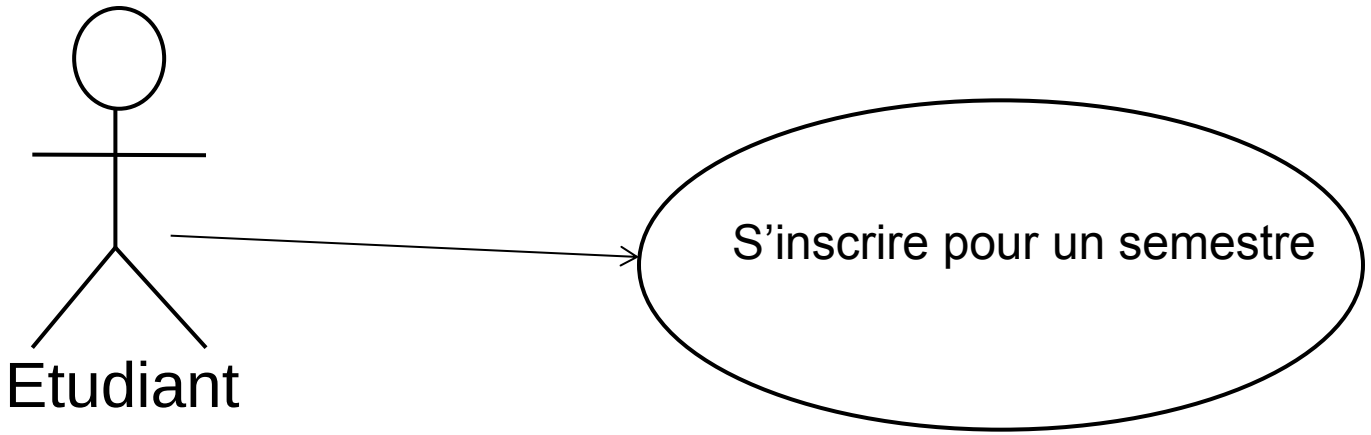


Diagramme des cas d'utilisation

- **Diagramme des cas d'utilisation:**
  - Permet de modéliser les services que le système doit proposer à ces utilisateurs (toute entité ou acteur interagissant avec le système)
  - C'est le diagramme utilisé pour modéliser les besoins (Étape d'analyse et de spécification des besoins)

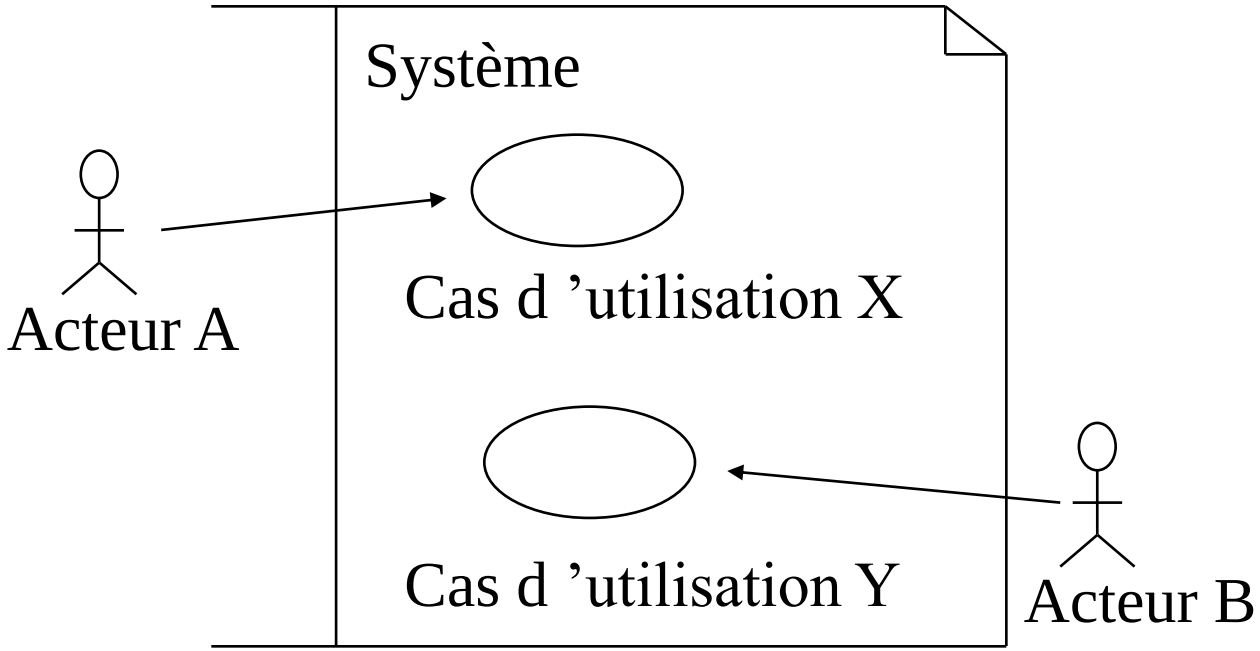


Diagramme des cas d'utilisation

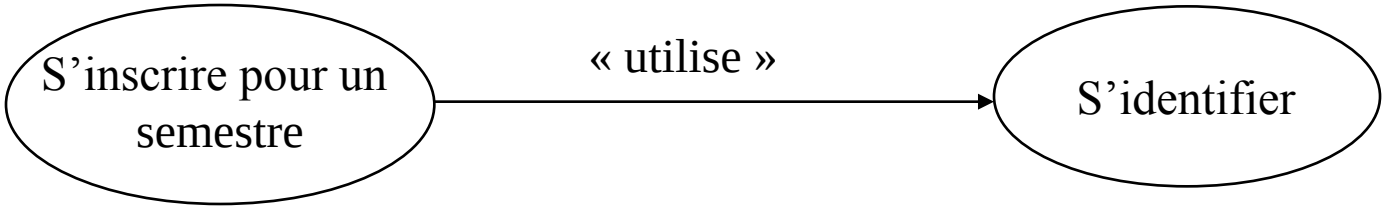
- **Acteur:**
  - Représente un rôle joué par une personne ou une entité qui interagit avec le système
  - Une même personne physique peut jouer le rôle de plusieurs acteurs

- **4 grandes catégories d 'acteurs:**
  - Acteurs principaux: Personnes qui utilisent les fonctions principales du système
  - Acteurs secondaires: Personnes qui effectuent des tâches administratives ou de maintenance
  - Matériel externe: Dispositifs matériels incontournables qui font partie du domaine d'application et qui doivent être utilisés
  - Autres systèmes avec lesquels le système doit interagir

tout acteur doit communiquer avec le système

Diagramme des cas d'utilisation

- **Relation d 'utilisation**: signifie qu'une instance du cas d'utilisation source comprend également le comportement décrit par le cas d'utilisation destination



- **Relation d 'extension**: signifie que le cas d 'utilisation source étend le comportement du cas d 'utilisation destination

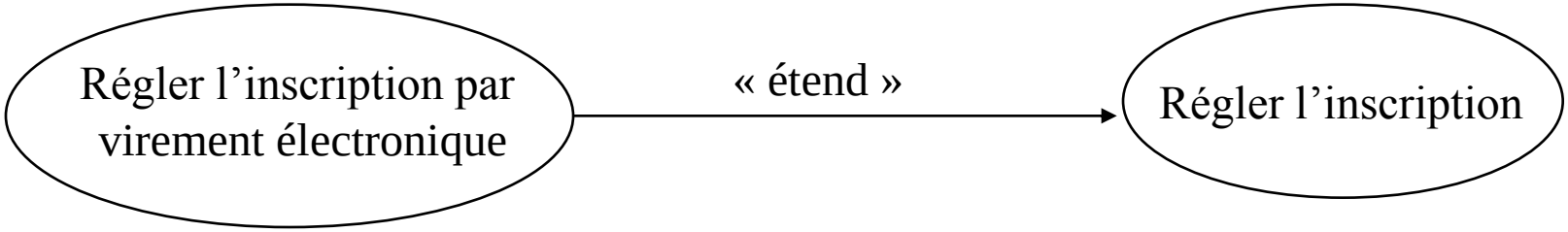


Diagramme des cas d’utilisation: notion de scénario

- Un scénario est une instance du cas d ’utilisation
- Le scénario se représente par une interaction entre les objets

Diagramme des cas d'utilisation: Aspect méthodologique

Etape 1

• ***Identification des acteurs*** **:** abstraction d 'un rôle joué par des entités externes qui interagissent avec le système.

- Utilisateurs humains, administrateur, opérateur de maintenance.
- Autres systèmes connexes qui interagissent avec le système étudié.

• ***Identification des messages*** **:** spécification d 'une communication entre objets qui transporte de l 'information avec l 'intention de déclencher une activité chez le Receveur (ici, c'est le système)

• ***Réalisation d 'un diagramme de contexte*** **:** représentation synthétique du système dans son environnement

Diagramme des cas d'utilisation: Aspect méthodologique

Etude de cas :Système de gestion d 'une bibliothèque municipale

Acteurs	**Evénements**
• Bibliothécaire	Classifie document
• Client	s 'inscrit, emprunte document loue livre, réserve livre, paie amande consulte index, retourne document
• Lecteur optique	lit code carte, lit code document

Diagramme des cas d'utilisation: Aspect méthodologique

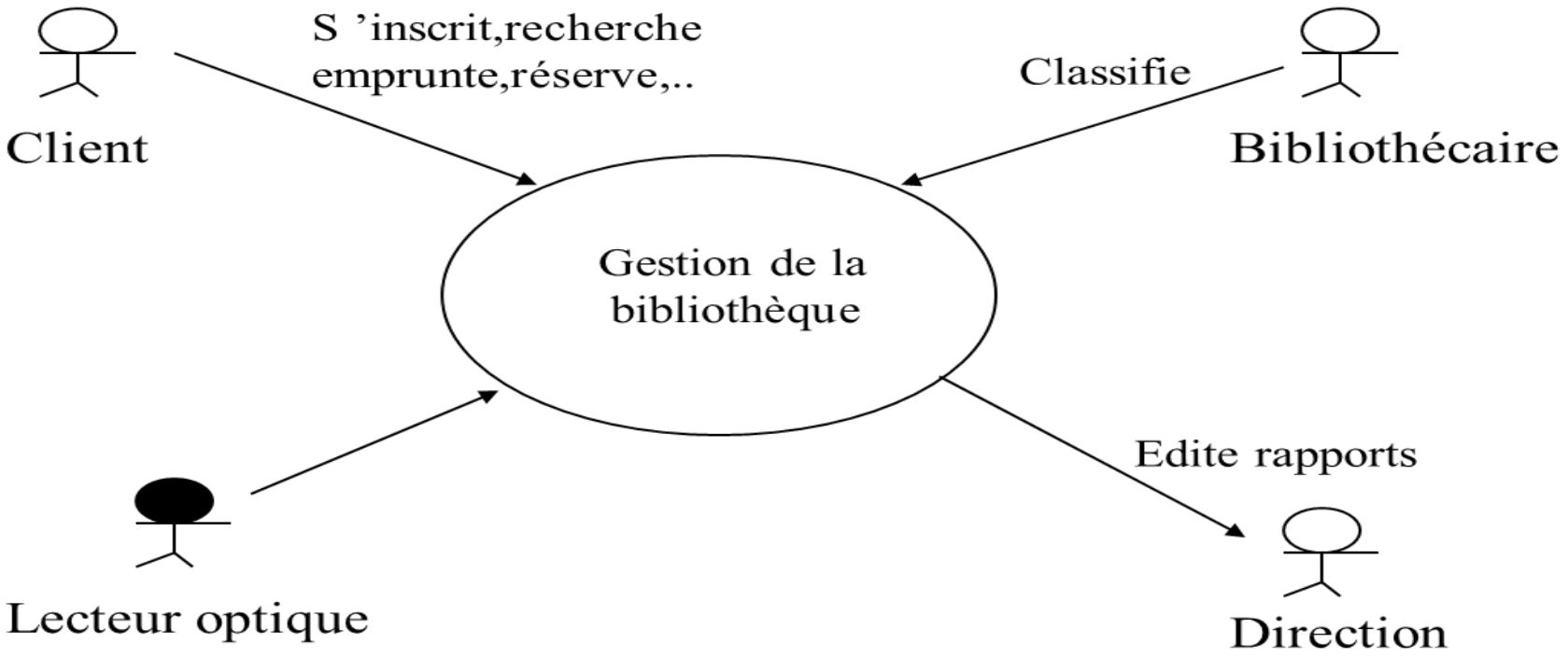


Diagramme de contexte

Diagramme des cas d'utilisation: Aspect méthodologique

Etape 2

• *Identifier les cas d 'utilisation*

- Analyser l 'intention fonctionnelle de l 'acteur par rapport au système dans le cadre de l 'émission ou de la réception de chaque message,

- Regrouper les intentions fonctionnelles en unités cohérentes.

