Les éléments de modélisation d'UML

Pierre-Alain Muller

ENSISA

pa.muller@uha.fr

+33 (0)3.89.33.69.65

Sommaire

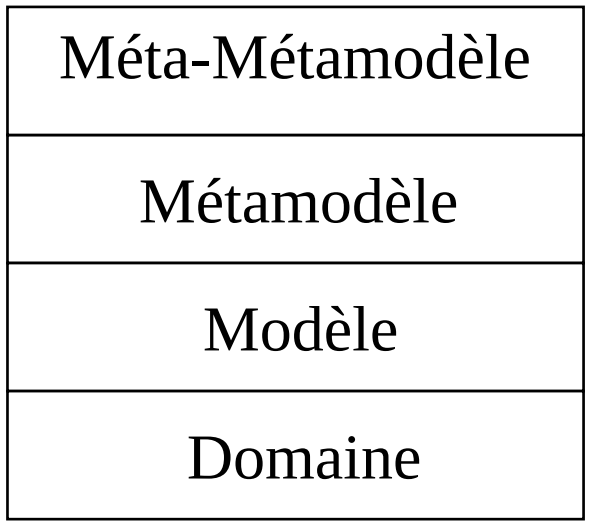
- Mécanismes généraux
- Les besoins, le comportement, la structure statique
- La réalisation, le déploiement
- l’articulation des diagrammes

Mécanismes généraux

- Introduction au métamodèle
- Les paquetages
- Les stéréotypes
- Les étiquettes
- Les notes
- Les contraintes

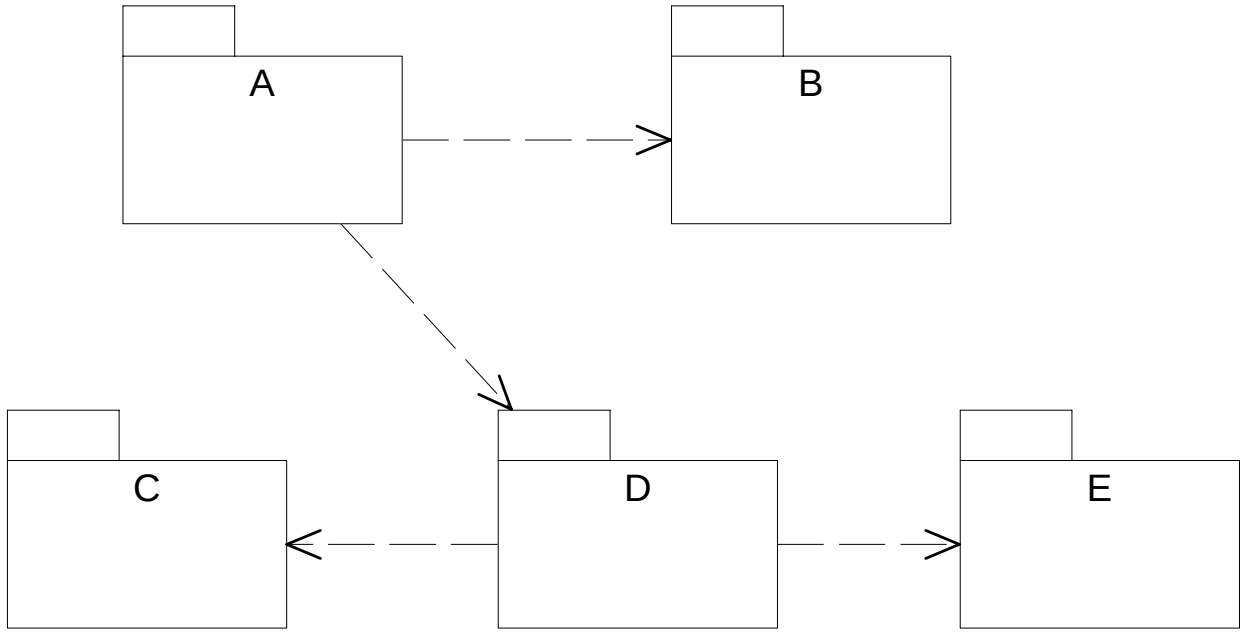
Le métamodèle

- Description formelle des éléments de modélisation



Les paquetages

- Structuration des modèles



Les stéréotypes

- Extension contrôlée des classes du métamodèle
- Spécialisation dont la profondeur est limitée à un niveau
- `<<Nom du stéréotype>>`
- Possibilité de modifier l'icone

Les étiquettes

- Extension des attributs des classes du métamodèle
- Paire **`(nom, valeur)`**

Nom de classe

<<Stéréotype>>

Propriété

Véhicule

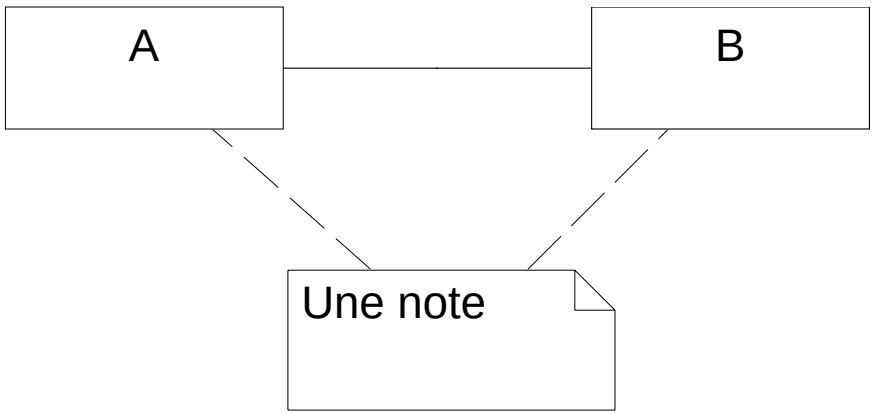
<<Utilitaire>>

Etat = testé

Auteur = pam

Les notes

- Commentaire attaché à un ou plusieurs éléments de modélisation
- Appartient à la vue, pas au modèle
- Peut être stéréotypée en contrainte



