

UML — The Unified Modeling Language

Introduction à la notation

Gerson Sunyé
Université de Nantes
gerson.sunye@univ-nantes.fr

Introduction

- UML - Langage de modélisation unifié
- Standard de l'OMG (Object Management Group)
- Une notation (syntaxe concrète)
- Un méta-modèle (syntaxe abstraite)
- Ce n'est pas une méthode (pas de processus de développement)

Un peu d'histoire

La jungle de méthodes des années 90

- OMT (Rumbaugh et al.), OOA/OOD (Booch), OOA (Coad), OOSE (Jacobson), OOA/OODLE (Schlaer & Mellor) , Fusion (HP), Classe-Relation (Desfray).
- et aussi: DFD (De Marco), Etat-transition (Harel), ER (Chen), CRC (Wirf & Brooks), JSD (Jackson)

Un peu d'histoire

- Les années 80-90:
 - Différents langage de modélisation
 - Différentes interprétations des constructeurs de modélisation
 - Peu d'outils, pas adaptés
 - Pas d'échange de modèles

Objectifs d'UML

- Proposer un langage visuel de modélisation, indépendant de l'implémentation et du processus de développement
- Proposer une base formel pour l'interprétation des modèles

Objectifs d'UML

- Intégrer des concepts de haut niveau: Composants, Patrons de conception (Patterns), Cadres d'applications (Frameworks)
- Encourager l'outillage

UML et le processus de développement

- Un processus est composé (au minimum) des étapes suivantes:
 - Expression des besoins
 - Analyse
 - Conception
 - Réalisation, validation, maintenance

Analyse et conception

- UML: une même notation, deux niveaux d'abstraction différents
- Une frontière floue, qui doit être précisée par une méthode de développement

Analyse avec UML

- Généralement:
- Le résultat de l'analyse est un modèle idéal de la solution
- Abstraction des particularités de la plate-forme d'implémentation (persistance, réseau, etc.)

Conception avec UML

- Le résultat de la conception est un modèle de la solution, adapté au monde réel
- Ce modèle reste indépendant du langage d'implémentation

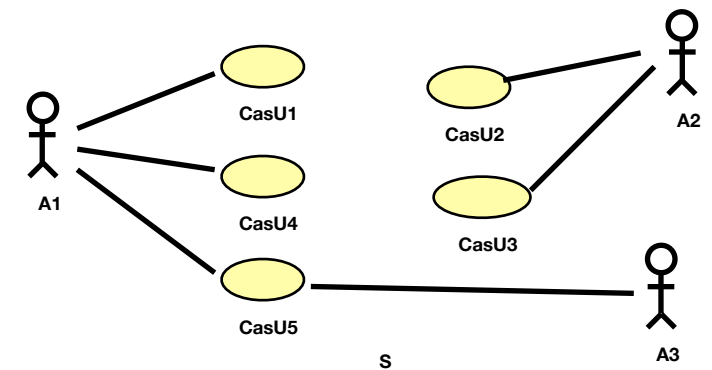
Modélisation avec UML

- Diagrammes de cas d'utilisation
- Diagrammes structurels
- Diagrammes comportementaux
- Mécanismes d'extension

Diagrammes de cas d'utilisation

Modèle des cas d'utilisation

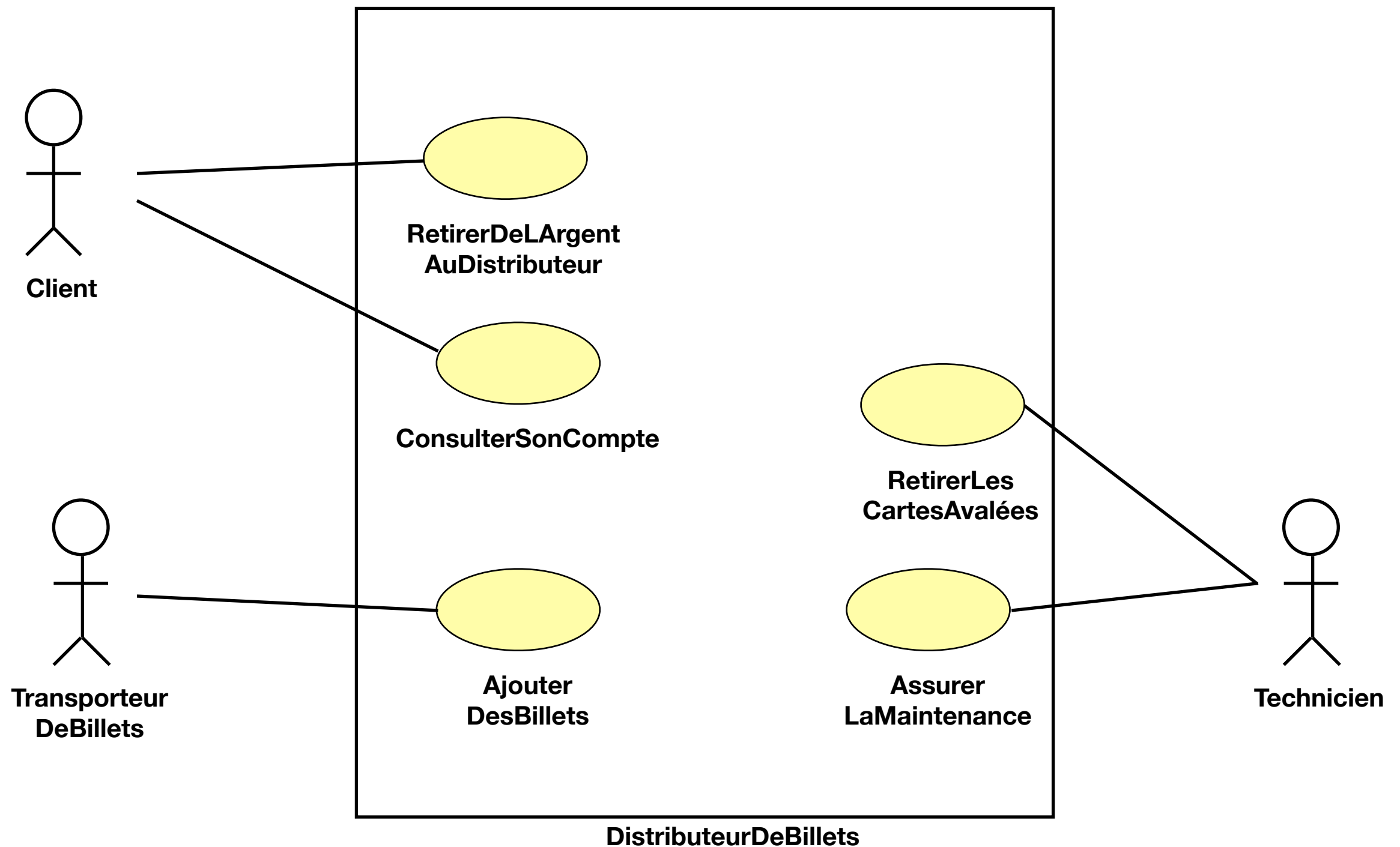
- Buts :
 - modéliser le point de vue des **utilisateurs**
 - définir les limites précises du **système**
- Notation très simple, compréhensible par tous, y compris le client
- Permet de structurer :
 - les besoins (cahier des charges)
 - le reste du développement
- Modèle (principalement) communicationnel



Modèle des cas d'utilisation

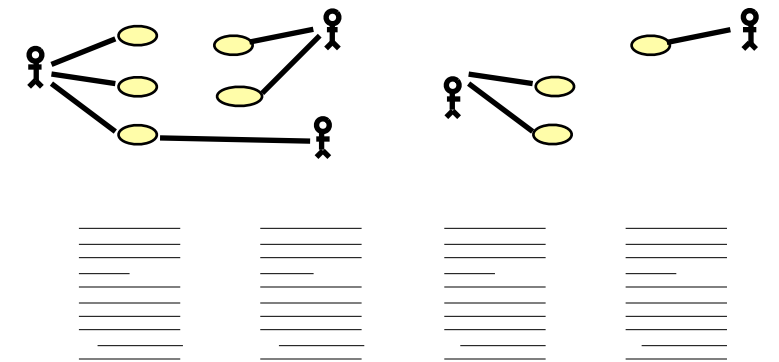
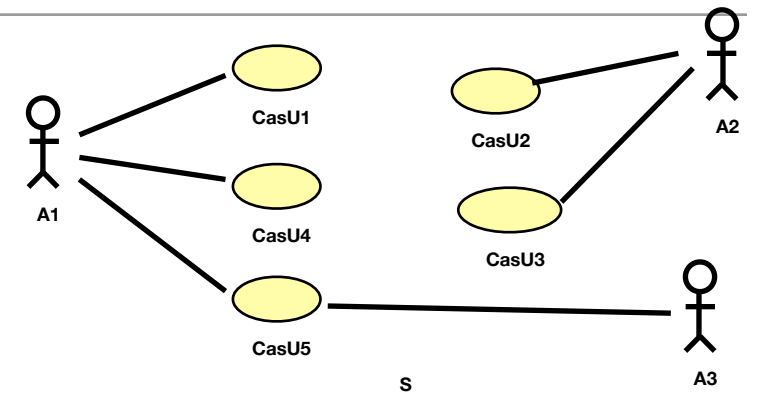
- Moyens :
 - Les acteurs UML
 - Les *use-cases* UML
 - Utilisation d'un dictionnaire du domaine

Use Case Diagram Example

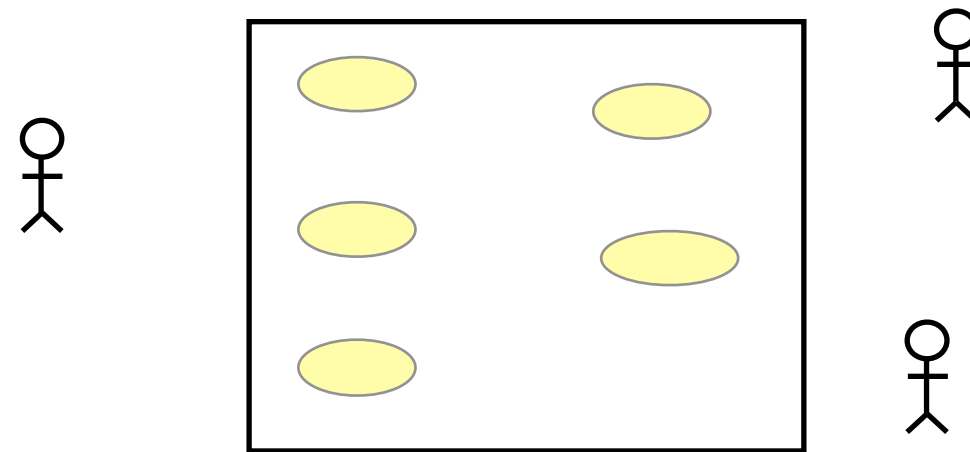


Modèle de cas d'utilisation

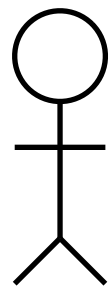
- Diagramme de cas d'utilisation
- Modèle de cas d'utilisation
 - descriptions textuelles,
 - diagrammes de cas d'utilisation
 - diagrammes de séquences
 - dictionnaire de données
 - ...



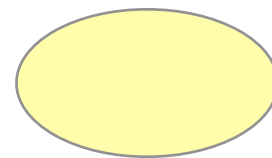
Éléments de base



Acteurs



Cas d'utilisation

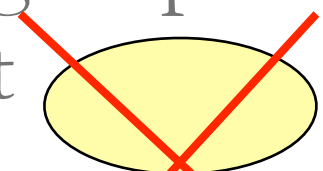


Systeme

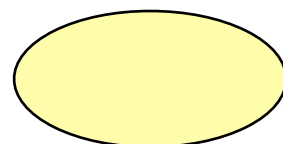


Cas d'utilisation (CU)

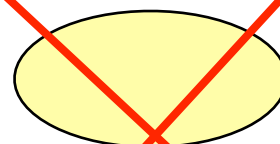
- Cas d'utilisation
 - une manière d'utiliser le système
 - une suite d'interactions entre un acteur et le système
- Correspond à une fonction du système visible par l'acteur
- Permet à un acteur d'atteindre un but
- Doit être utile en soi
- Regroupe un ensemble de scénarii correspondant à un même but



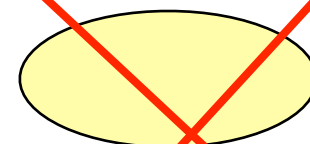
EnregistrerEntrée



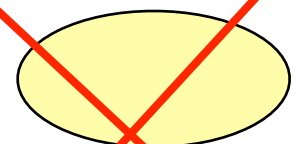
**RetirerDeLArgentAu
Distributeur**



S'identifier



**EntrerPendant
LesHeuresDOuverture**



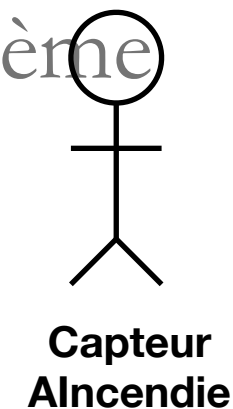
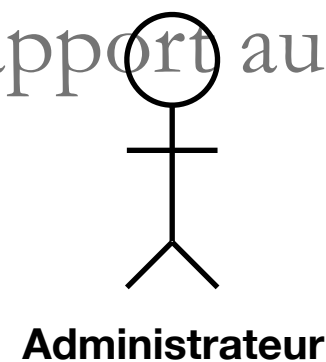
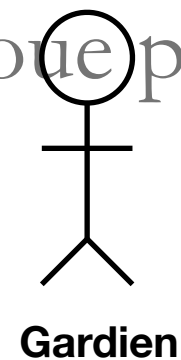
TaperSonCode

Systeme

- Le système est
 - modélisé par un ensemble de cas d'utilisation
 - vu comme une **boîte noire**
- Le système contient :
 - les cas d'utilisation,
 - mais pas les acteurs.
- Un modèle de cas d'utilisation permet de définir :
 - les fonctions essentielles du système,
 - les **limites du système**,
 - le système par rapport à son environnement,
 - **délimiter le cadre du projet !**

Acteurs

- Un Acteur =
 - élément *externe*
 - qui *interagit* avec le système (prend des décisions, des initiatives. Il est "actif".)
- *Rôle* qu'un "utilisateur" joue par rapport au système



Acteurs vs. utilisateurs

Ne pas confondre les notions
d'acteur et de personne utilisant le système

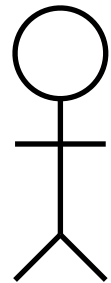
- Une même personne peut jouer plusieurs rôles
ex: Maurice est directeur mais peut jouer le rôle de guichetier
- Plusieurs personnes peuvent jouer un même rôle
ex: Paul et Pierre sont deux clients
- Un rôle par rapport au système plutôt qu'une position dans l'organisation
ex: PorteurDeCarte plutôt qu'Enseignant
- Un acteur n'est **pas forcément** un être **humain**
ex: un distributeur de billet peut être vu comme un acteur

Différents types d'acteurs

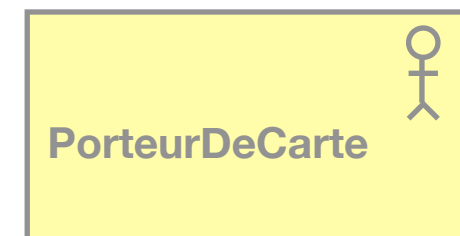
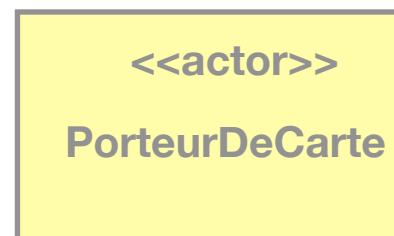
Ne pas oublier d'acteurs :

- Utilisateurs principaux
ex: client, guichetier
- Utilisateurs secondaires
ex: contrôleur, directeur, ingénieur système, administrateur...
- Périphériques externes
ex: un capteur, une horloge externe, ...
- Systèmes externes
ex: système bancaires

Notation

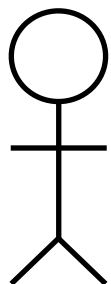


PorteurDeCarte

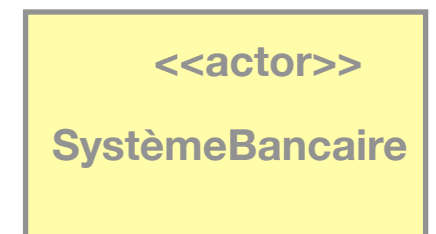
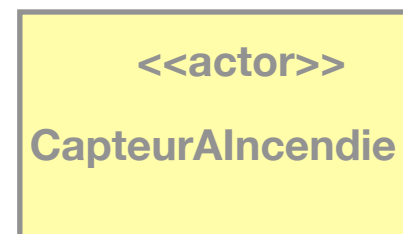


- Notations alternatives pour les acteurs
- **Note de style :**

utiliser plutôt le stéréotype `<<actor>>` pour les acteurs non humains



PorteurDeCarte

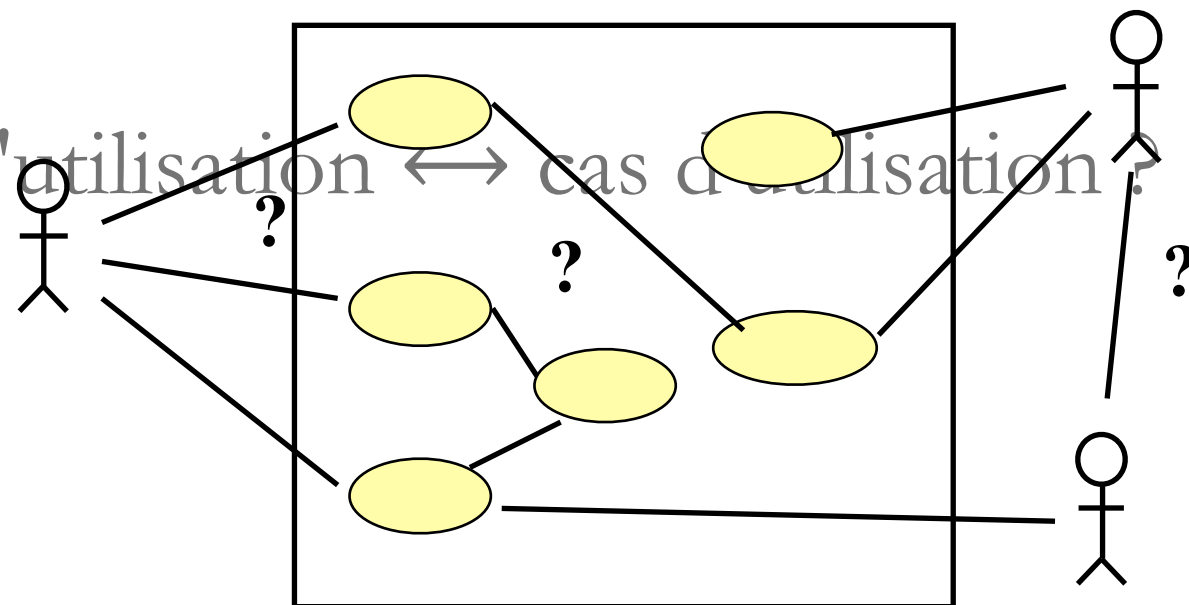


Relations entre les éléments de base

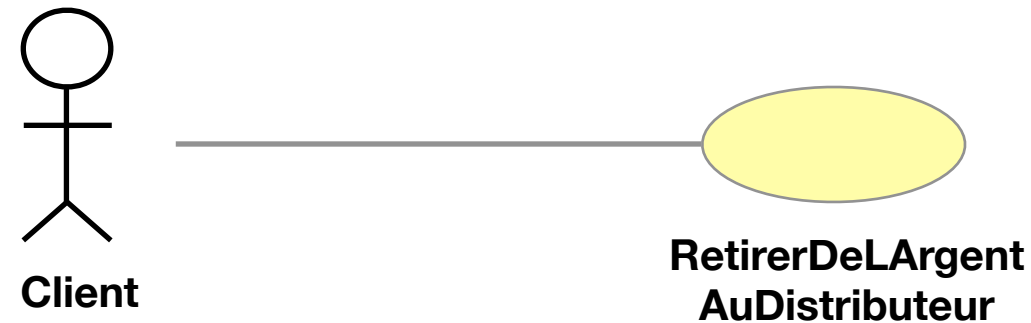
- Relations acteurs \leftrightarrow cas d'utilisation ?

- Relations acteurs \leftrightarrow acteurs ?

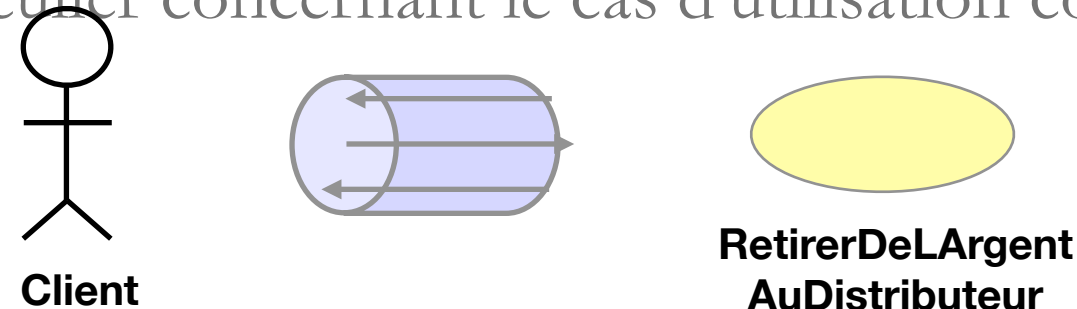
- Relations cas d'utilisation \leftrightarrow cas d'utilisation ?



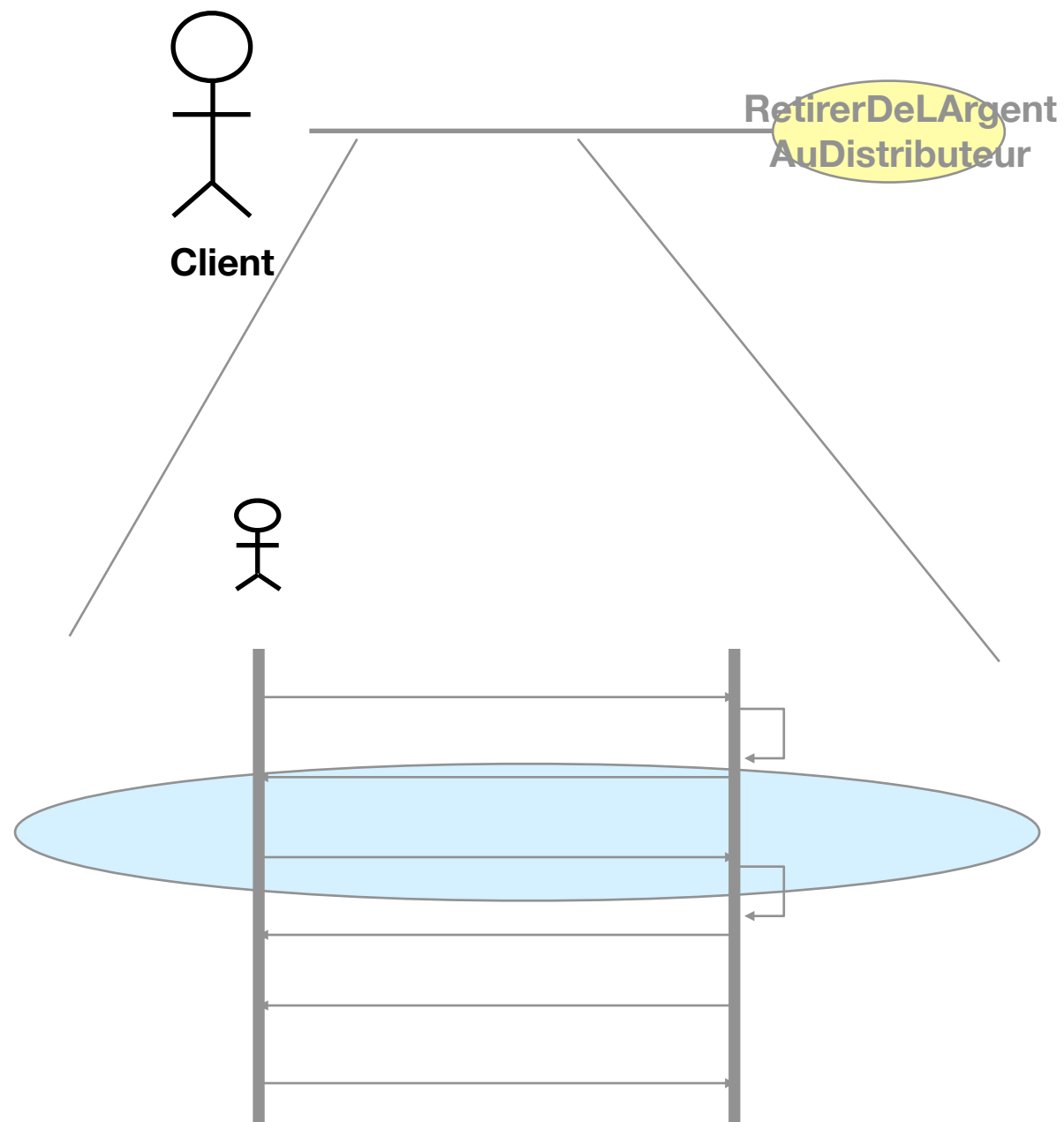
Relation acteur \leftrightarrow cas d'utilisation



- Point de vue besoin: représente la possibilité d'atteindre un but
- Point de vue système: représente un canal de communication
 - Echange de messages, potentiellement dans les deux sens
 - Protocole particulier concernant le cas d'utilisation considéré

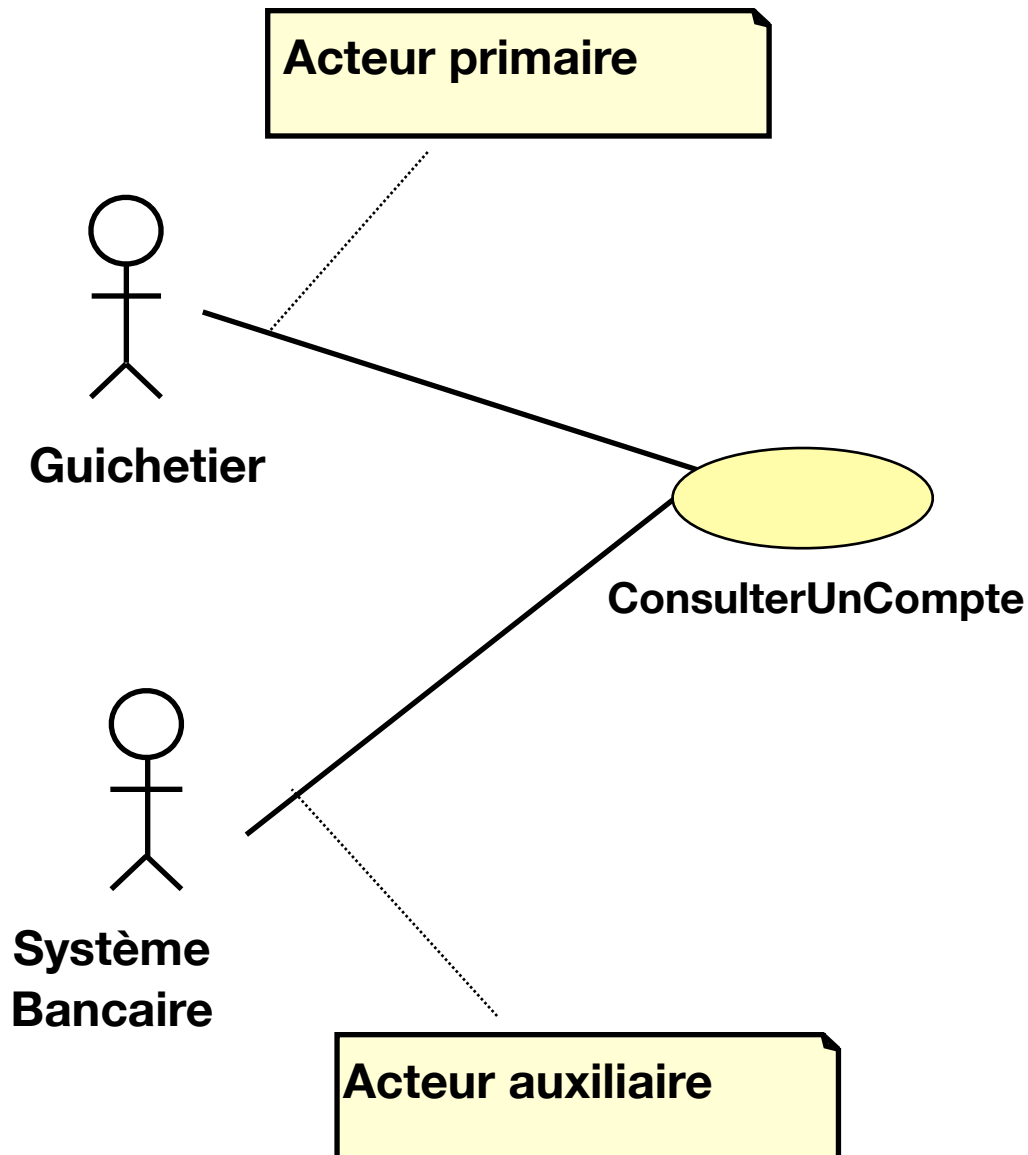


Relation acteur \leftrightarrow cas d'utilisation



- Description via des diagrammes de séquences "systèmes"
- Plus tard dans le cours ...