• *Organiser les cas d 'utilisation*

•*Identifier les classes candidates*

Diagramme des cas d'utilisation: Aspect méthodologique

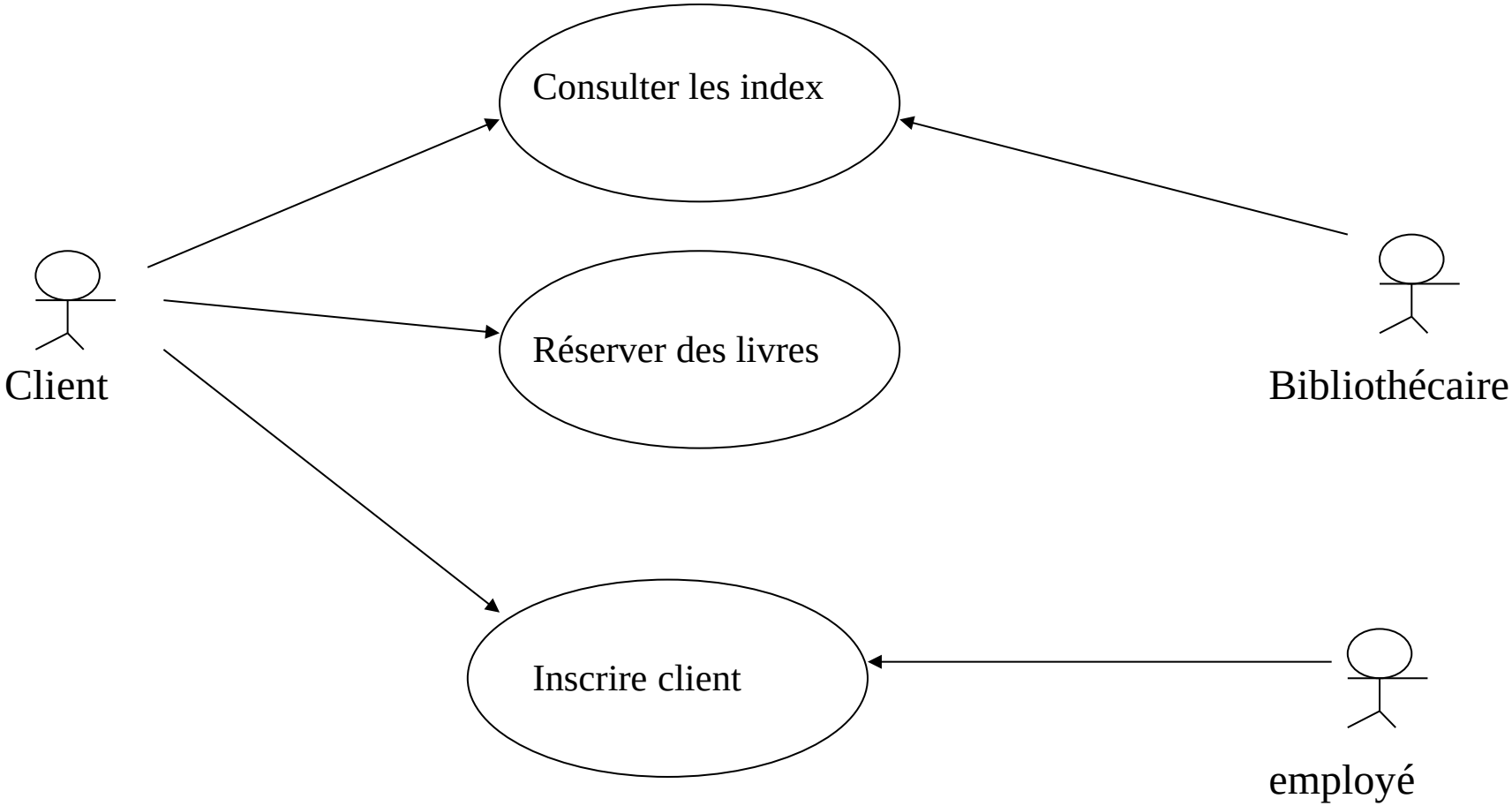


Diagramme des classes

- Montre la structure du système sous la forme d'un ensemble de classes et de relations entre ces classes
- Le pilier des étapes de conception et de réalisation du système: spécifie la structure et les liens entre les objets qui composent le système
- Les concepts: Classe, Association

Diagramme des classes: Objet et Classe

- Un **objet** est un concept, une abstraction ou une chose ayant un sens précis dans le contexte du problème.

- Une **classe** décrit un groupe d'objets ayant :
  - des propriétés similaires : attributs,
  - un comportement commun: opérations,
  - des relations communes avec les autres objets,
  - une même sémantique.

- Un **attribut** est une valeur de donnée représentant une caractéristique d'un objet
  Son existence dépend de celle de l'objet auquel il appartient

- Une **opération** est une fonction applicable aux objets d'une classe.
  Toute opération a un objet cible.

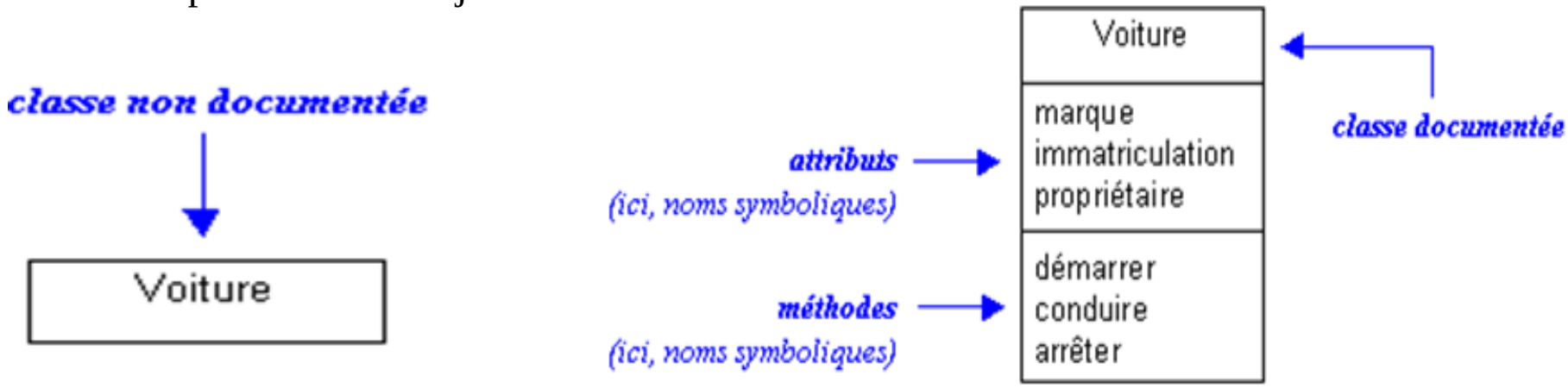


Diagramme des classes: Lien et Association

- Un **lien** est une relation physique ou conceptuelle entre des objets
  C'est une instance d'association.

- Une **association** est un groupe de liens ayant une structure et une sémantique communes
  - fondamentalement bidirectionnelle
  - peut être binaire, ternaire,...
  - toute relation entre classes doit être modélisée sous forme d'association et non pas par un attribut de type pointeur.

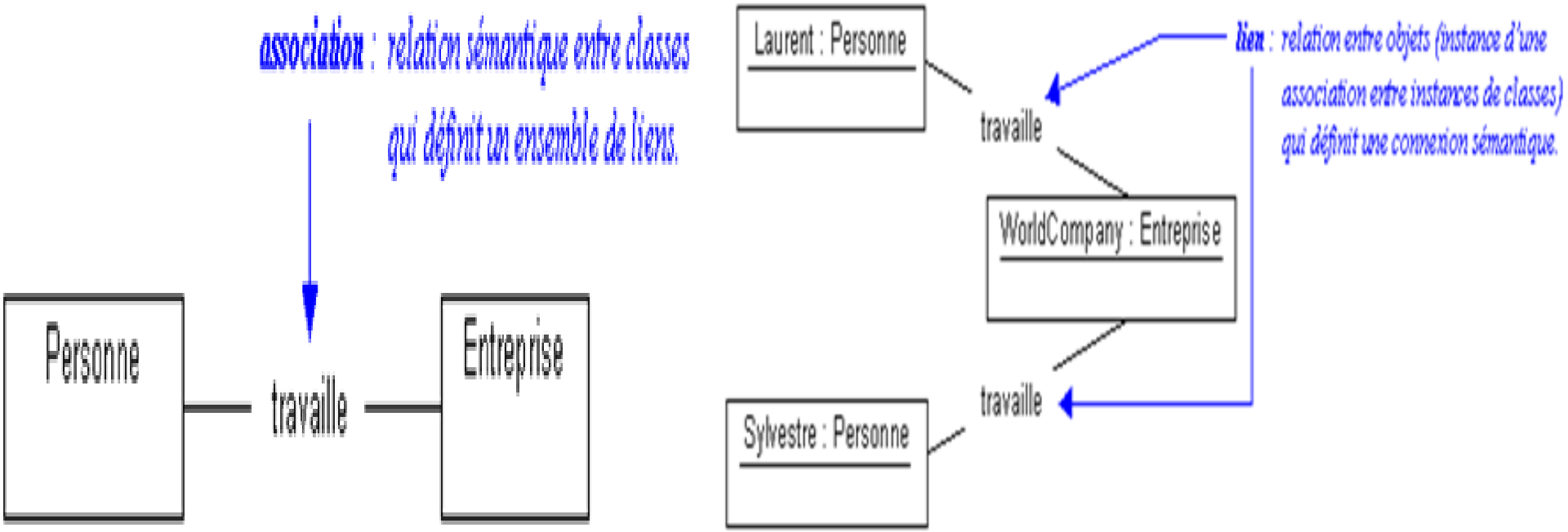
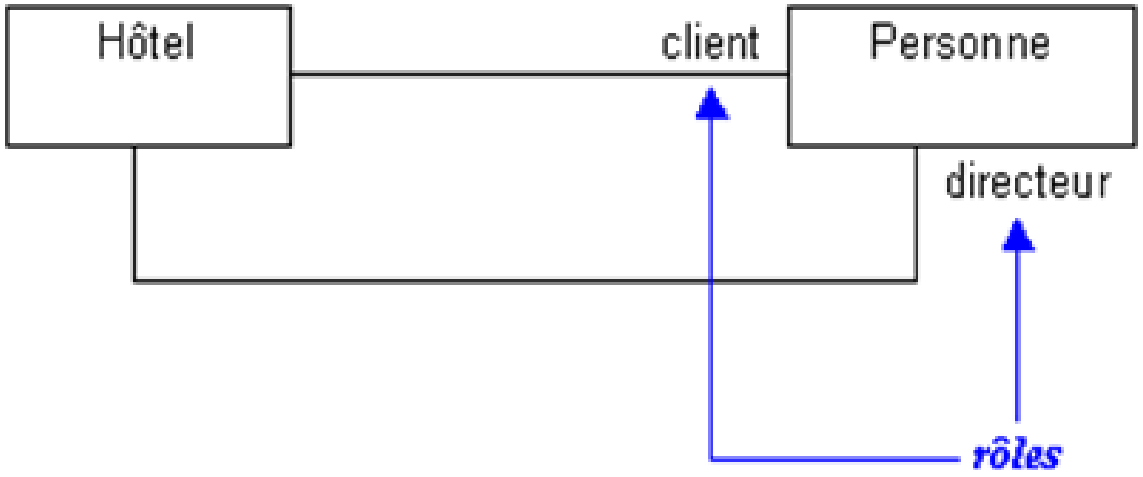


Diagramme des classes: Rôle

- - Rôle : identifie de façon unique une extrémité d'une association lorsque la même classe participe plusieurs fois à la même association.



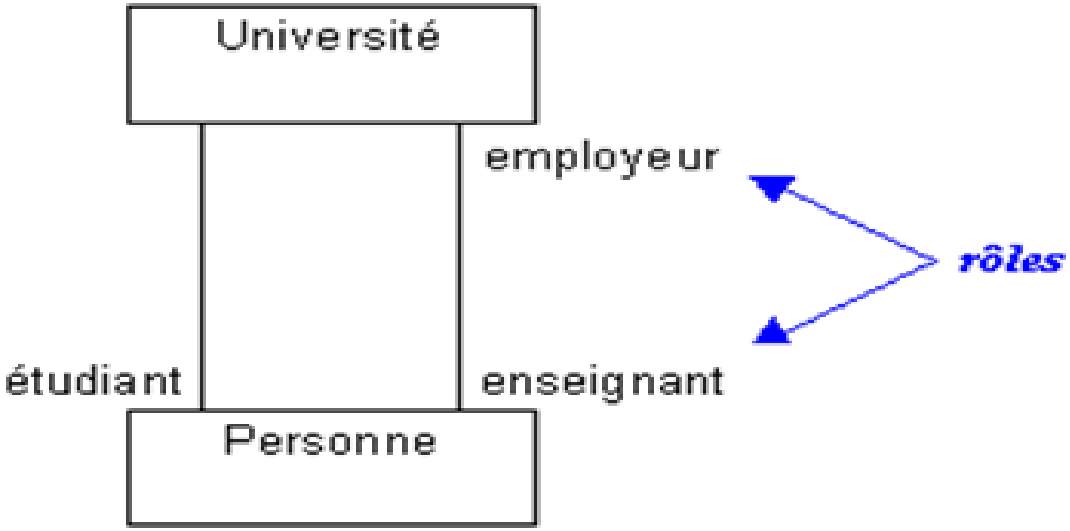


Diagramme des classes: Cardinalité

- **Cardinalités** : précise le nombre d'instances qui participent à une relation

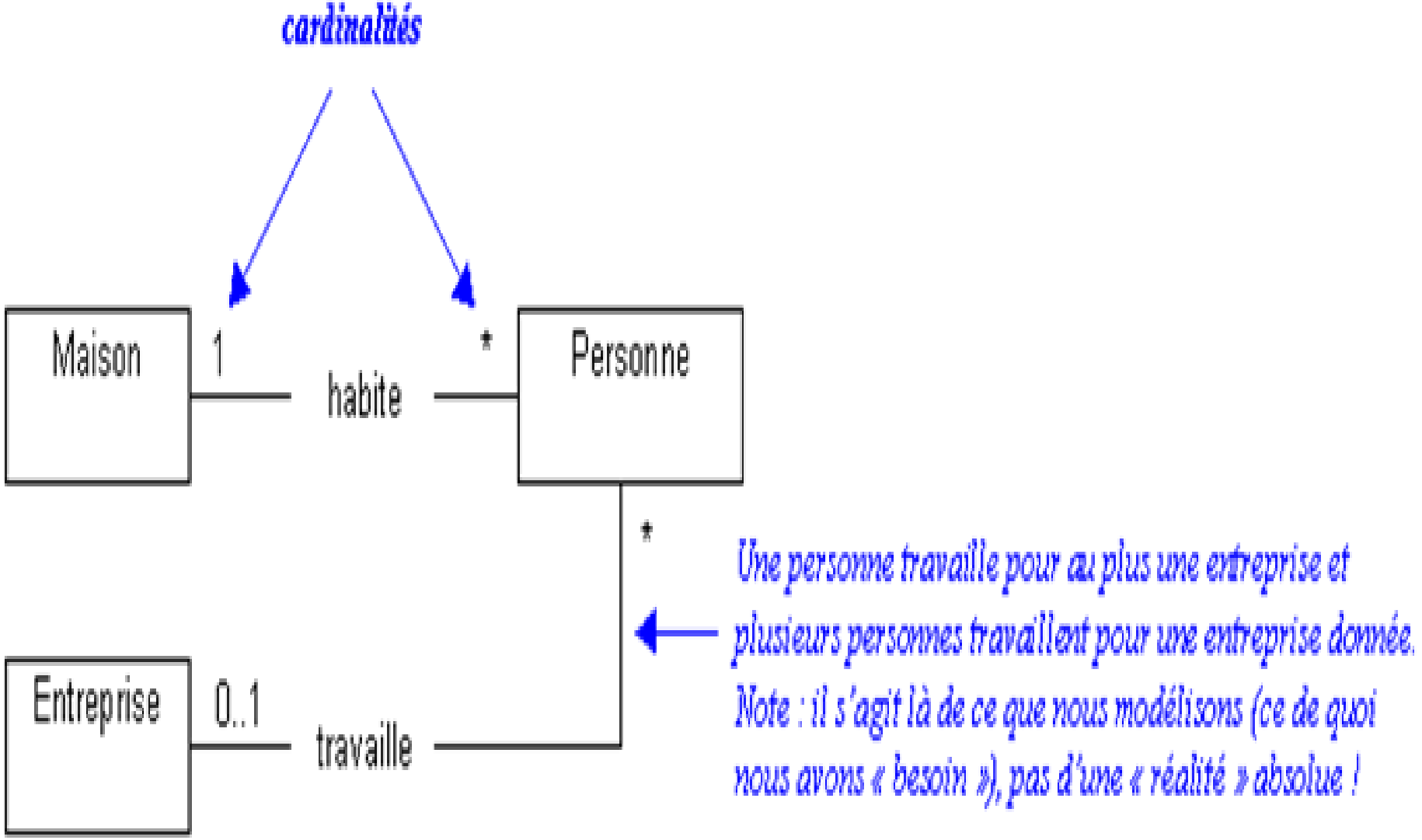


Diagramme des classes: Agrégation/Composition

- **L'agrégation: une forme particulière d'association**

  - Exprime une association de type "composé-composant" ou "partie de": une relation entre un ensemble et ses parties