Les contraintes

- Relation sémantique quelconque entre éléments de modélisation
- Exprimée en OCL (Object Constraint Language) ou en langage naturel
  - {Contrainte}

OCL Object Constraint Language

- Développé par IBM
- Pas d'effets de bord, langage de spécification
- Spécification formelle de la sémantique statique
  - Métamodèle (invariant d'une instance de métaclasse)
  - Modèle (règles sur les attributs et les relations)
- Utilisable par tout le monde, pas seulement par des mathématiciens

Exemples OCL

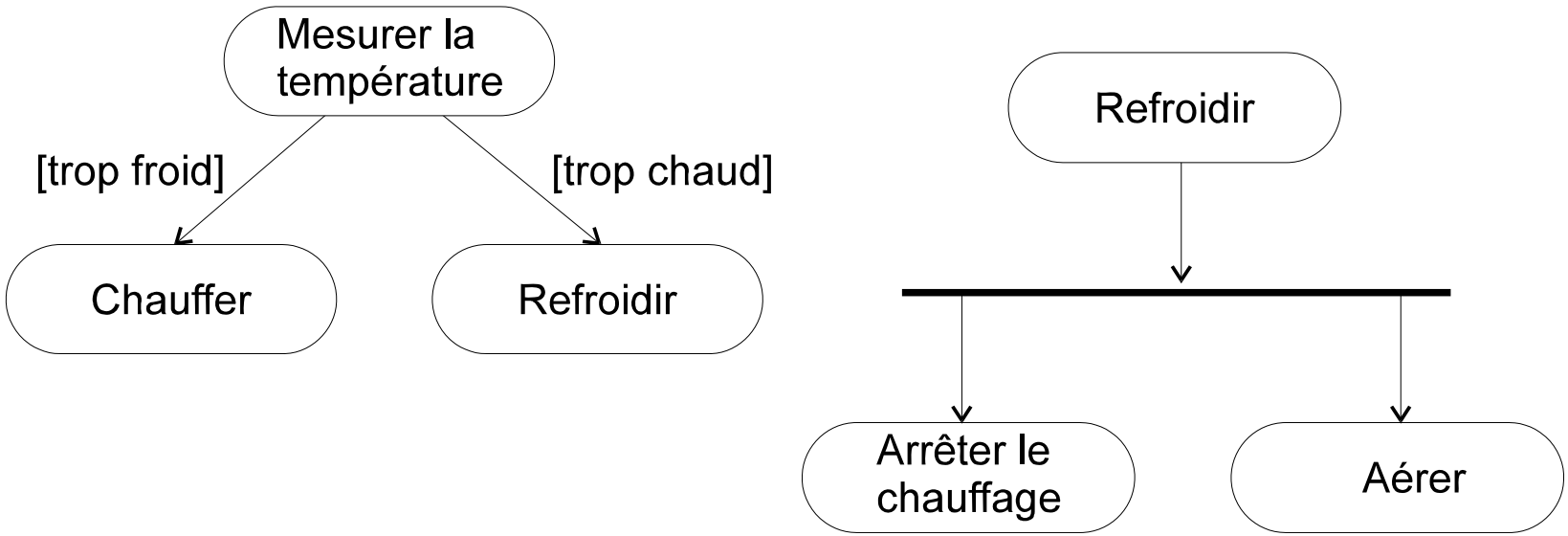
- Types de base
  - `Boolean, Integer, String, Set, Sequence`
- Opérations
  - `and, or, not, implies`
  - `+, -, >, =, ...`
- Exemple
  - `self.allRoles.forAll (r1, r2 | r1.name = r2.name implies r1 = r2)`

Expression des besoins

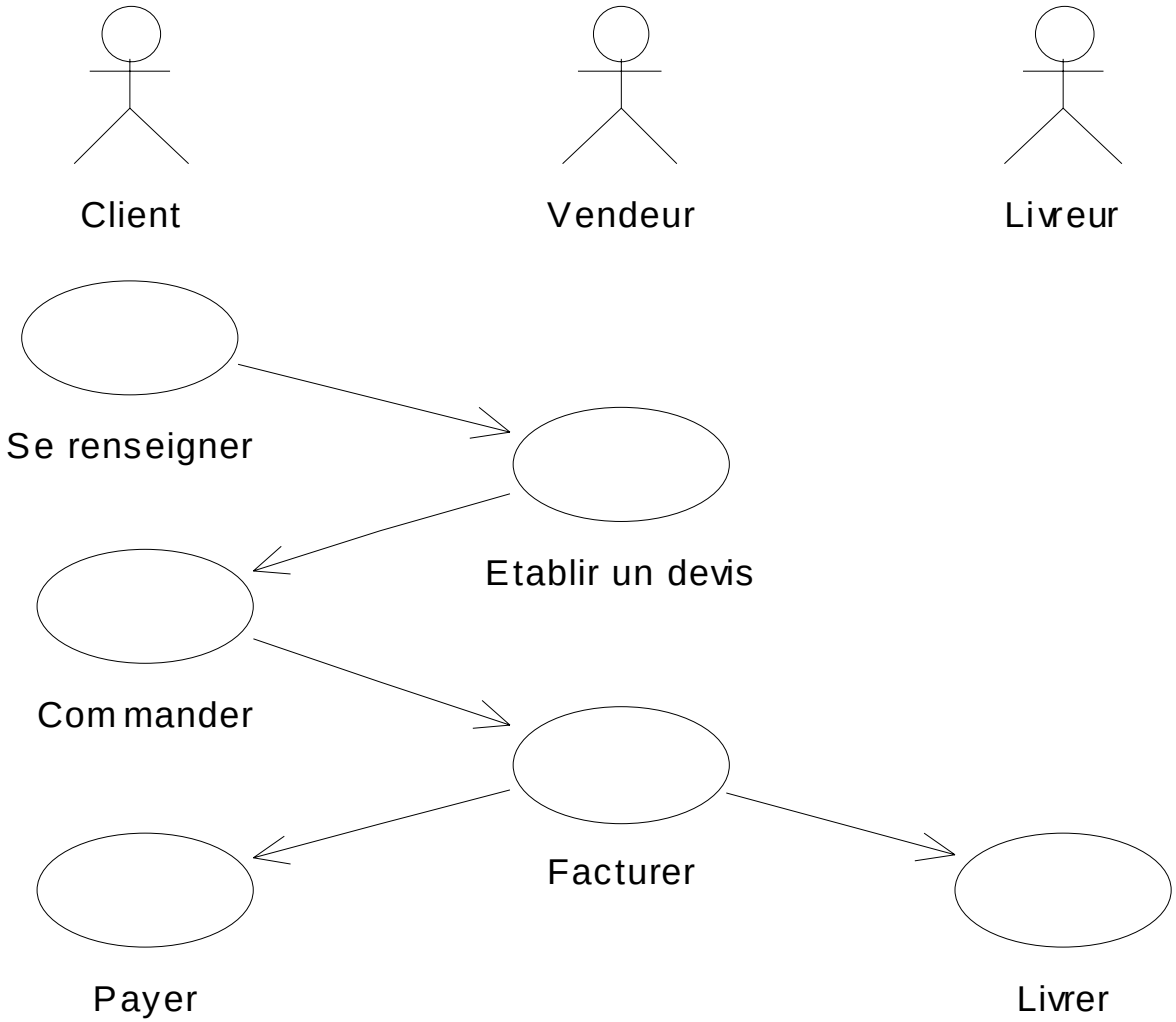
- Les activités
- Les cas d’utilisation et les scénarios