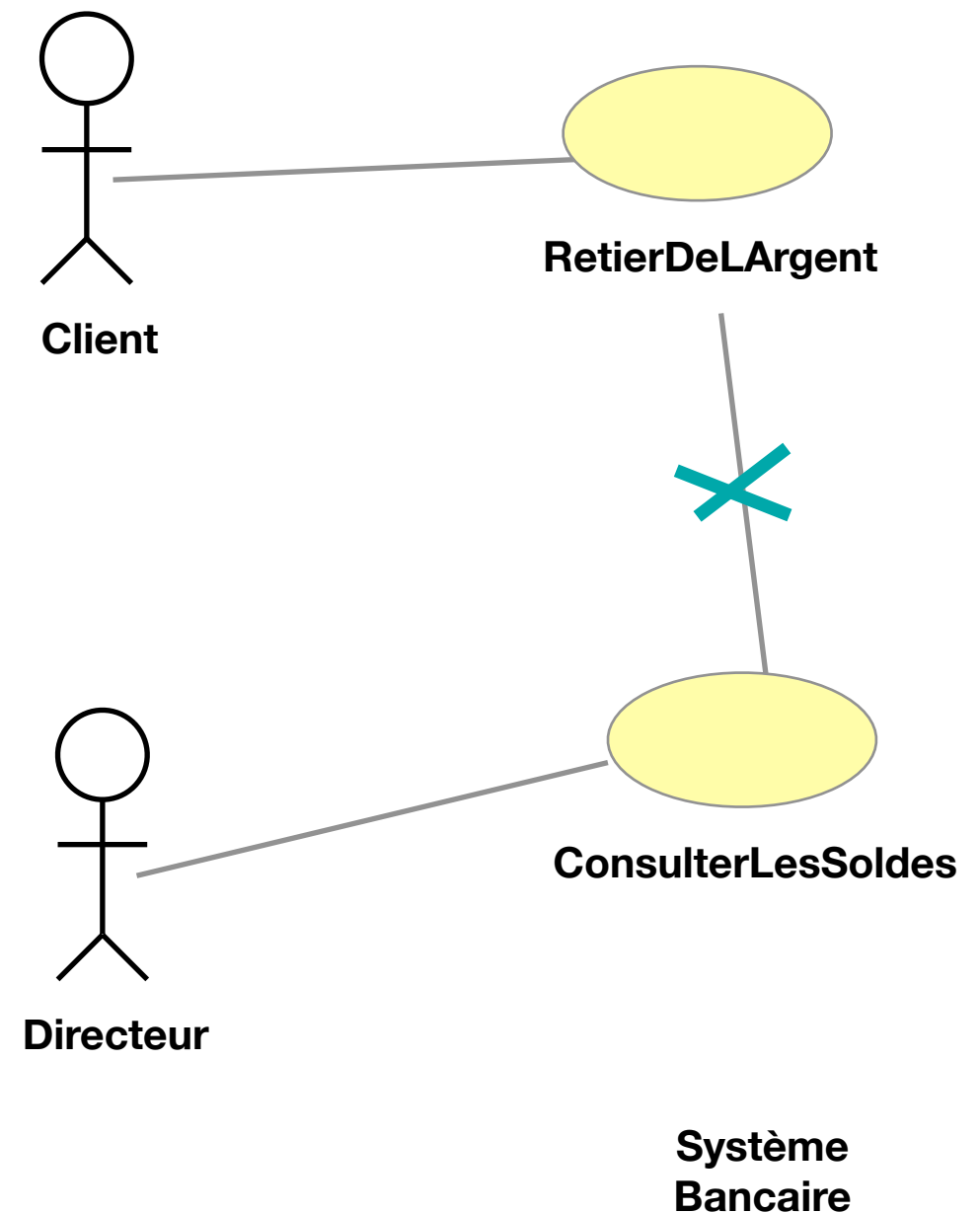
Relation acteur \leftrightarrow cas d'utilisation



- Acteur "primaire »
 - utilise le système comme outil pour réaliser son but
 - initie généralement la communication
- Acteur(s) "auxiliaire(s)"
 - interviennent suite à l'intervention de l'acteur primaire
 - offrent généralement leurs services au système

Relation cas d'utilisation \leftrightarrow cas d'utilisation

- Communications internes non modélisées.
- UML se concentre sur la description du système et de ses interactions avec l'extérieur
- Formalisme bien trop pauvre pour décrire l'intérieur du système. Utiliser les autres modèles UML pour cela.
- Autres relations possibles entre CU (slides suivants)

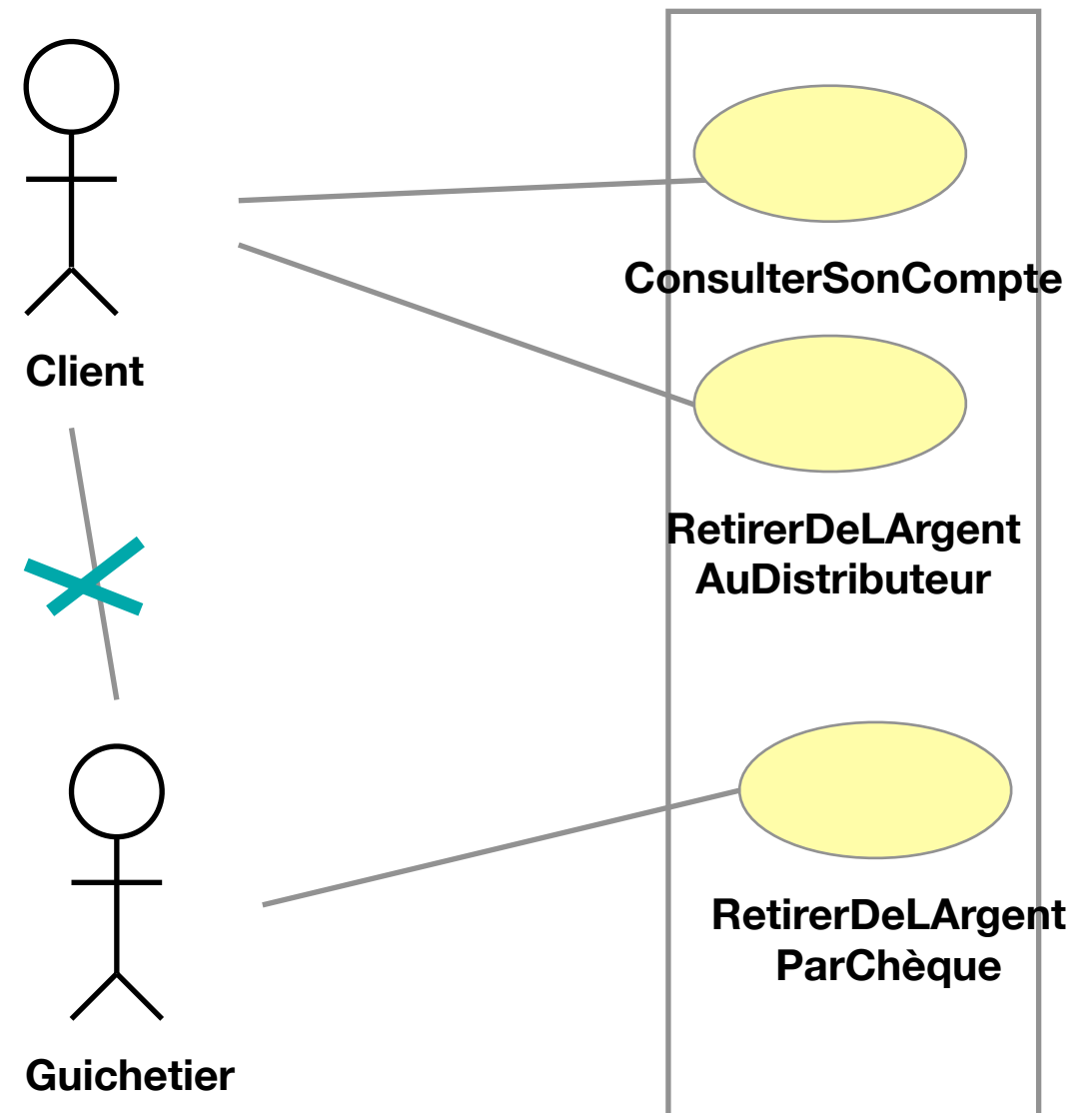


Relation cas d'utilisation \longleftrightarrow cas d'utilisation

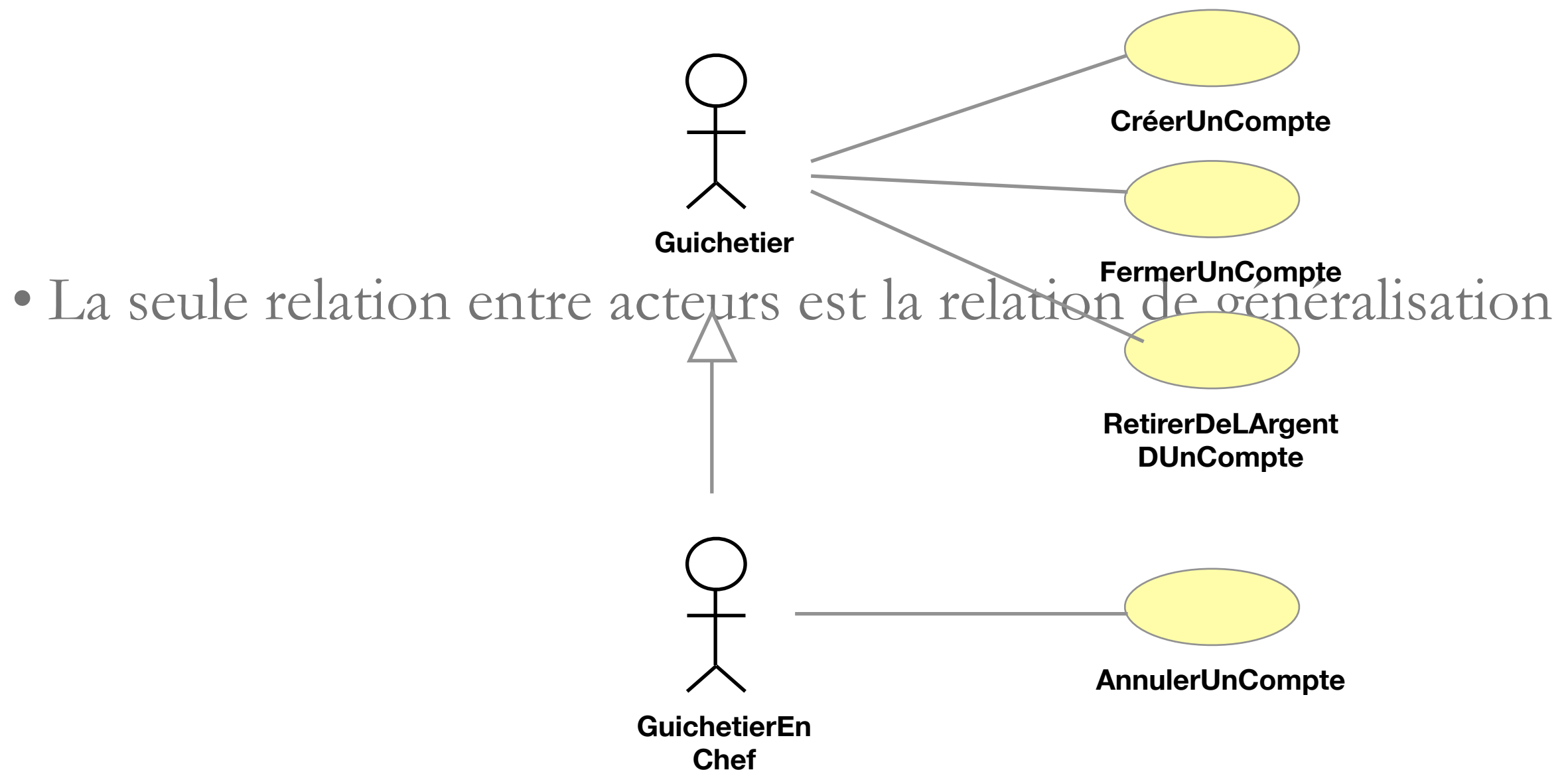
- Utilisation (*«include»* / *«uses»*)
 - Utilisation d'autres use-cases pour en préciser la définition
- Extension (*«extend»* / *«extends»*)
 - Un use-case étendu est une spécialisation du use-case père

Relation acteur \leftrightarrow acteur

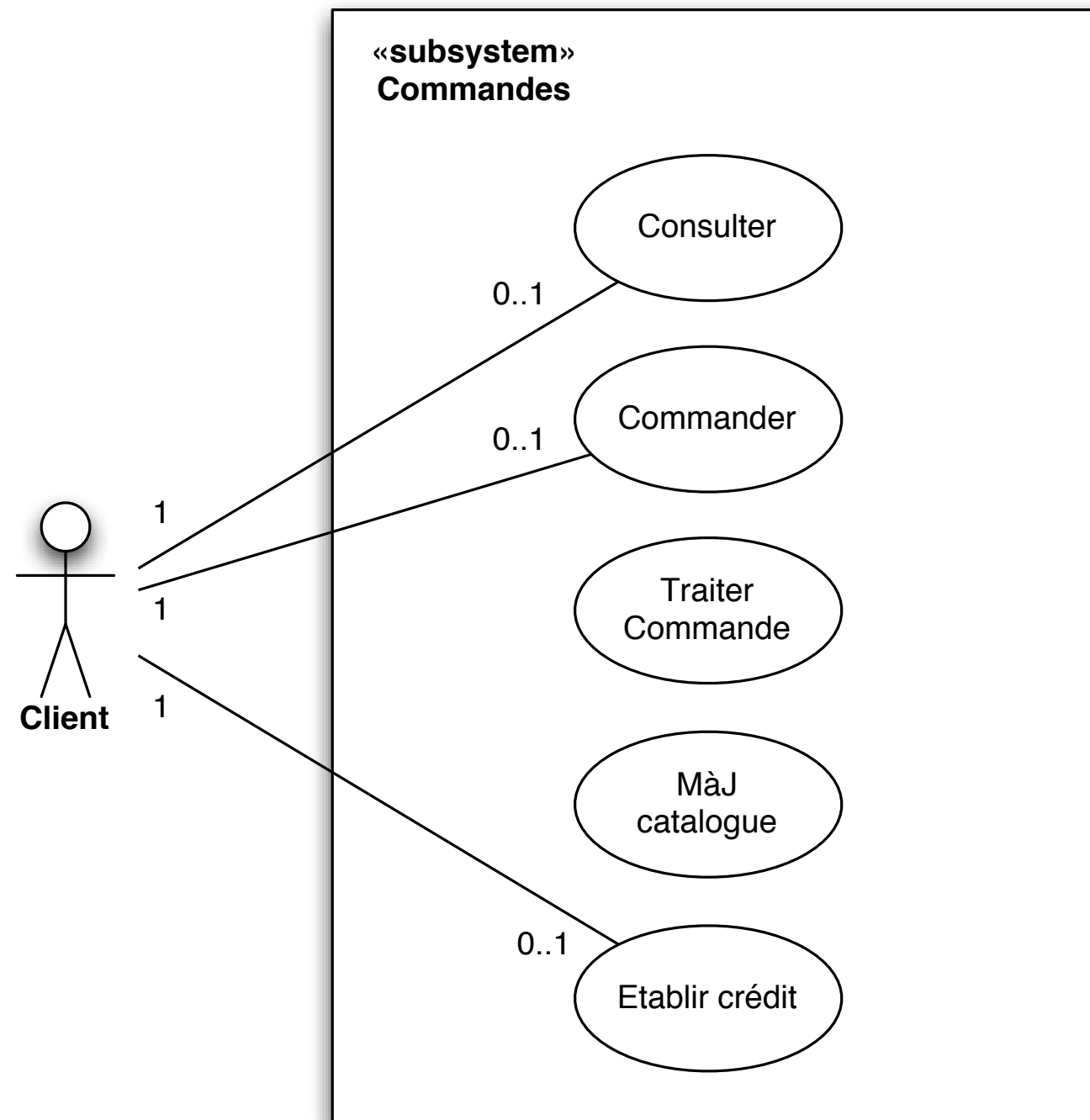
- Communications externes non modélisée
- UML se concentre sur la description du système et de ses interactions avec l'extérieur



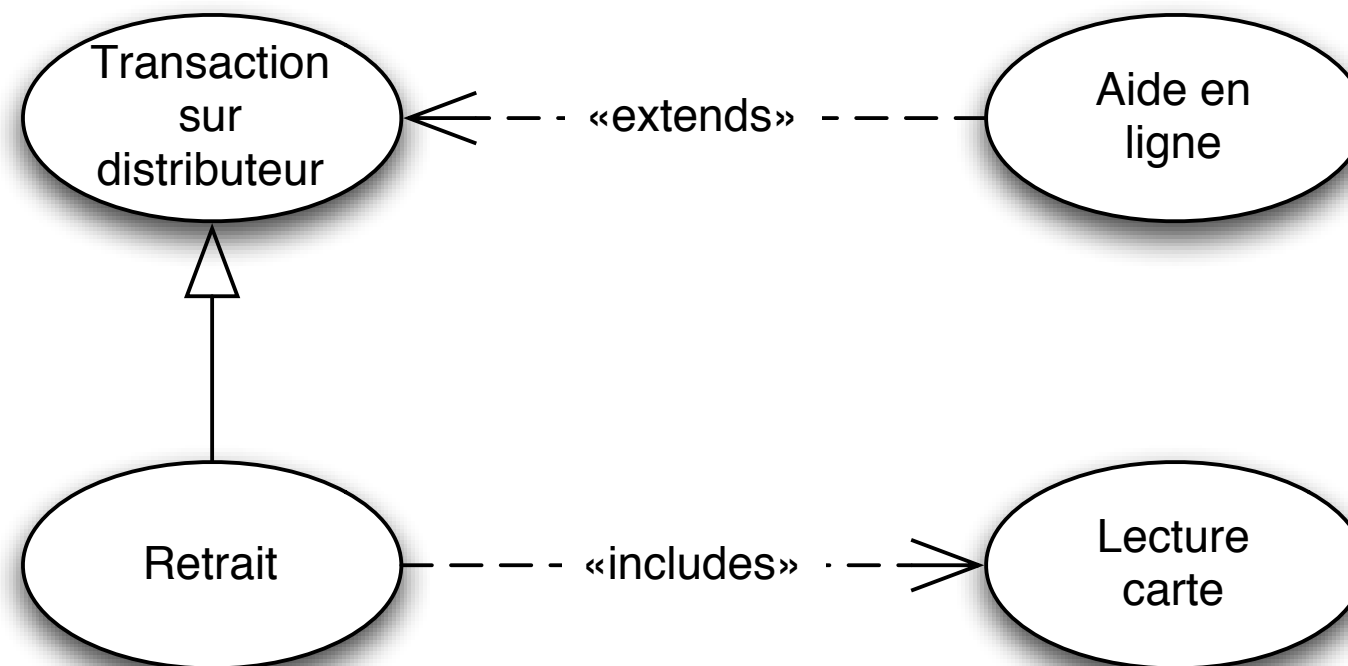
Relation acteur \leftrightarrow acteur : généralisation



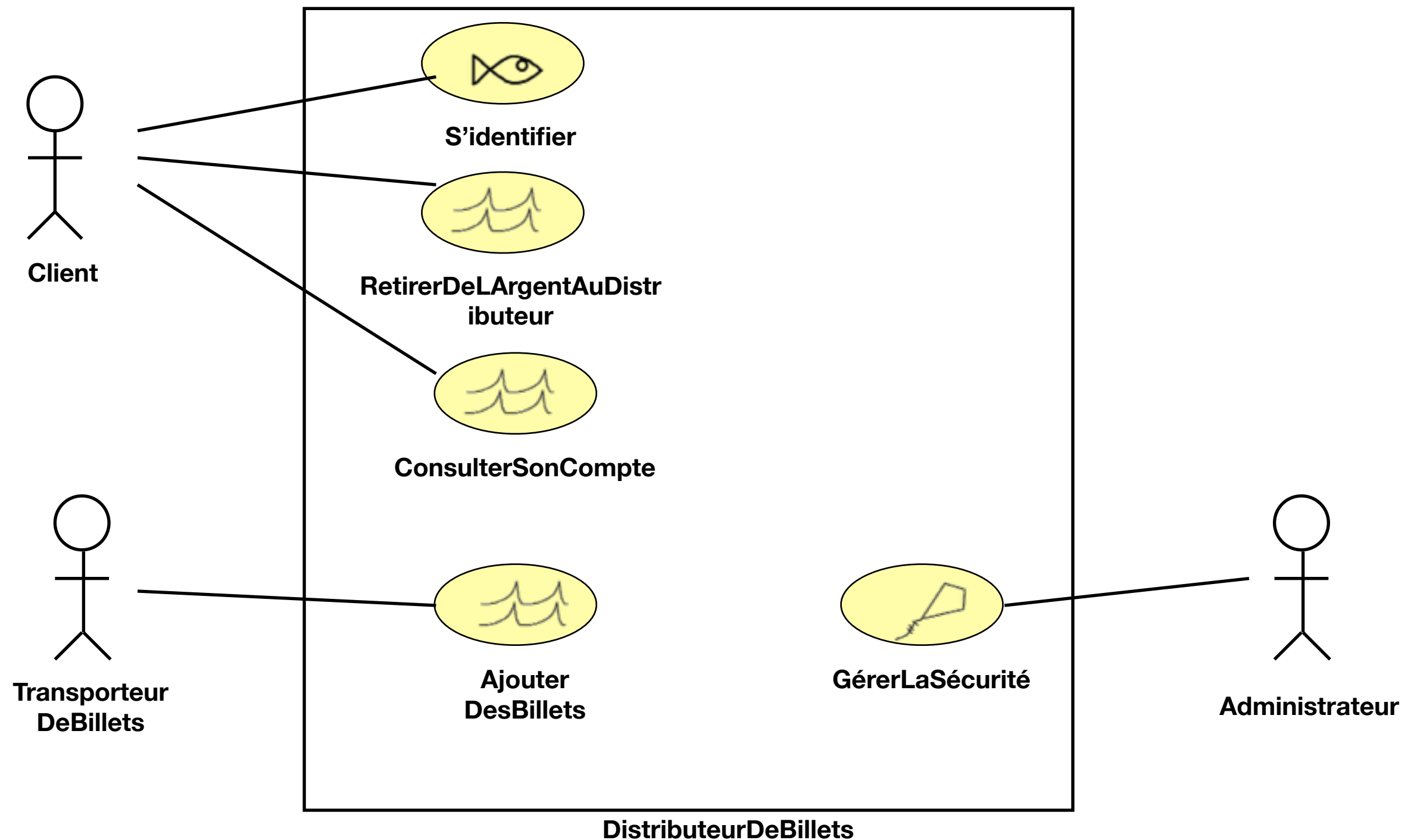
Notation — résumé



Notation — résumé




Exemple de marquage



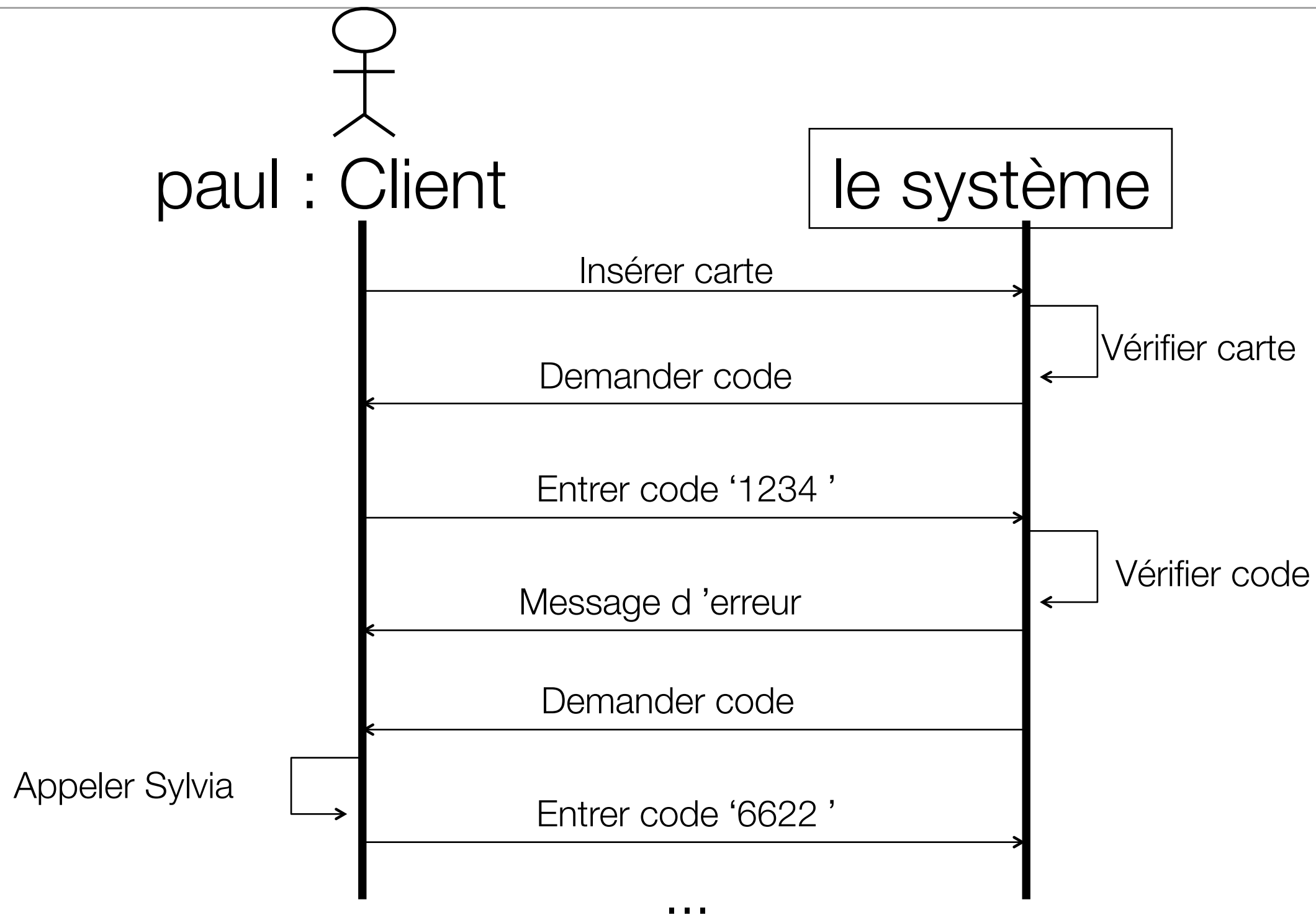
Cas d'utilisation *vs.* Scénarii

- Un scénario est un exemple :
 - une manière particulière d'utiliser le système ...
 - ... par un acteur particulier ...
 - ... dans un contexte particulier.
- cas d'utilisation = ensemble de scénarios
- scénario = une exécution particulière d'un CU

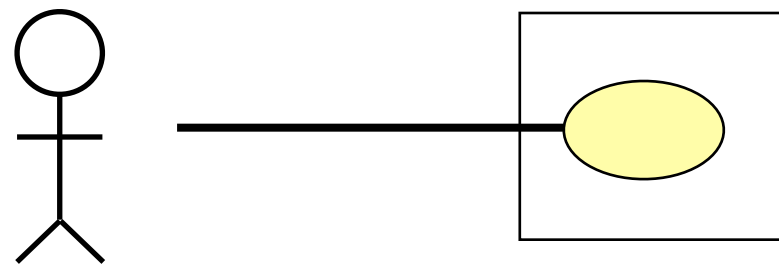
Diagramme de séquences système

- Pour décrire un scénario : un diagramme de séquences
 - Diagramme de séquences :
 - L'une des notations UML, une notation générale
 - Peut être utilisée dans de nombreux contextes
 - Permet de décrire une séquence de messages échangés entre différents objets
 - Différents niveaux de détails
 - Pour décrire  un scénario simple, deux objets : l'acteur et le système
- ⇒ « Diagramme de séquences système »

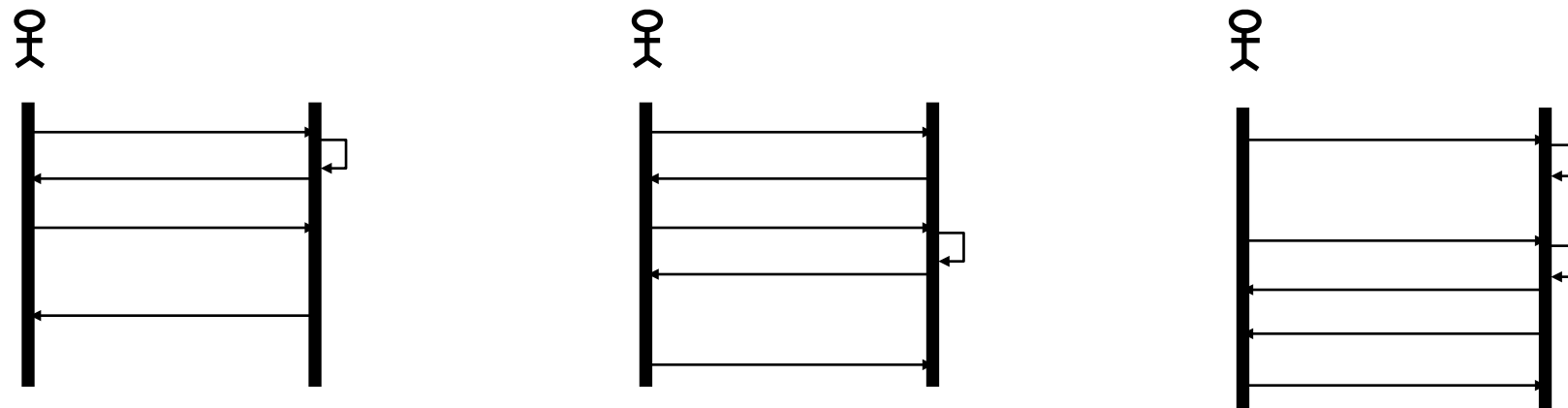
Exemple de diagramme de séquence système



Cas d'utilisation *vs.* scénarii



Niveau modèle



Niveau instances

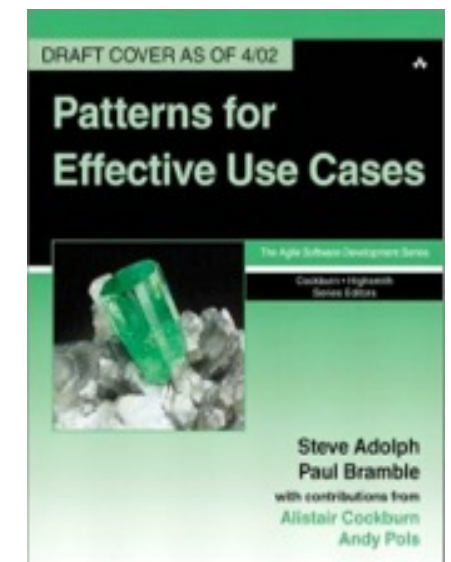
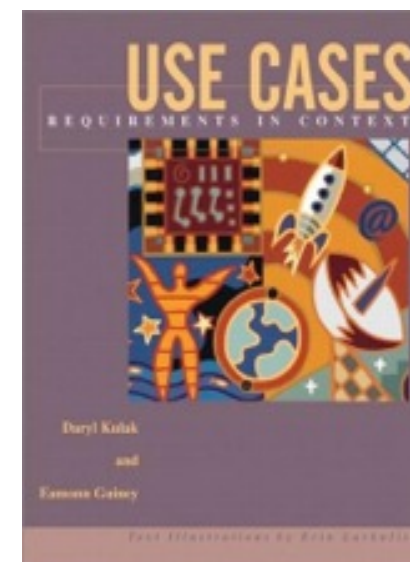
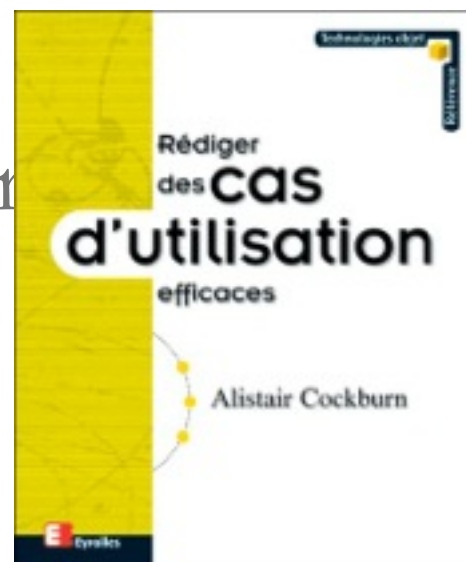
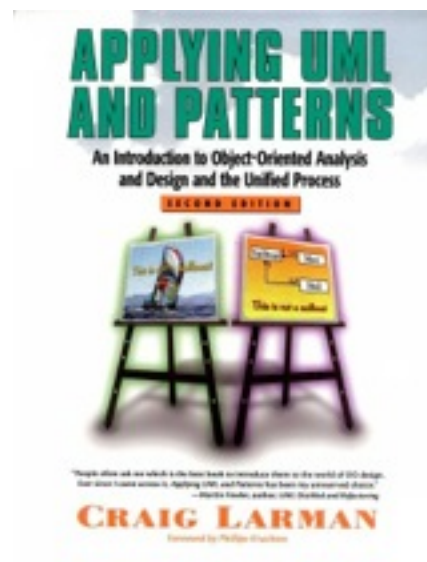
Un peu de méthodologie...