  - Propriétés : transitivité, non-symétrie, propagation de certaines propriétés

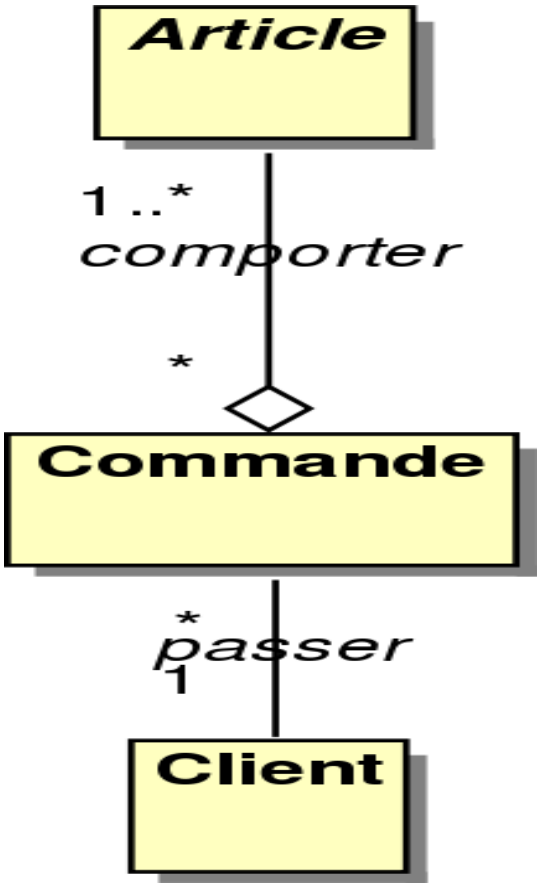


Diagramme des classes: Agrégation/Composition

- La relation de composition décrit une contenance structurelle entre instances. On utilise un losange plein: **agrégation forte** .

- La destruction et la copie de l'objet composite (l'ensemble) impliquent respectivement la destruction ou la copie de ses composants (les parties).

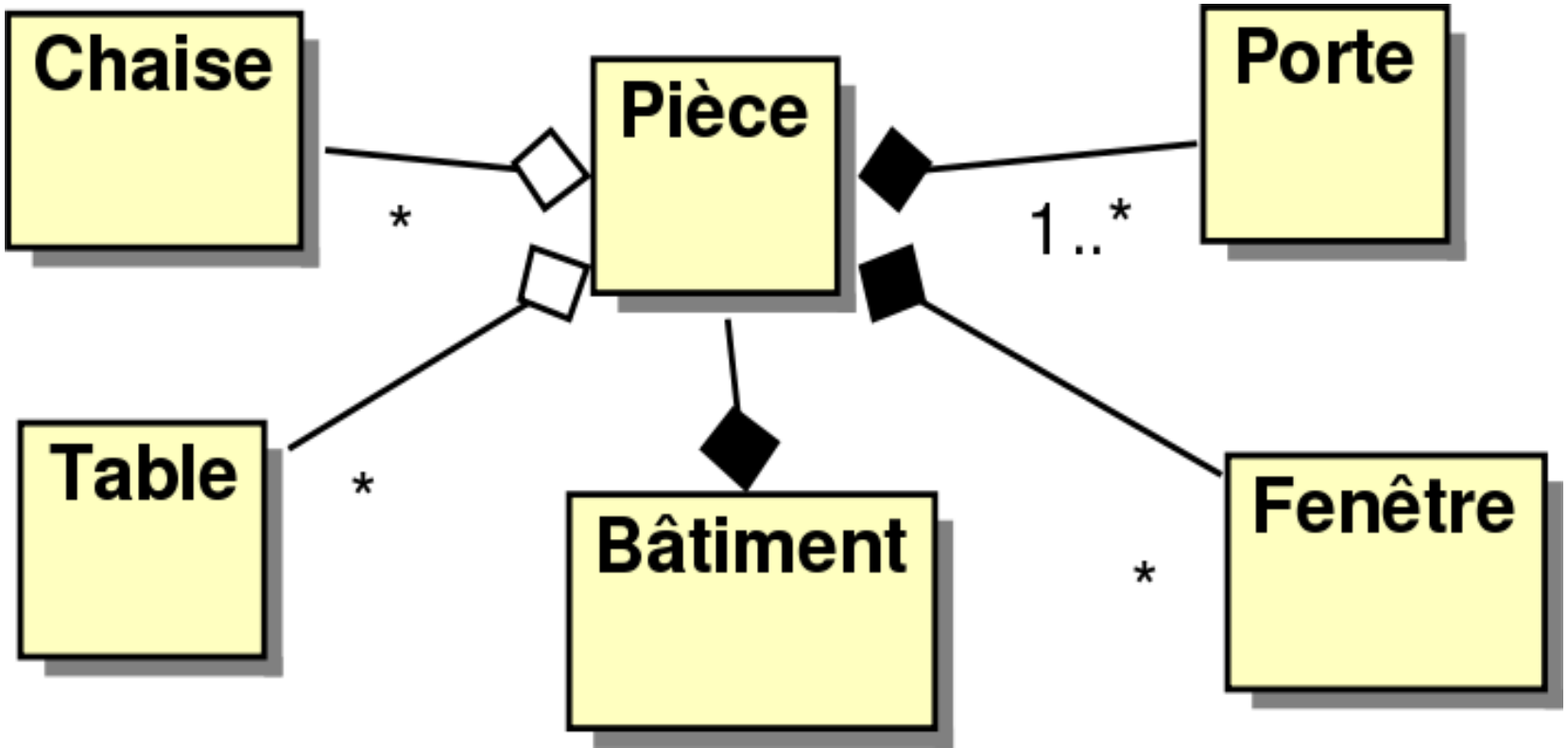


Diagramme des classes: Héritage

Généralisation et héritage

- Constituent des abstractions puissantes permettant aux classes de partager leurs points communs tout en préservant leurs différences.
- La généralisation est la relation entre une classe (super-classe) et une ou plusieurs versions affinées (sous-classes) de cette classe
- Une sous-classe hérite les attributs et les opérations de sa super-classe.
- Une instance de sous-classe est aussi une instance de toutes les classes ancêtres.

Propriétés : Transitifs

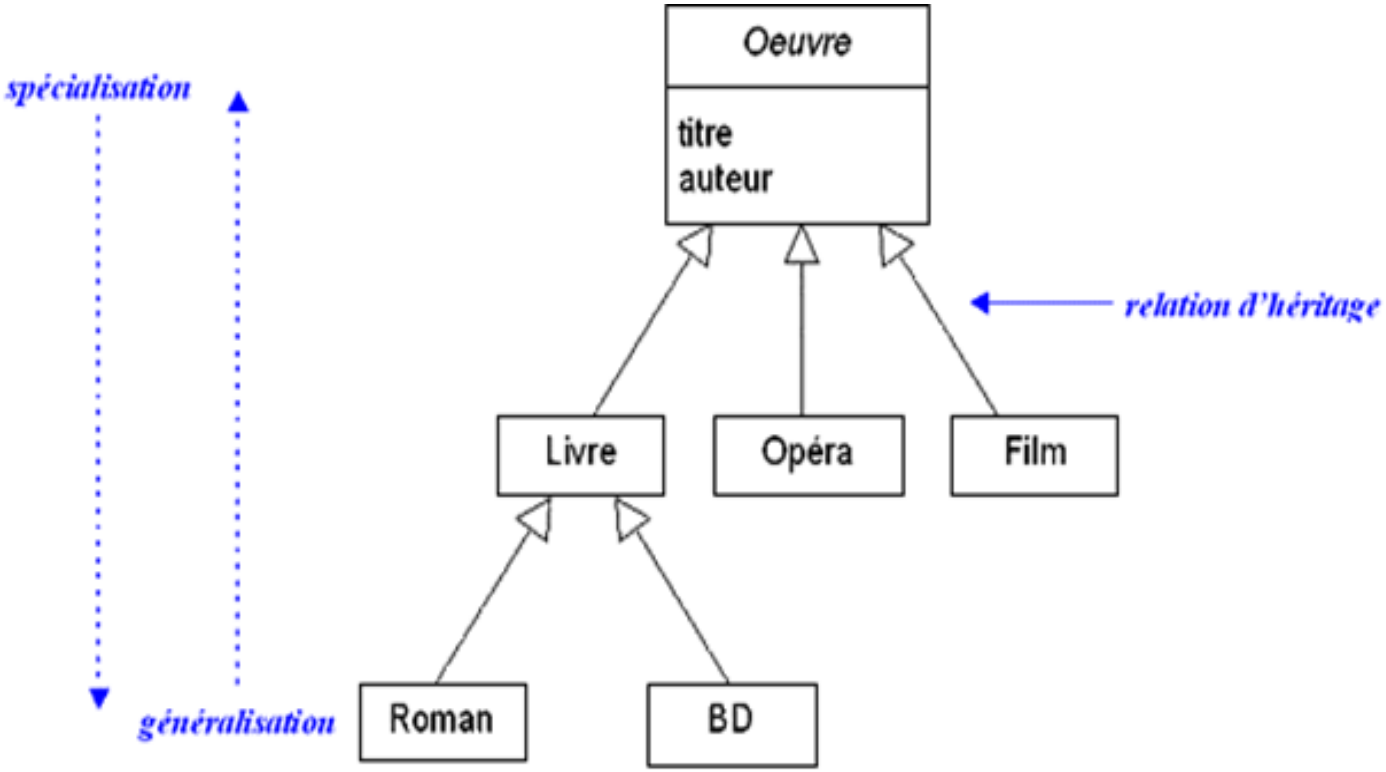


Diagramme des classes: quelques conseils

- Trouver les classes du domaine étudié: concepts et substantifs du domaine.
- Trouver les associations entre classes: verbes mettant en relation plusieurs classes.
- Trouver les attributs des classes: substantifs correspondant à un niveau de granularité plus fin que les classes. Les adjectifs et les valeurs correspondent souvent à des valeurs d'attributs.
- Organiser et simplifier le modèle en utilisant l'héritage
- Tester les chemins d'accès aux classes
- Itérer et raffner le modèle.

Diagramme des classes: Exemple

- Un gérant de bibliothèque désire automatiser la gestion des prêts. Il commande un logiciel permettant aux utilisateurs de *connaître* les livres présents, d'en *réserver* jusqu'à 2. L'adhérent peut connaître la liste des livres qu'il *a empruntés* ou *réservés*. L'adhérent possède un mot de passe qui lui est donné à son inscription.

- . L'emprunt *est* toujours *réalisé* par les employés qui *travaillent* à la bibliothèque. Après avoir identifié l'emprunteur, ils savent si le prêt est possible (nombre max de prêts = 5), et s'il a la priorité (il est celui qui a réservé le livre).

- Ce sont les employés qui mettent en bibliothèque les livres rendus et les nouveaux livres. Il leur est possible de connaître l'ensemble des prêts *réalisés* dans la bibliothèque

Les classes retenues:

-Gérant - non pertinente

-Bibliothèque - oui, responsabilité: gérer les livres, les adhérents, les prêts

-Gestion – vague

-prêts – oui, responsabilité: contenir les infos sur les prêts

-Logiciel – vague

-Utilisateurs – oui: idem adhérent, emprunteur

-Livres – oui, responsabilité: permettre de connaitre son état

-Liste – non, implantation ou conception

-Mot de passe – non, attribut

Etc…….

Diagramme des classes: Dictionnaire de données

- bibliothèque : organisme gérant une collection de livres qui peuvent être empruntés par ses adhérents. Une bibliothèque est gérée par ses employés.
- prêt : un prêt est caractérisé par le numéro du livre, la date, la durée. Il ne peut être fait que par un adhérent.
- livre ouvrage pouvant être emprunté.
- adhérent personne inscrite à la bibliothèque.
- employé personne travaillant à la bibliothèque.