Les activités

- Représentation des activités réalisées par les différents acteurs



Exemple de diagramme d'activités



Les cas d'utilisation

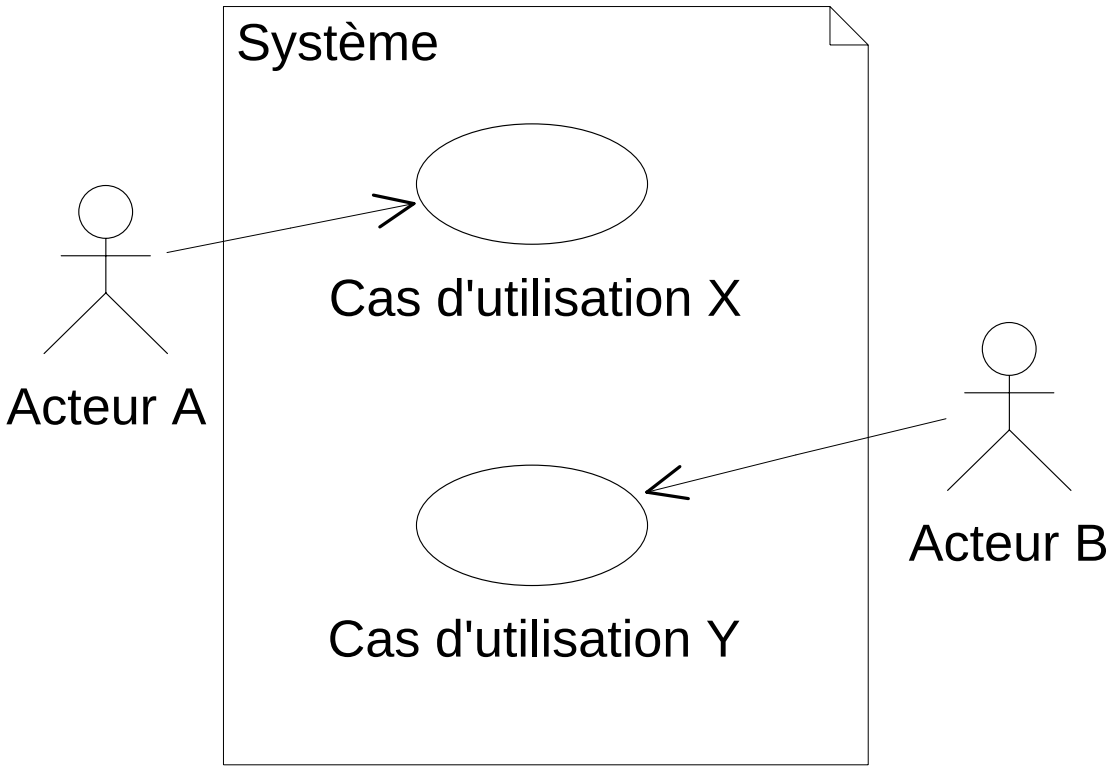
- Formalisés par Ivar Jacobson : Object-Oriented Software Engineering (Addison-Wesley, 1992)
- Expression du comportement du système (actions et de réactions), selon le point de vue de l'utilisateur
- Décrivent le système et les relations entre le système et l'environnement

Intérêt des cas d'utilisations

- Constituent un moyen de déterminer les besoins d'un système
- Utilisés par les utilisateurs finaux pour exprimer leur attentes et leur besoins
- Permettent d'impliquer les utilisateurs dès les premiers stades du développement
- Constituent une base pour les tests fonctionnels

Les cas d’utilisation

- Représentation graphique



Les acteurs

- Un acteur est une personne ou un système qui interagit avec un système, en échangeant de l'information (en entrée et en sortie)
- On trouve les acteurs en observant les utilisateurs directs du système, ceux qui sont responsable pour sa maintenance, ainsi que les autres systèmes qui interagissent avec le système