Stabilisation du modèle par **consensus** **grandissant**

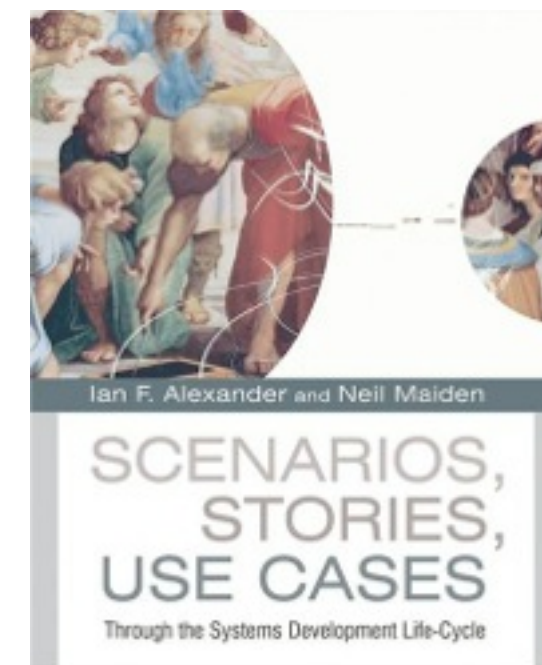
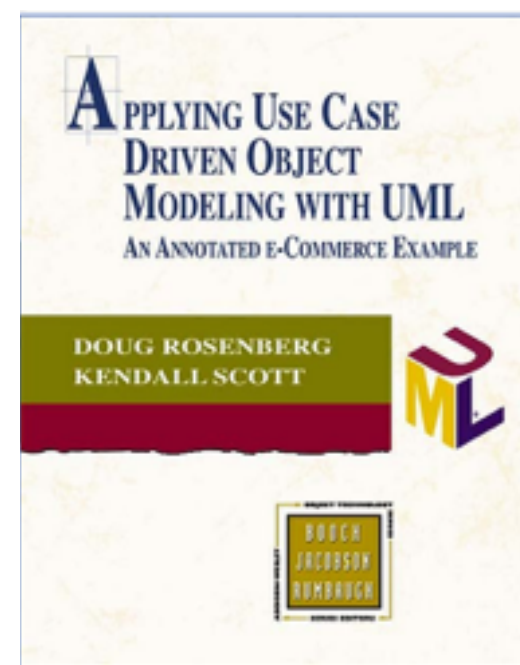
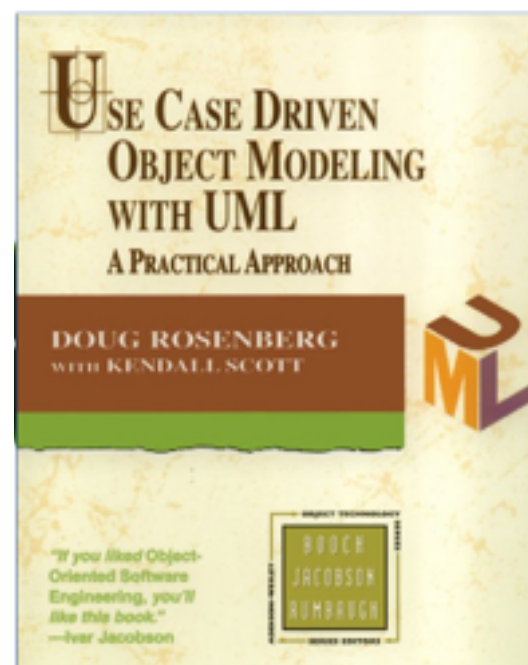
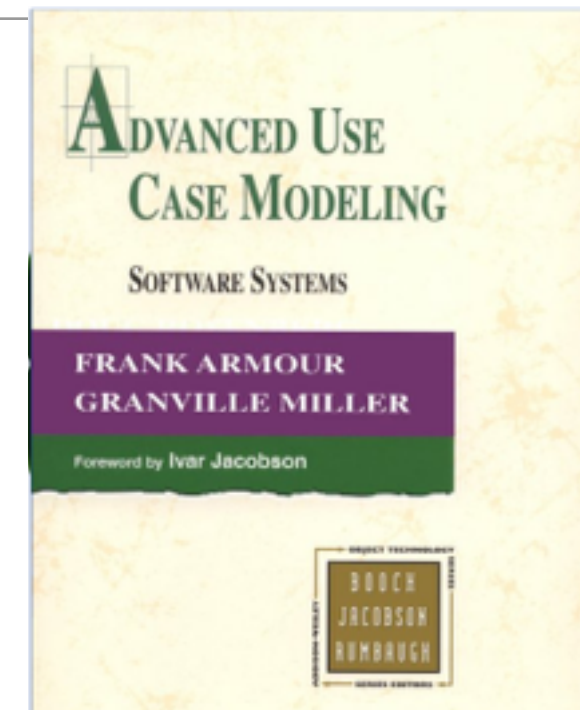
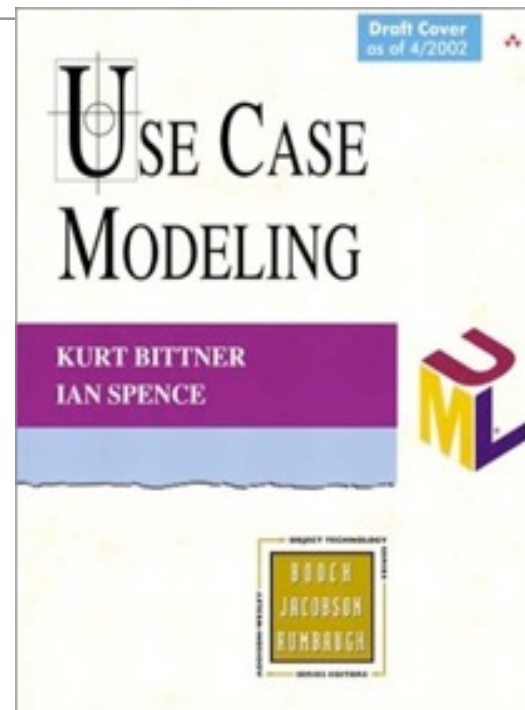
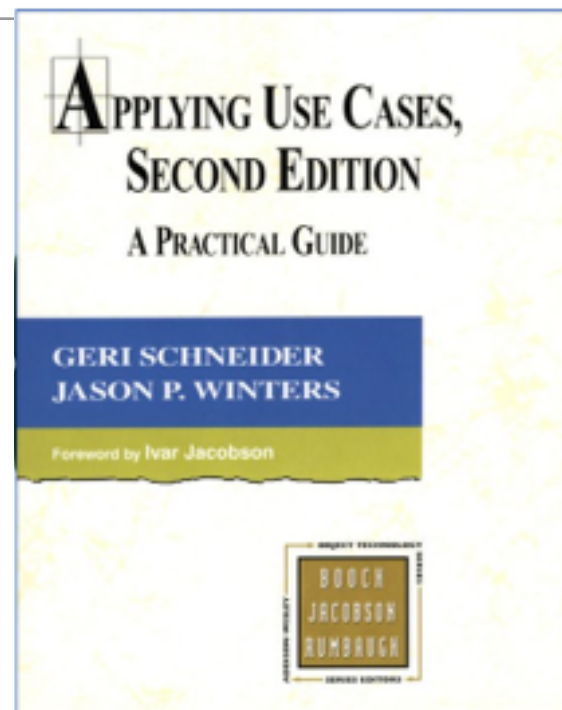
- Équivalent à définir une **table des matières** et des résumés pour chaque chapitre
 - Pas de règles strictes
 - Effectuer les meilleurs regroupement possibles
 - Rester simple !
 - Structuration possible en termes de paquetages
 - Culture d'entreprise

Pour en savoir plus

- Pour un template "standard" de description de cas d'utilisation : <http://alistair.cockburn.us/Basic+use+case+template>



Pour en savoir *encore* plus



Diagrammes structurels

Diagrammes Structurels

- Classes
- Objets
- Structures composite

Diagramme de classes

- Classificateurs, attributs et opérations
- Associations
- Généralisation
- Dépendances
- Paquetages

Classificateurs

Rappel: objet

- Encapsulation d'un état et d'un ensemble d'opérations (qui s'appliquent à cet état)
 - Abstraction d'une entité réelle
 - Identité unique
 - Existence temporelle (création, modification, destruction)

Rappel: classe

- Description du comportement et des propriétés communs à un ensemble d'objets
 - Un objet est une instance d'une classe
 - Chaque classe a un nom

Notation

- Sans détails
- Analyse
- Conception
- Propriétés groupées selon leur visibilité

HTML Page

HTML Page

title : String
size : Integer
render()
save()

HTML Page

+ title : String
+ /size : Integer
~ version : Integer
contents : String
- visibility : Boolean = true
- tags : String [1..*]
+ render()
+ save()
- optimize()

HTML Page

public
title : String
/size : Integer
package
version : Integer
protected
contents : String
private
visibility : Boolean = true
tags : String [1..*]
public
render()
save()
private
optimize()

Attributs

- Caractéristique structurelle typée d'un classificateur, qui spécifie la structure des instances de ce classificateur
- Peuvent être dérivés d'autres attributs

Attributs

Syntaxe: [visibility][/]name[:type][multiplicity]
[=default][{prop-modifier}]

- Propriétés:
 - {readOnly}, {union}, {subsets <p>}, {redefines <p>}, {ordered}, {unique}, {id}, {non-unique}, <prop-constraint>.
- Visibilité: [+ , ~ , # , -]
- Peuvent être soulignés (attributs de classe)

Opération

- Caractéristique comportementale d'un classificateur qui spécifie le nom, les types, les paramètres et les contraintes pour invoquer un comportement associé

Opération

Syntaxe: [visibility]name (parameter-list):return-type {oper-properties}-

parameter: [direction]name : type-expression [multiplicity] =
default-value [param-properties]

- Peut être soulignée (opération de classe)
- Propriétés: redefines<operation-name>, query, ordered, unique
- direction: in, out, inout

Autres classificateurs

- Classes abstraites
- Datatypes
- Interfaces
- Classes paramétrées

Classes abstraites

- Encapsulent un comportement commun
- Ne possèdent pas d'instances
- Possèdent des opérations abstraites

Etudiant

**Etudiant
{abstract}**

Classes abstraites

- deferred en Eiffel
- abstract en Java
- pure virtual en C++
- méthodes `^self shouldNotImplement` en Smalltalk

Datatypes

- Type, dont:
 - les valeurs n'ont pas d'identité
 - les opérations sont de fonctions “pures”

«dataType» Date
day : Integer month : Integer year : Integer

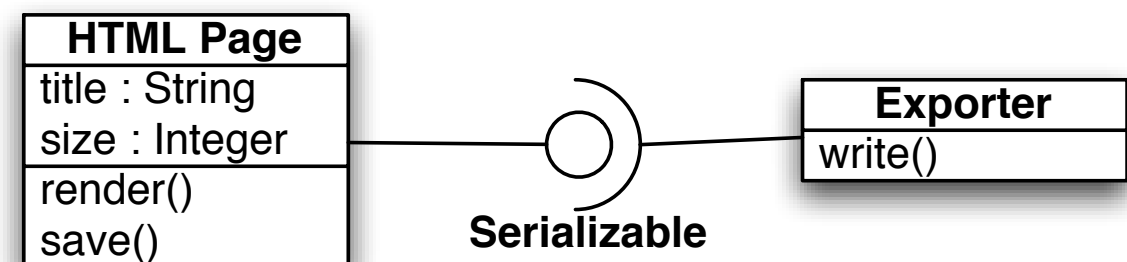
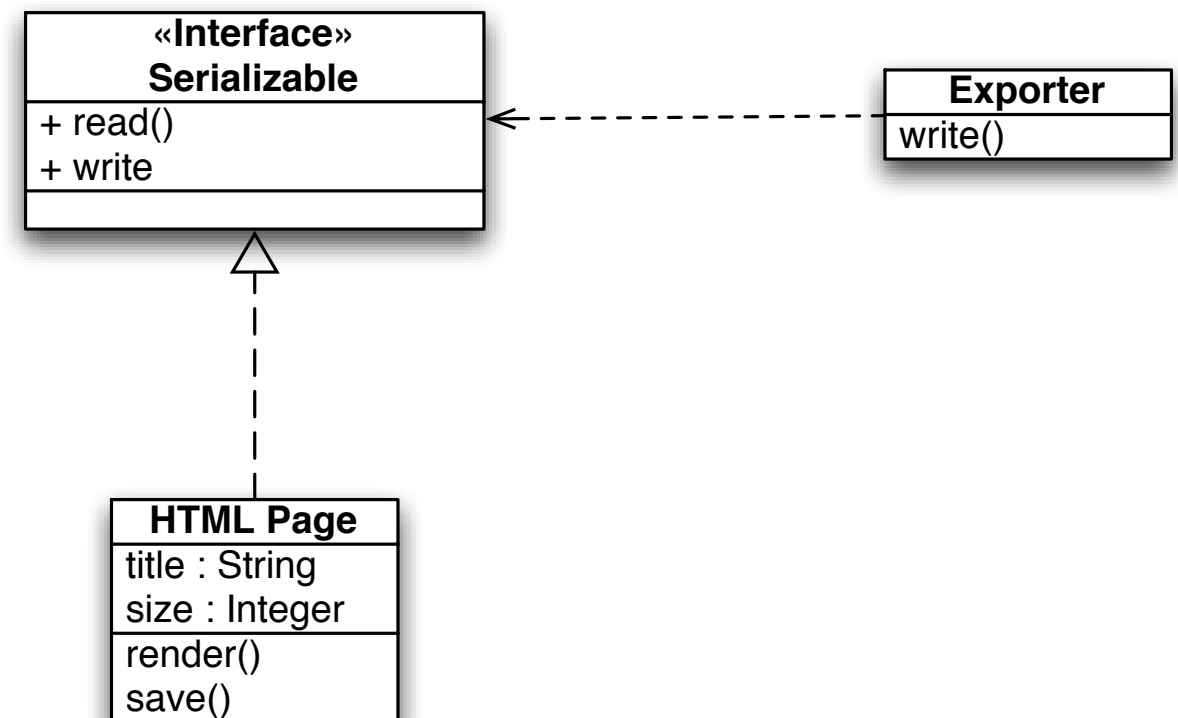
Datatypes

- Enumeration
- Types primitifs:
 - Boolean, Integer, UnlimitedNatural, String

«enumeration» Temperature
Cold
Hot

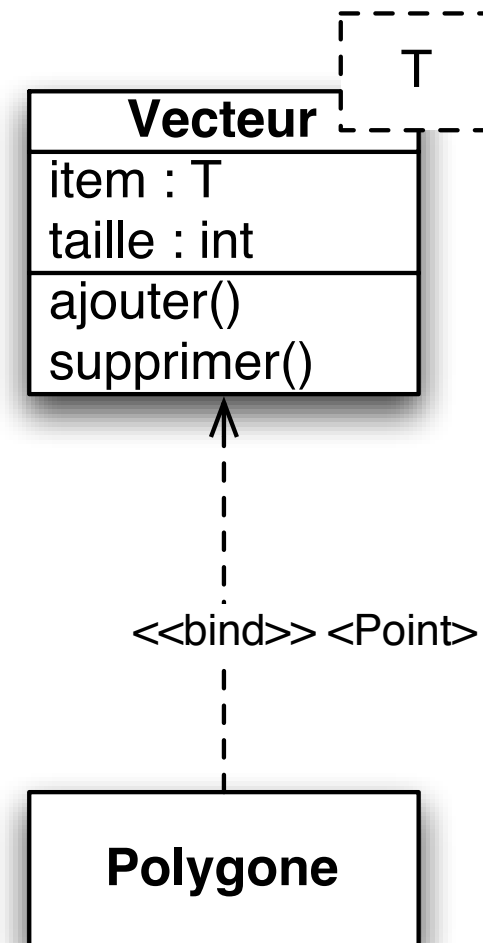
Interfaces

- Une déclaration d'un ensemble cohérent d'opérations
- Une interface peut être implémentée ou demandée



Classes paramétrées

- Indispensable pour les classes conteneurs (e.g. Listes)
- Existe en Ada, Eiffel, C++ (templates) et Java (>1.5)
- N'est pas nécessaire en Smalltalk et Python



Associations

Rappel: association

- Une association entre deux classes représente un lien stable entre deux objets (instances de ces classes)

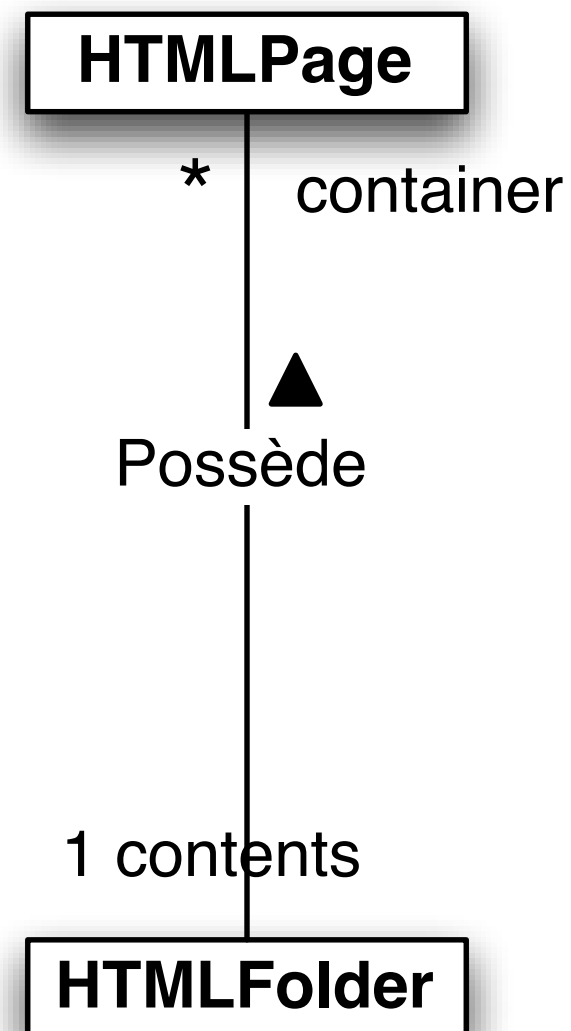
Rappel: association

```
feature {VISITOR}  
  accept(v: VISITOR) is  
    do  
      v.visitExemple(Current)  
    end -- accept
```

- Un lien d'utilisation, temporaire, n'est pas une association

Notation

- Nom
- Rôles
- Cardinalités
- Direction de lecture
- Navigabilité



Cardinalités - Notation

- Exacte : 1
- Plusieurs (0 à n): *
- Optionnelle: 0,1
- Spécifiée: 1, 2, 4
- Intervalle: 1-10

Propriétés des rôles

- `{set}` : chaque élément est présent au plus une fois (par défaut)
- `{bag}` : un élément peut être présent plus d'une fois
- `{ordered}`: set ordonné
- `{sequence}` or `{seq}`: bag ordonné

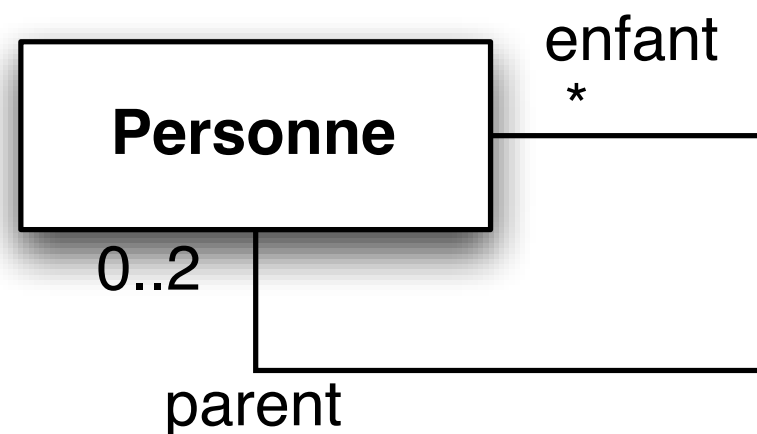
Propriétés des rôles

- {redefines <end-name>} : redéfinition d'un autre rôle
- {subsets <property-name>} : le rôle représente un sous-ensemble
- {union} : union dérivée de tous ses sous-ensembles

Cas particuliers (1 / 2)

- Association réflexive
- Association n-aire
- Association qualifiée
- Association-classe
- Agrégation partagée
- Agrégation composite

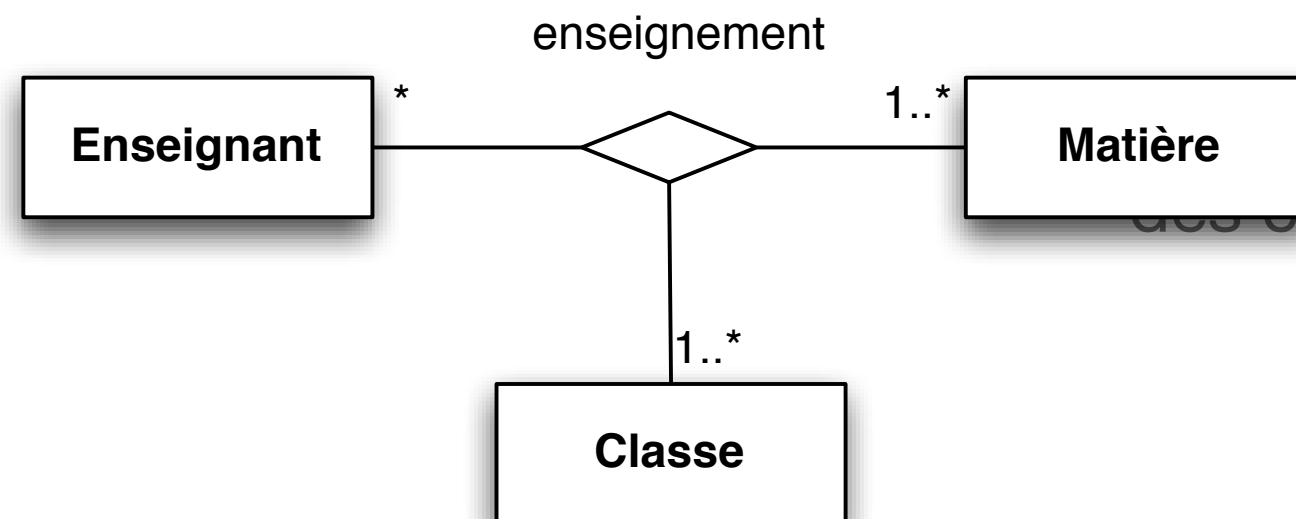
Association réflexive



- Liaison entre objets de la même classe

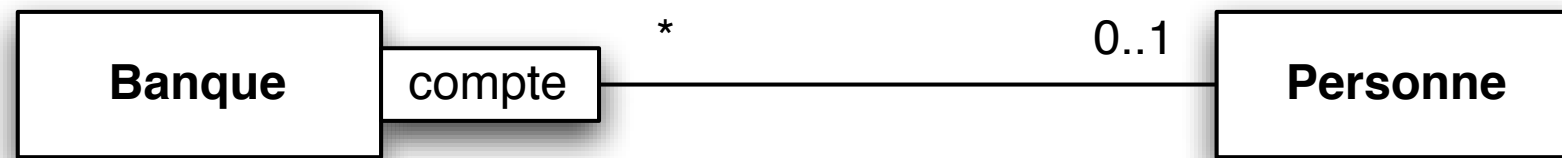
Association n-aire

- Association entre plus de deux classes



interprétation confuse
des cardinalités

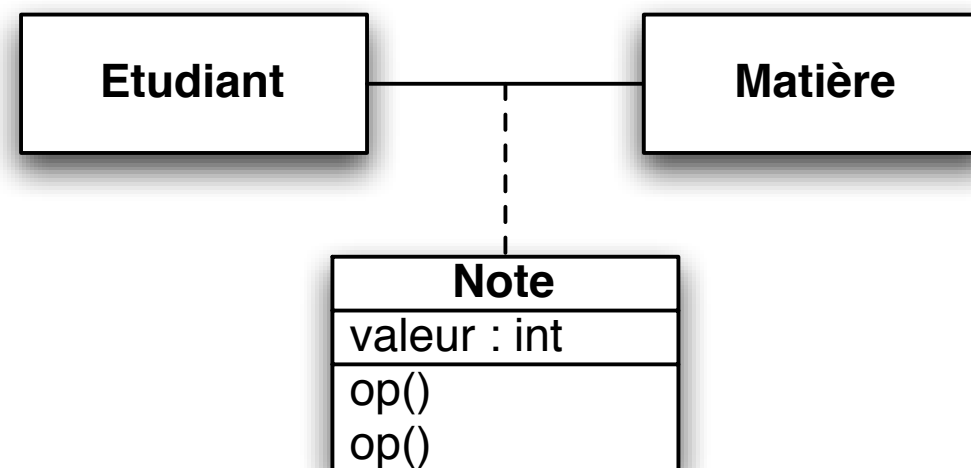
Association qualifiée



- Qualificatif: attribut spécial, qui permet de réduire la cardinalité (de n à 1)
- Une clef permettant de distinguer un objet de façon unique

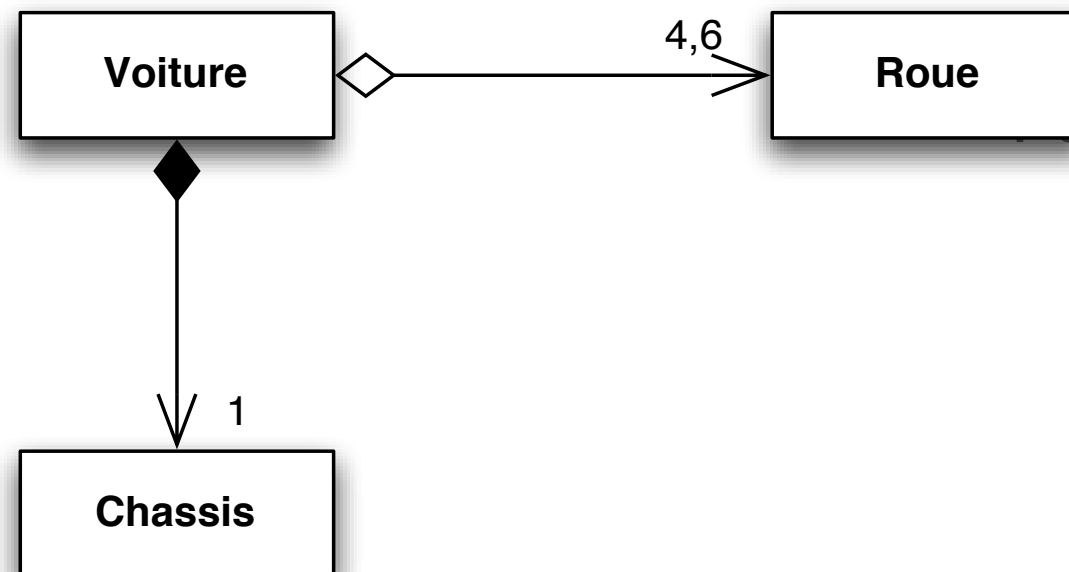
Association-classe

- Aussi appelée classe de liaison



Agrégation

- Notion de “tout” et “partie”



Partagée et composite

Généralisation

Généralisation

- Héritage, caractéristique des langages à objets
- Partage de structure et comportement entre classes

Utilisation (1 / 2)

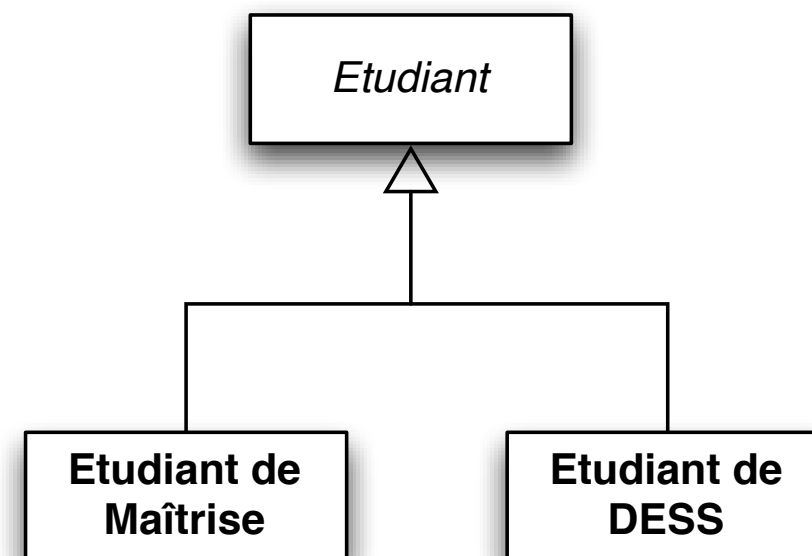
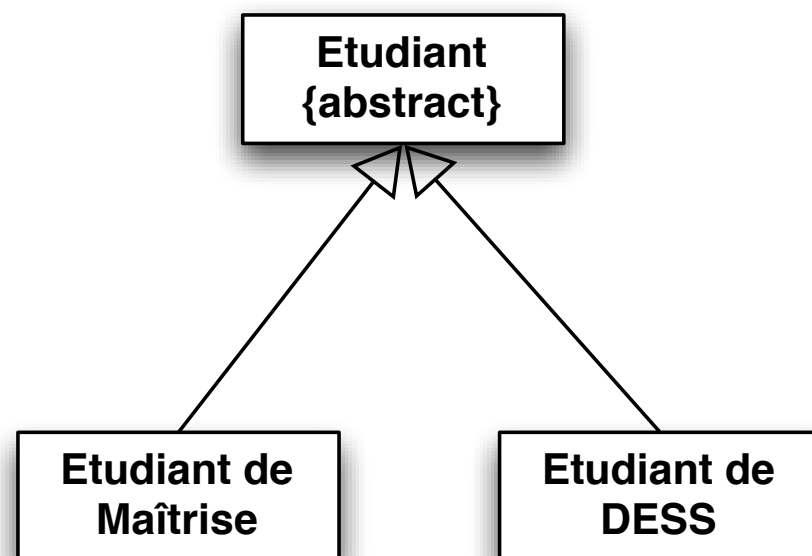
- Mécanisme d'extension de module
 - ajout de fonctionnalités dans la sous-classe
 - réutilisation de code

Utilisation (2/2)

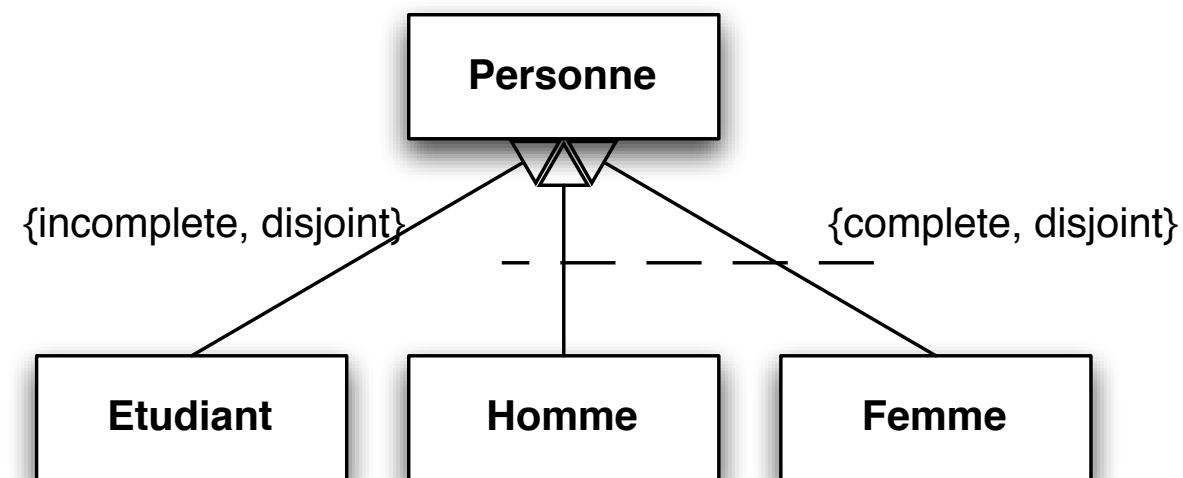
- Mécanisme de classification : sous-typage
 - relation X est-une-sort-de Y (est substituable à)
 - organisation des systèmes complexes (cf. Linnaeus)
 - réutilisation d'interface

Notation

- Relation “est une sorte de”, “est remplaçable par”



Generalization Set



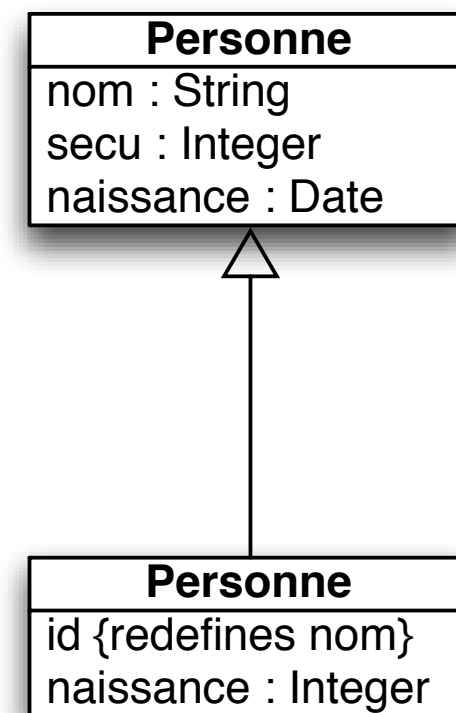
- complete/**incomplete**: couverture. Une personne est soit un homme, soit une femme.
- **disjoint**/overlapping: si les classificateurs partagent des instances. Une personne peut être aussi un étudiant.