Diagramme des classes: les associations

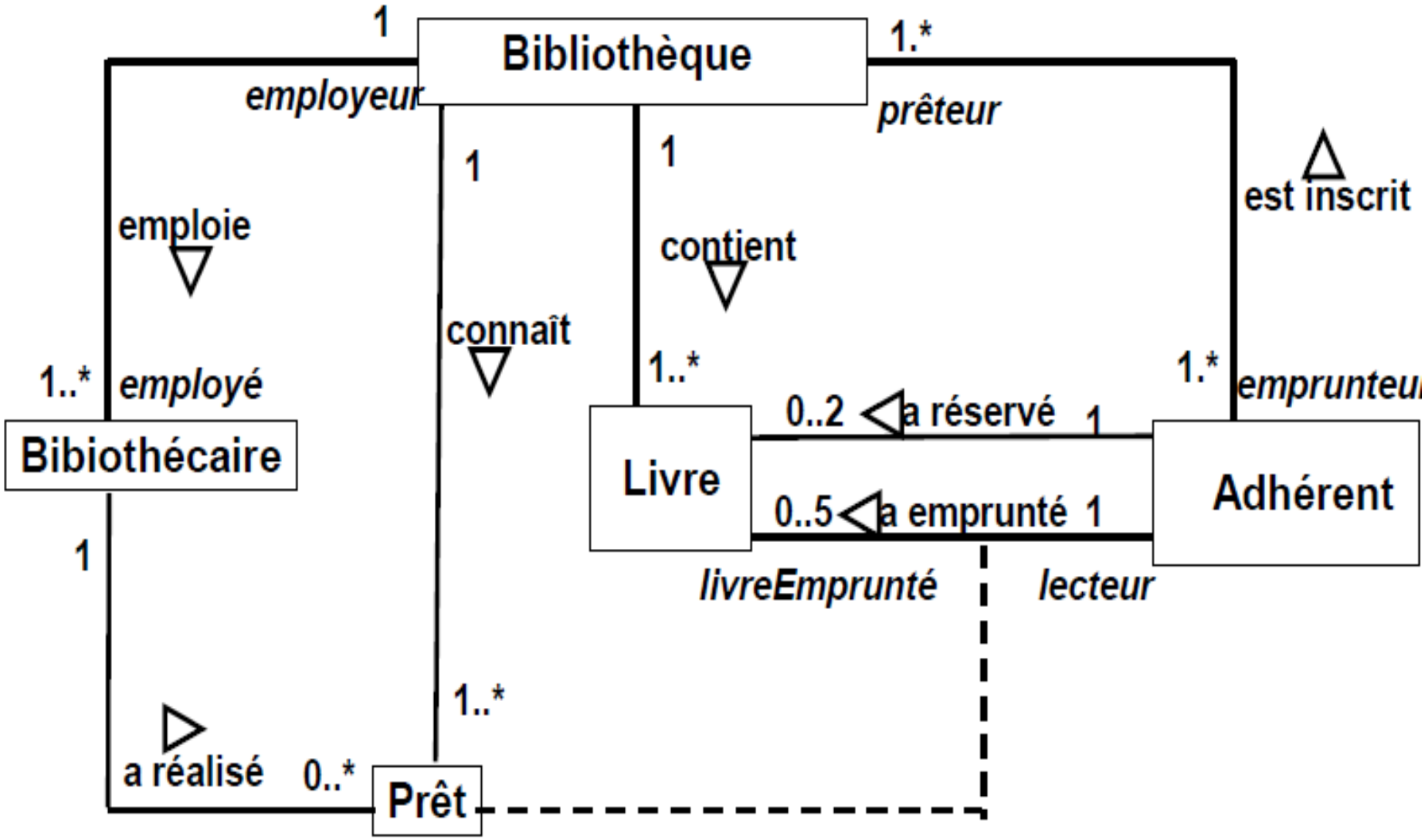


Diagramme des classes: les attributs

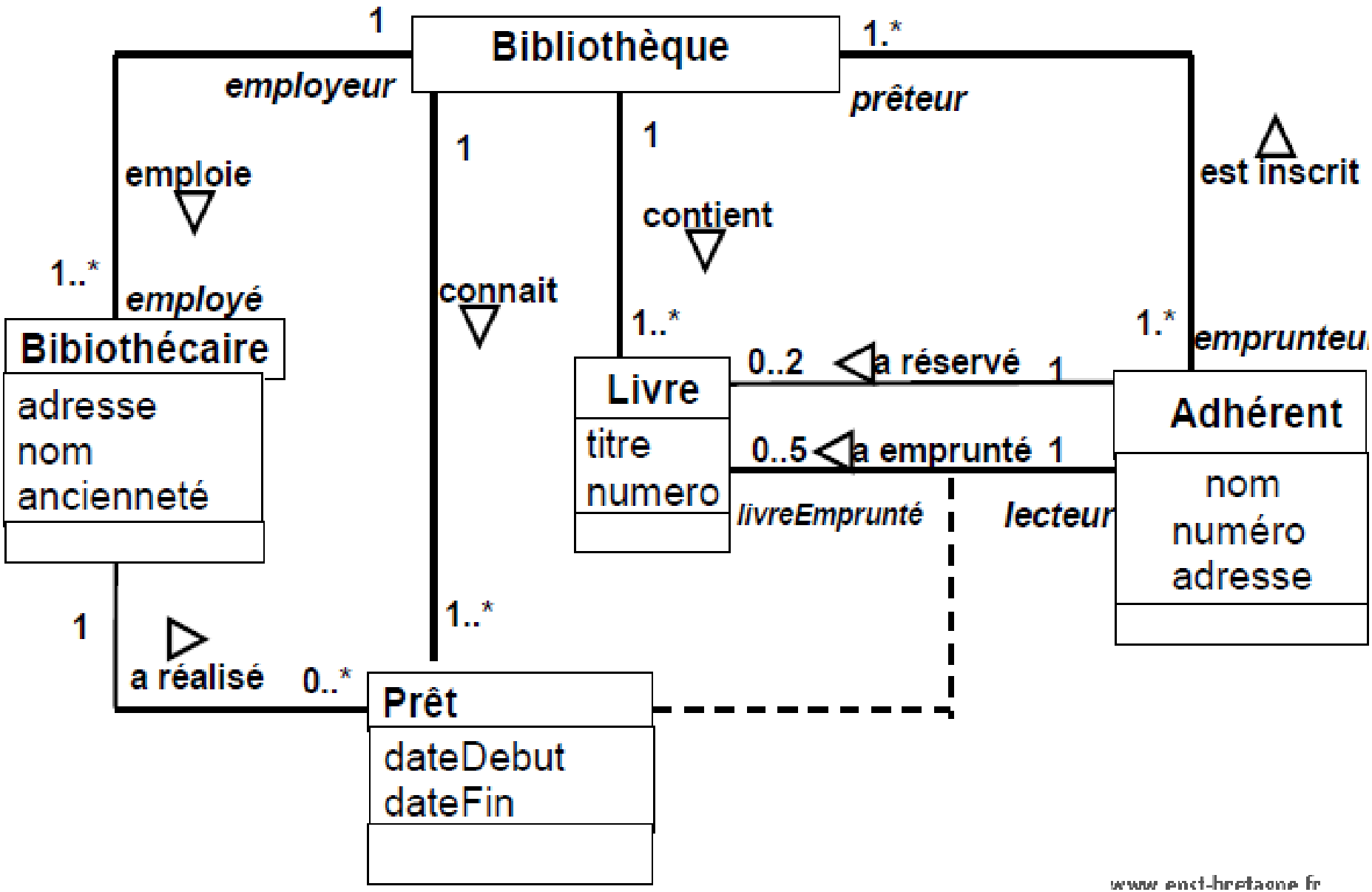


Diagramme des classes: Introduction de l'héritage

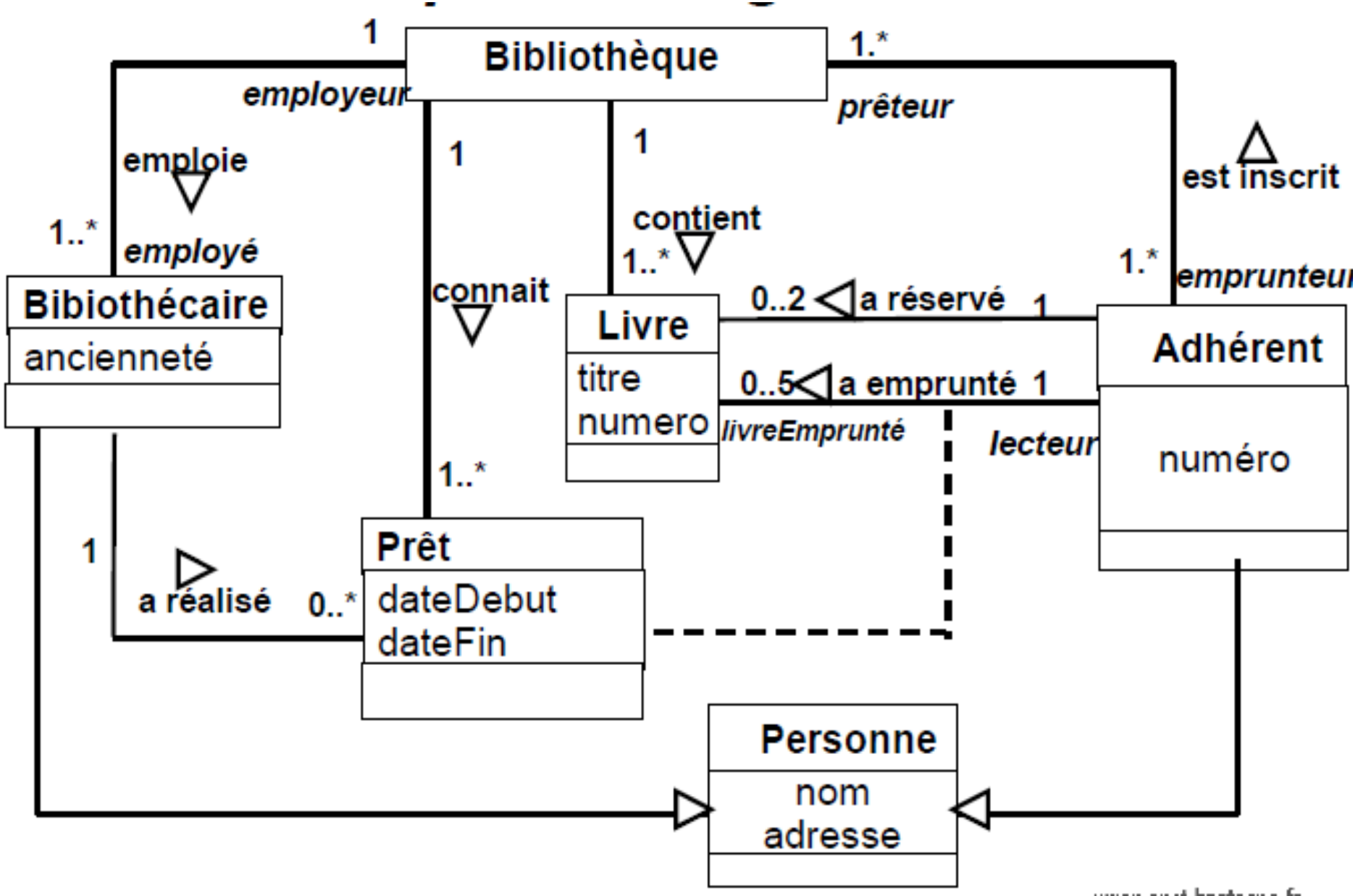


Diagramme de collaboration - Diagramme de séquences

- Représentent la **collaboration entre objets** pour
  - la réalisation d'un cas d'utilisation ou la
  - réalisation d’une opération
- . Mettent en évidence l'expéditeur et le récepteur de l'événement.
- Chaque **événement** reçu par un objet devra se traduire par une **opération** pour le gérer dans le diagramme de classes.

Diagramme des séquences

- Montre les interactions entre les objets selon un point de vue temporel

- 2 cas d ’utilisation selon cycle de vie et niveau de détail
  - Documentation des cas d ’utilisation
  - Représentation précise des interactions entre objets