Les utilisateurs

- Un acteur représente un rôle joué par un utilisateur qui interagit avec le système
- La même personne physique peut jouer le rôle de plusieurs acteurs (vendeur, client)
- D’autre part, plusieurs personnes peuvent également jouer le même rôle, et donc agir comme le même acteur (tous les clients)

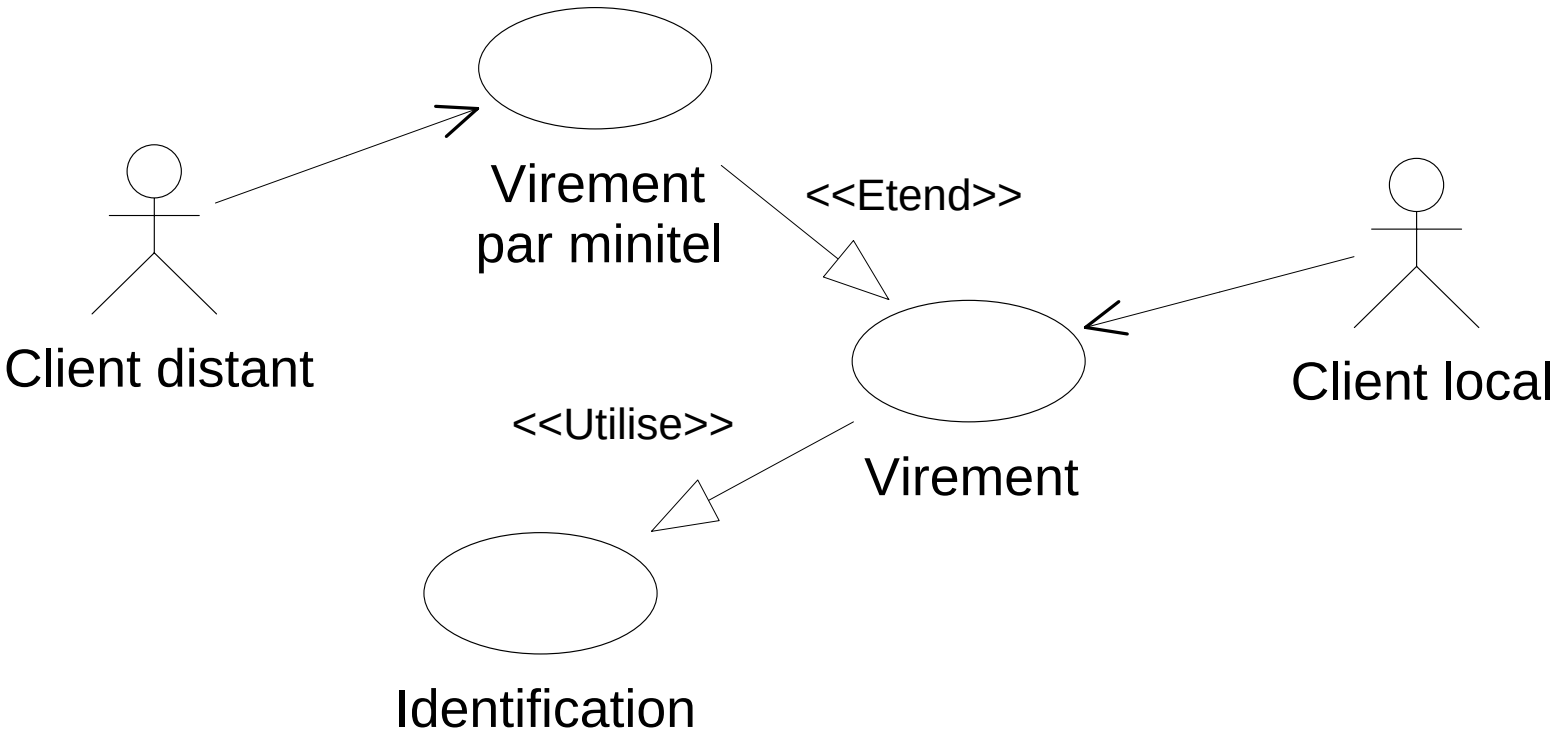
Acteurs et cas d’utilisation

- Les cas d’utilisation représentent le dialogue entre l’acteur et le système de manière abstraite
- Ensemble de scénarios au sein d’une description unique
- Les cas d’utilisation doivent être vus comme des classes de scénarios