Généralisation

- Attributs
- Opérations
- Associations
- Contrats, états, etc. (mais cela, nous verrons plus tard)

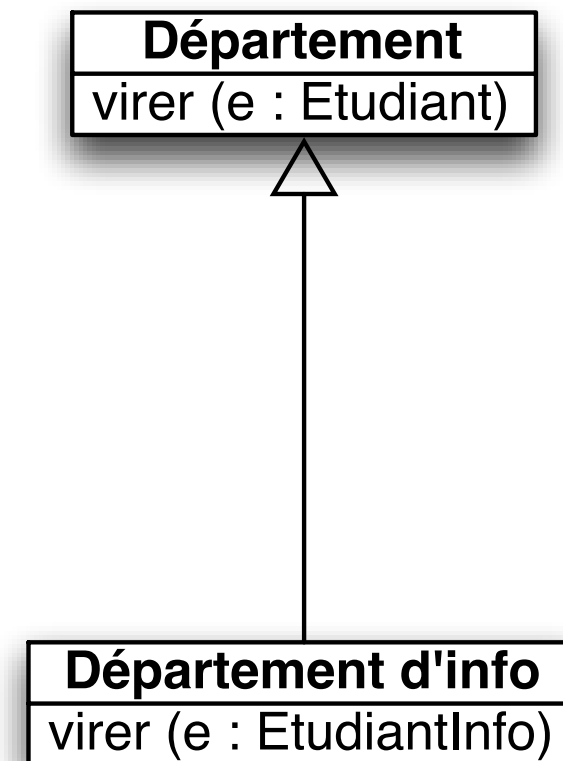
Attributs

- Un attribut qui a le même nom qu'un attribut hérité est une redéfinition

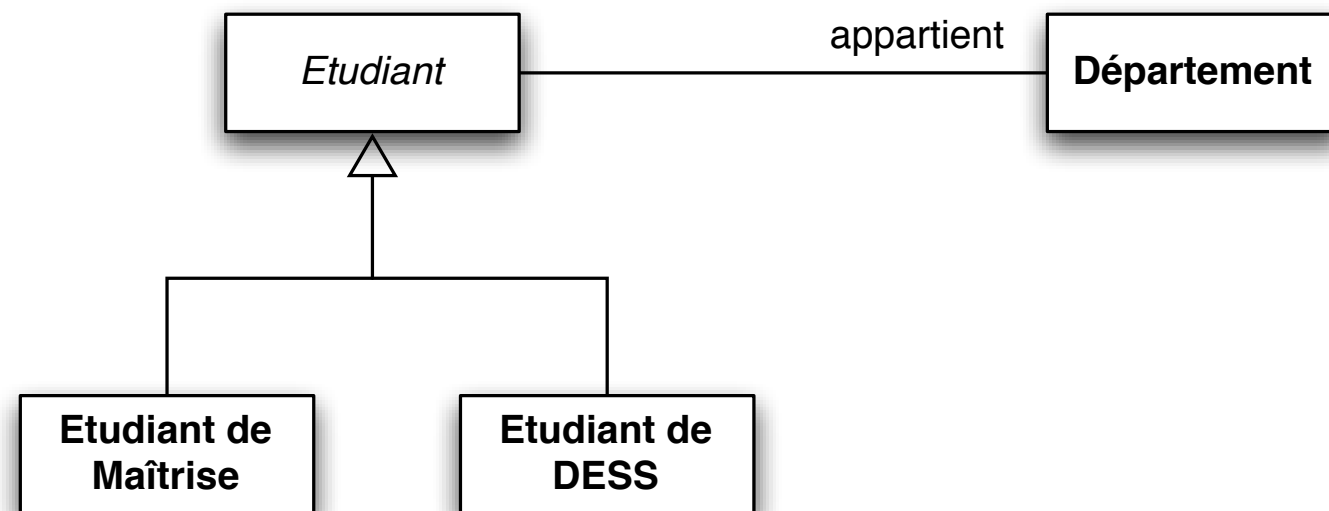


Opérations

- Une opération peut être redéfinie si:
 - elle a le même nombre de paramètres
 - Le type de chaque paramètre est conforme au paramètre correspondant



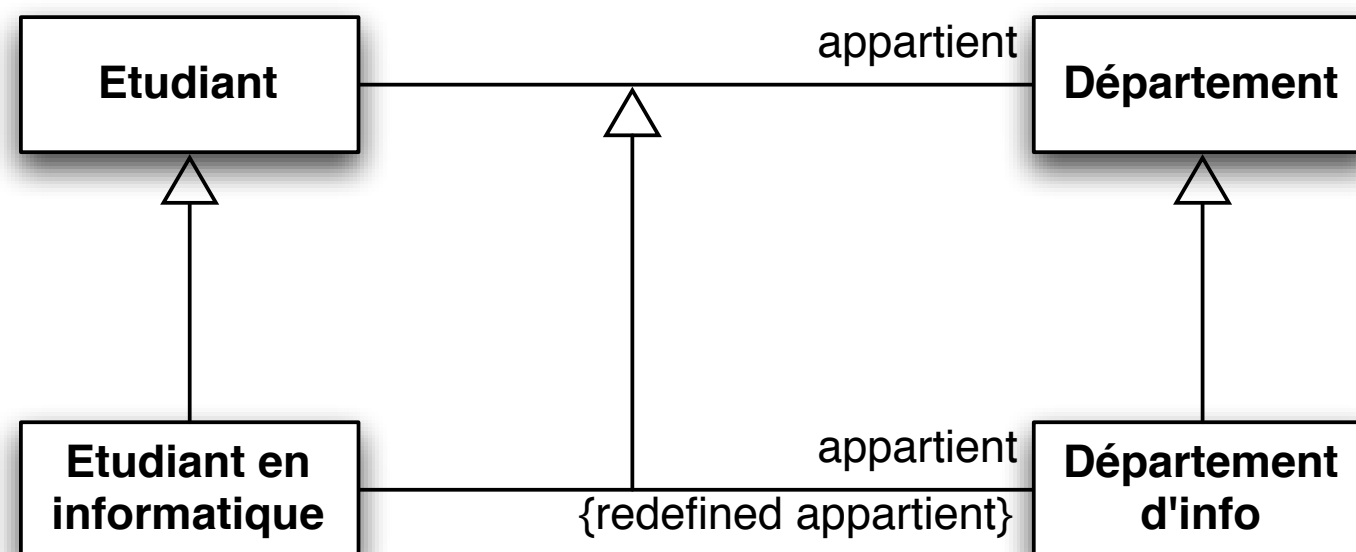
Associations



- Les associations sont héritées par les sous-classes

Associations

- Il est possible de spécialiser une association
- Double emploi avec `<redefined>` ?



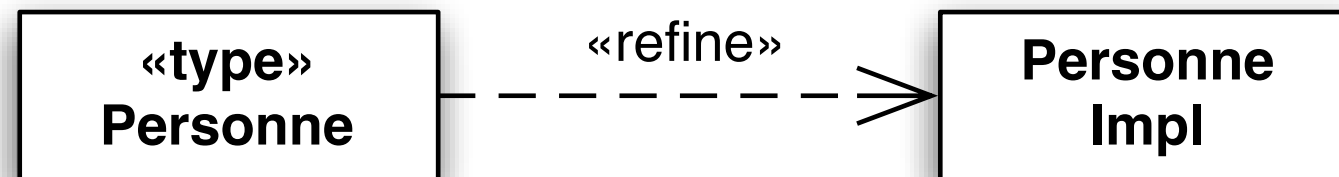
Dépendances

- Relation entre deux éléments (client et fournisseur)
 - Abstraction
 - Permission
 - Réalisation
 - Substitution

Abstraction (1 / 2)

- Entre éléments de différents niveaux sémantiques
 - «derive» : éléments dérivés, pas forcément du même type
 - «refine» : raffinement entre modèles
 - «trace» : (aussi) - utilisé pour marquer les changements entre modèles.

Abstraction (2/2)

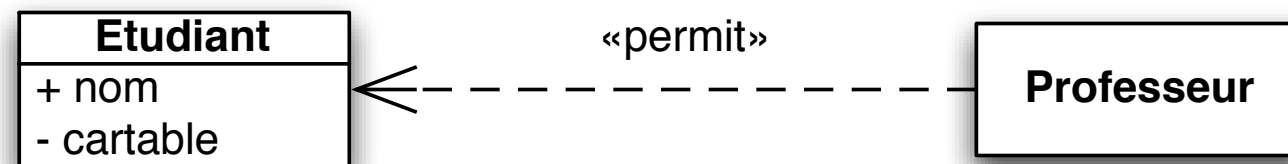


- Relation entre un «type» du modèle d'analyse avec une classe du modèle de conception

Intérêt de l'abstraction

- Outillage de méthodes de développement (RUP, Catalysis, etc.)

Permission



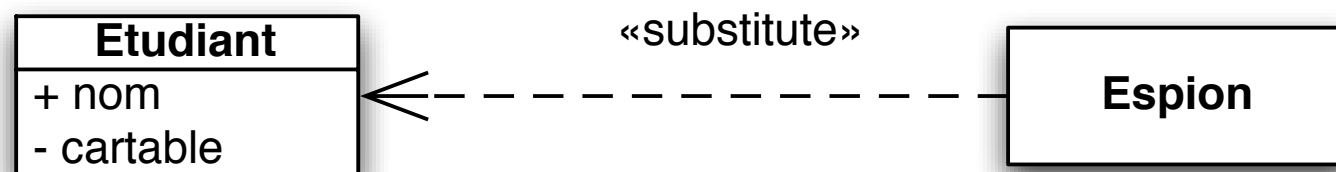
- Droits d'accès aux propriétés privées

Réalisation

- Relation entre deux ensembles d'éléments, de spécification et d'implémentation
- Utilisé pour représenter: raffinement, optimisation, transformations, synthèses, etc.
- «realize»

Substitution

- Relation entre classificateurs. Les instances du fournisseur peuvent remplacer celles du client



Usage

- «call» : dépendance entre opérations (où classes)
- «create» : le client crée des instances du fournisseur (classificateurs)
- «instantiate» : les opérateurs du client créent des instances du fournisseur

Usage (2/2)

- «send» : entre une opération et un signal

Intérêt de l'usage

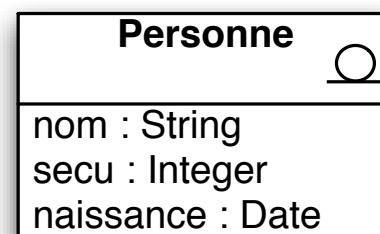
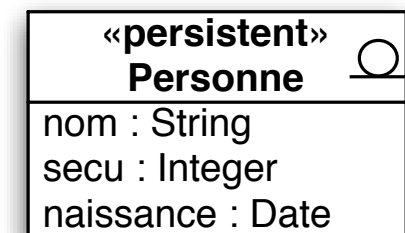
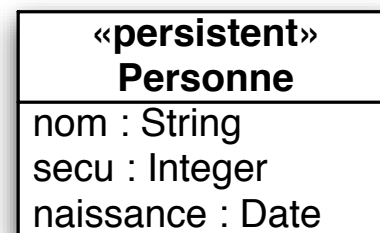
- Représentation des associations non permanentes
- Très utiles pour estimer la testabilité d'un modèle ou l'impact d'un changement

Stéréotypes

- Principal mécanisme d'extension d'UML
- Permet l'extension (ou plutôt la restriction) de la plupart des éléments de modélisation: classe, objet, opération, acteur, cas d'utilisation, diagrammes, etc.

Notation

- Peuvent introduire:
 - une nouvelle notation
 - des contraintes d'utilisation
- Très utiles pour l'interprétation d'un modèle



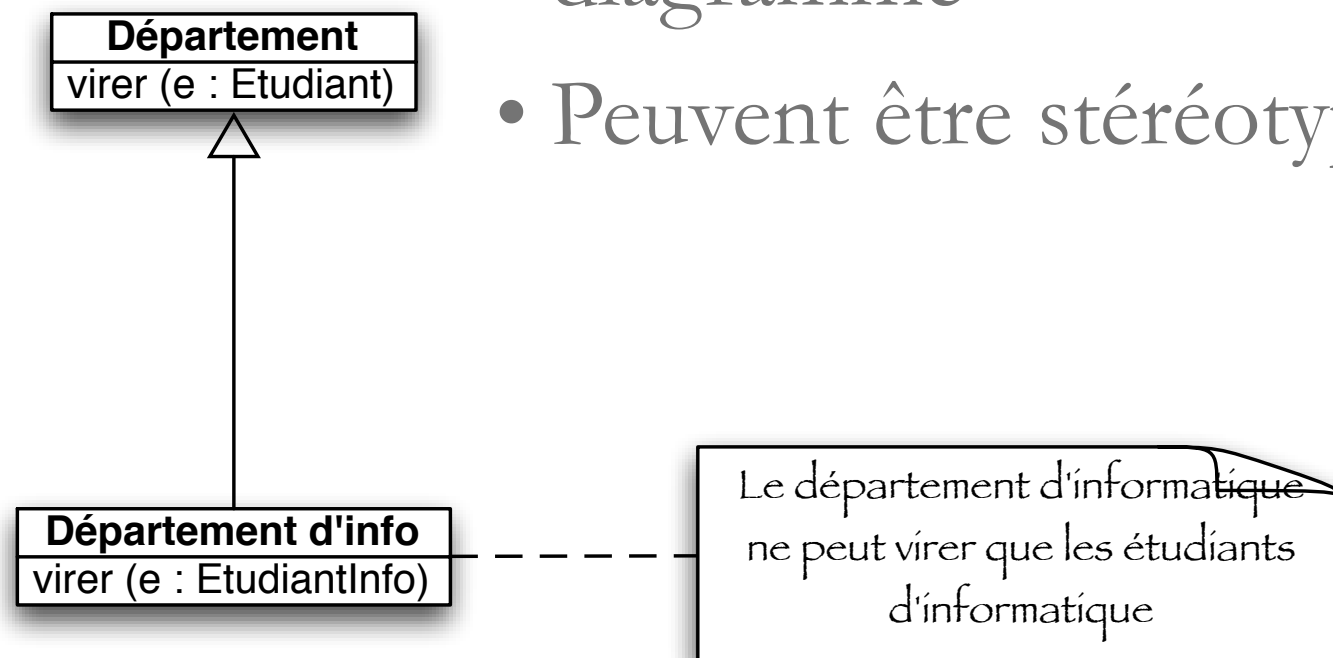
Etiquettes (tagged values)

- Pair de la forme “étiquette=valeur”, attachée aux éléments de modélisation

Personne
{author = G. Sunyé, status = incomplete}
nom : String secu : Integer naissance : Date

Notes

- Compléments de modélisation sous forme textuelle, attachées à un élément ou un diagramme
- Peuvent être stéréotypées



Profils

- Mécanisme d'extension, permettant d'adapter UML à l'aide d'un ensemble de stéréotypes
- Le profil “Standard” contient les stéréotypes utilisés dans UML
- Autres profils: J2EE/EJB, COM, .NET, CCM, etc.

Paquetage (1 / 3)

- Groupement d'éléments, qui leur fournit un espace de nommage commun

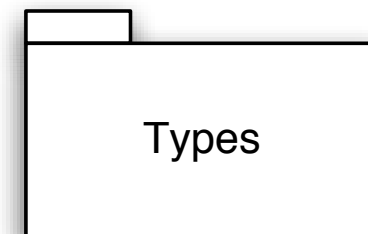
Paquetage (2/3)

- Utilisé pour structurer une application
 - Il se compose de classes et d'autres paquetages
 - Il référence des classes ou d'autres paquetages

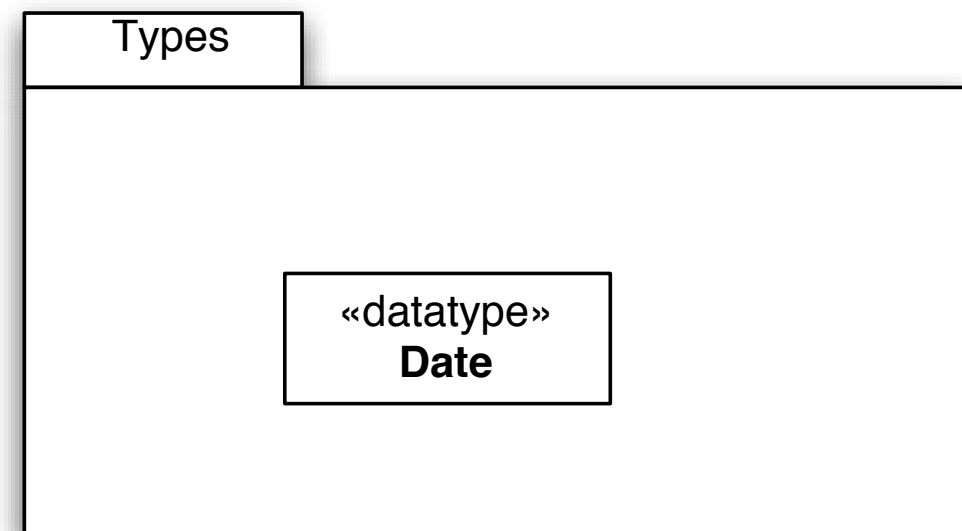
Paquetage (3/3)

- Utilisé pour régler la visibilité des classes et paquets qui le composent

Notation

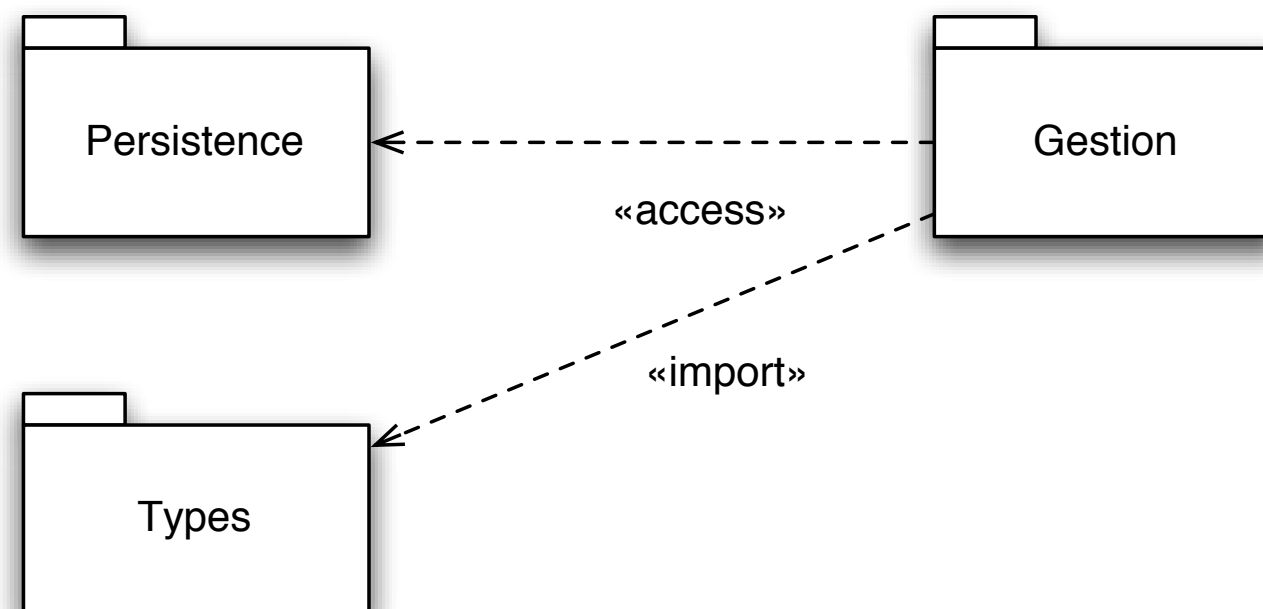


- Vues externe et interne



Dépendances entre paquetages

- Importation («import»): les éléments de l'espace de nommage sont importés
- Accès («access»): les éléments sont simplement utilisés

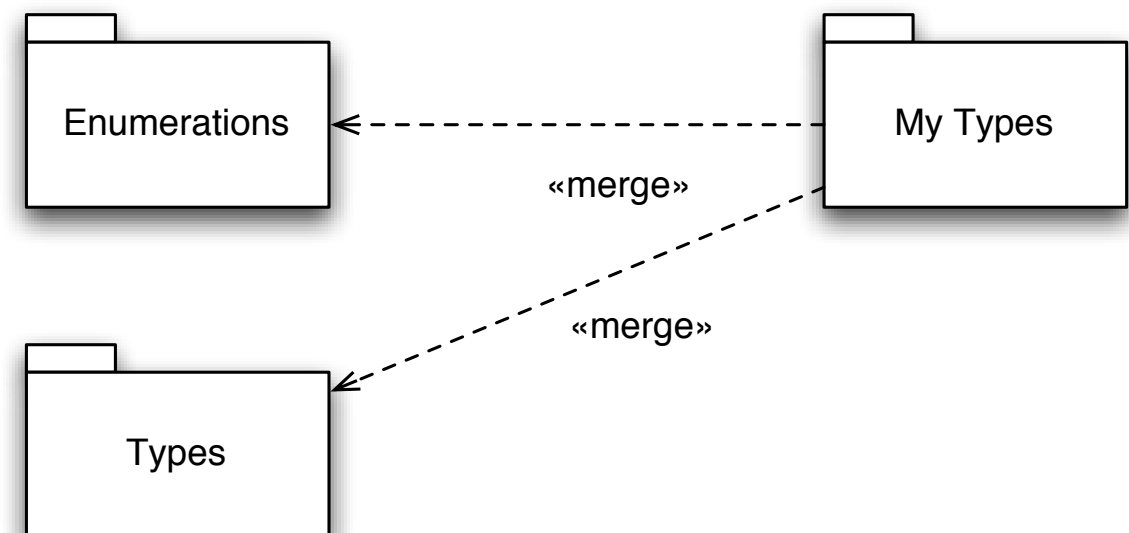


Fusion de paquetages

- Une fusion définit la manière dont un paquetage étend un autre
- Composé de liens de généralisation et de redéfinition

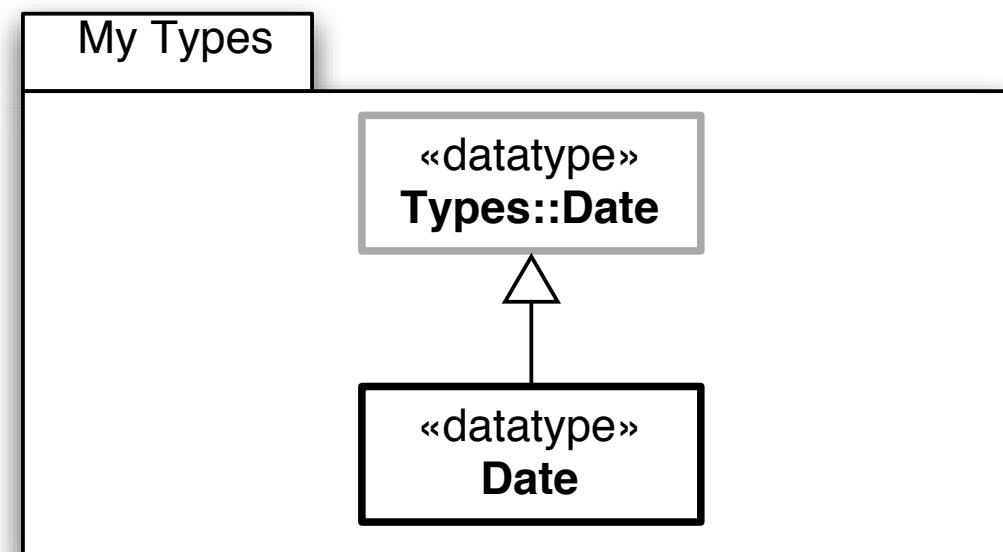
Notation

- Vue externe



Notation

- Vue interne



Diagrammes Structurels

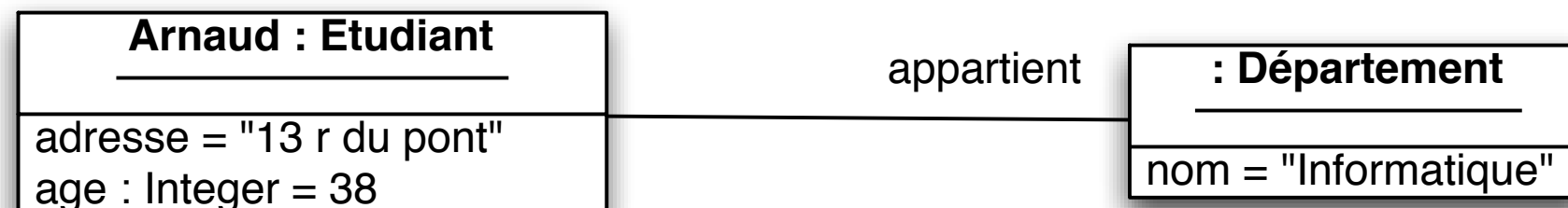
- Classes
- Objets
- Structures composite

Diagramme d'instances

- Représentation d'objets, de liens et de valeurs.
- Très utiles pour expliquer les associations entre classes ou le résultat d'une opération

Notation

- Similaire à celle d'une classe
- Le nom de l'instance et celui de la classe sont soulignés et séparés par “:”
- Les deux noms sont facultatifs



UML — Composants

Gerson Sunyé
gerson.sunye@univ-nantes.fr

Éléments du diagramme de composants

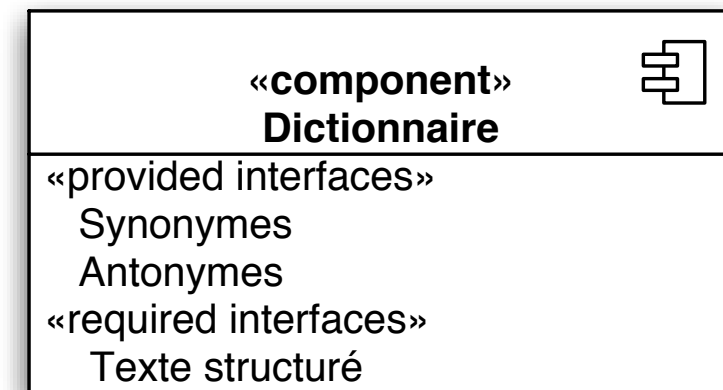
- Composant
- Connecteur
- Artefact
- Port

Définition

- Composants:
 - Partie remplaçable d'un système.
 - Son comportement est spécifié en terme d'interfaces requises et fournies.

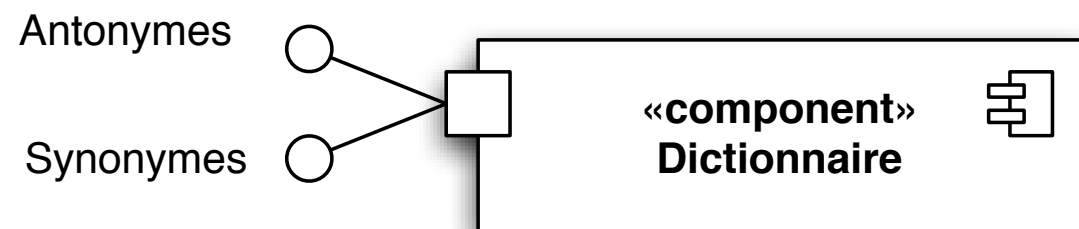
Notation

- Interfaces requises et fournies.



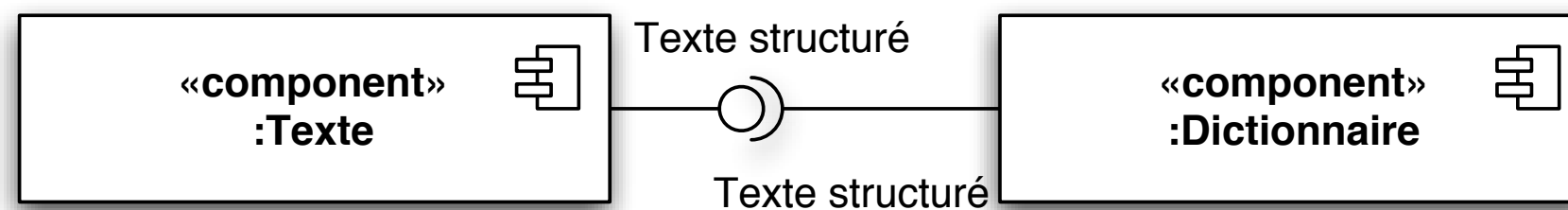
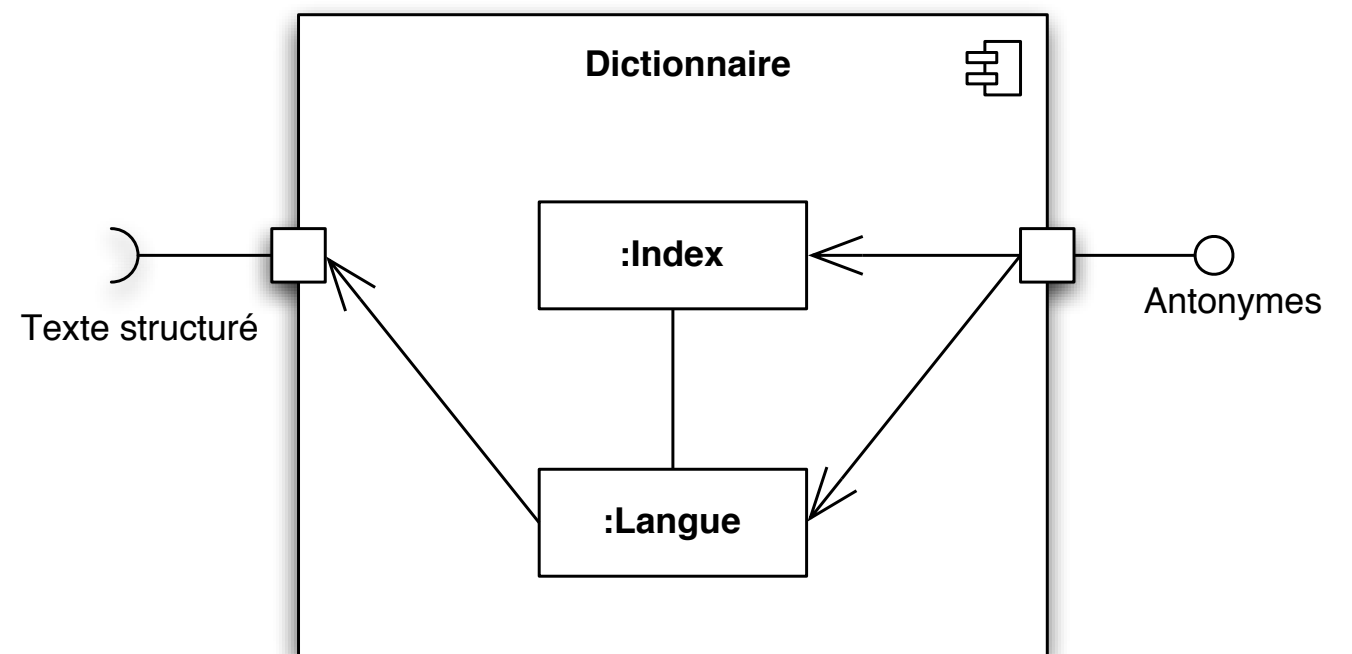
Port

- Point d'interaction entre un composant et son environnement.
- La nature de ces interactions est spécifiée par des interfaces.



Connecteur

- Liaison entre un contrat externe (port) et la réalisation.
- Assemblage, délégation.



Diagrammes Structurels

- Classes
- Objets
- Composants
- Déploiement
- Structures composite

UML – Composants

Éléments du diagramme de composants

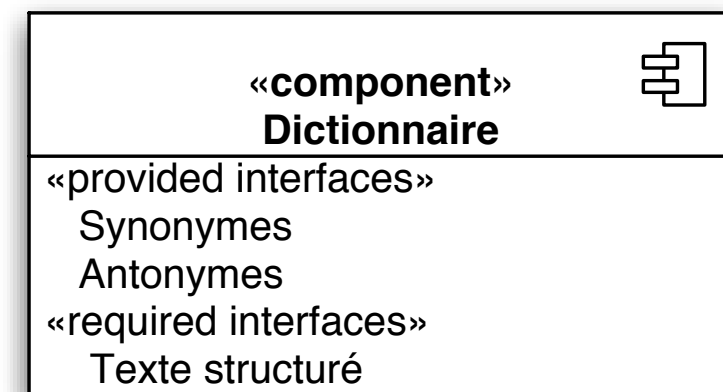
- Composant
- Connecteur
- Artefact
- Port

Définition

- Composants:
 - Partie remplaçable d'un système.
 - Son comportement est spécifié en terme d'interfaces requises et fournies.

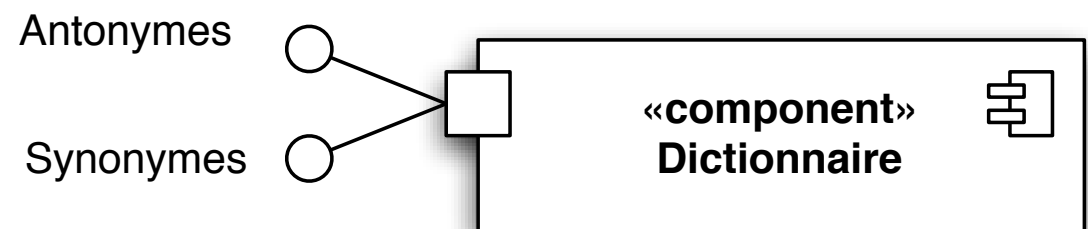
Notation

- Interfaces requises et fournies.



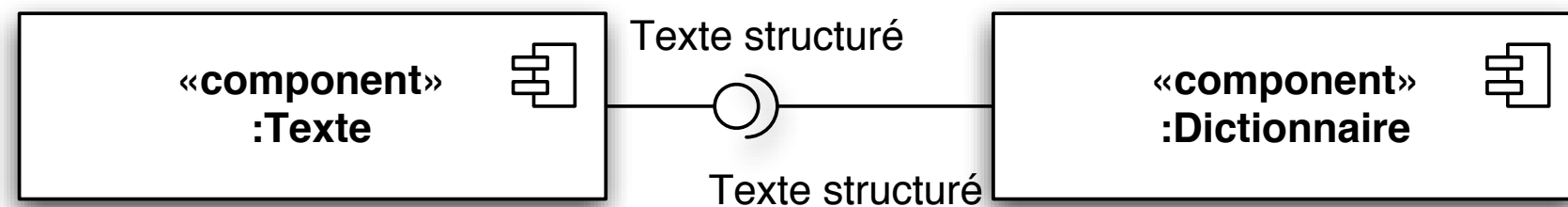
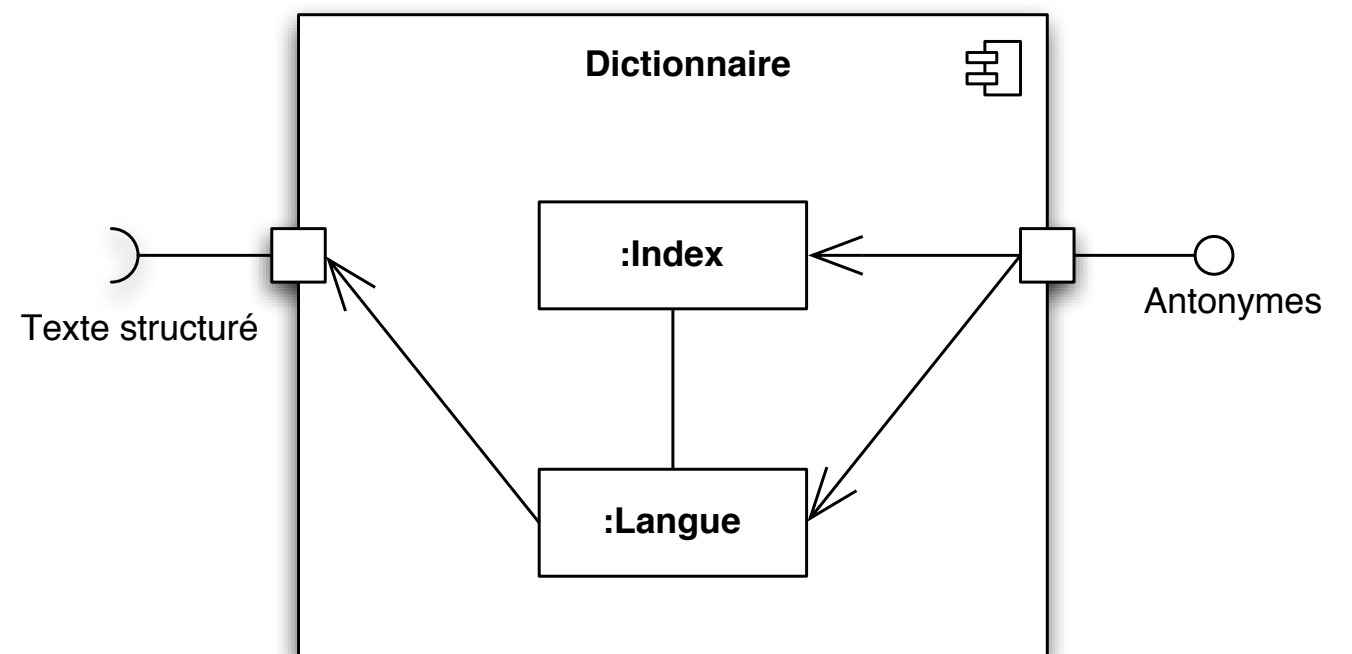
Port

- Point d'interaction entre un composant et son environnement.
- La nature de ces interactions est spécifiée par des interfaces.