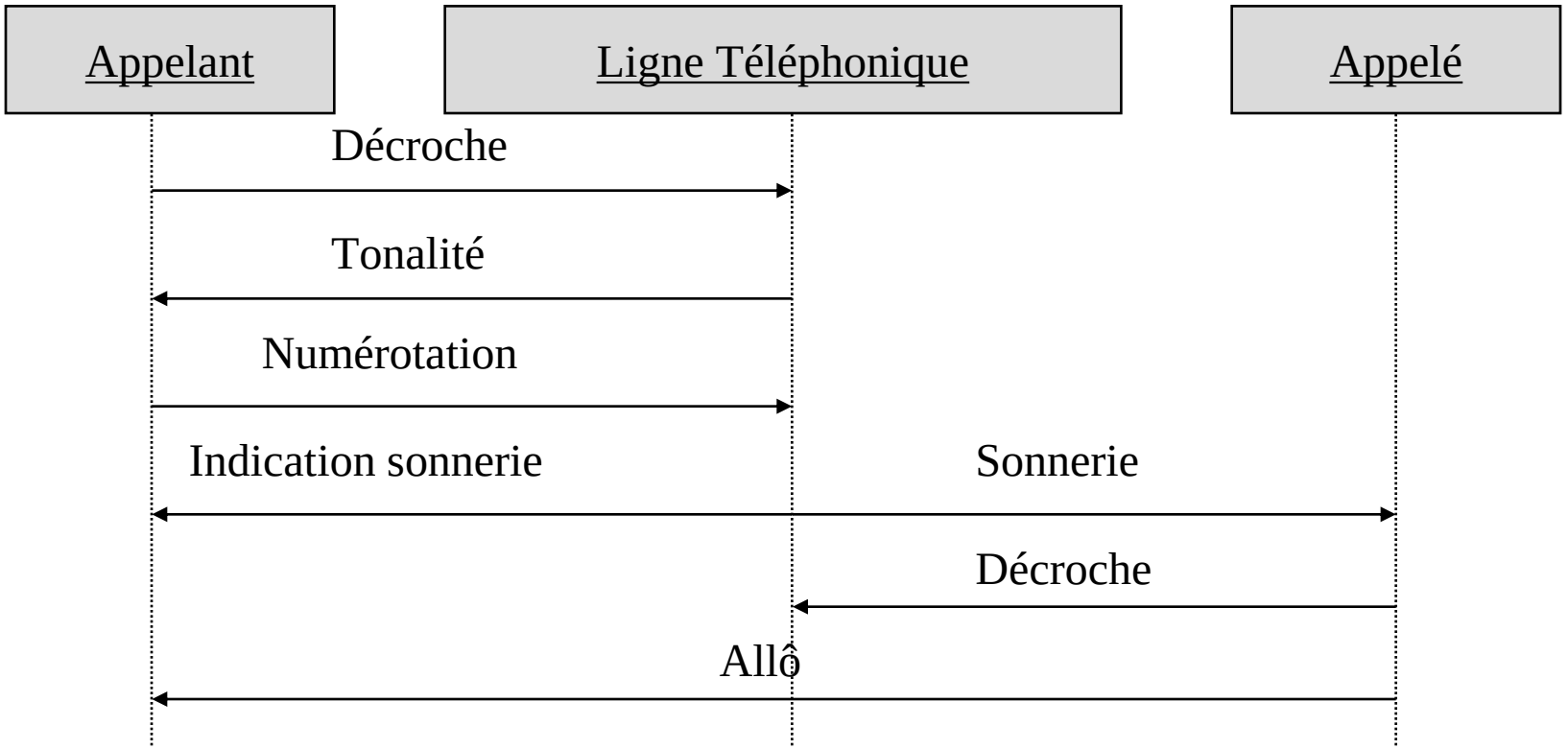


Diagramme des séquences

- Message de Création d ’objets et de Destruction d ’objets

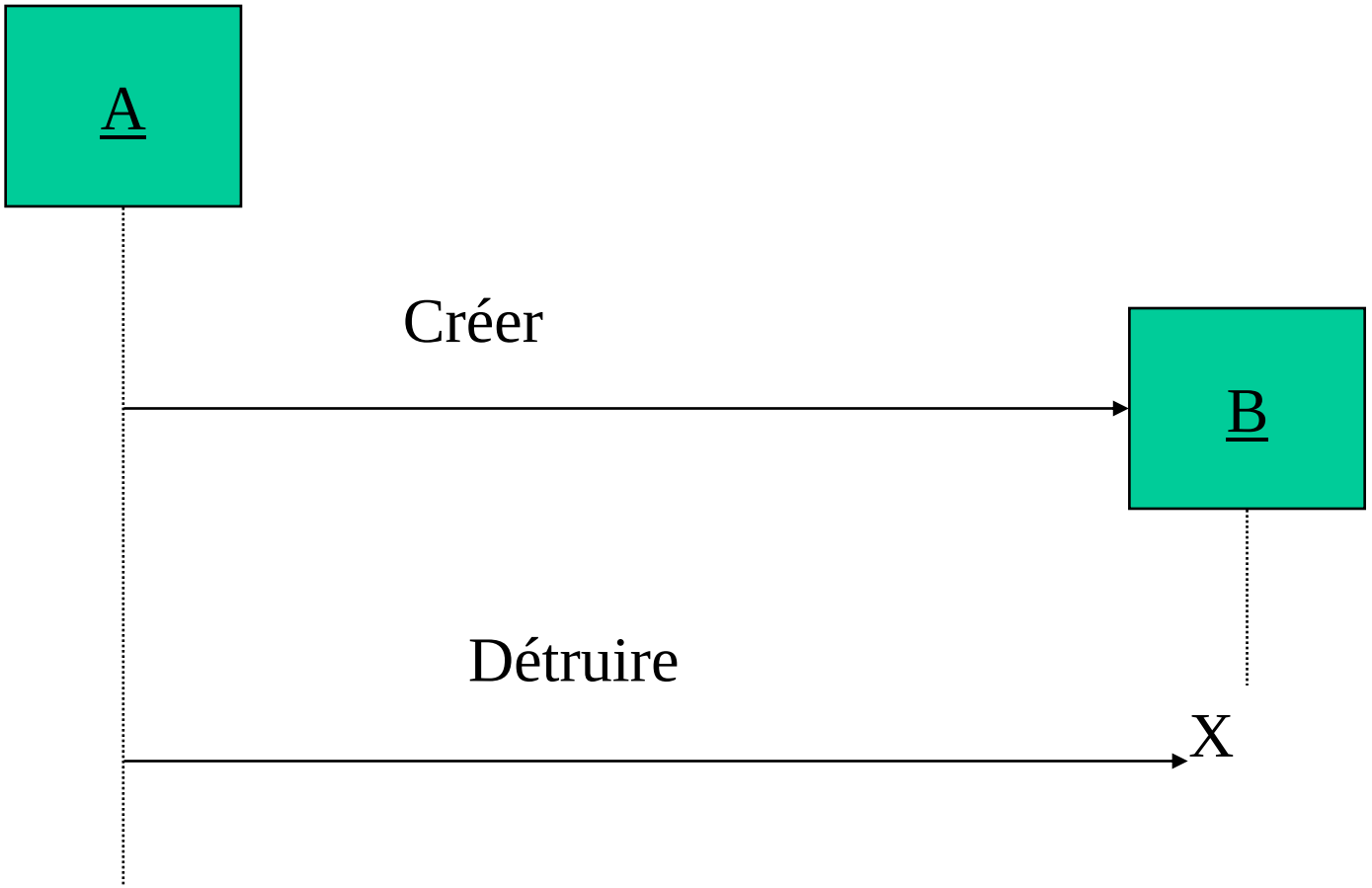


Diagramme des séquences

- Branchements Conditionnels

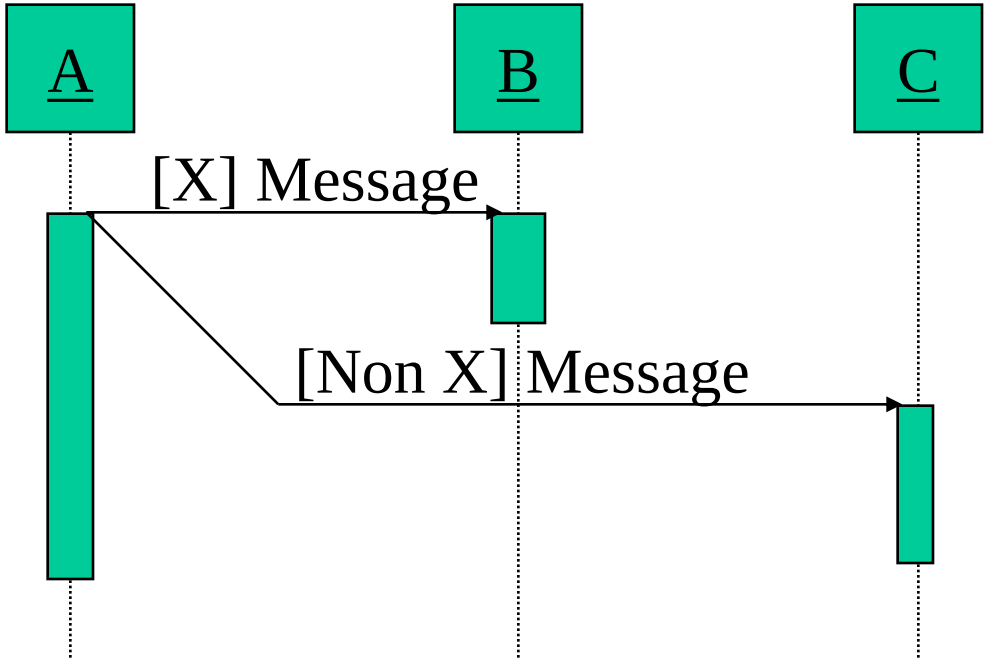


Diagramme de collaboration

- Modélise les interactions entre les objets par des échanges de messages
- Même objectif que les diagrammes de séquence, en faisant abstraction de l'aspect temporel des interactions:
  - Le temps n'est pas représenté explicitement
  - Les différents messages sont numérotés pour indiquer l 'ordre des envois

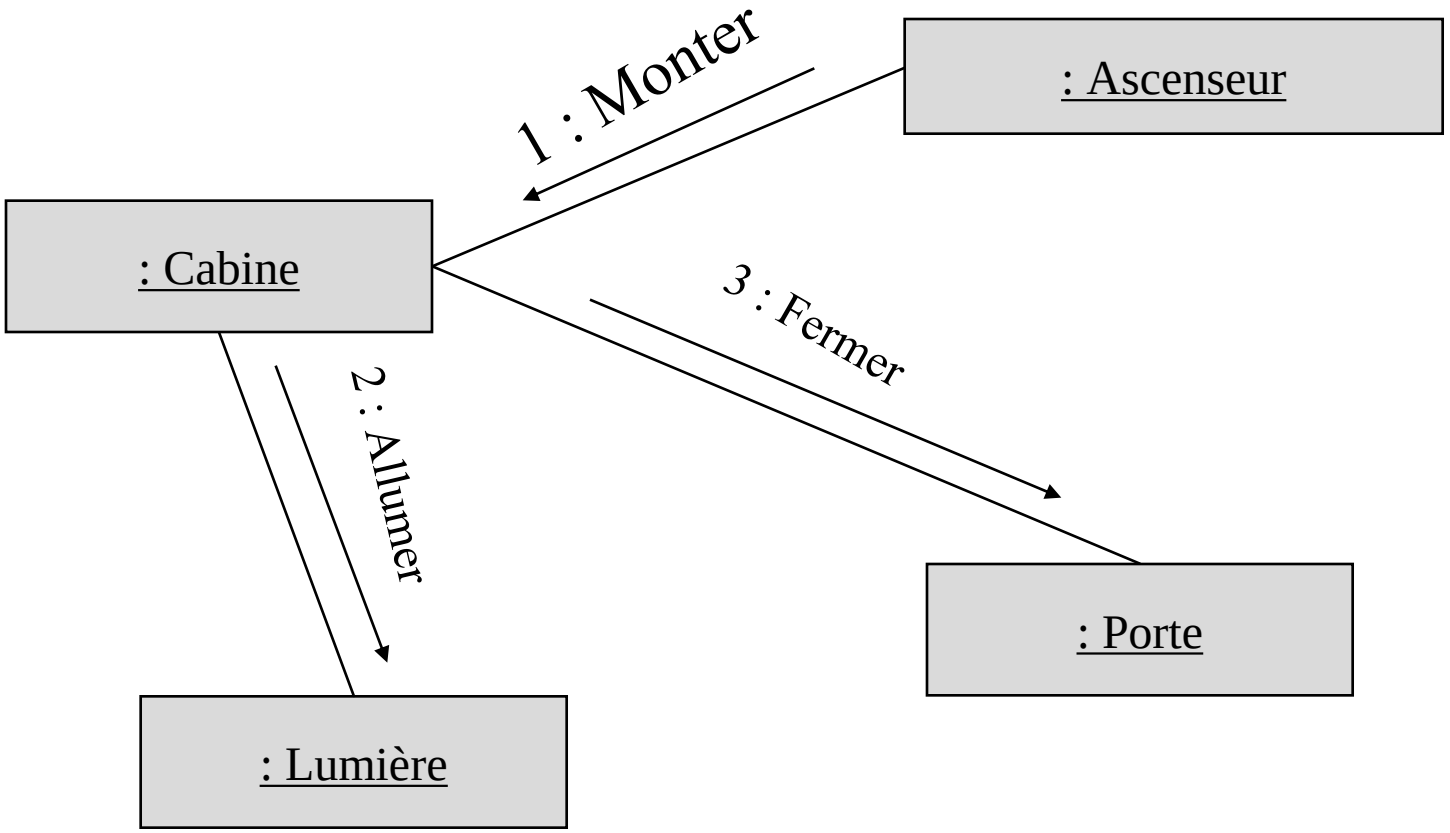


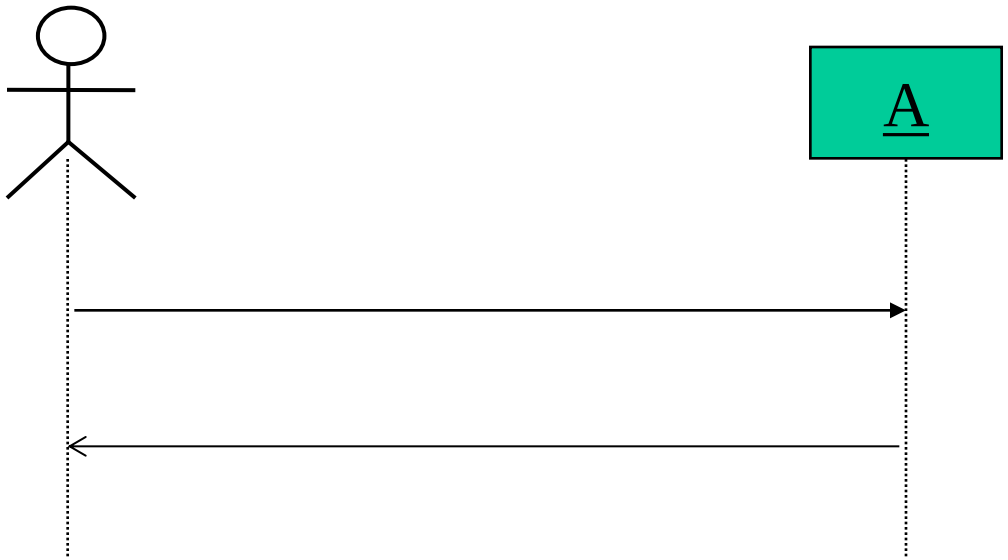
Diagramme des cas d'utilisation: Aspect méthodologique

Etape 3

- Pour chaque cas d'utilisation, choisir les scénarios significatifs
- Détailler chacun des scénarios
  - Interactions entre le système et l'utilisateur : forme textuelle ou diagramme de séquences
  - Interactions entre les objets de l'application : diagramme de séquence ou diagramme de collaboration

Diagramme des cas d'utilisation: notion de scénario

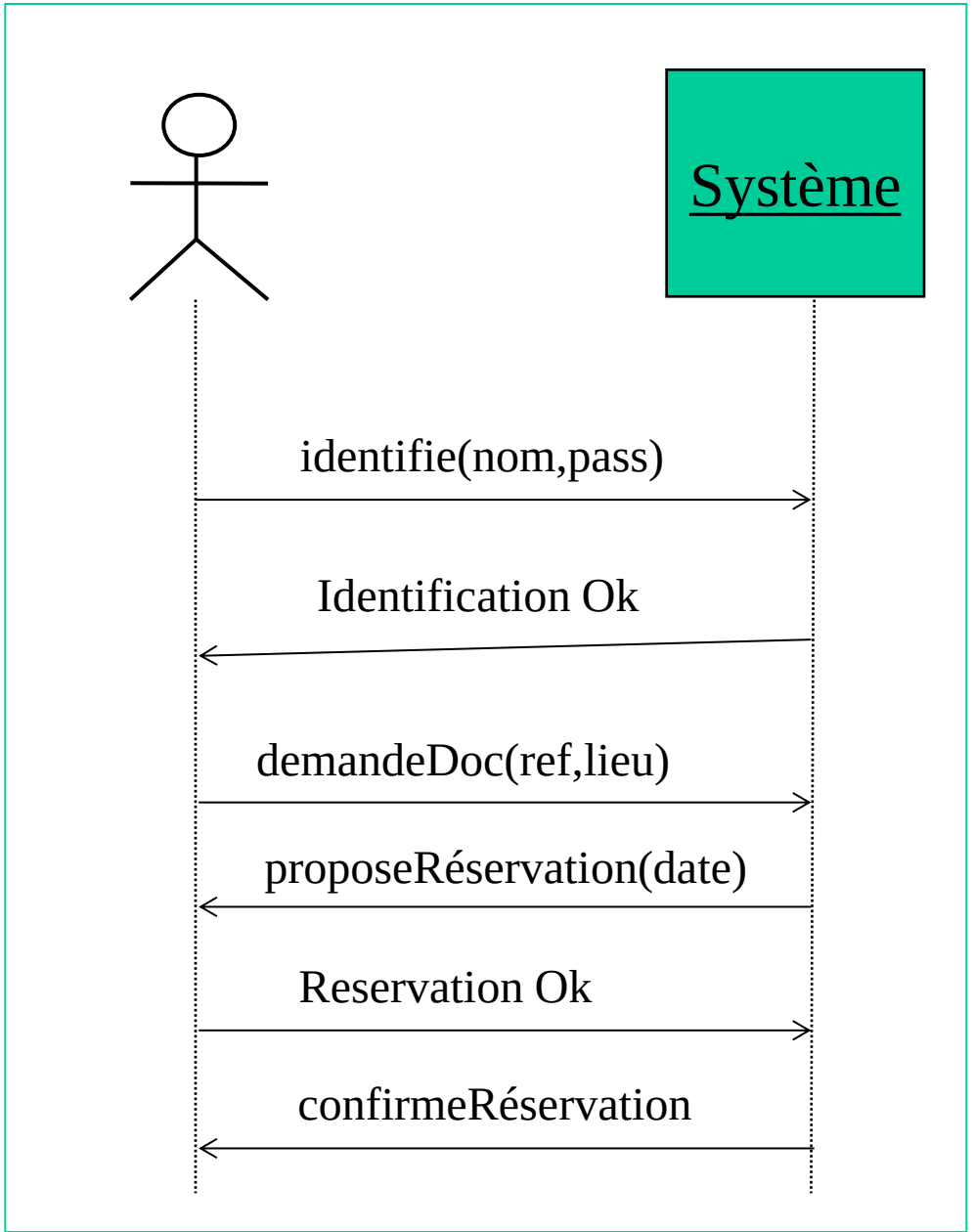
- La description d'un cas d'utilisation se fait par des scénarios qui définissent la suite logique des interactions:
  - Entre l'acteur et le système dans son ensemble: Description de haut niveau
  - Entre les objets qui participent à la réalisation du cas d'utilisation: Description détaillée