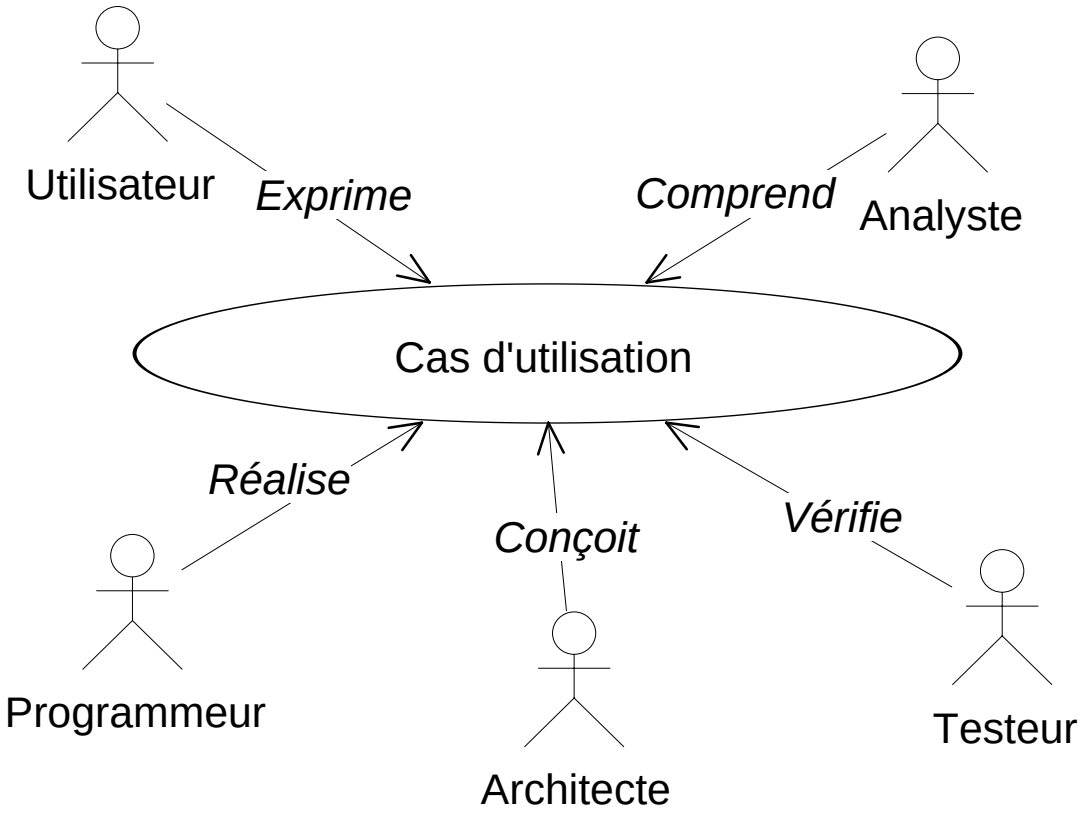
Détermination des cas d'utilisation

- Quelles sont les tâches de l'acteur ?
- Quelles informations l'acteur doit-il créer, sauvegarder, modifier, détruire ou simplement lire ?
- L'acteur devra-t-il informer le système de changements externes ?
- Le système devra-t-il informer l'acteur de conditions internes au système ?