Connecteur

- Liaison entre un contrat externe (port) et la réalisation.
- Assemblage, délégation.



Diagrammes Structurels

- Classes
- Objets
- Composants
- Déploiement
- Structures composite

UML — Déploiement

Diagramme de déploiement

- Éléments de modélisation:
 - Artefact
 - Chemin de communication
 - Déploiement
 - Dispositif
 - Noeud

Artefact

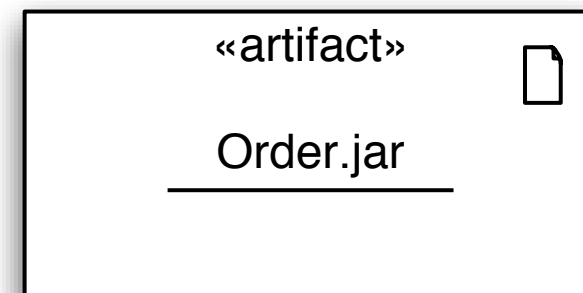
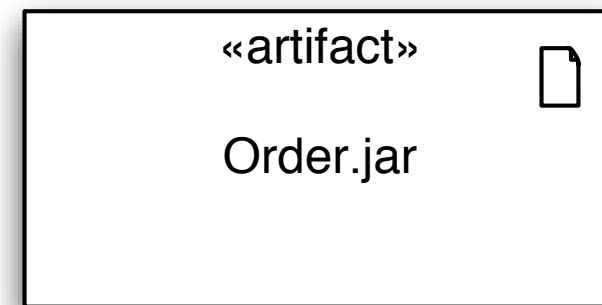
- Spécification d'une pièce physique d'information utilisée dans le processus de développement
- Fichiers source, modèles, scripts, binaires, document, etc.

Artefact

- Un artefact est un classificateur: il possède des propriétés et des opérations
- Il peut s'associer avec d'autres artefacts

Notation

- Stéréotypes standards:
«source», «executable»

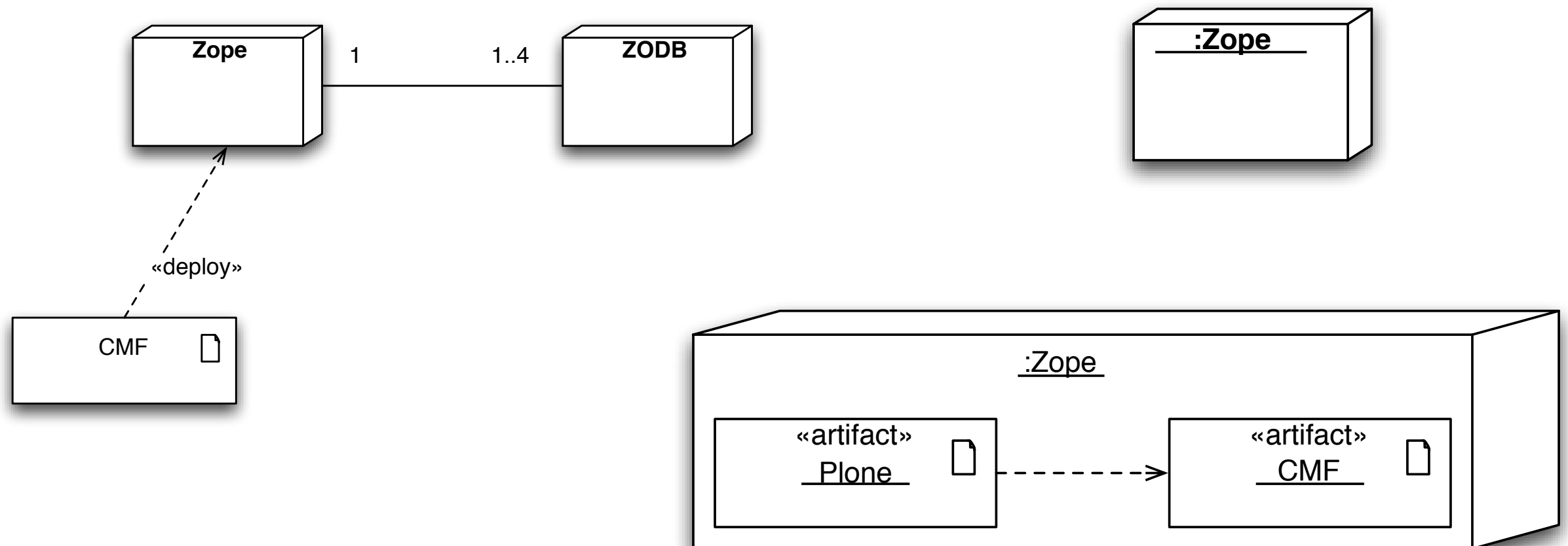


Noeud

- Ressource logique sur laquelle les artefacts sont déployés.

Notation

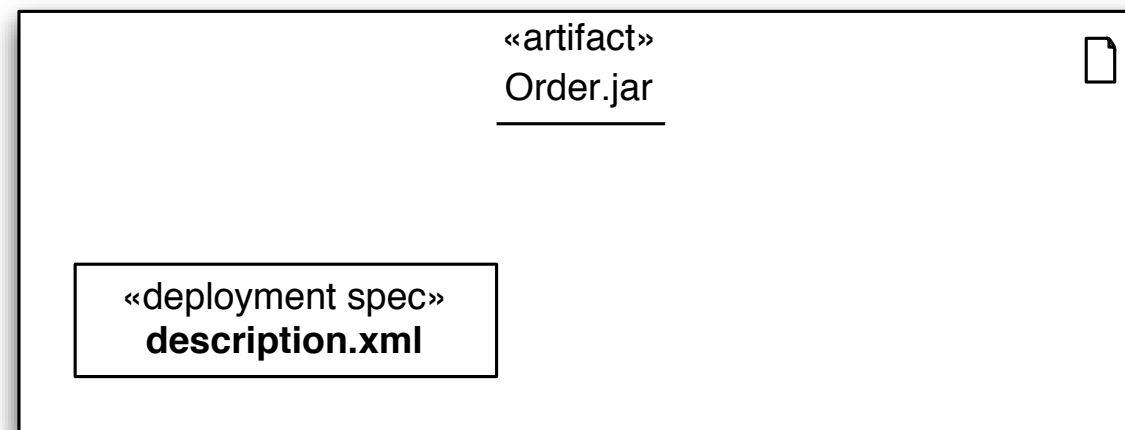
- Connexion, instances



Chemin de communication

- Association entre deux noeuds, permettant l'échange de signaux et de messages.

Spécification de déploiement



- Ressource (artefact) déterminant les paramètres d'exécution d'un artefact.

Diagrammes structurels

- Classes
- Objets
- Structures composite

Structures composite

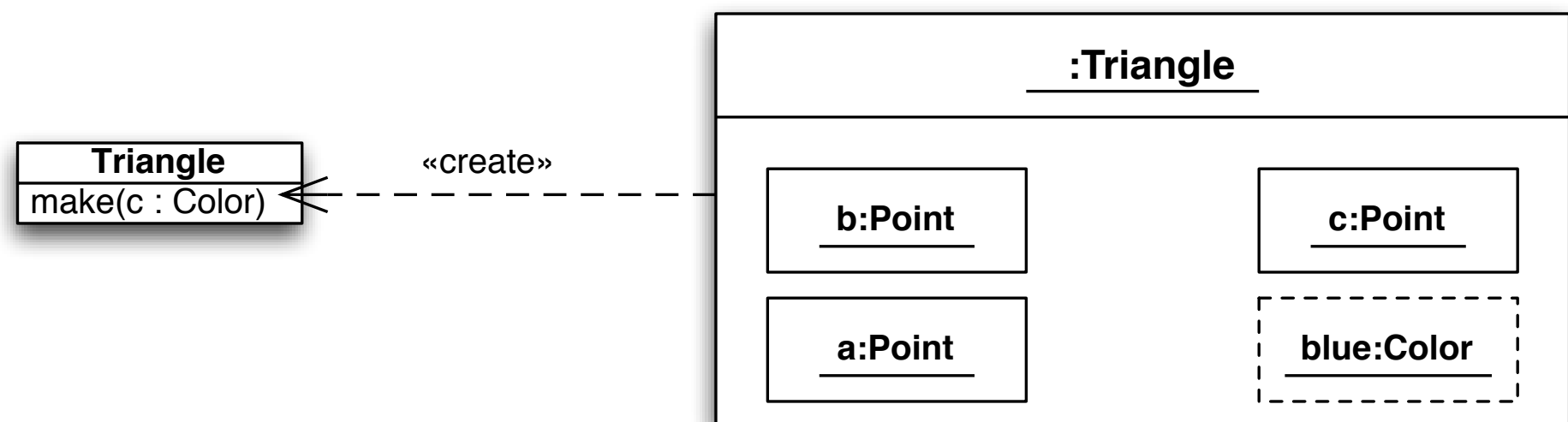
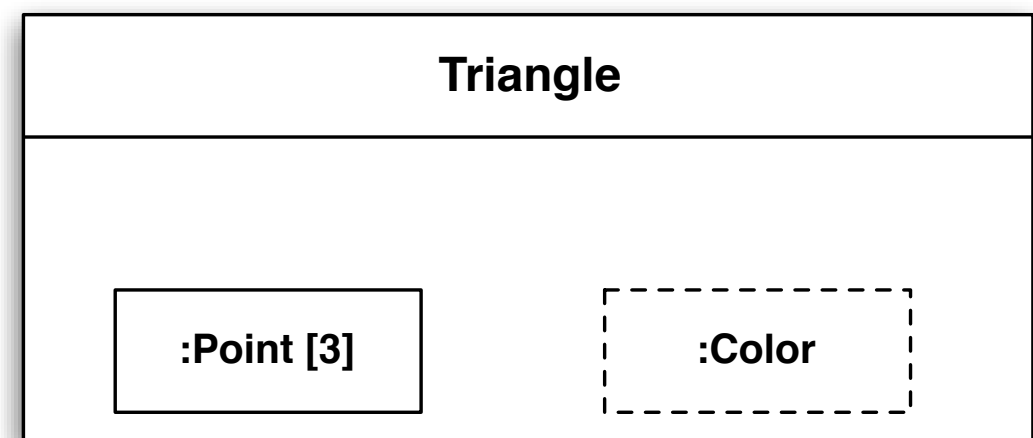
- Structures internes
- Collaborations

Structures internes

- Spécification d'éléments créés à l'intérieur d'un classificateur

Notation

- Propriétés
 - Instances
 - Références

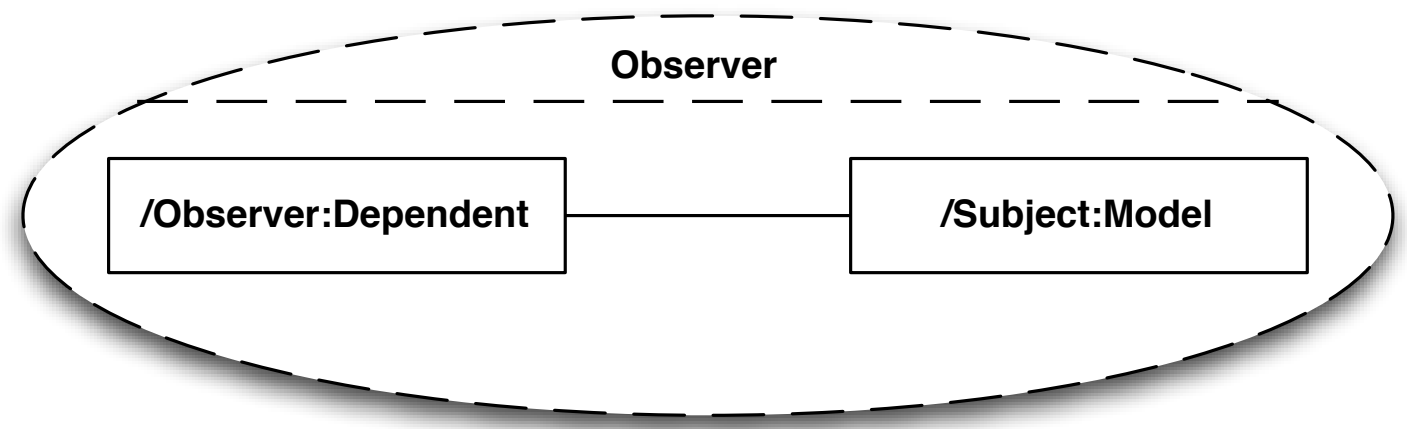


Collaborations

- Spécification de la structure d'un ensemble d'éléments (rôles) qui réalisent une tâche commune
- Représentation de la structure d'un patron de conception

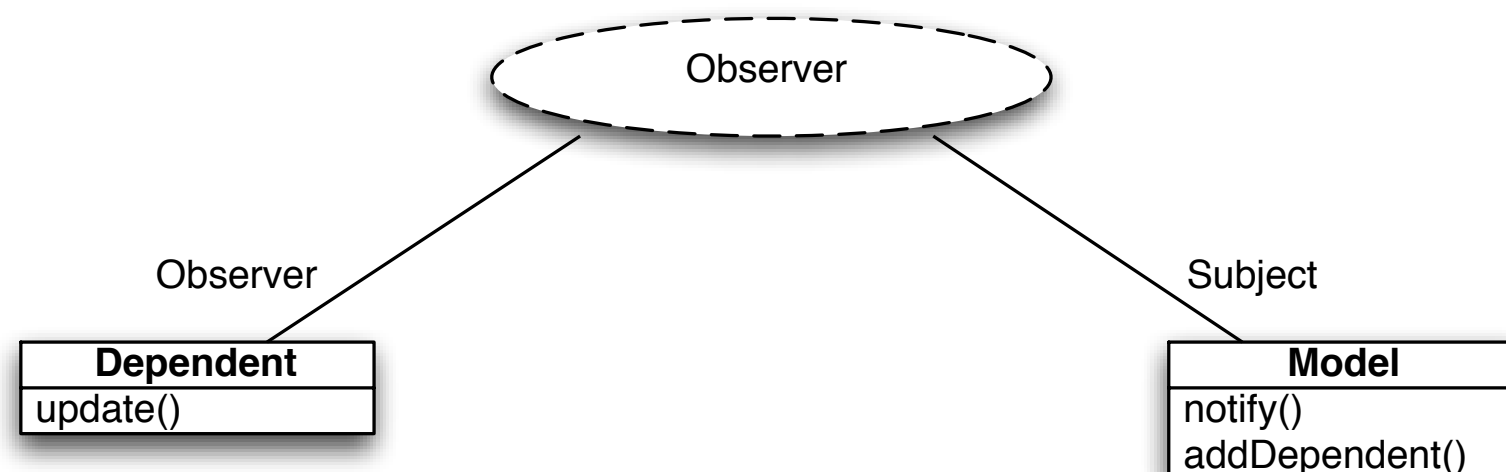
Notation (1 / 3)

- Rôle: rôle d'une classe participant à une collaboration
- Connecteur: lien permettant la communication entre deux classes



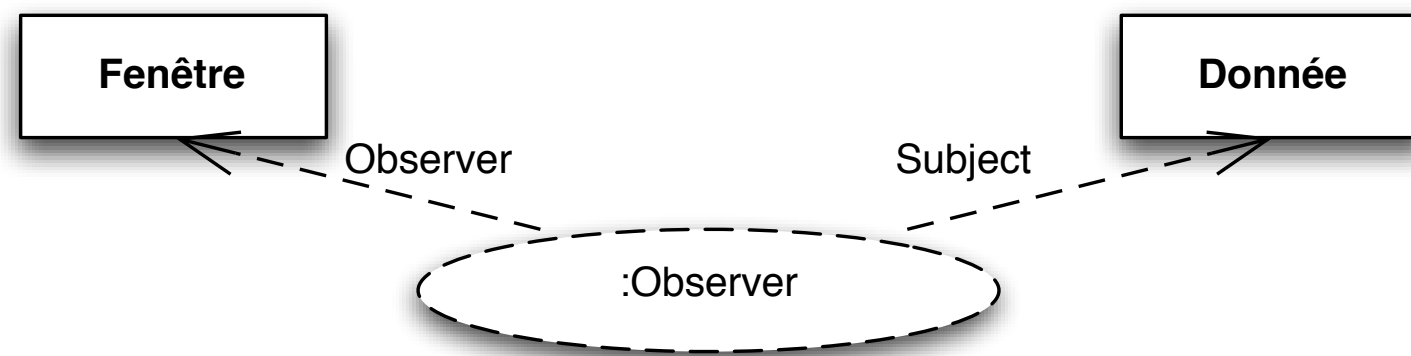
Notation (2/3)

- Les propriétés définies par les classificateurs doivent être possédées par les classes jouant leurs rôles



Notation (3/3)

- Occurrence d'une collaboration



Diagrammes comportementaux

Diagrammes de comportement

- Etat-transition
- Interaction
- Activités

Diagrammes état-transition

- Modélisation d'un comportement par un système fini d'états et de transitions
- Il existe deux sortes de diagrammes d'état:
 - De comportement
 - De protocole

Diagramme de d'état de comportement

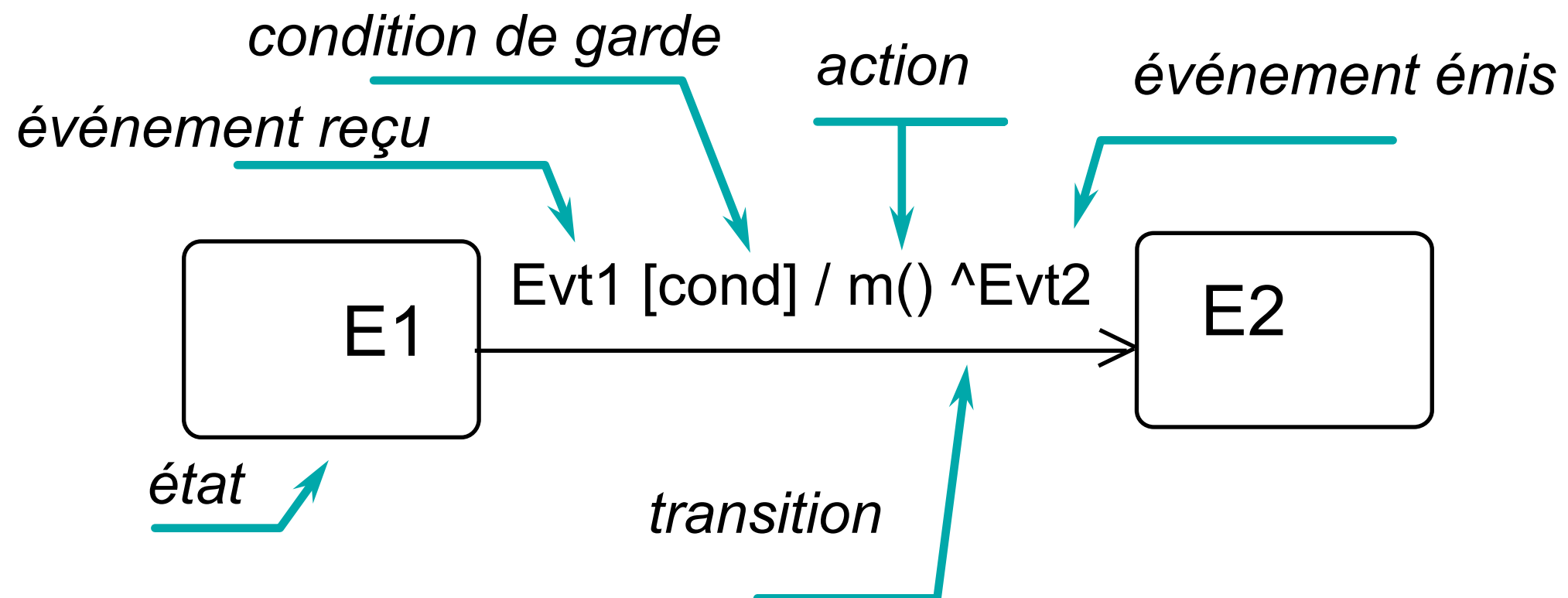
Machines à états

- Décrivent un comportement à l'aide d'un graphe d'États, interconnectés par des Transitions, qui sont déclenchées par des Événements
- En d'autres termes, décrivent les réactions d'un objet aux changements de son environnement.
- Essentiellement, un diagramme de Harel
- Attachés à un classificateur (sauf interfaces) ou à une opération
 - Sont hérités, et peuvent être spécialisés

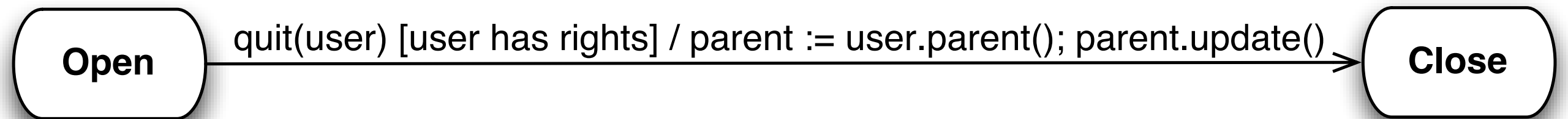
Transition

- Relation unidirectionnelle entre deux états (ou entre deux machine d'état)
- Interne à un état
- Externe: implique un changement d'état
- Locale: s'applique à tous les états d'un état composite

Transition — notation



Transition — exemple



Syntaxe : trigger [guard]/ trigger

Déclencheur (Trigger)

- Événement qui peut causer l'exécution d'un comportement qui lui est associé. Exemples:
 - Message: Événement ou Opération
 - Temporel
 - Changement

Événement/Signal

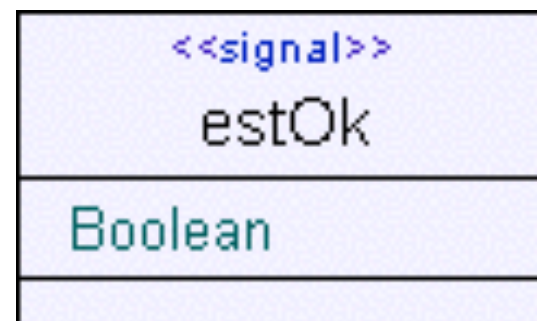
- Un événement est un type de message pouvant être échangé entre deux objets.
- Asynchrone (contrairement aux opérations, qui sont synchrones)
- Peut transporter des données.
- Correspond à un message asynchrone dans les diagrammes de séquence.

Événement

- Stimuli auxquels réagissent les objets
 - Occurrence déclenchant une transition d'état
- Abstraction d'une information instantanée échangée entre des objets et des acteurs
 - Un événement est instantané
 - Un événement correspond à une communication unidirectionnelle
 - Un objet peut réagir à certains événements lorsqu'il est dans certains états.
 - Un événement appartient à une classe d'événements (classe stéréotypée «signal»).

Événement

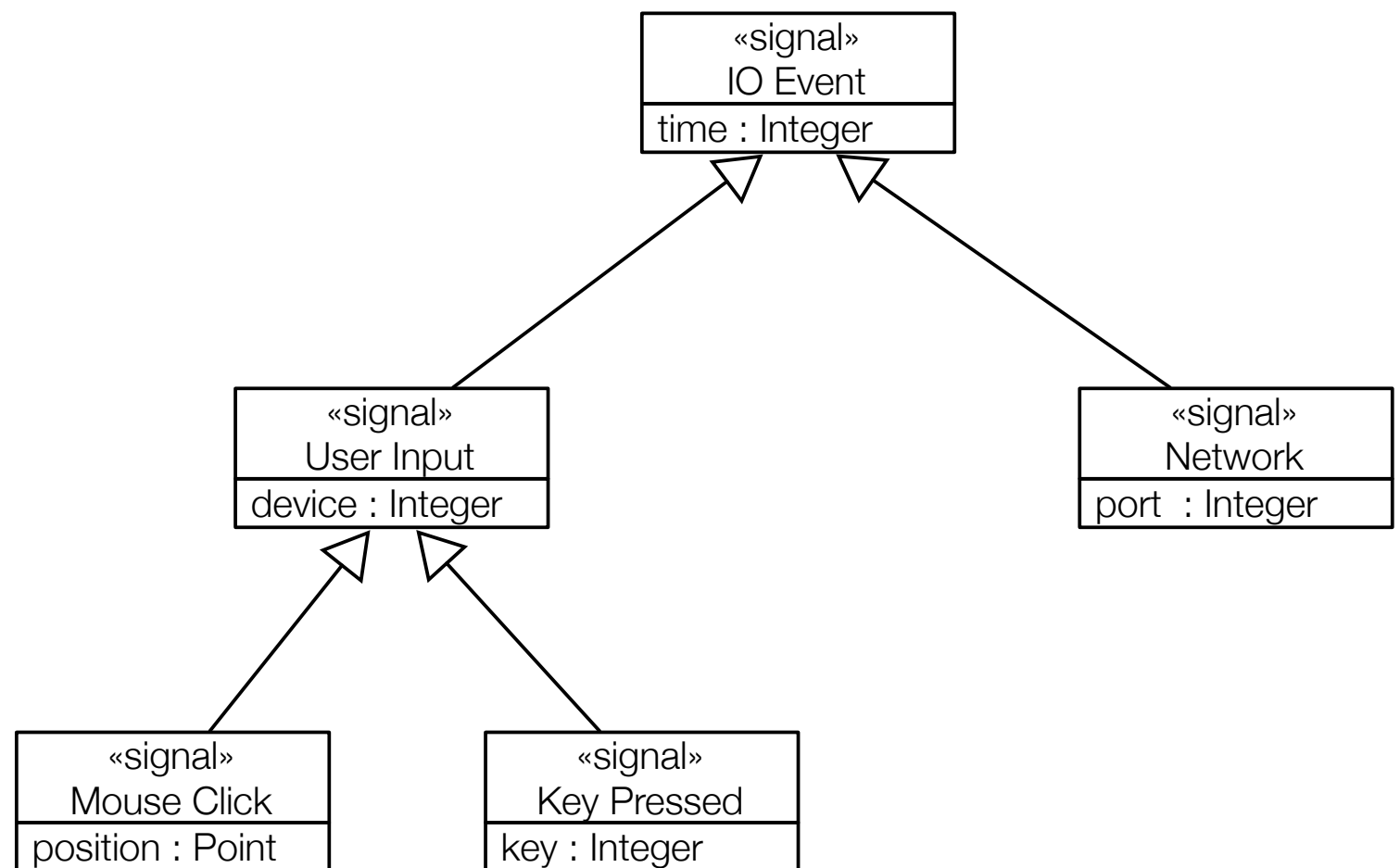
- Déclaration au sein d'un diagramme de classe au moyen d'une Classe stéréotypée « signal »



- Utilisation au sein
 - d'un diagramme de séquence pour représenter les interactions dynamiques entre objets,
 - des diagrammes d'états pour représenter les déclencheurs de transitions.

Notion d'événements

- Les événements sont considérés comme des objets



Types d'événements

- Réalisation d'une condition arbitraire
 - transcrit par une condition de garde sur la transition
- Réception d'un signal issu d'un autre objet
 - transcrit en un événement déclenchant sur la transition
- Réception d'un appel d'opération par un objet
 - transcrit comme un événement déclenchant sur la transition
- Période de temps écoulée
 - transcrit comme une expression du temps sur la transition

Garde

- Contrainte évaluée avant le déclenchement d'une transition

Activité

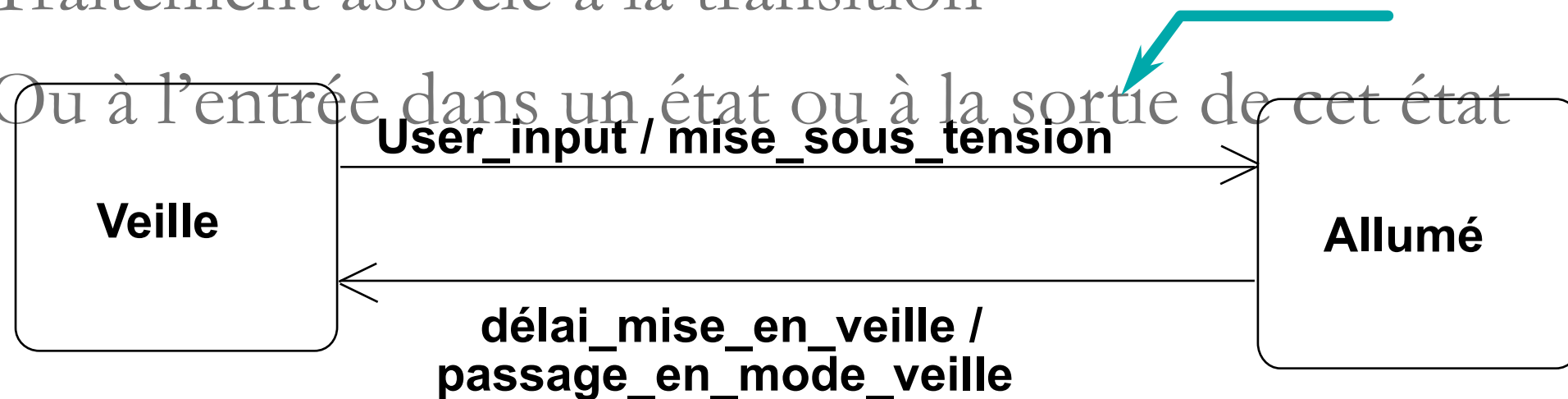
- Déclenchée par une transition ou interne à un état
- Définit un comportement, spécifié dans n'importe quel langage (!)

Notion d'action

- Action : opération instantanée (conceptuellement) et atomique (ne peut être interrompue)
- Déclenchée par un événement

- Traitement associé à la transition

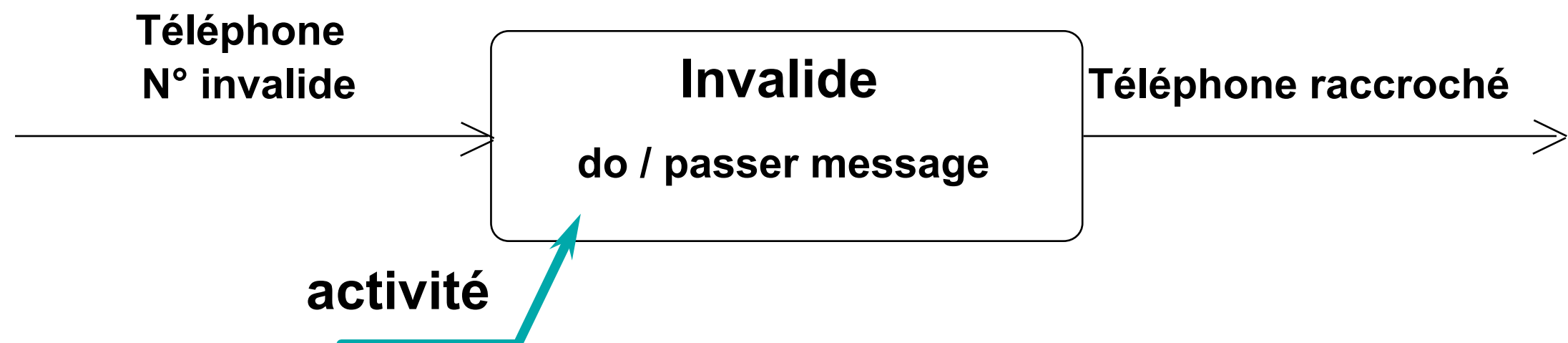
- Ou à l'entrée dans un état ou à la sortie de cet état



action

Notion d'activité interne à un état

- Activité : opération se déroulant continuellement tant que l'on est dans l'état associé
 - do/ action
- Une activité peut être interrompue par un événement.