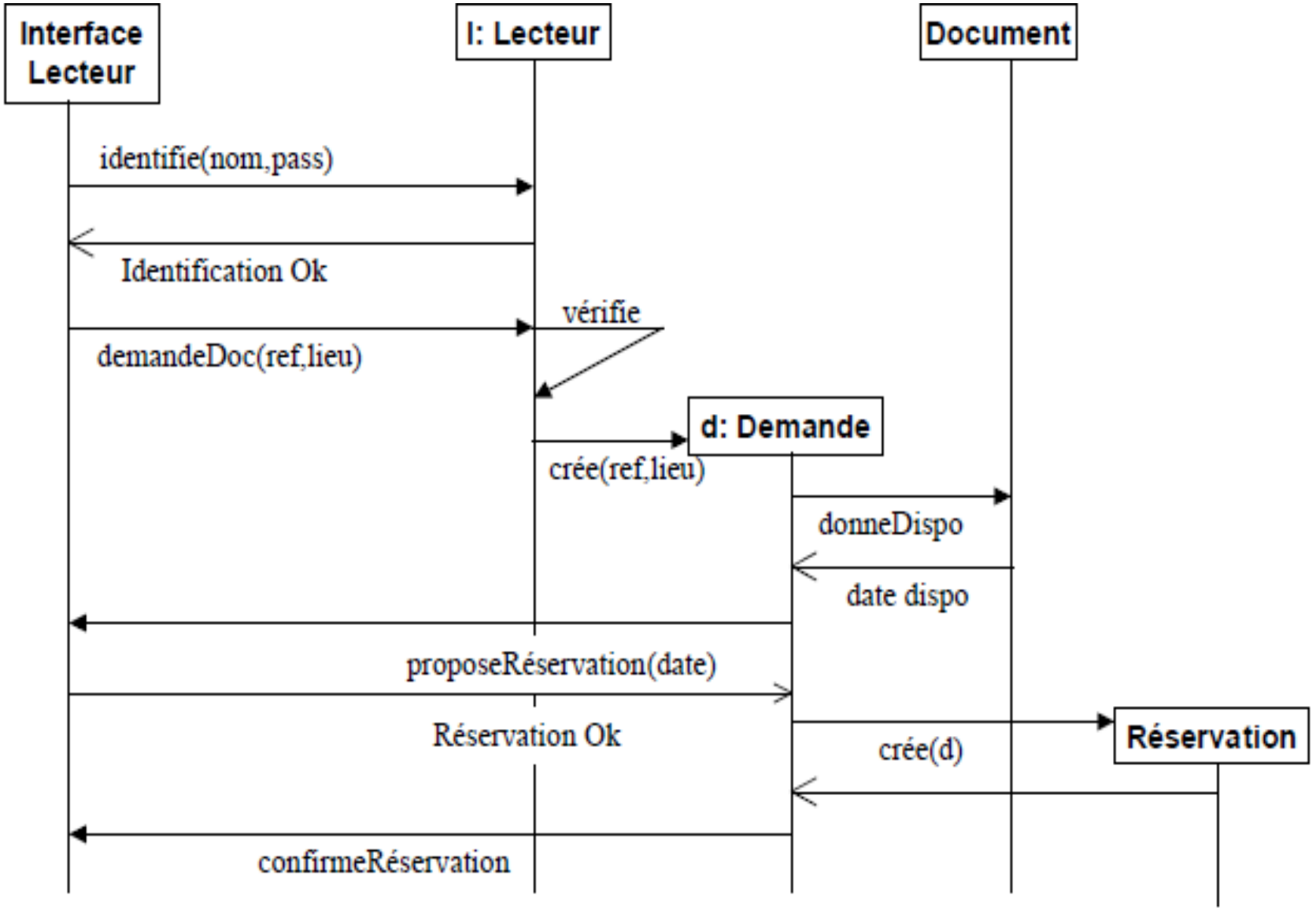
Description de haut niveau d'un scénario: demande d'un document

Ce type de description permet de concevoir une 1ère version de l'IHM (des écrans de dialogue pour l'exécution d'un Cas d'Utilisation)

- le lecteur s'identifie
- Après accord, il effectue sa demande de document
- Le système propose une réservation
- Le lecteur accepte la réservation
- Le système confirme la réservation par la délivrance d'une fiche



Description détaillée d'un scénario: Diagramme de séquence



Description détaillée d'un scénario: Diagramme de collaboration

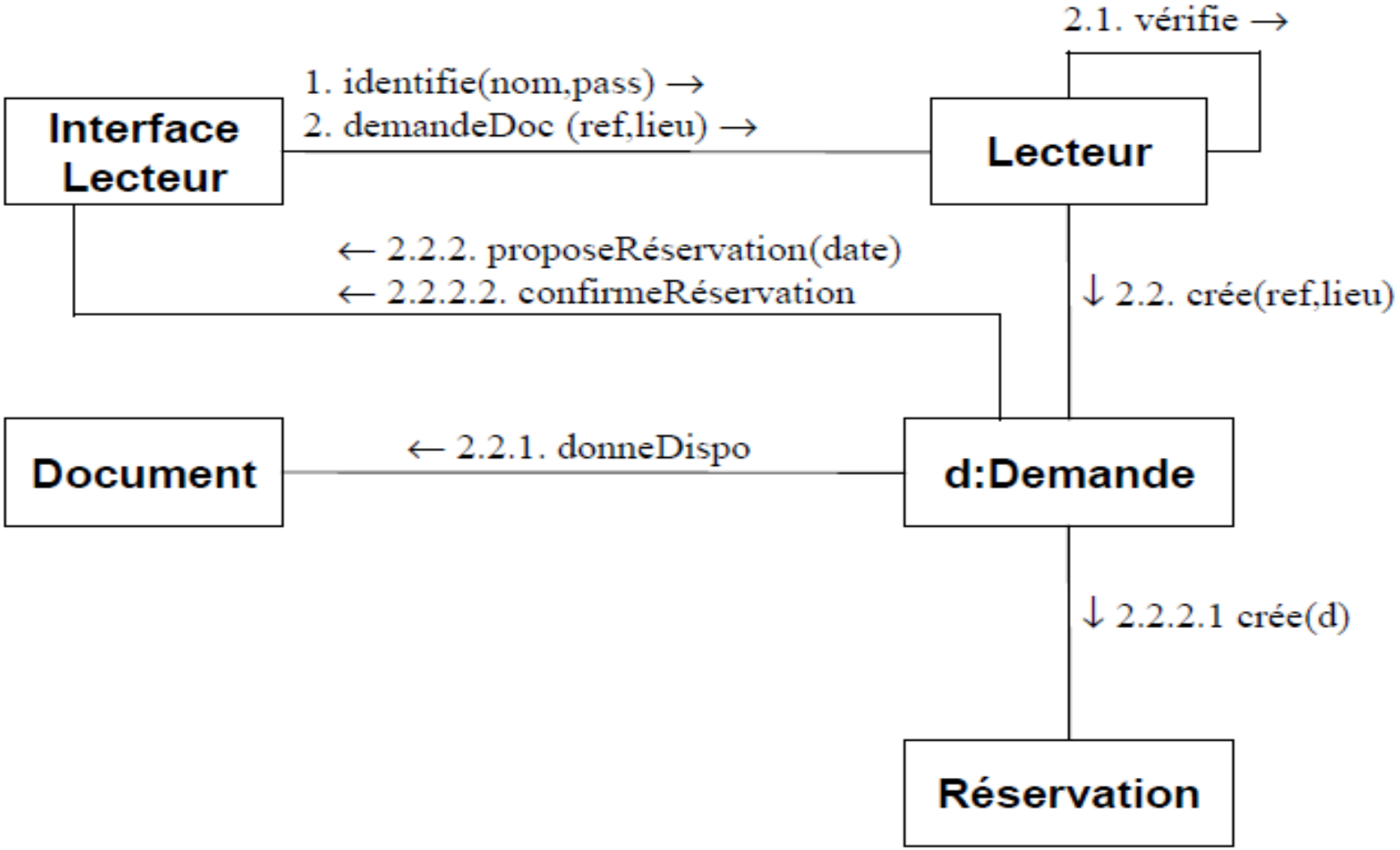


Diagramme d'Etats-Transitions: cycle de vie des objets

- permet de décrire les changements d'états d'un objet ou d'un composant, en réponse aux interactions avec d'autres objets ou composants ou avec des acteurs

- Fondé sur les automates: graphes d'états, reliés par des arcs orientés qui décrivent les transitions

« Si le système est dans l'état A et que l'événement E survient alors le système passe à l'état B »

Diagramme d'Etats-Transitions

- Décrit le **comportement** des objets d'une classe au moyen d'un automate d'états associé à la classe

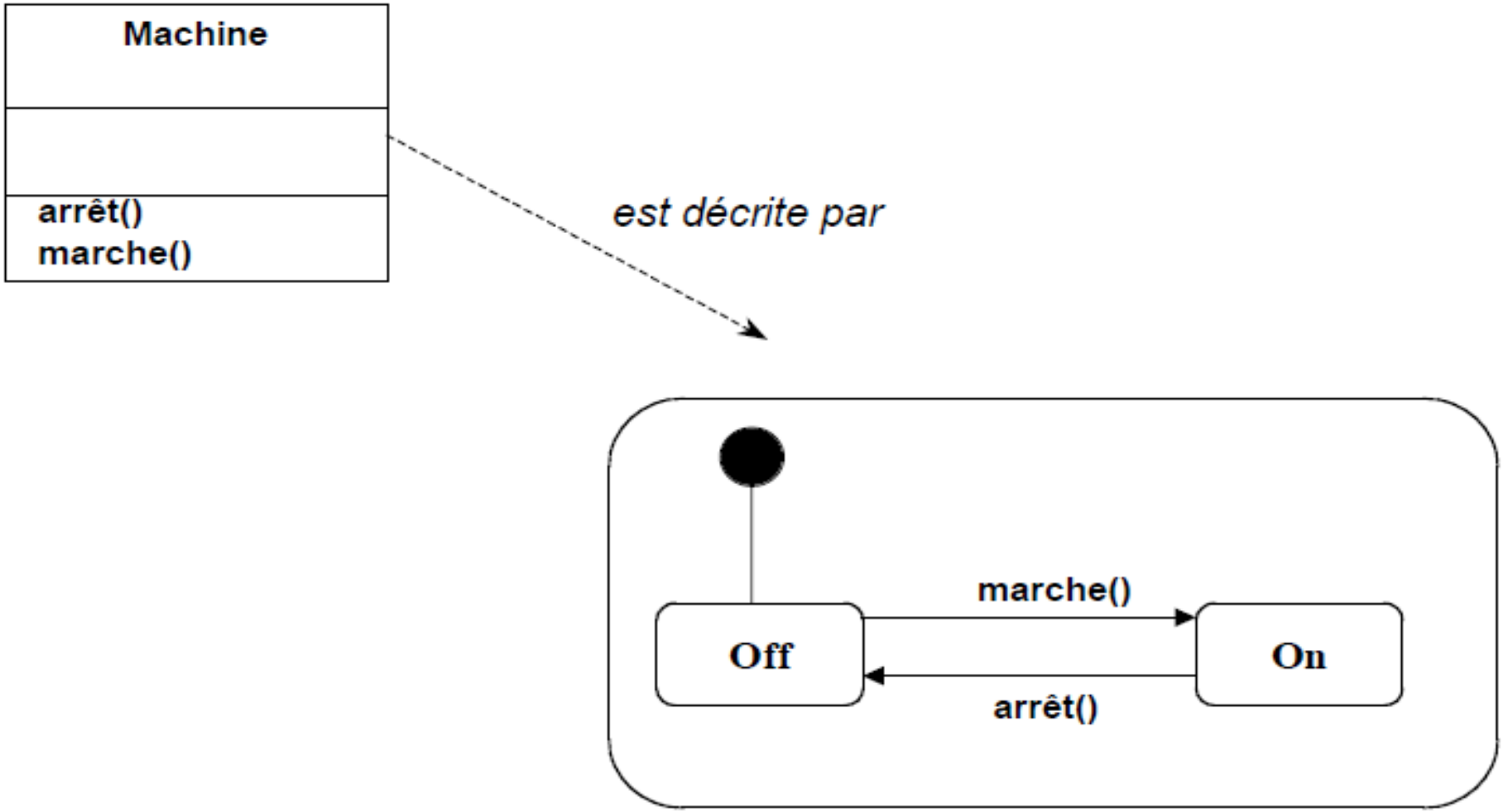


Diagramme d'Etats-Transitions: les concepts

un état :

• Représente une conjonction instantanée des valeurs des attributs d'un objet = une étape dans le cycle de vie d'un objet durant lequel:

- il réalise certaines actions
- il attend certains événements

• un état est stable et durable

Un objet de la classe Livre peut se trouver dans 3 états:

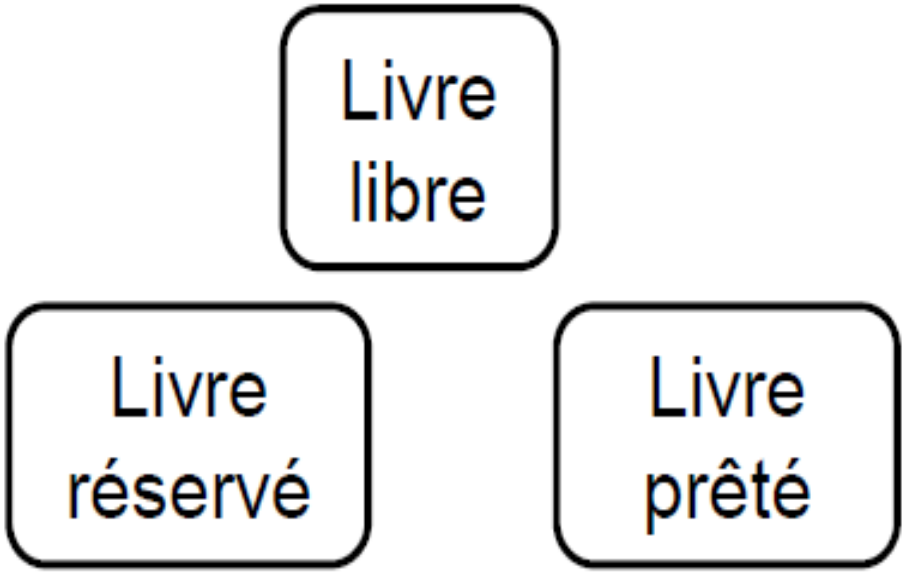


Diagramme d'Etats-Transitions: les concepts

Evénement :

- représente un phénomène sans durée, survenant à un instant donné
- déclenche une opération, ou signale la fin d'une opération.
- cause la transition d'un état vers un autre état.