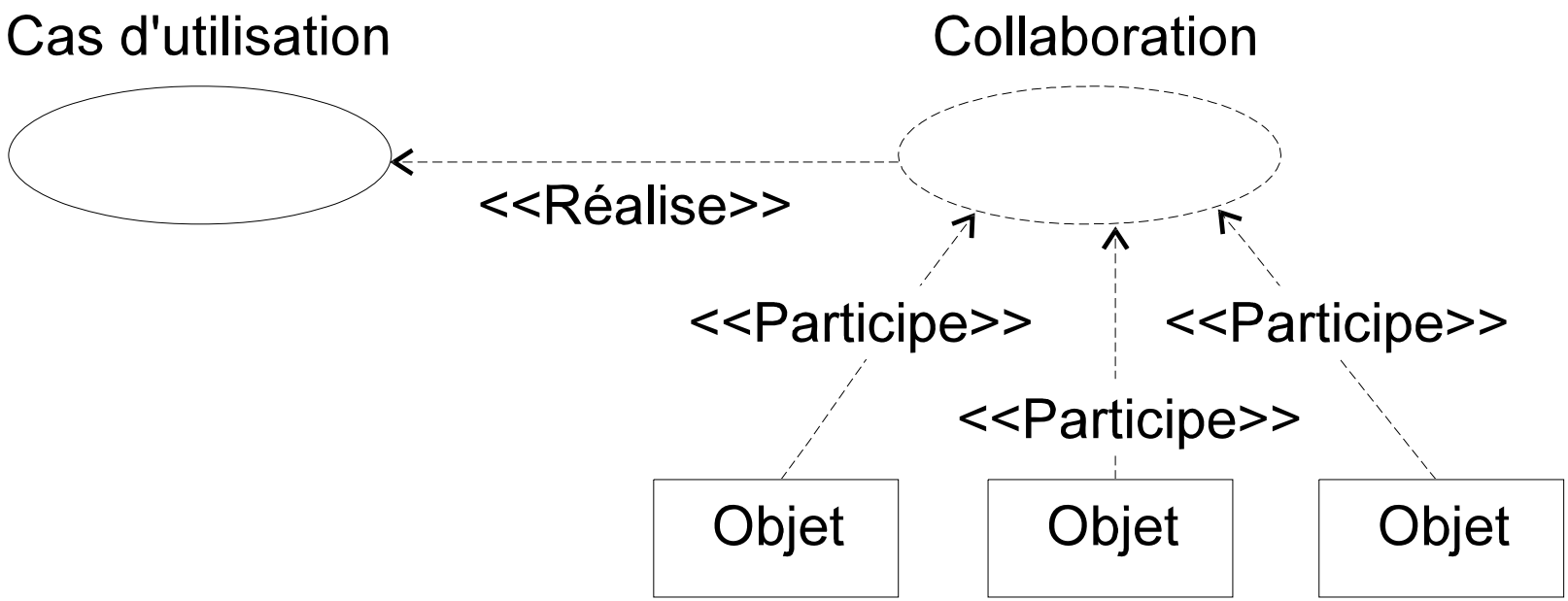
Diagramme de cas d’utilisation



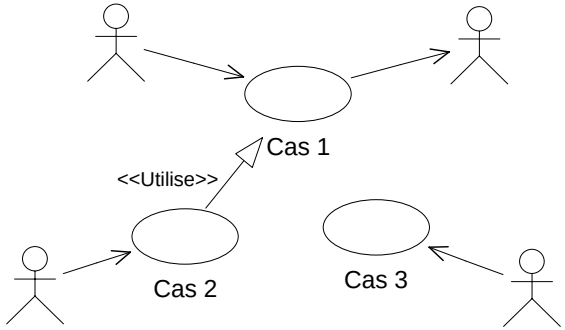
Fil conducteur du projet



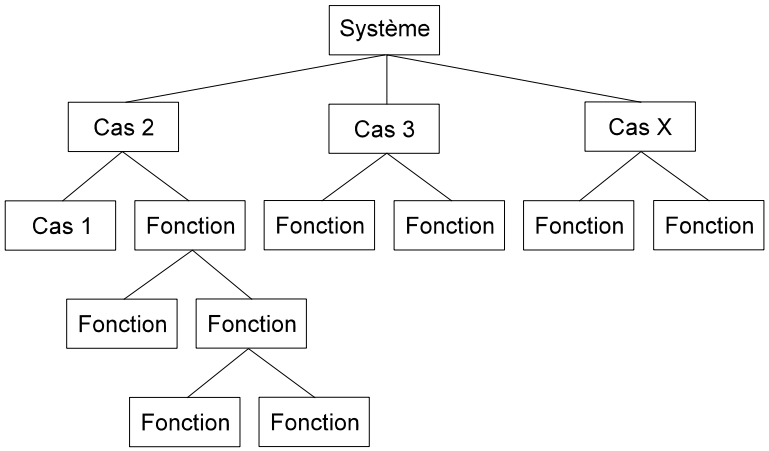
Transition vers les objets



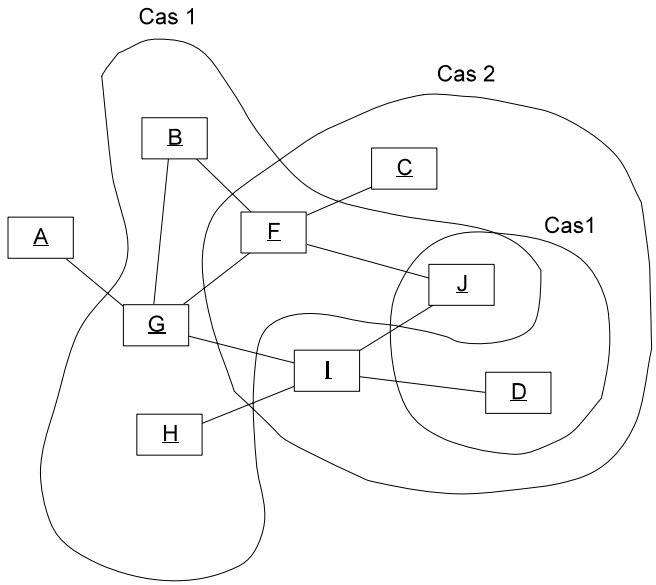
Le «virage» vers l'objet



Décomposition structurée



Décomposition objet



Communication entre objets

- Application = société d'objets collaborants
- Les objets travaillent en synergie afin de réaliser les fonctions de l’application
- Le comportement global d’une application repose donc sur la communication entre les objets qui la composent

Expression du comportement

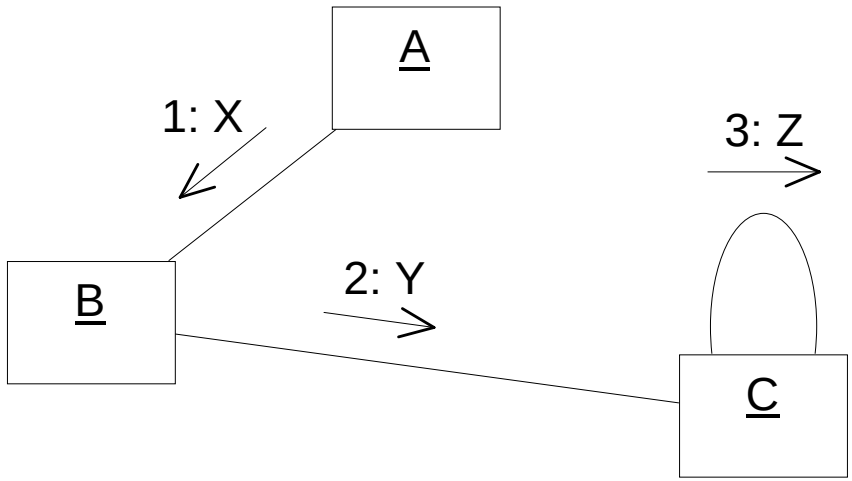
- Les collaborations
  - Diagramme de collaboration
  - Diagramme de séquence

Les diagrammes de collaboration

- Des objets dans une situation donnée
- Des liens relient les objets qui se connaissent
- Les messages échangés par les objets sont représentés le long de ces liens
- L’ordre d’envoi des messages est matérialisé par un numéro de séquence

Exemple

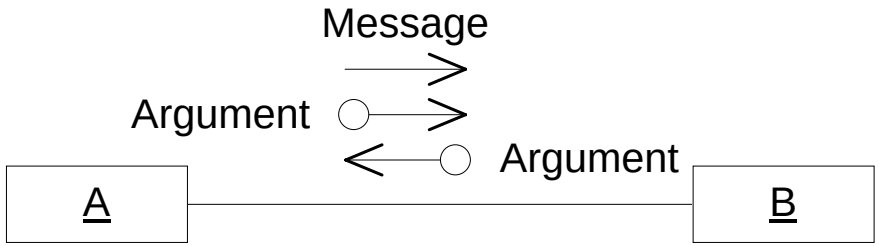
- *Un objet **A** envoie un message **X** à un objet **B**, puis l'objet **B** envoie un message **Y** à un objet **C**, et enfin **C** s'envoie un message **Z**.*



Le concept de messages

- L’unité de communication entre objets
- Concept très général pouvant être mis en œuvre suivant de nombreuses variantes
- Regroupe les flots de contrôle et les flots de données
- Représente également les événements