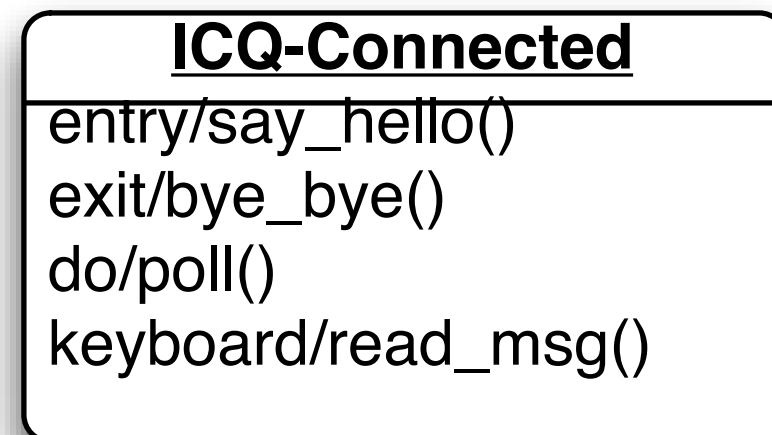


Etat

- Situation stable d'un objet:
 - en attente d'un événement, ou
 - en train de réaliser une tâche
- Conditionne la réponse de l'objet à des événements
 - programmation réactive / « temps réel »
- Peut avoir des variables internes
 - Attributs de la classe attachée à ce diagramme d'états
- Trois sortes: simple, composite et sous-machine

Notation

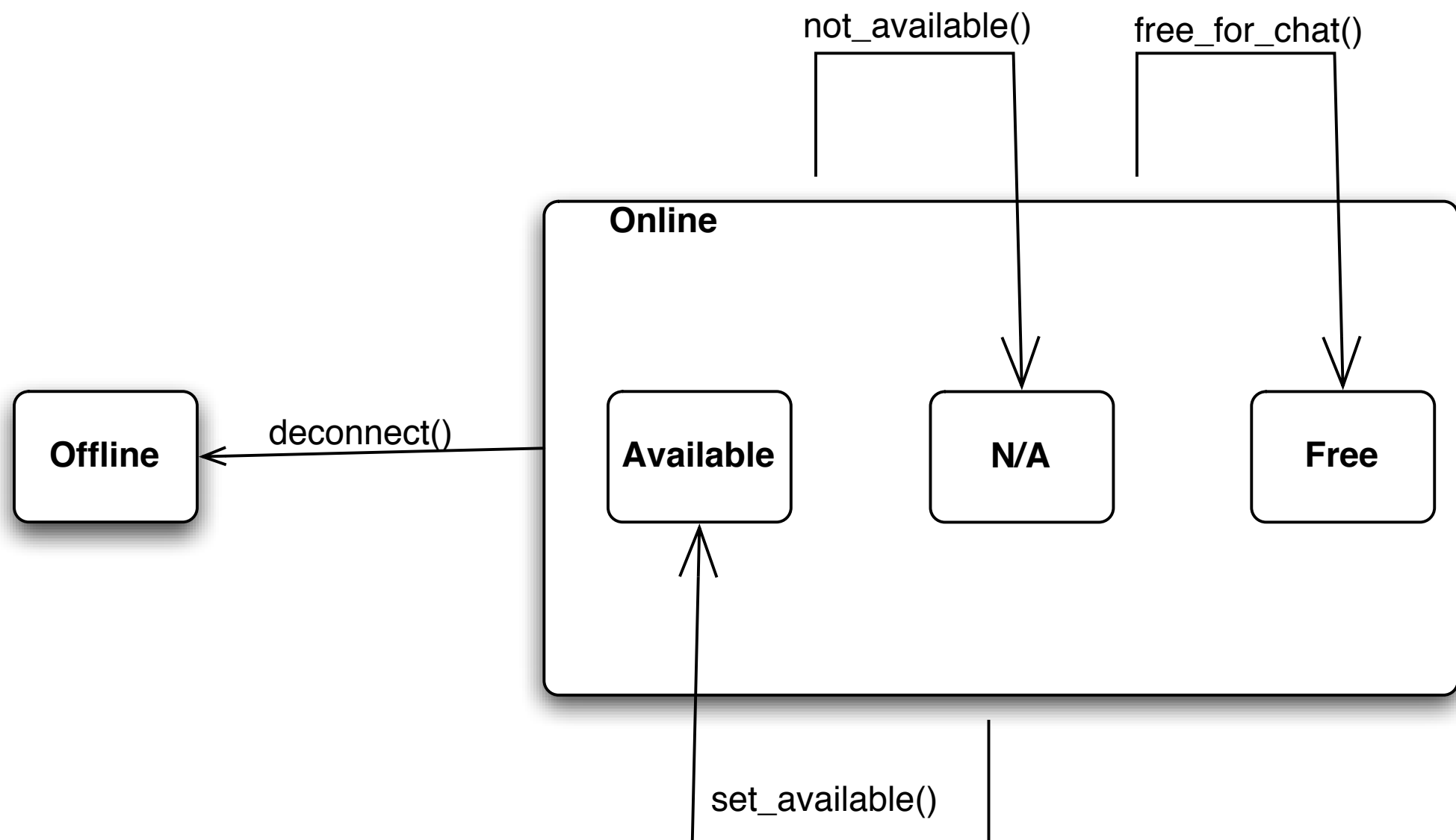
- Compartiments:
 - Nom
 - Activités internes
 - Transitions internes



Etat composite

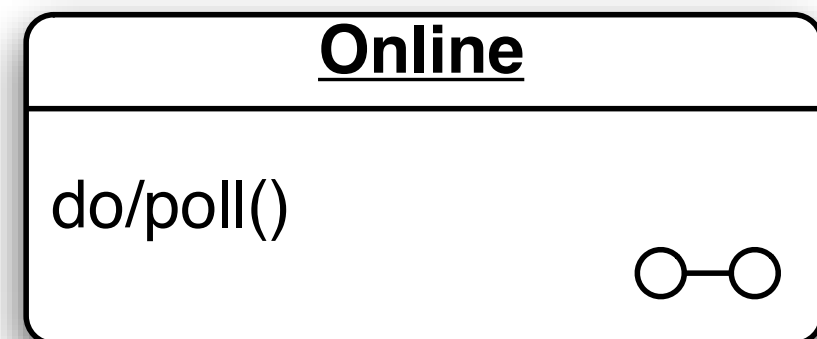
- Etat englobant différents sous-états
- Possède une entrée et une sortie
- Transition arrivant \rightarrow entrée
- Transition sortant \rightarrow sort de tous les états

Notation



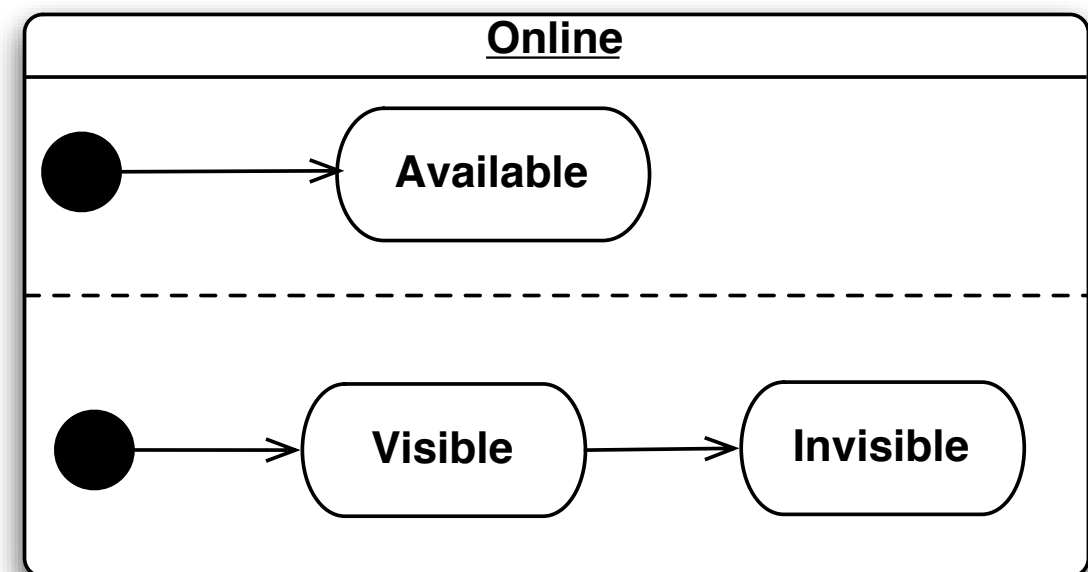
Notation

- Etat composite avec indicateur de décomposition

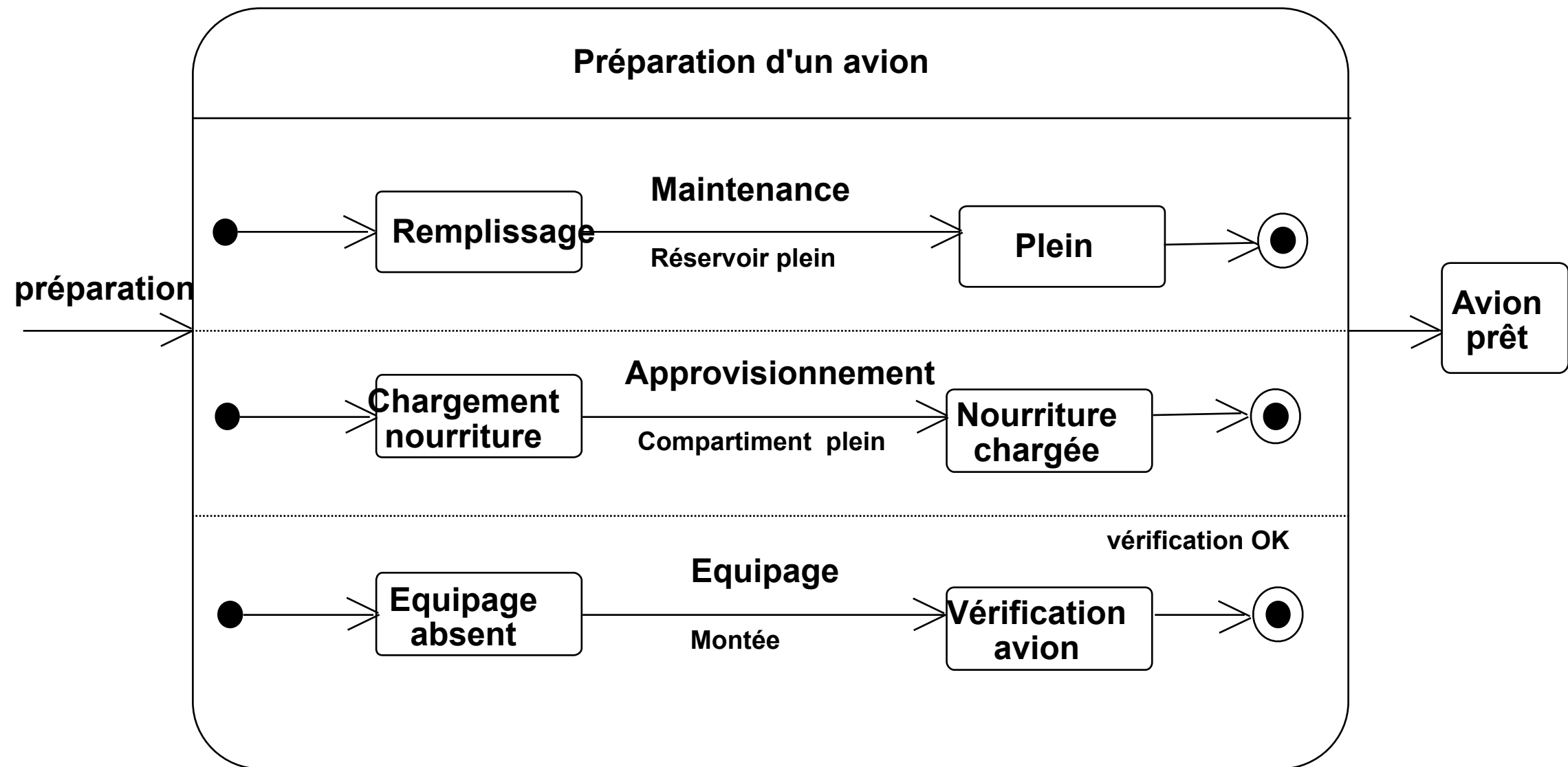


Composite Orthogonal

- Régions concurrentes, représentant 2 (ou n) aspects indépendants d'un objet.
- Activités parallèles
- Evitent l'expression d'un produit cartésien



Diagrammes d'états concurrents : exemple

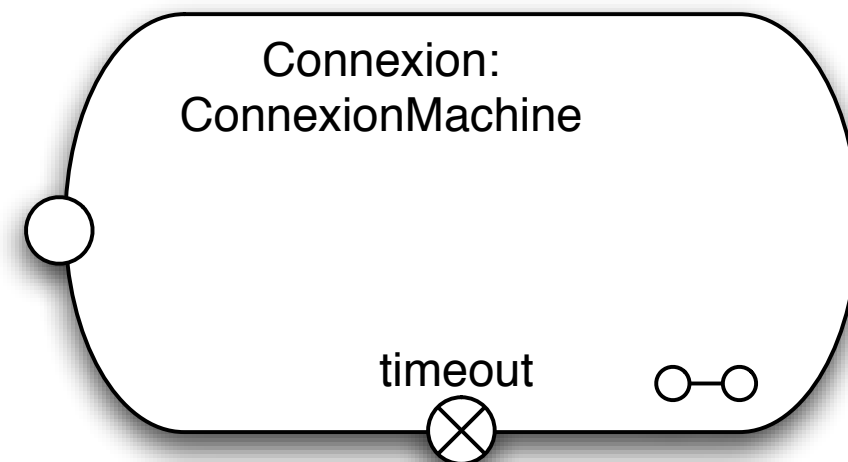


Sous-machine

- Référence à une machine d'états

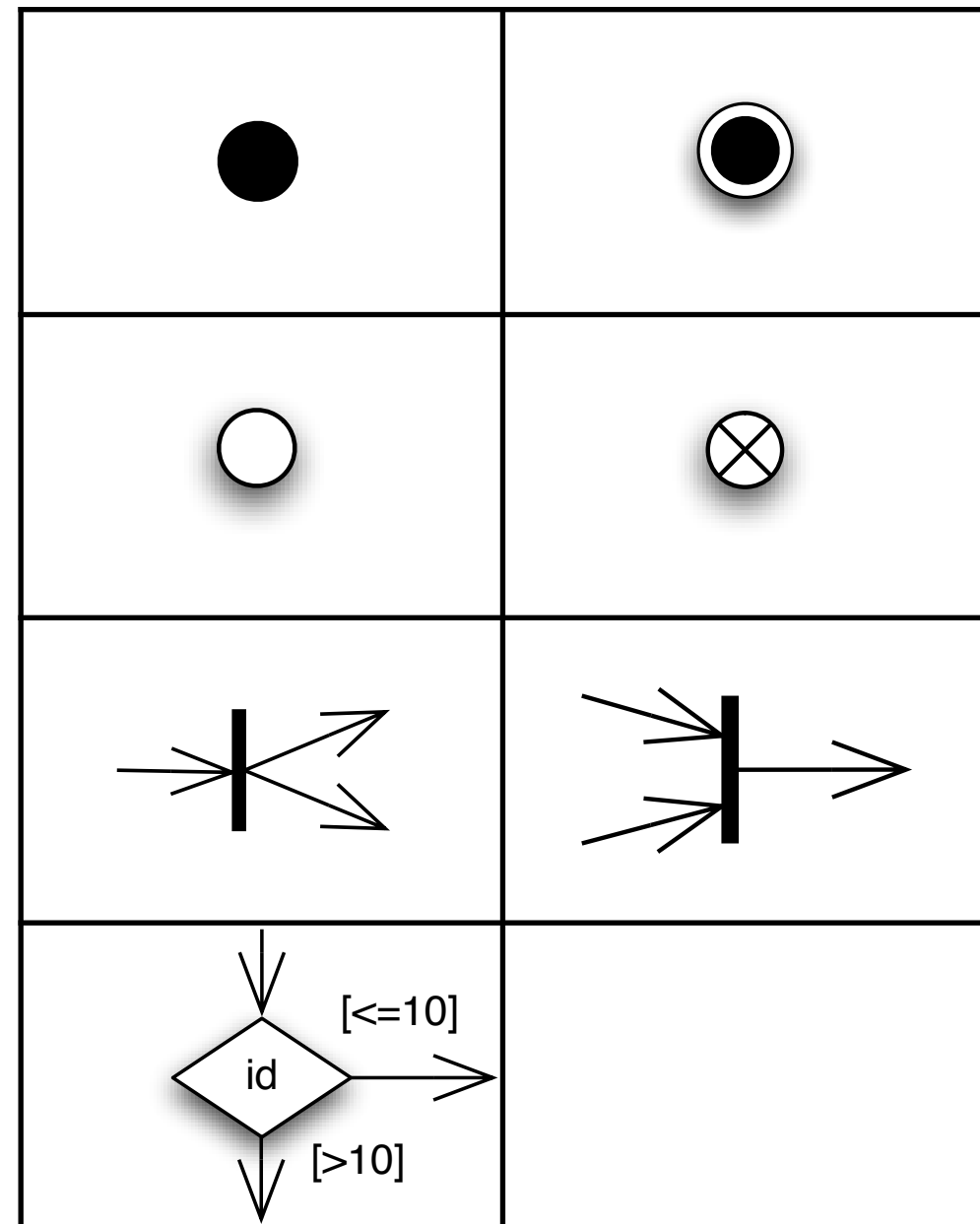
Notation

- Référence à une machine d'états
- Points de connexion:
 - Entrée
 - Sortie



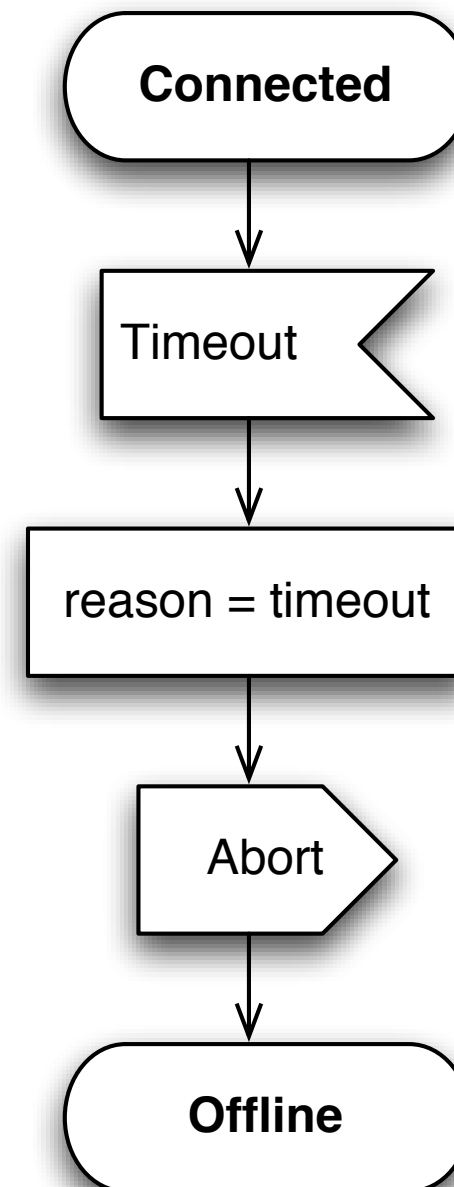
Pseudo-états

- Initial, Final
- Entrée, sortie
- Fourchette, jonction (régions orthogonales)
- Choix



Transitions particulières

- Envoi et réception d'événements
- Séquence d'actions
- Pour une vue “orientée transition”



Spécialisation

- Ajout de nouvelles transitions ou de nouveaux états
- Spécialisation d'états (transformation d'un état simple en composite)
- Un état peut être {final}

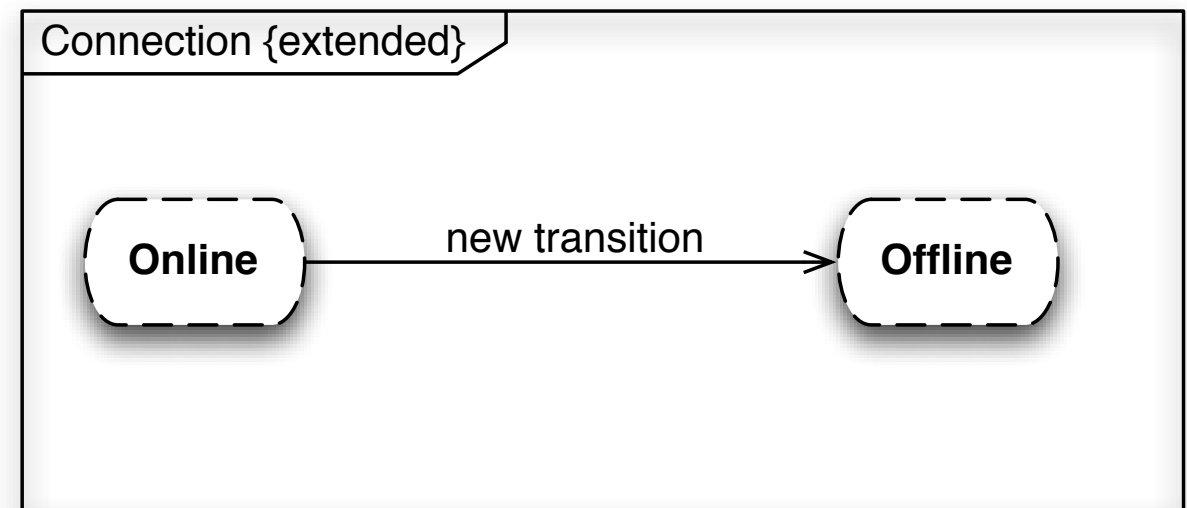


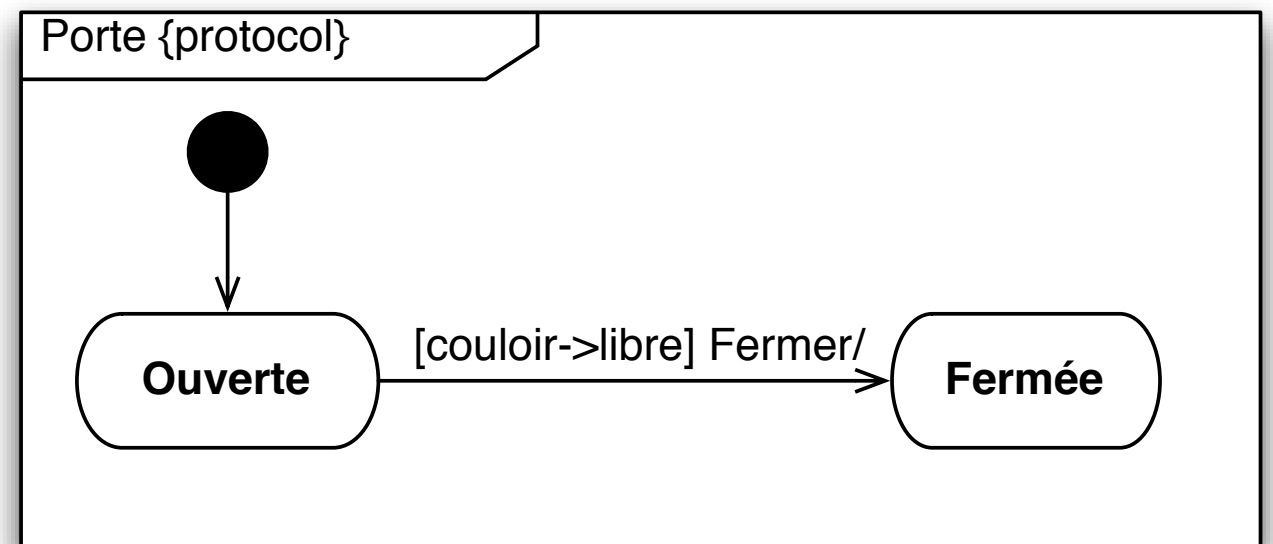
Diagramme d'état de protocole

Machines d'état de protocole

- Spécialisation des machines d'états
- Toujours attachées à un classificateur
- Représentent un cycle de vie d'un objet et spécifient quels messages sont acceptés à chaque état

Notation

- Similaire aux machines d'états
- Notée {protocol}



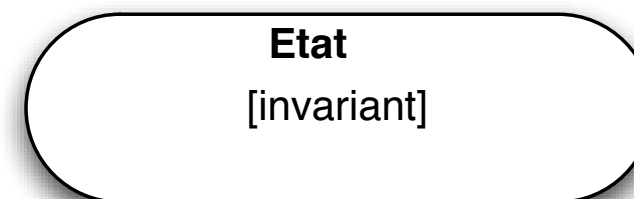
Transitions

- Spécifient une pré et une post condition

 [pré-condition] événement / [post-condition] ➤

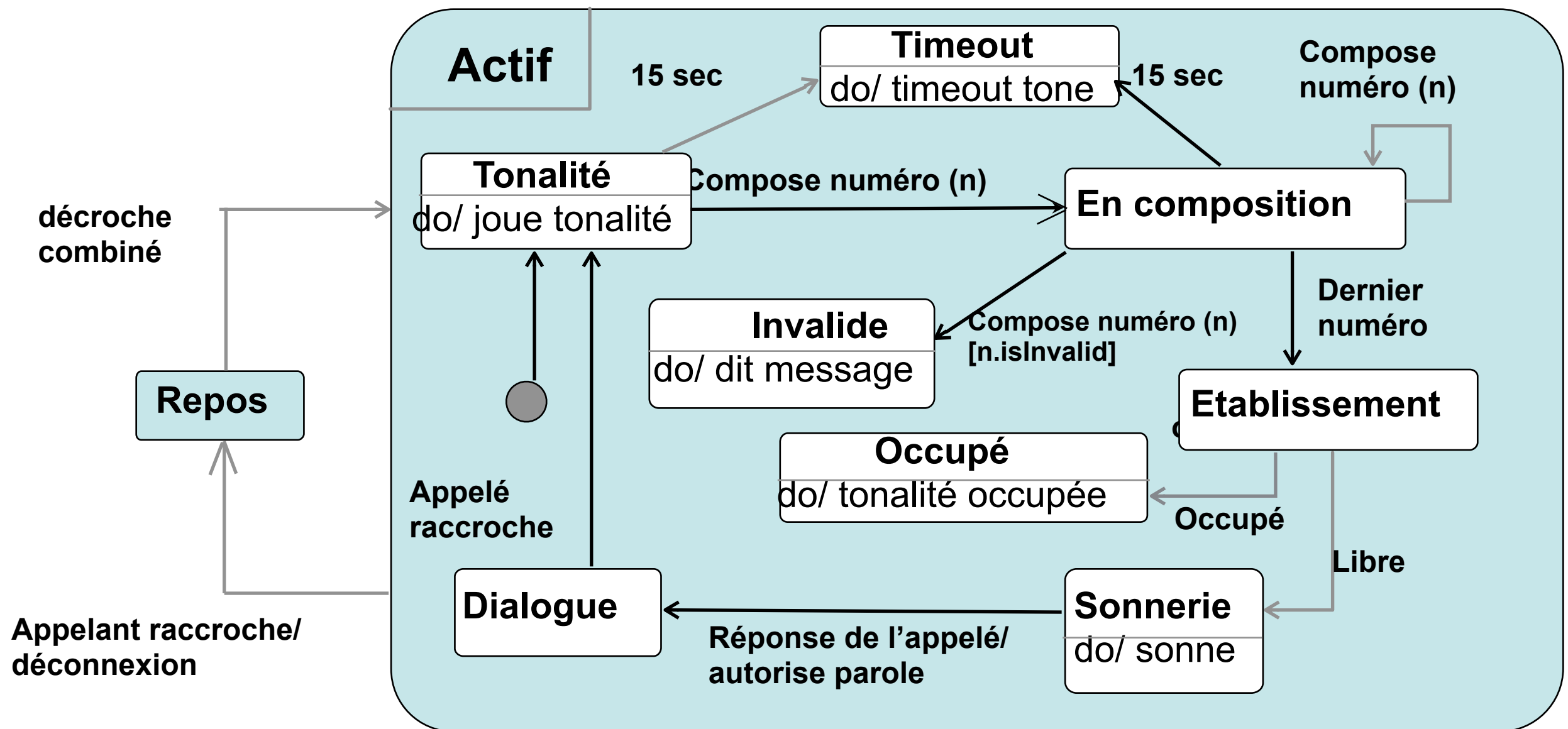
Etats

- Spécifient un invariant

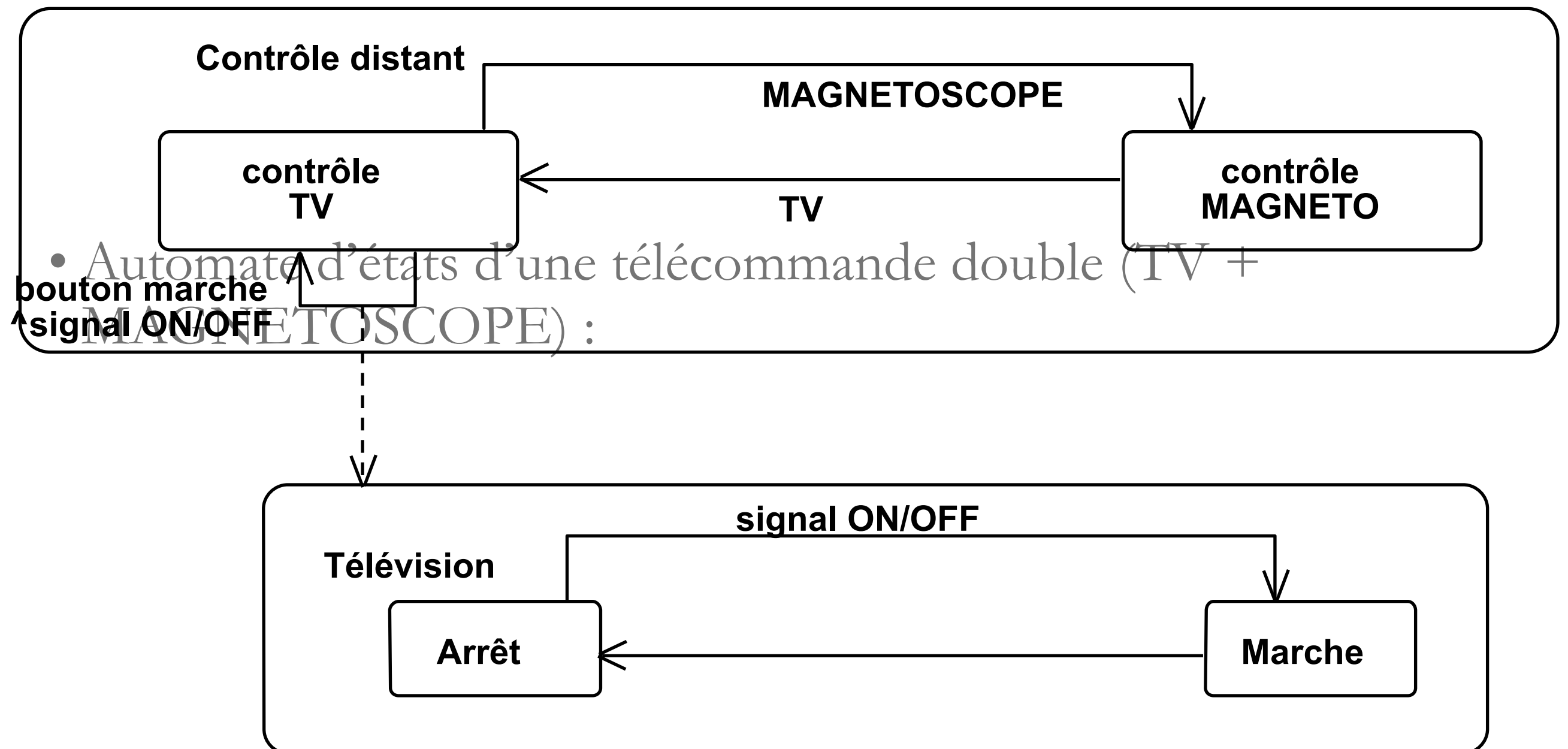


Résumé

Exemple de diagramme d'états



Émission d'événements



Etat-transition (résumé)

- Format :
 - événement (arguments) [conditions] / action ^événements provoqués
- Déclenchement :
 - par un événement (peut être nul).
 - Peut avoir des arguments.
 - Conditionné par des expressions booléennes sur l'objet courant, l'événement, ou d'autre objets.
- Tir de la transition :
 - Exécute certaines actions instantanément.
 - Provoque d'autres événements ; globaux ou vers des objets cibles.