A chaque objet, on peut associer au moins un événement de création et un événement de fin de vie.

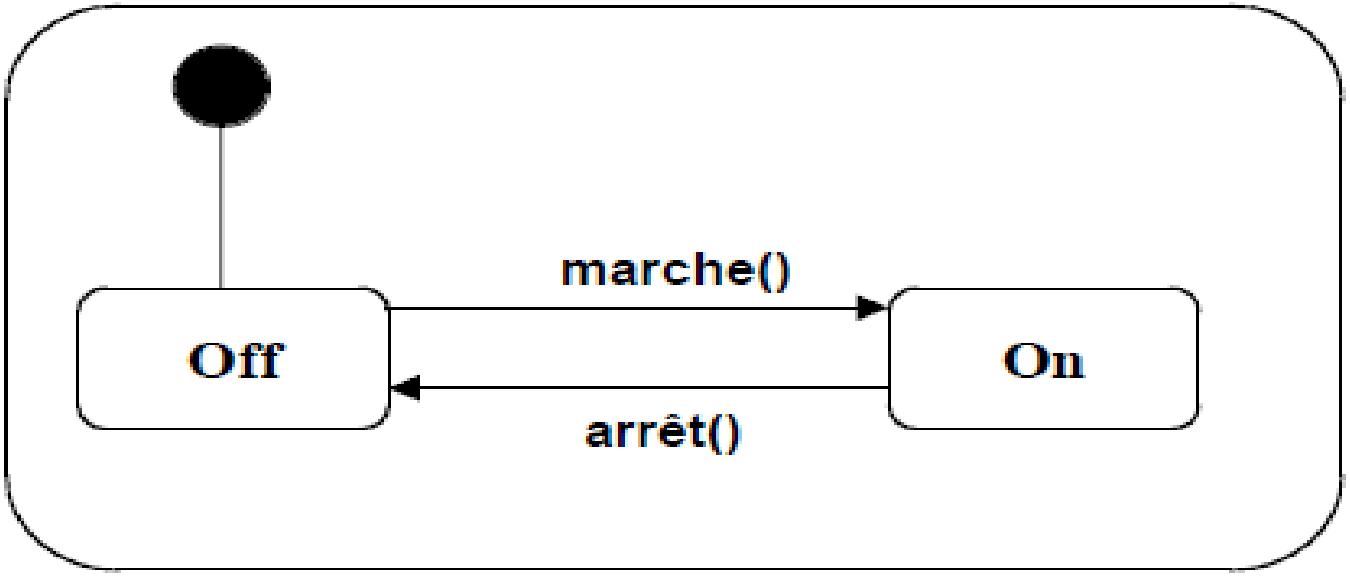


Diagramme d'Etats-Transitions: les concepts

Transition : représente le passage instantané d'un état vers un autre.

- est déclenchée par un événement : c'est l'arrivée d'un événement qui conditionne la transition.
- peut aussi être automatique, lorsqu'on ne spécifie pas l'événement qui la déclenche.
- peut être conditionnée à l'aide de "gardes" : expressions booléennes, exprimées en langage naturel et encadrées de crochets.

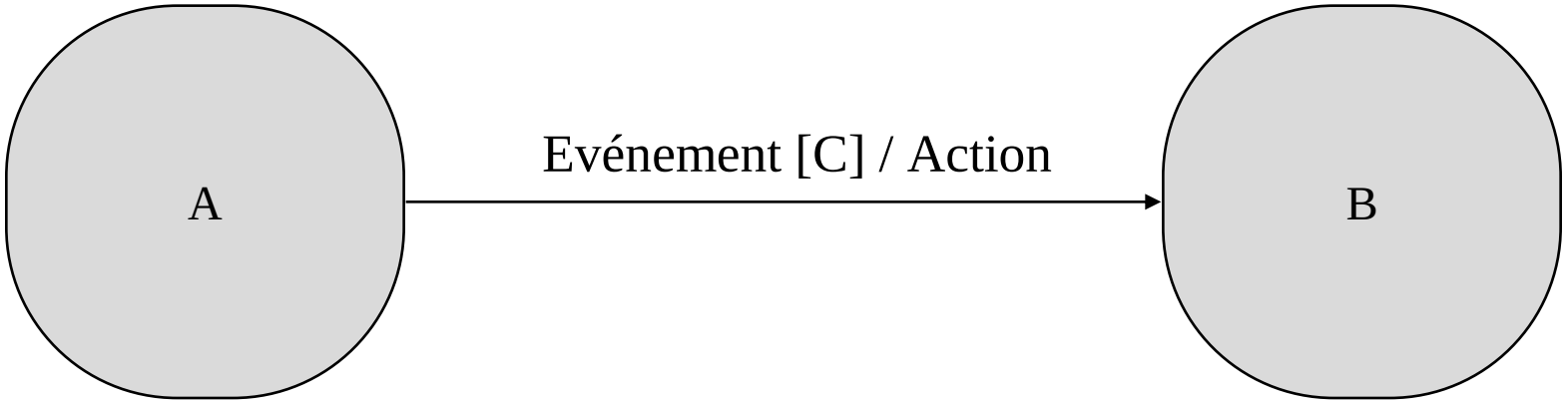


Diagramme d'Etats-Transitions: Statechart

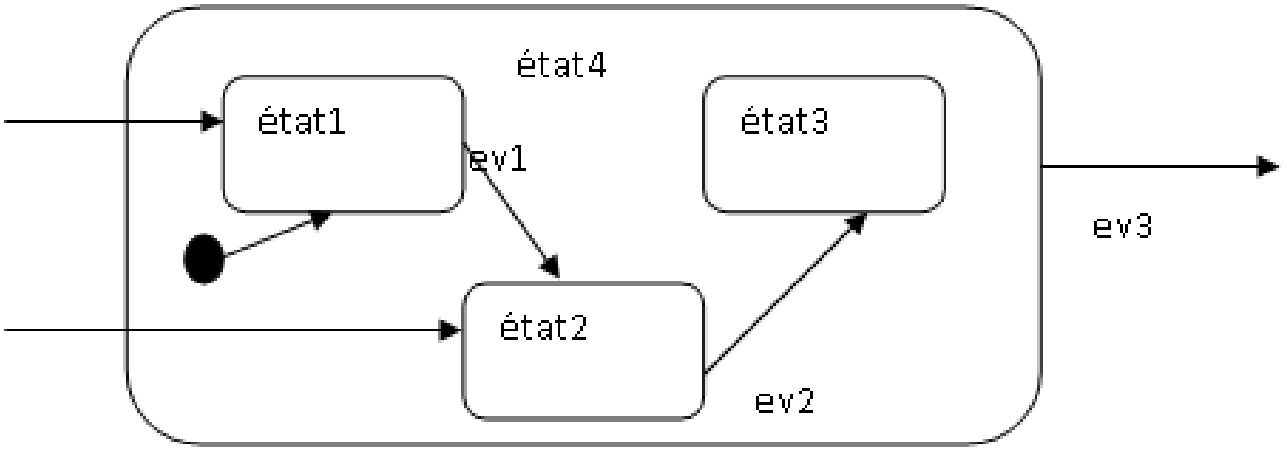
Extension des automates :

- **Hiérarchie**: Un état peut se décomposer en plusieurs sous états disjoints
- **Orthogonalité**: description explicite du parallélisme entre les composants d’un système
- **Communication par diffusion**: un évènement peut affecter plusieurs états

Diagramme d'Etats-Transitions: Statechart

Hiérarchie :Super-Etat

élément de structuration des diagrammes d'états-transitions : un état qui englobe d'autres états et transitions.



- état 4 est composé des états: état1, état2, état3
- L'objet est dans l'état 4 si et seulement si il est dans l'un de ses sous-états (état1, état2, état3): état4 = état1 XOR état2 XOR état3
- état1 est l'état initial

Diagramme d'Etats-Transitions: Statechart

Hiérarchie :Super-Etat:

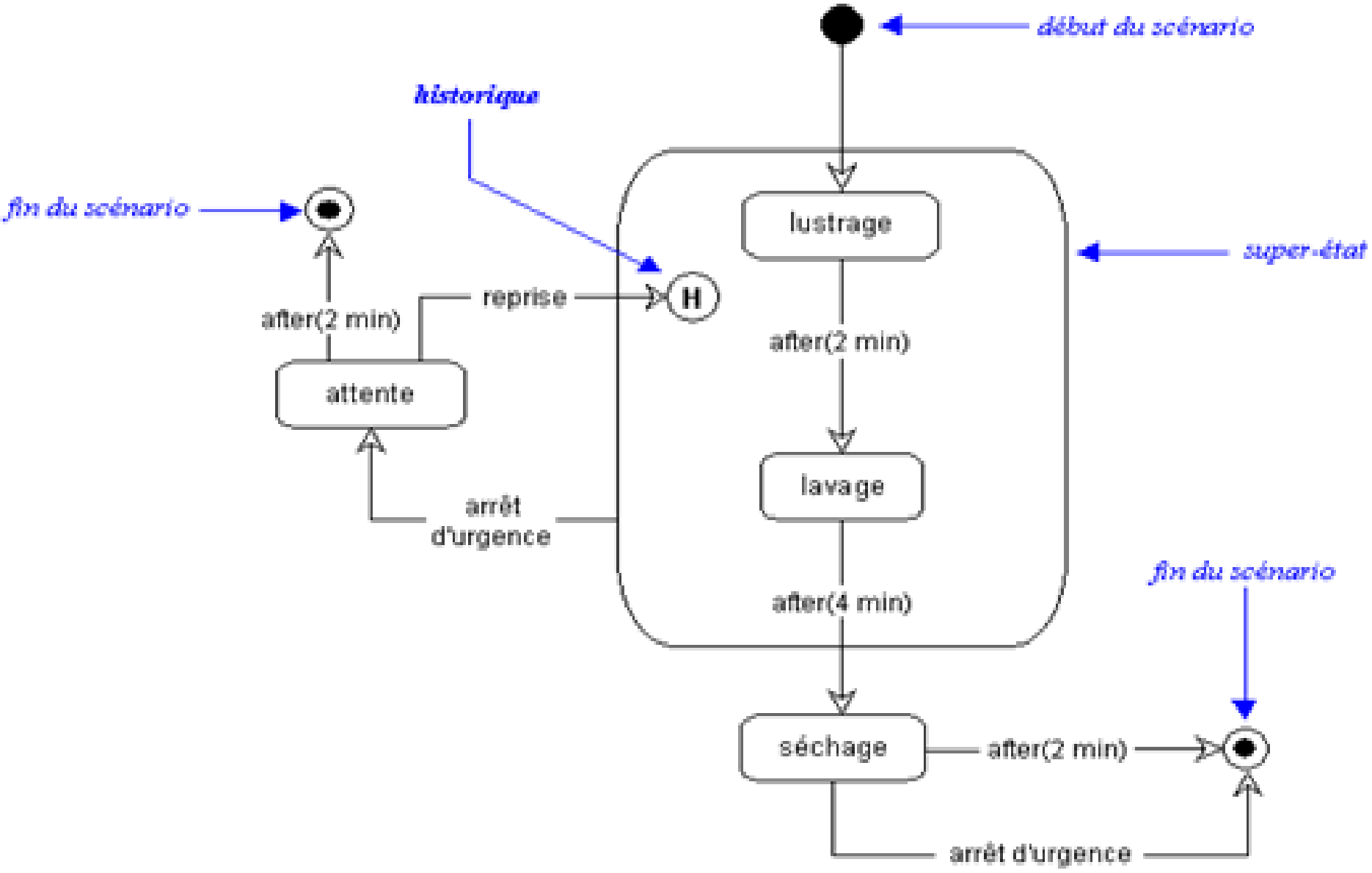
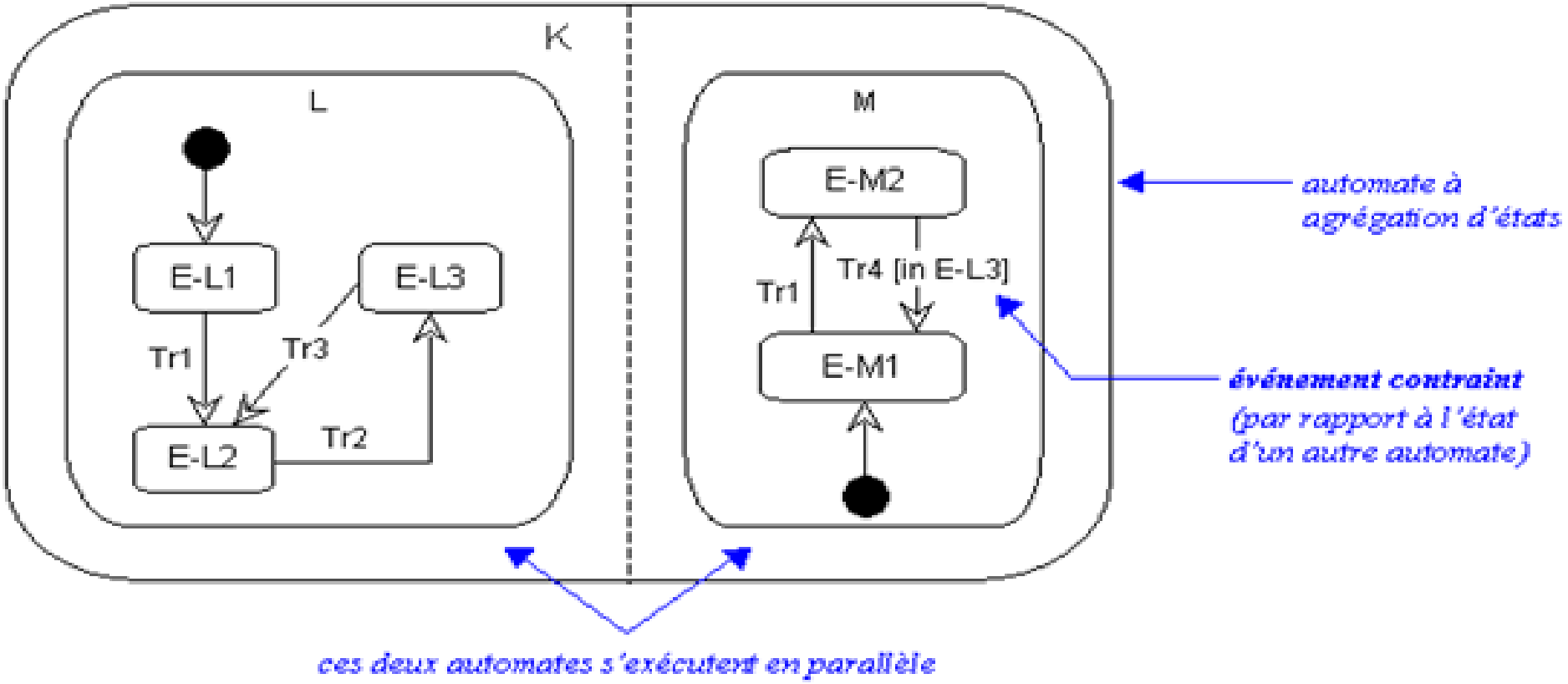


Diagramme d'Etats-Transitions: Statechart

Orthogonalité: Etats concurrents

Permet de modéliser le comportement d'objets concurrents sur un même diagramme d'états-transitions.