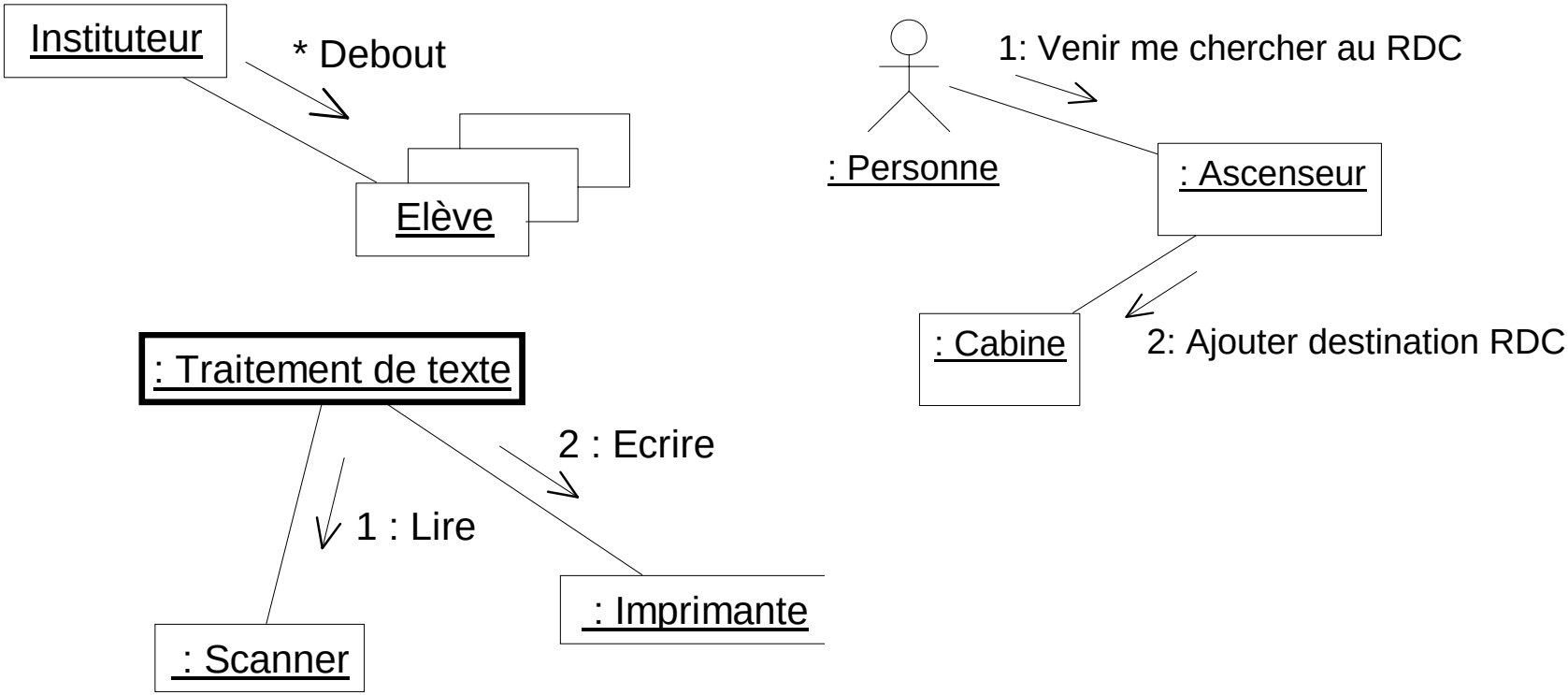
Envoi des messages



- 4 : Afficher (x, y) -- message simple
- 3.3.1 : Afficher (x, y) -- message imbriqué
- 4.2 : âge := Soustraire (Aujourd'hui, DateDeNaissance)
  - -- message imbriqué avec valeur retournée
- [Age >= 18 ans] 6.2 : Voter ()
  - -- message conditionnel
- a.4, b.6 / c.1 : Allumer (Lampe)
  - -- synchronisation avec d'autres flots d'exécution
- 1 * : Laver () -- itération
- 3.a, 3.b / 4 *||[i := 1..n] : Eteindre ()
  - -- itération parallèle