Diagrammes de comportement

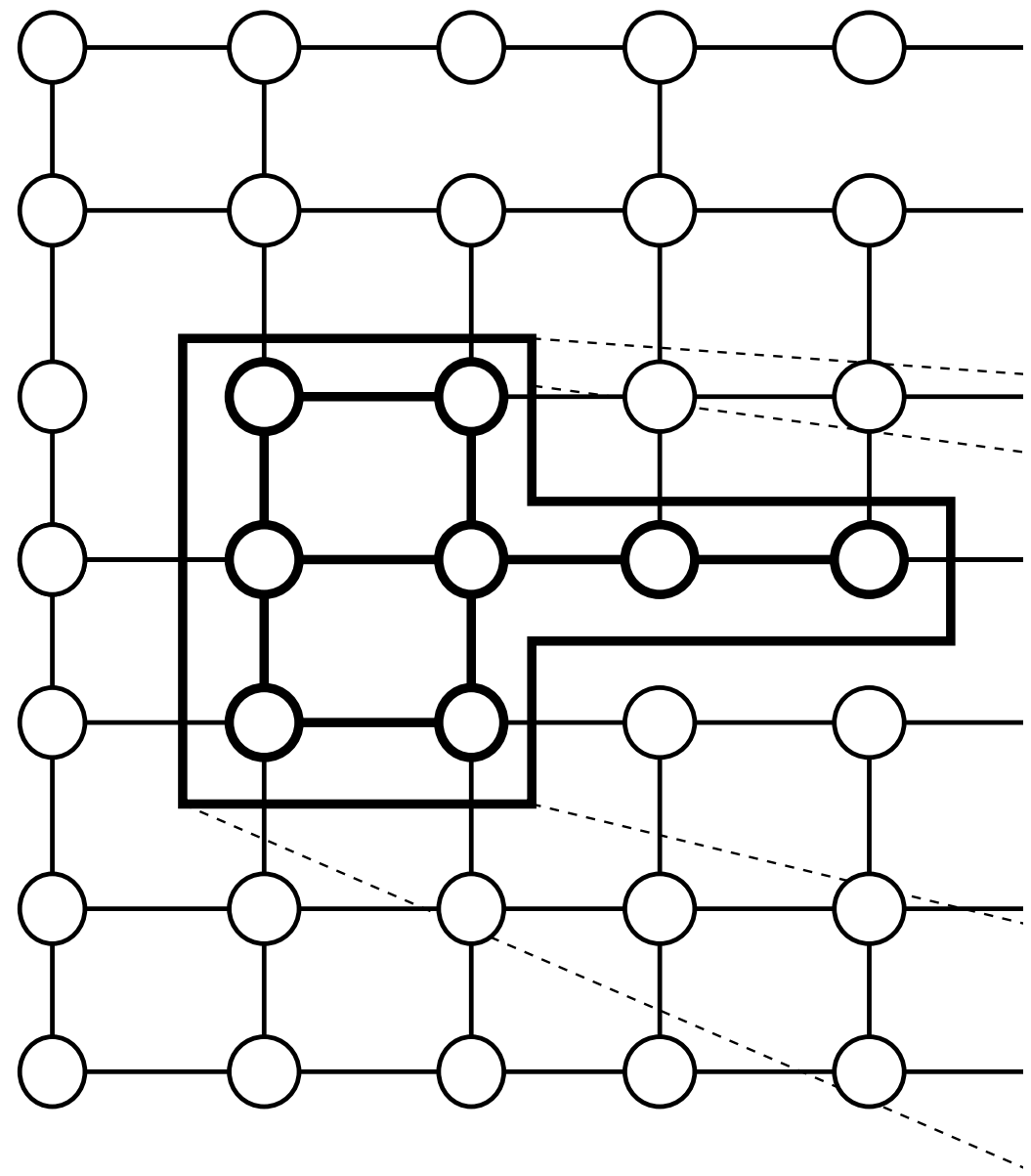
- Interaction
- Activités

Interactions

- Définissent le comportement d'un Classificateur
- Le comportement est défini par deux ensembles de traces:
 - Valides
 - Invalides

Interactions

- Décrivent l'échange de messages entre objets.
Diagrammes:
 - Séquence
 - Communication
 - Aperçu d'interaction



Interaction

Diagramme de Séquences

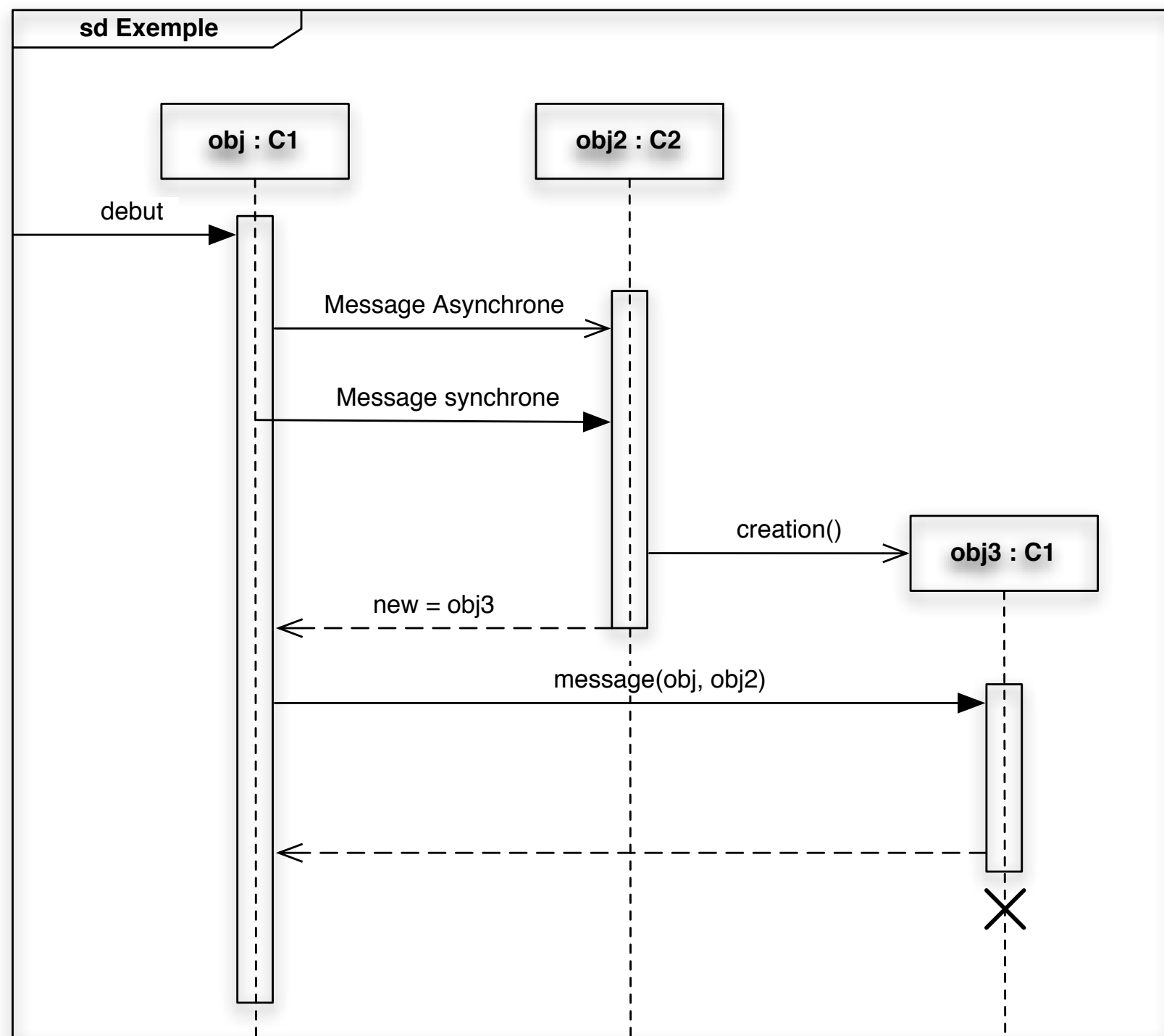
Diagramme de séquences (scénario)

- Exemples de coopération entre objets
- Illustrent la dynamique d'enchaînement d'un traitement à travers les messages échangés entre objets
- Représentation du temps comme une dimension explicite (verticale)

Diagramme de séquences

- Constitution:
 - Un message initiateur
 - Un ensemble d'objets et de messages
 - Des cadres (frames)
 - Des contraintes de temps

Notation

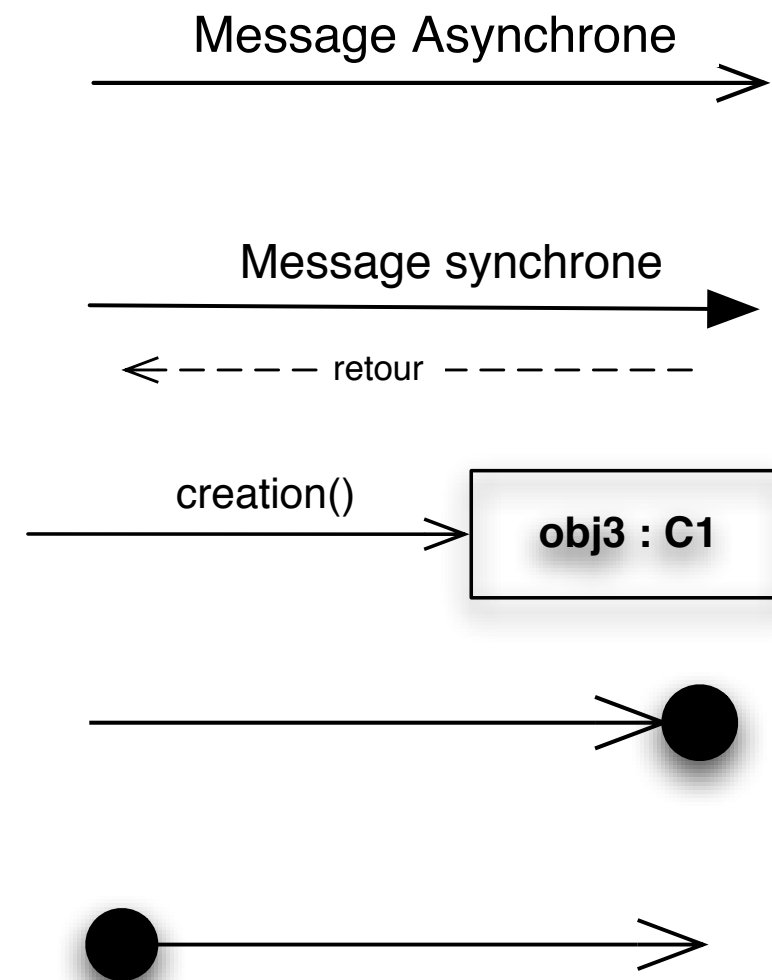


Messages

- Communication (synchrone ou asynchrone) entre deux lignes de vie. Sortes de message:
 - Complet (événements d'envoi et de réception connus)
 - Trouvé (événement d'envoi inconnu)
 - Perdu (événement de réception inconnu)
 - Inconnu

Messages - Notation

- Asynchrone
- Synchrone
- Création
- Perdu
- Trouvé



Messages - Syntaxe

message ::= [attribut =] signal-ou-operation [(arguments)][: valeur-retour] | '*'
arguments ::= argument [, arguments]

argument ::= [paramètre=]valeur-argument | attribut = paramètre-sortie [valeur-argument] |

Exemples:

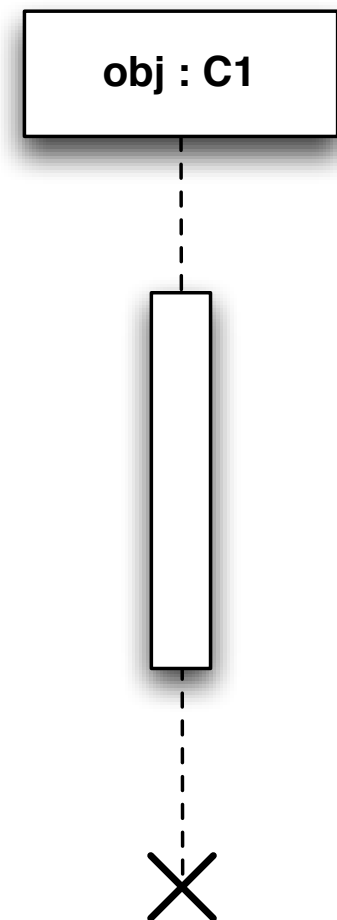
message(14, -, 3.14, "hello")

v=message(16, variable):96

message(id=16)

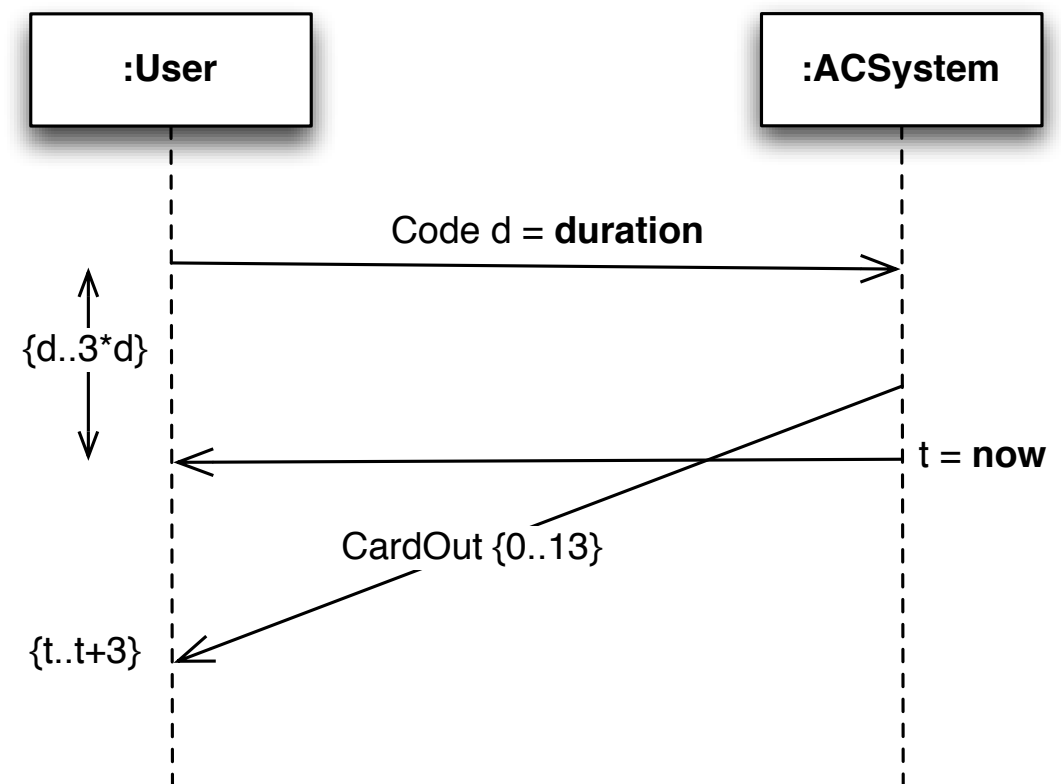
Ligne de vie, Activation

- Ligne de vie:
 - Représente l'existence d'un objet à un moment particulier
 - Entre la création et la destruction
- Activation:
 - Période d'exécution d'une activité (lui-même, délégation)



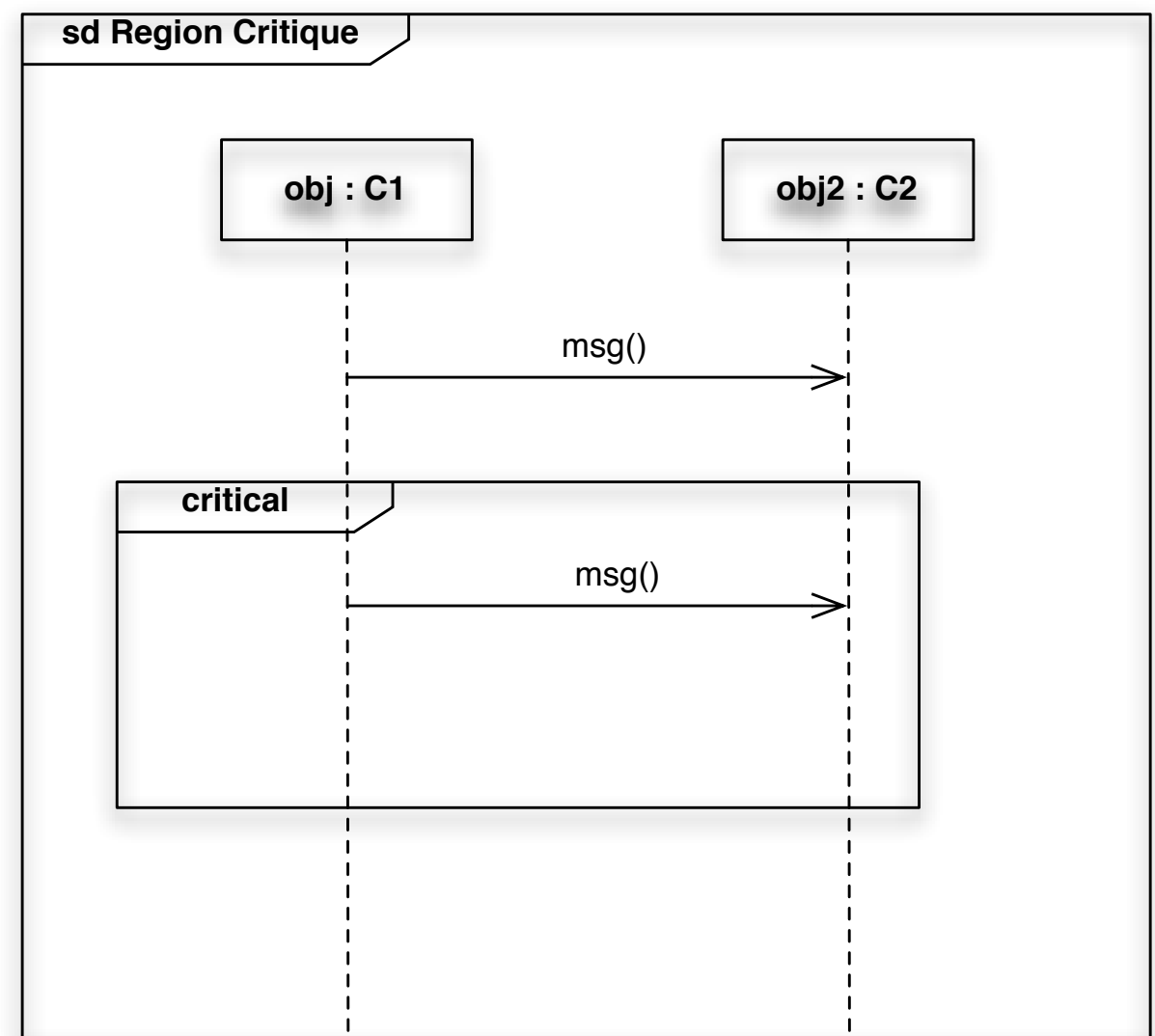
Contraintes temporelles

- Observation de durée, contraintes
- Observation de temps (instants), contraintes



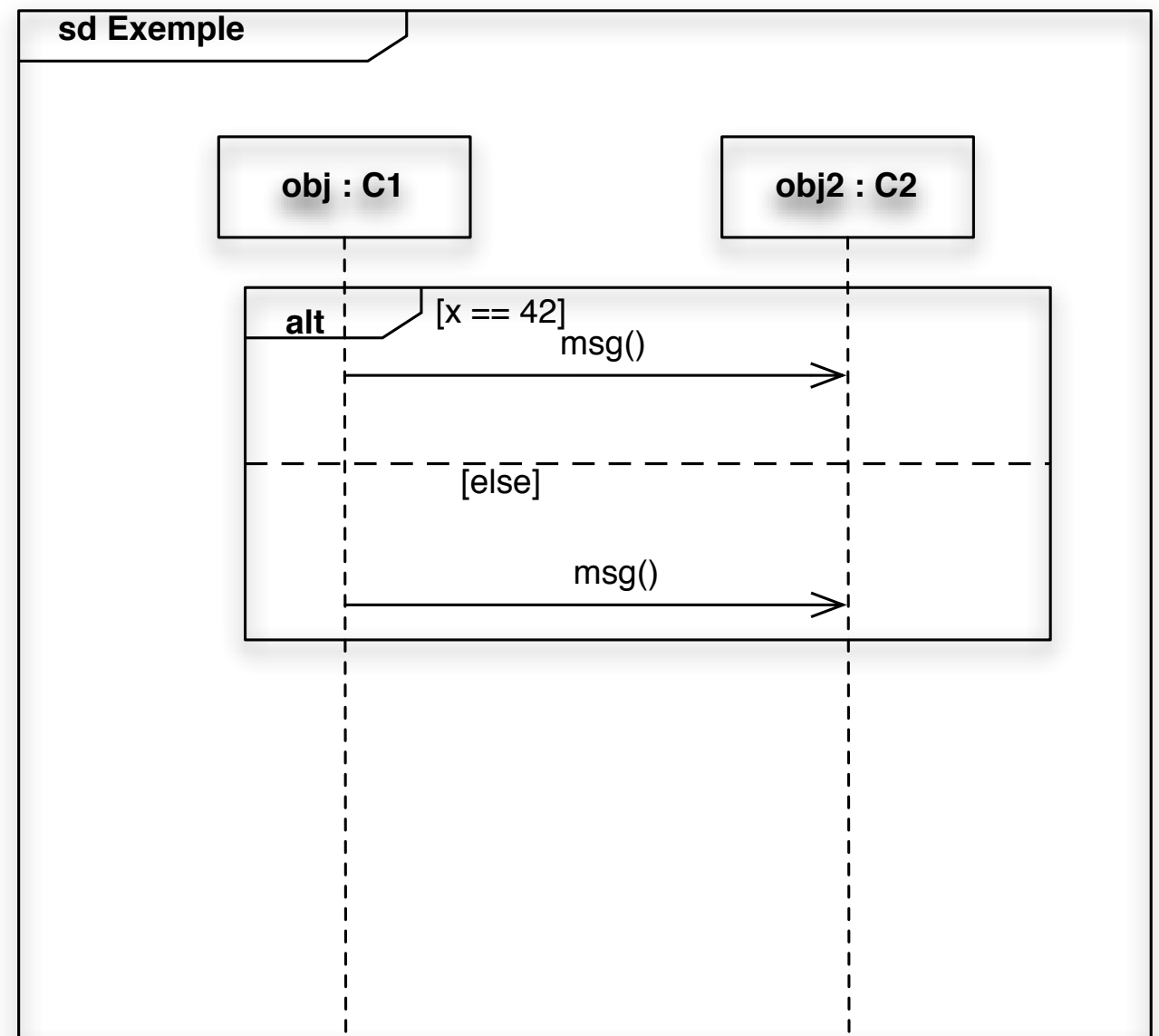
Fragments

- Région définissant une sémantique précise à l'intérieur d'une interaction
- Raccourcis pour l'écriture de traces
- Composé de $[1..*]$ "Opérandes"



Fragments

- Alternatives (alt): choix de comportement, entre deux opérandes



Fragments

- Option (opt): l'opérande est optionnelle. Elle peut ne pas exister
- Break (break): l'opérande est exécutée, à la place du reste de l'interaction (raccourci pour alternative)

Fragments

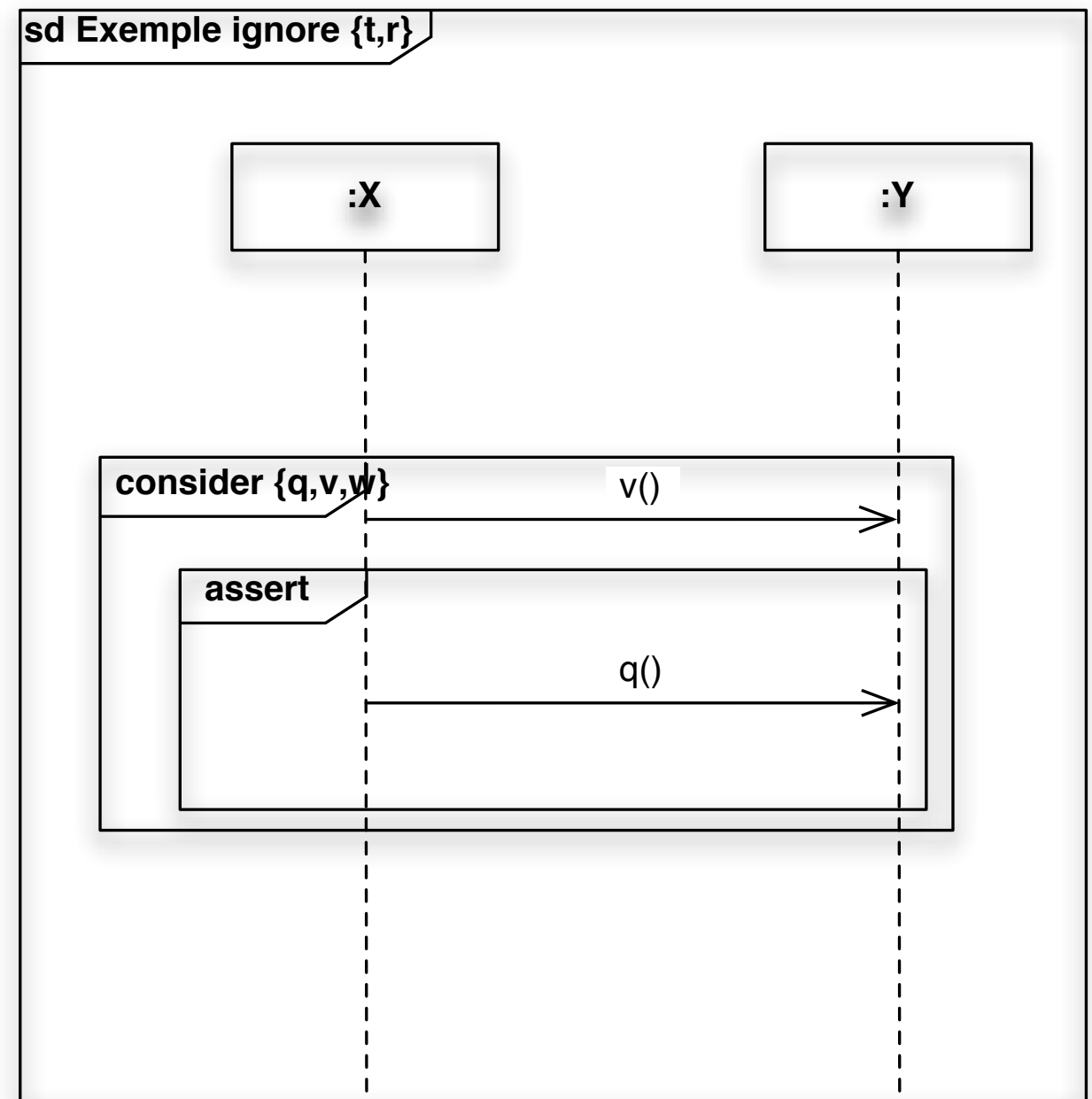
- Parallèle (par): le comportement spécifié par les deux opérandes peut s'exécuter en parallèle (fusion entre les messages)
- Négatif: le fragment représente des traces invalides

Fragments

- Région critique (critical): les traces spécifiées ne peuvent pas être intercalées

Fragments

- Assertion (assert):
spécification de la seule
continuation valable
- Ignorer et Considérer
(ignore | consider):
spécification des messages
significatifs



Fragments

- Boucle (loop): l'opérande sera répétée autant de fois que défini par une garde
- Autres: strict, ordonnancement strict, ordonnancement faible

Interaction

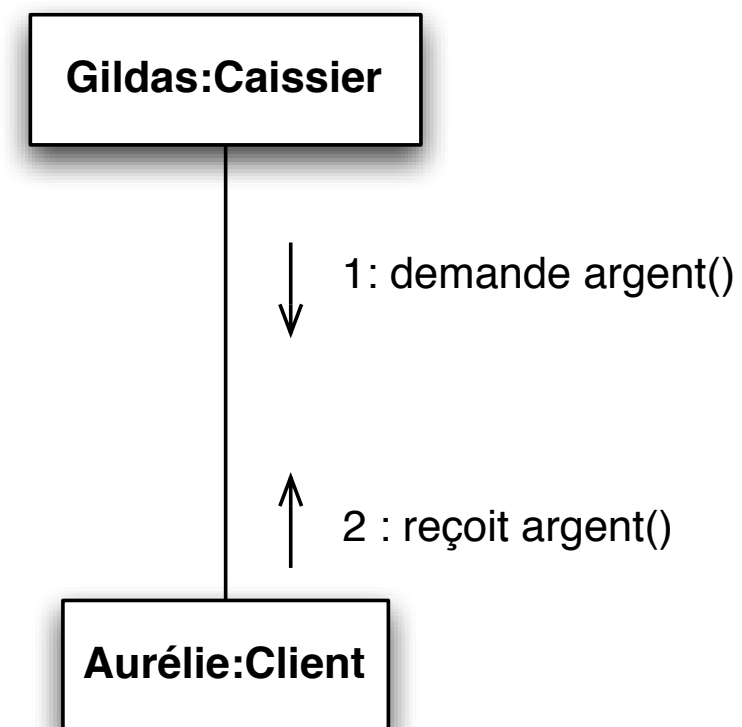
- Diagramme de Communication

Diagramme de Communication

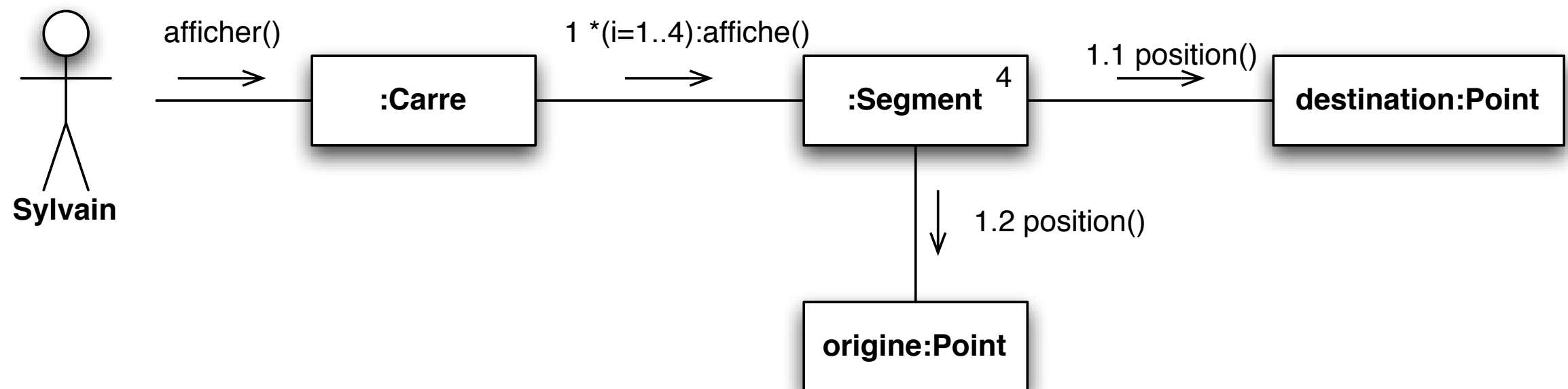
- Comme le diagramme de séquences, sert à représenter des exemples de coopération d'objets
- Cependant: la dimension temporelle est représentée par une séquence et les liens entre objets sont visibles

Notation

- Equivalent à un diagramme de séquences
- Le chemin utilisé par les messages est visible



Notation



Séquences consécutives et imbriquées

Diagrammes de comportement

- Interaction
- *Activités*

Diagramme Activités

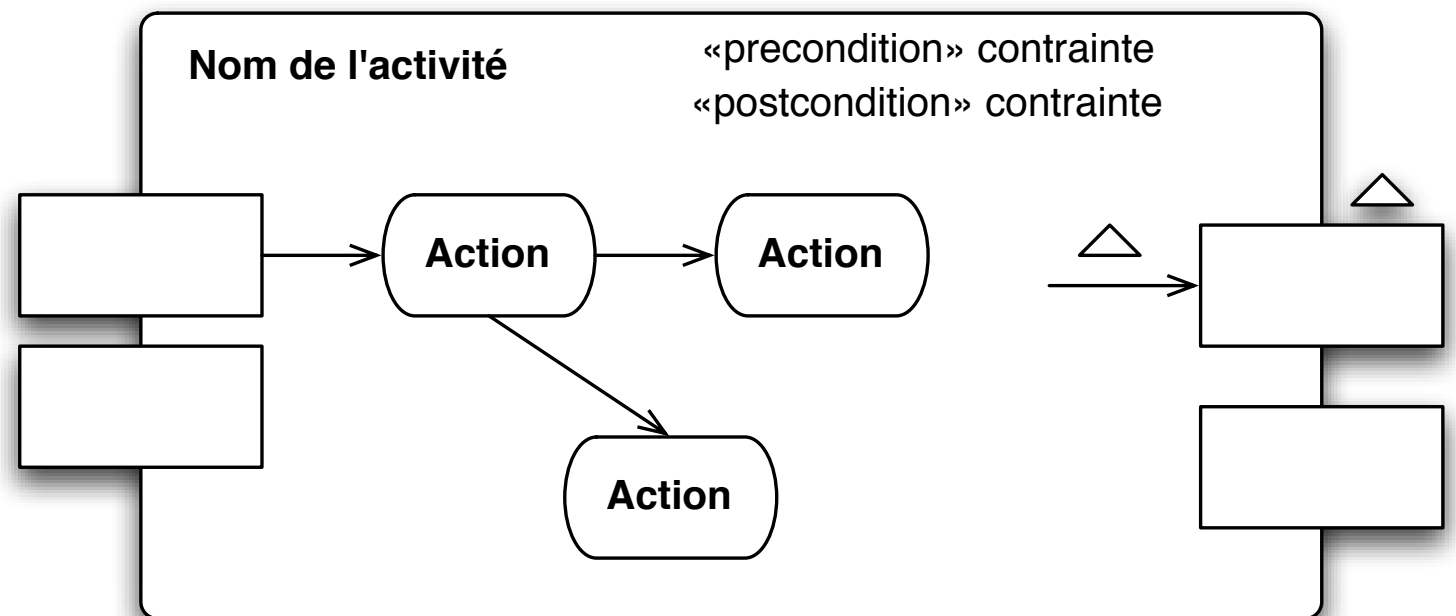
- Spécifie la séquence et les conditions pour coordonner différents comportements (plutôt que les classificateurs contenant ces comportements)
- Attachés à un Classificateur