K = L * M

Diagramme d'Etats-Transitions

- **Actions dans un état**

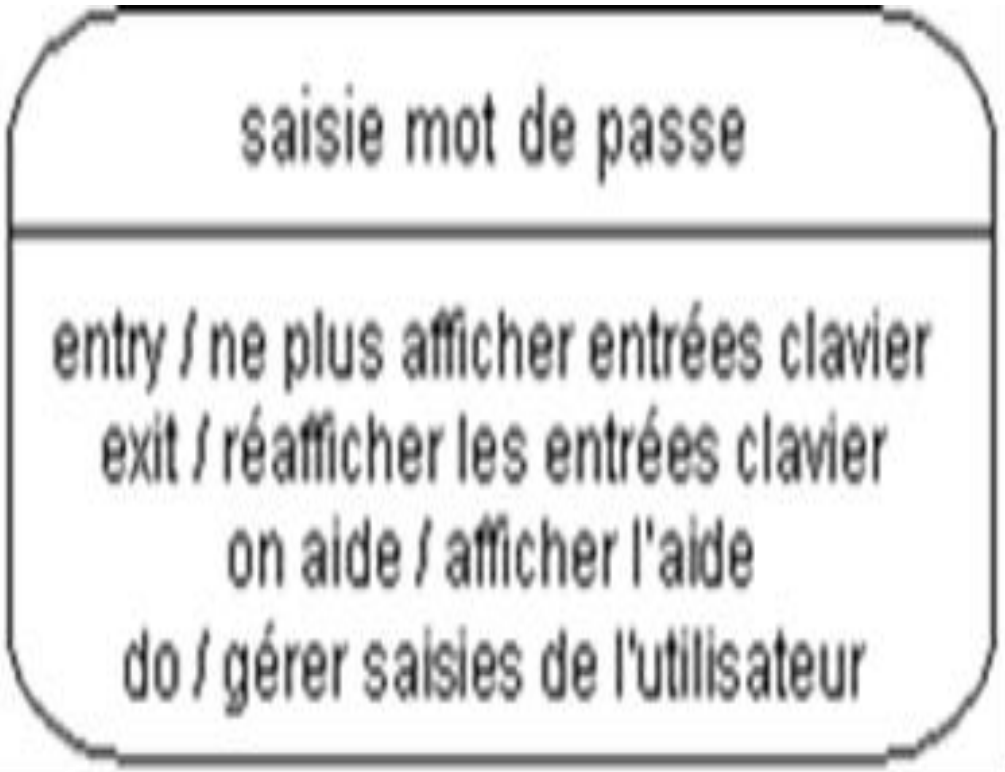


Diagramme d'Activités

- Permet de représenter le comportement interne d'une méthode, d'une opération ou d'un cas d'utilisation

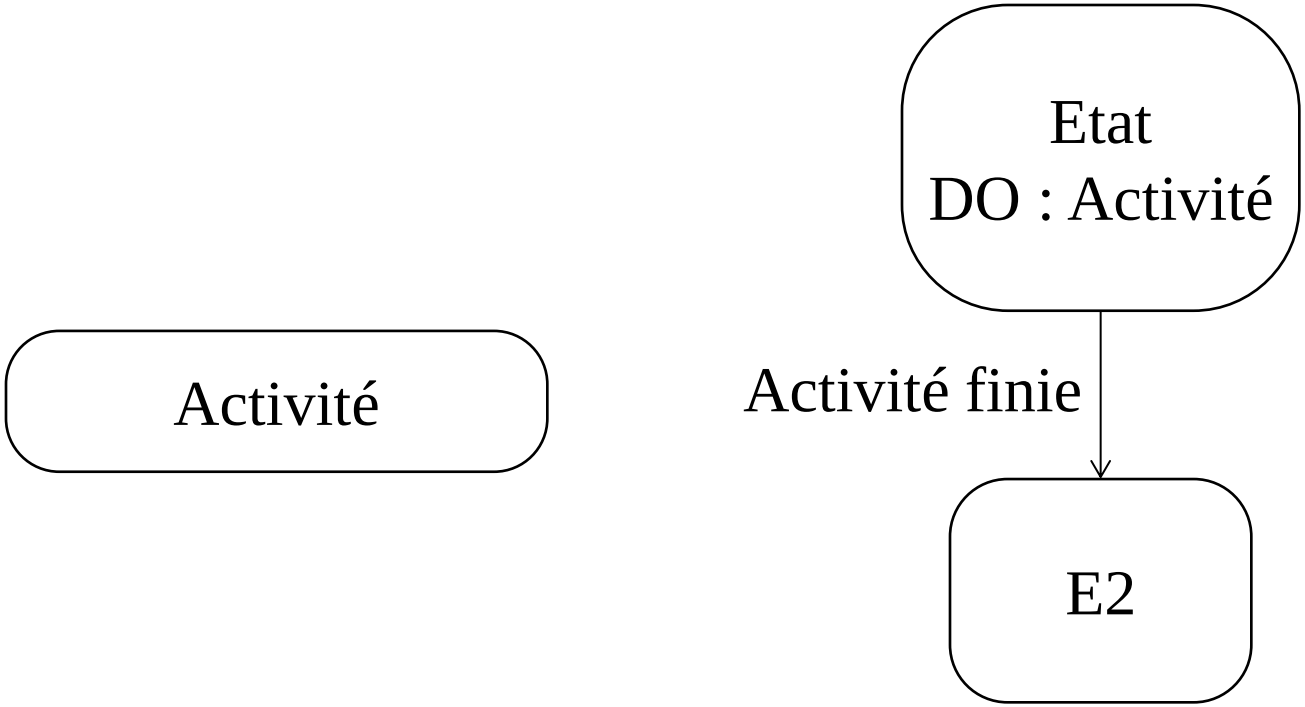
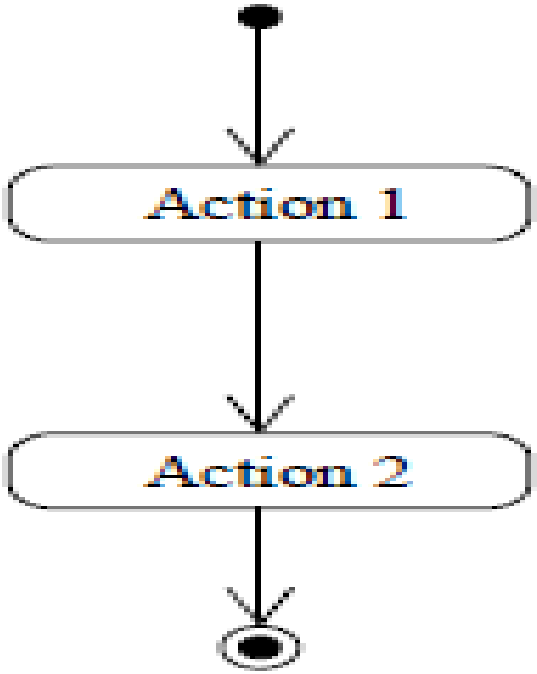
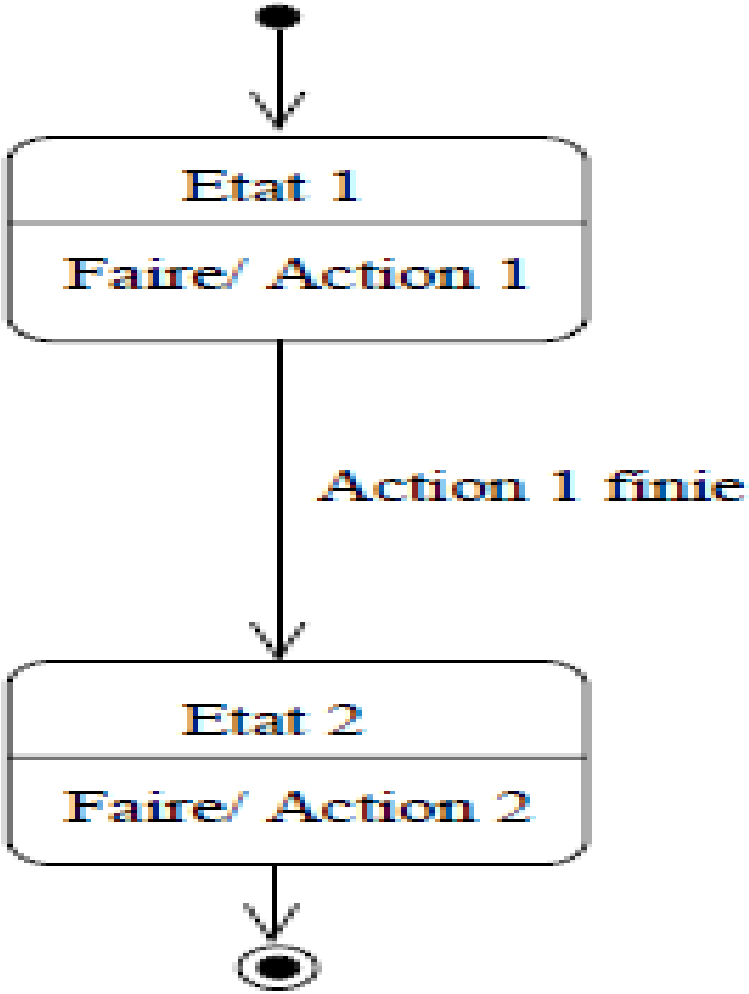


Diagramme d'Activités

- Elles sont reliés par des transitions automatiques
- Les transitions peuvent être gardées par des conditions booléennes mutuellement exclusive





Remarques sur les diagrammes d'activités

- Niveau simple :
  - permet, pour un objet donné, d'exprimer graphiquement une activité du diagramme d'état en un algorithme
  - traduction directe possible dans un langage de programmation OO (Java, C++, …)

- Niveau global :
  - donne une vue macroscopique de l'enchaînement des fonctionnalités
  - correspond à la traditionnelle décomposition hiérarchique fonctionnelle

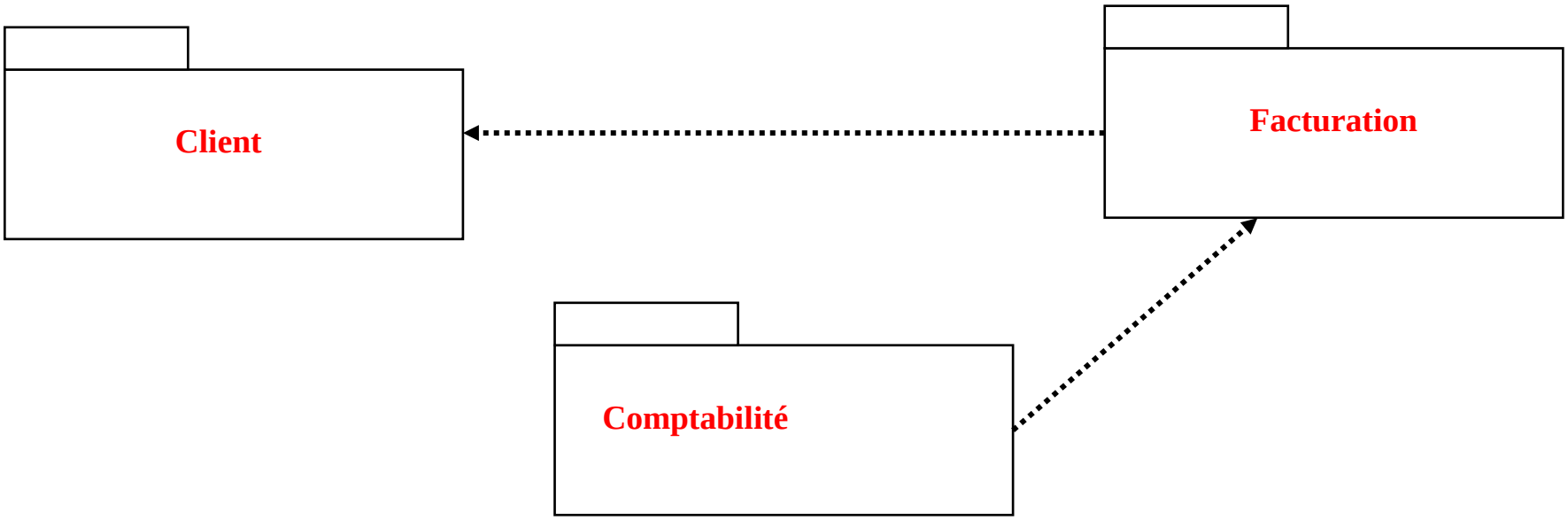
Notion de **paquetage**

Un **paquetage** permet de:

• regrouper un ensemble de classes, d'associations,… et éventuellement d'autres **paquetage**s pour des raisons logiques.

• découper un système en plusieurs parties représentées chacune par un **paquetage**.

• organiser les modèles de la même manière que les répertoires organisent les fichiers.



Paquetages

- Une classe peut apparaître dans différents packages (avec le même nom).
- On y trouve même des classes qui n'appartiennent pas au package mais qui sont référencées par les classes propres.
- On désigner une classe d'un package par: ***nomPackage :: nomClasse***