Diagramme Activités

- Composés d'un ensemble d'**actions** et de **flots** (de contrôle et de données)
- Spécifient une opération (conception)
- Peuvent spécifier un workflow
- Sémantique basée sur les réseaux de Petri

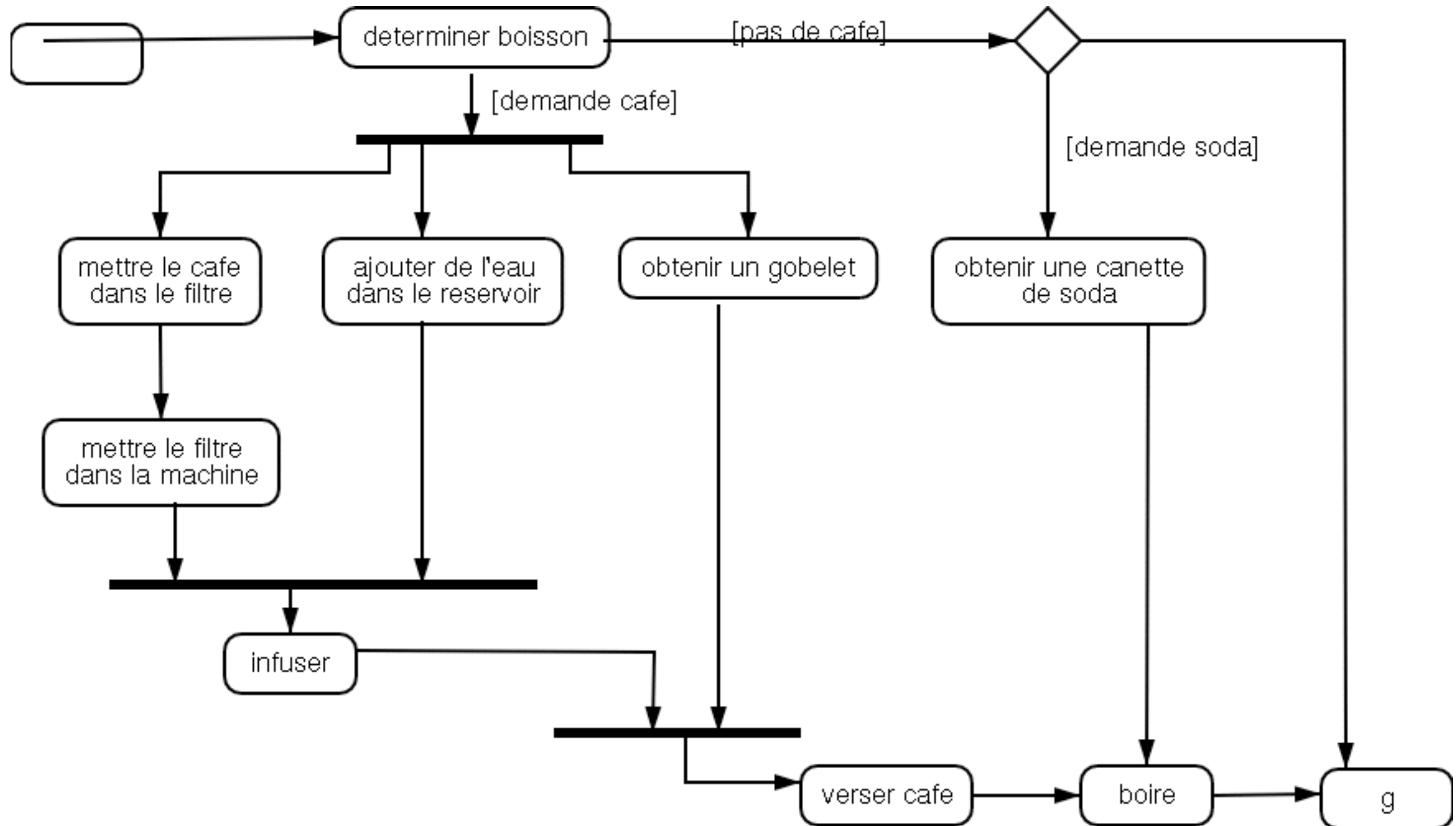
Notation

- Paramètres d'entrée et de sortie
- Pré et post-conditions
- Exceptions



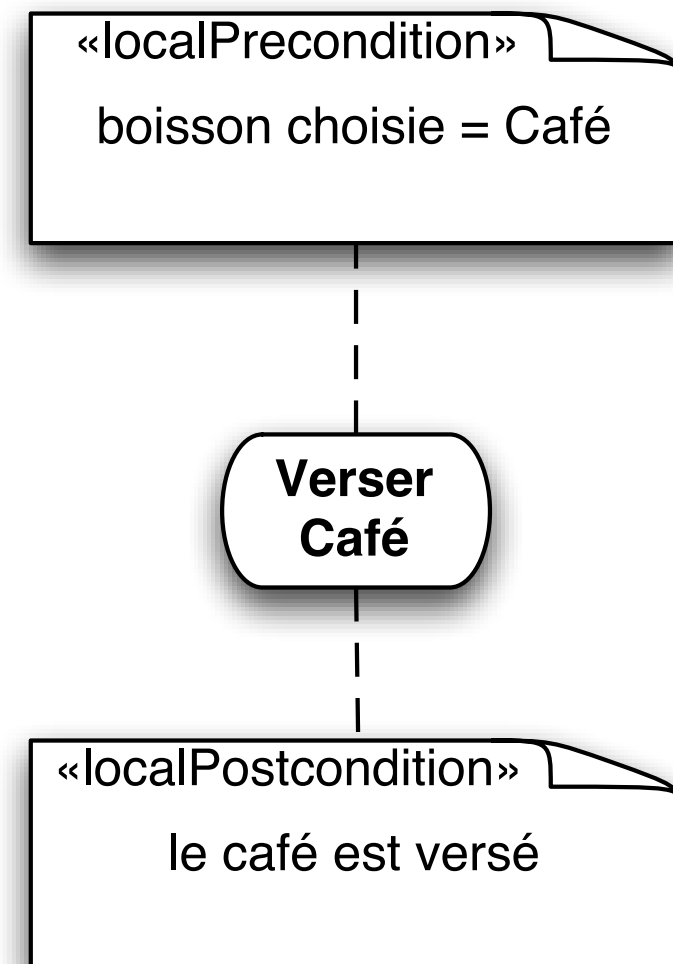
{activity} Activité
Attribut : type
Attribut : type
Operation(param)
Operation(param)

Notation



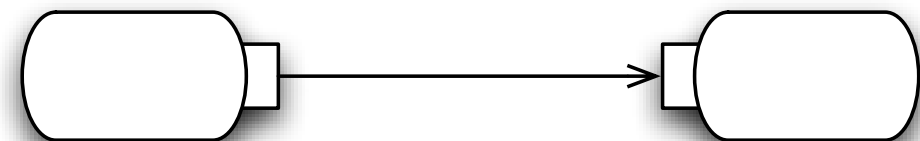
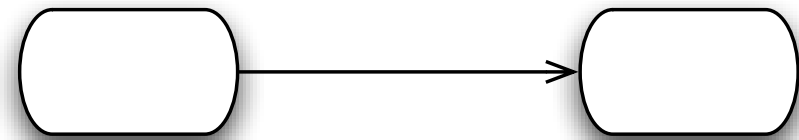
Actions

- Représentent un comportement d'exécution (une étape dans l'exécution d'un algorithme)
- Peuvent avoir une pré et une post conditions
- Types: fonctions primitives, appel de comportement, envoi d'événements, manipulation d'objets (lecture/écriture d'attributs)



Flots

- De contrôle: déclenche une action une fois que l'action précédente est finie
- D'objet: chemin par lequel peuvent passer des objets et des données

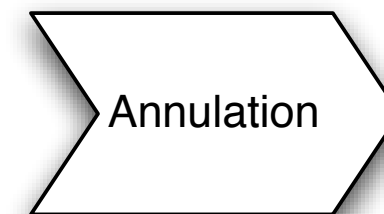


Objet

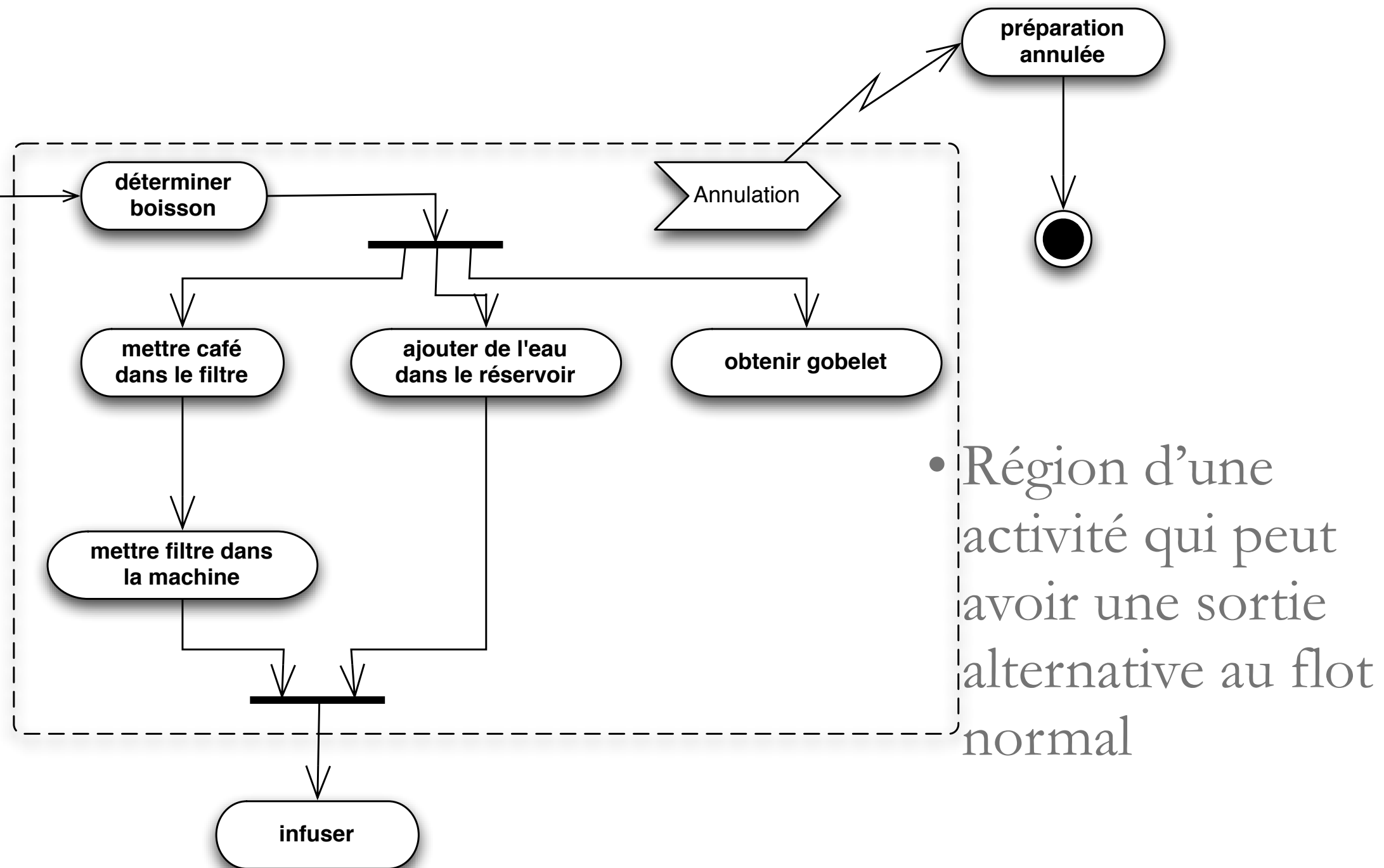
- Instance d'un classificateur, potentiellement dans un état particulier
- Événements



{ordering = LIFO}

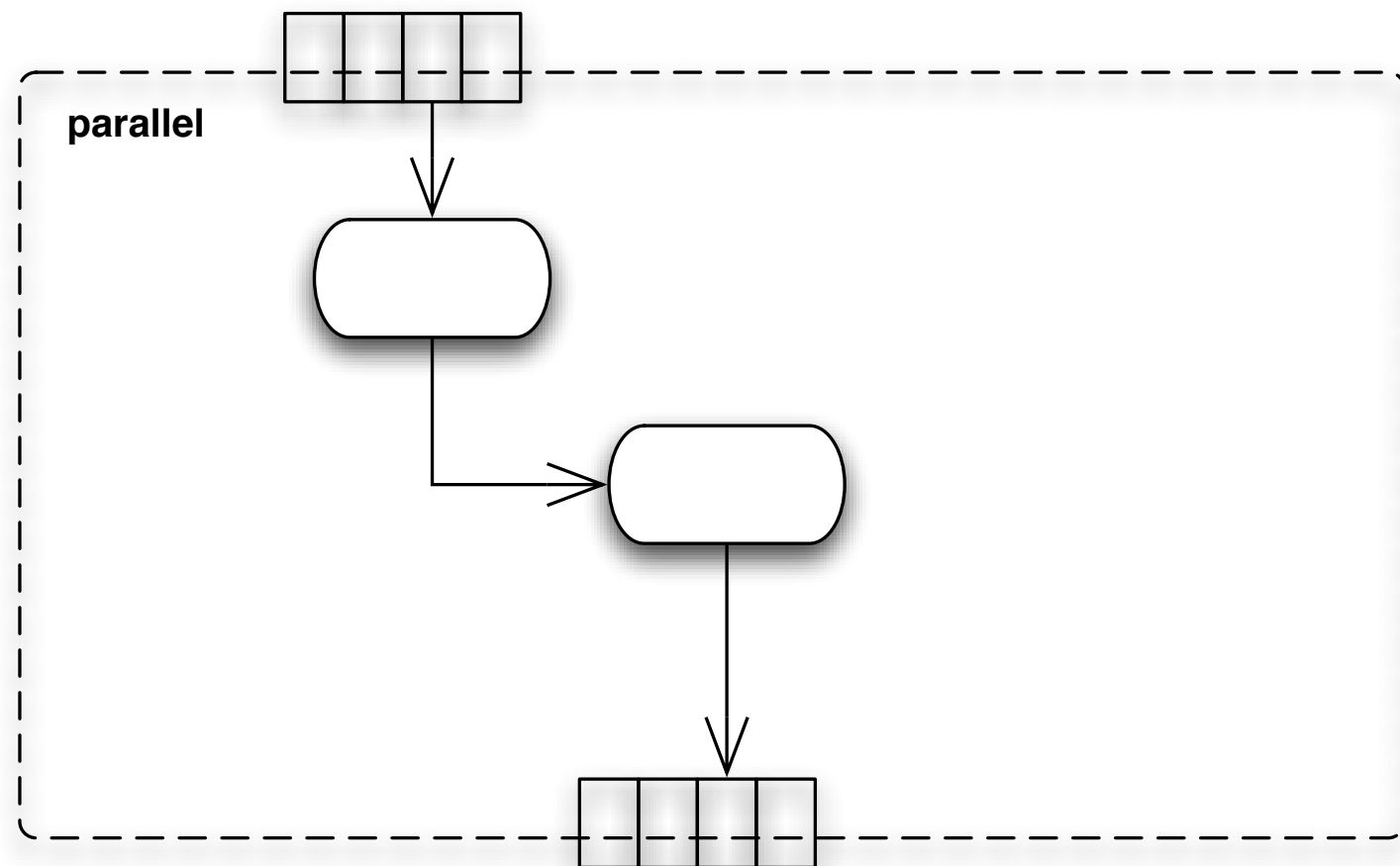


Région interruptive



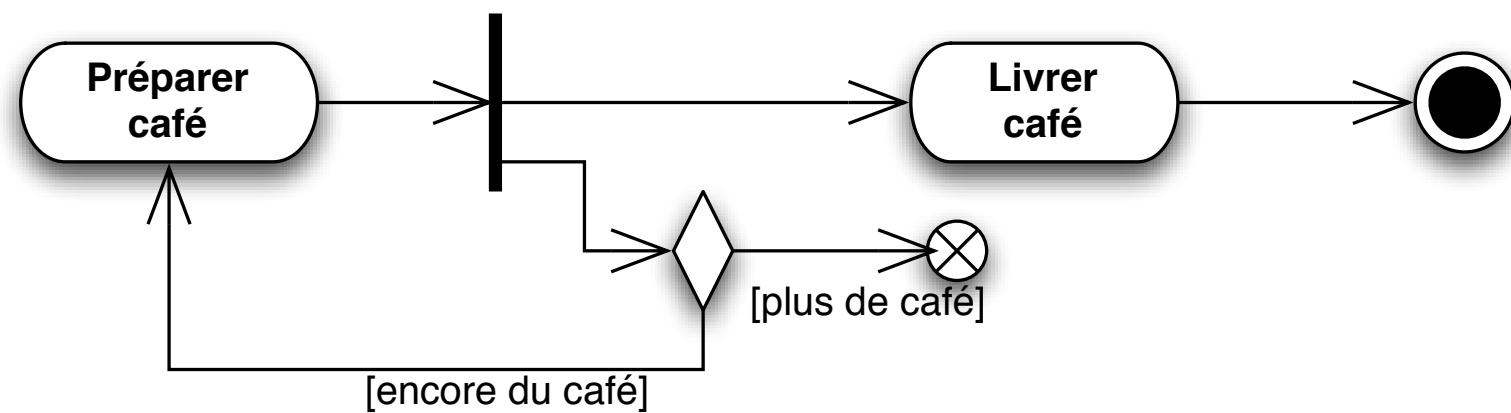
Région d'expansion

- Région qui s'exécute plusieurs fois, selon les éléments de la collection passée en entrée
- Types d'exécution: parallèle, itérative ou stream



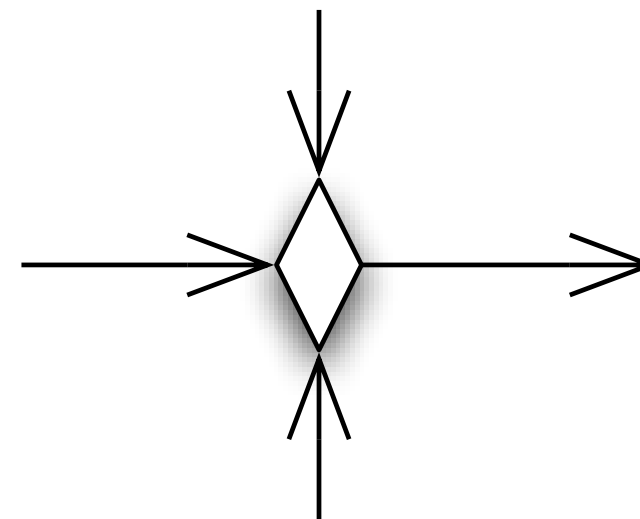
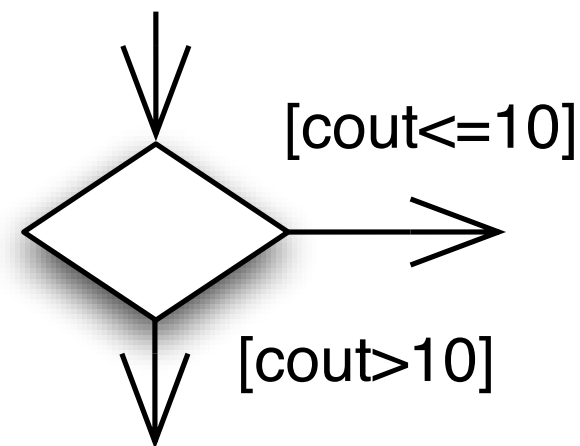
Autres noeuds

- Fin d'activité, fin de flot
- Fourchette



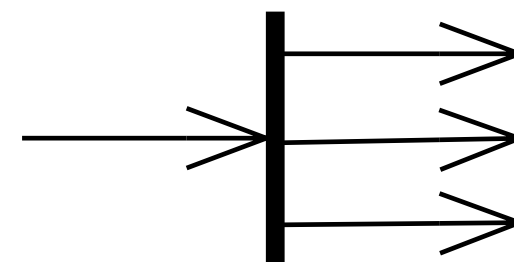
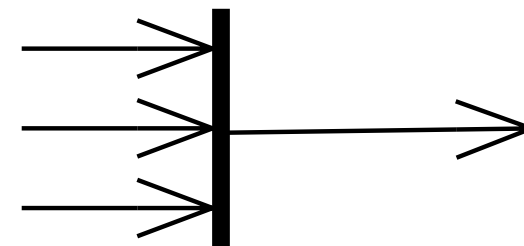
Autres noeuds

- Décision: branchement sur plusieurs transitions
- Fusion: accepte un flot parmi plusieurs



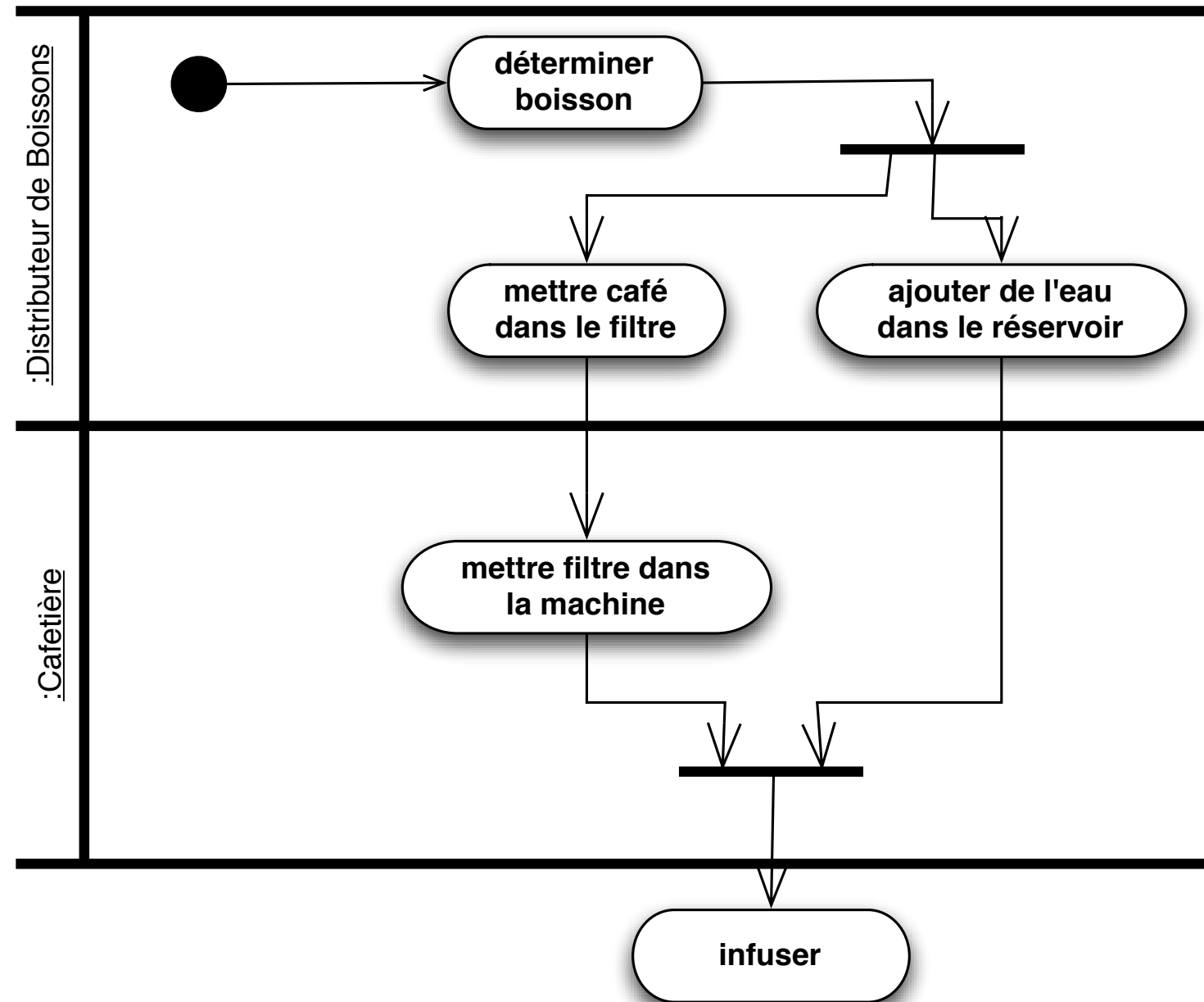
Autres Noeuds

- Jointure
- Fourchette



Partition

- Groupement d'actions ayant des caractéristiques communes
- Utilisées pour affecter des actions à des ressources différentes (Classificateurs, Noeuds)



Mécanismes d'extension

Motivation

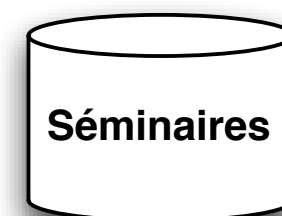
- Malgré sa richesse, UML n'est pas adapté à tous les domaines.
- Formes d'extension:
 - Ajout de nouveaux éléments.
 - Création de nouvelles propriétés.
 - Spécification d'une nouvelle sémantique.
- Mécanismes disponibles:
 - Stéréotypes, étiquettes et contraintes.

Stéréotypes

- Utilisés pour créer des éléments de modélisation spécifiques à un domaine.
- Par exemple, pour modéliser une base de données:
 - «Entity», «Relationship»
 - «Table», «Index»

Stéréotypes - notation

- Textuelle: entre guillemets (« »)
- Icône



Étiquettes

- Définition de propriétés pour n'importe quel élément de modélisation.
- Attachées à des éléments existants ou à des stéréotypes.
- Présentées sous la forme d'une paire étiquette=valeur

Étiquettes

- Utiles pour ajouter des informations concernant:
 - La documentation
 - Le contrôle de version
 - La génération de code
 - etc.

Étiquettes - notation

- Présentées entre crochés.

{version = 1.1} Cours
Nom : String Durée : Integer

Méta-modèle

Méta-modélisation

- Spécification d'un langage de modélisation grâce à un autre langage de modélisation, appelé «métalangage» ou «métalangue».

Métalangage

- Une métalangue est une langue qui prend pour objet une autre langue et qui la formalise.
- Un métalangage n'est rien d'autre qu'un langage.
- Même s'il s'agit d'un méta-méta-méta-méta-méta-méta-méta-méta-méta-méta-méta-métalangage.

Langage

- Un langage est composé d'une syntaxe et d'une sémantique.
- La syntaxe spécifie les sujets (concepts, entités) et les verbes (liens entre ces concepts) d'un langage.
- La sémantique donne une interprétations aux sujets et aux verbes

Syntaxe abstraite

- Définition des liens possibles
- Exemple: une association lie deux classes, une classe contient des attributs, etc.

Syntaxe concrète

- Syntaxe concrète: définition de la représentation des sujets et des verbes.
- Par exemple: graphique (diagrammes), textuelle (HUTN).

Cours
Nom : String
Durée : Integer

```
Class "Cours" {  
  attribute: "nom" {  
    type = "String"  
  }  
  attribute: "durée" {  
    type = "String"  
  }  
}
```

Méta-modèle

- Modèle représentant la syntaxe abstraite d'un langage.
- Celui de UML est représenté en MOF et organisé en différents paquets.
- Il est complété par des règles de bonne formation (well formedness rules) sous la forme de contraintes OCL.

Profils

Profil

- Mécanisme d'extension d'UML permettant la création de modèles dans des domaines spécifiques:
 - Systèmes de temps-réel, EJB, Corba, Test, etc.

Spécification

- Un profil est un paquetage qui contient un ensemble de mécanismes d'extension: stéréotypes, étiquettes, contraintes, etc.
- Ces mécanismes s'appliquent à des éléments UML existants:
 - les profils ne modifient pas le méta-modèle de UML.

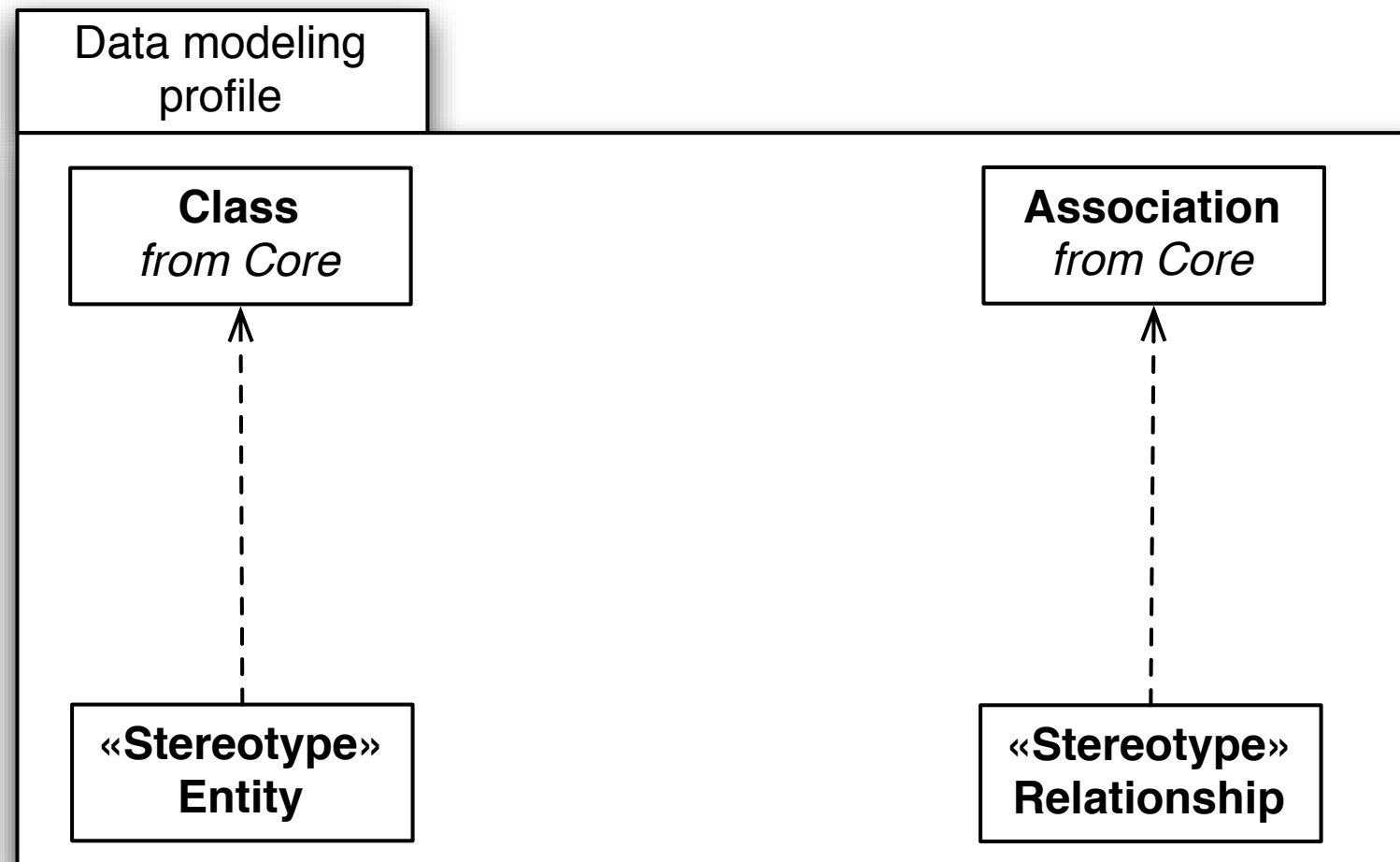
Rôle

- Identifier un sous-ensemble d'UML
- Spécifier des stéréotypes et/ou des étiquettes.
- Spécifier d'autres règles de bonne formation.
- Spécifier une sémantique (toujours en langage naturel).

Exemple

- Profil UML pour la modélisation de données.
- Éléments:
 - Entités
 - Associations
- Contraintes: les entités sont des classes qui n'ont pas de comportement

Le profil de modélisation



context Stereotype

inv:

```
self.name = "Entity" implies  
self.baseClass.ownedOperation->isEmpty()
```

Example