Diagramme de collaboration

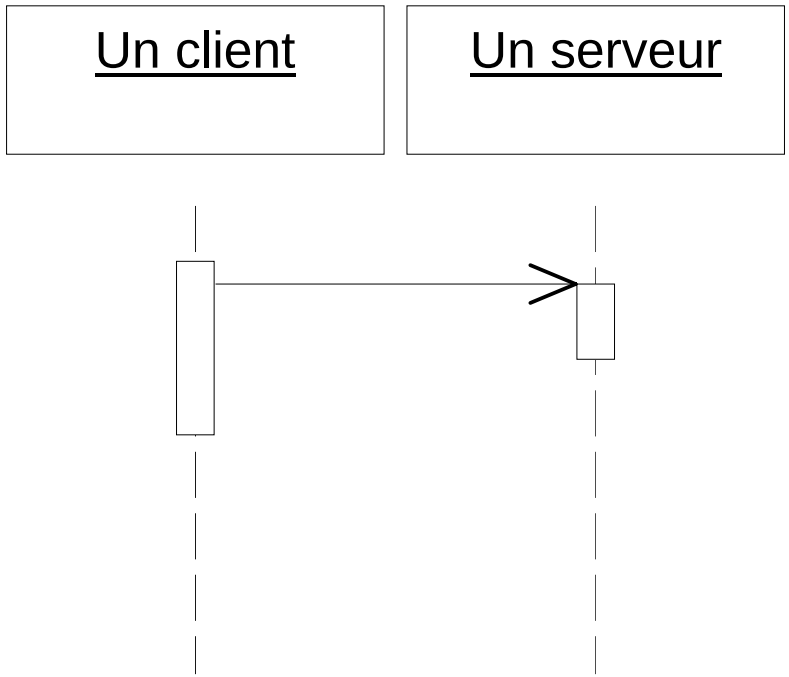
- Représentation spatiale



Les diagrammes de séquence

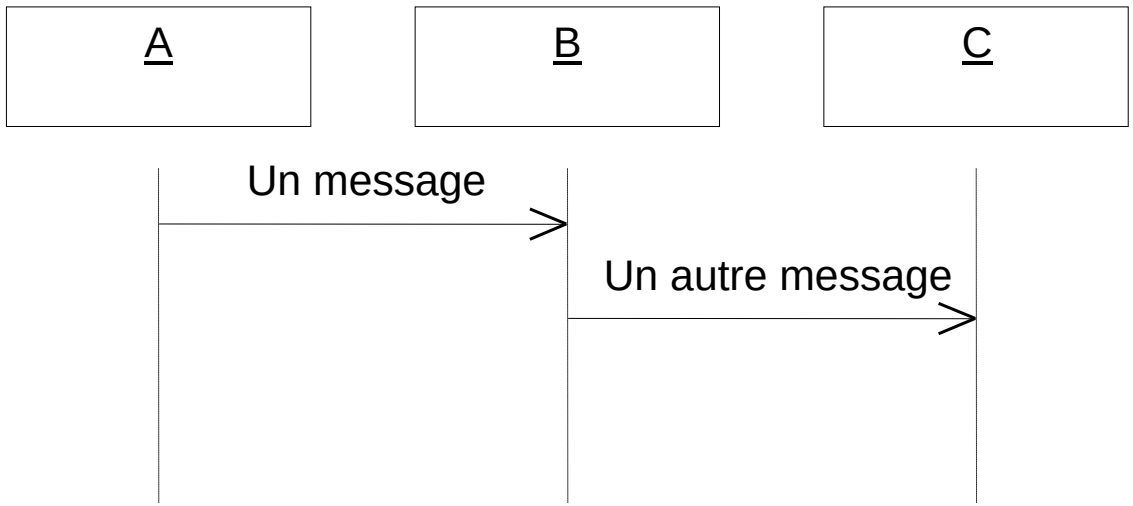
- L'accent est mis sur la communication, au détriment de la structure spatiale
- Chaque objet est représenté par une barre verticale
- Le temps s'écoule de haut en bas, de sorte que la numérotation des messages est optionnelle.

Exemple

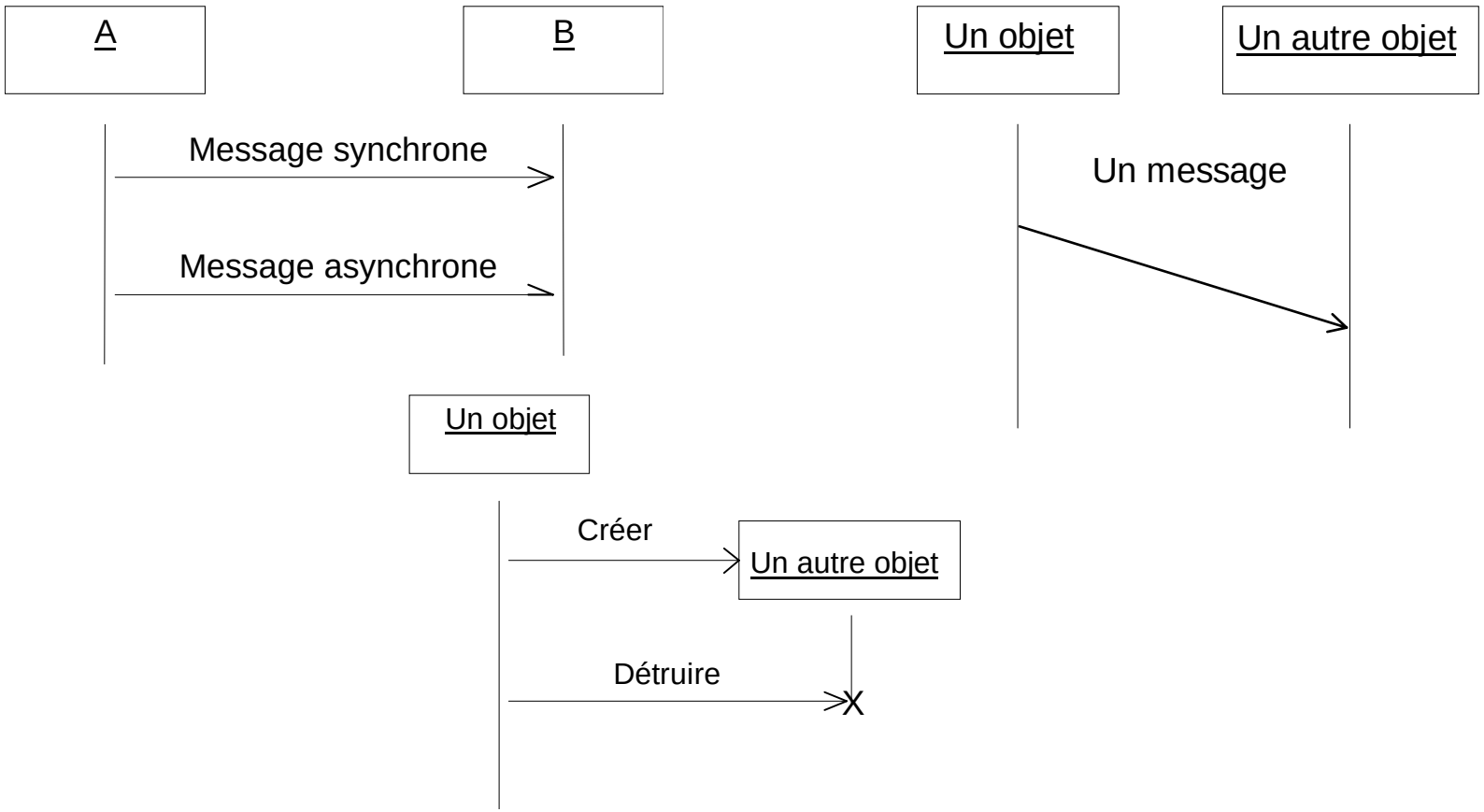


Diagrammes de séquence

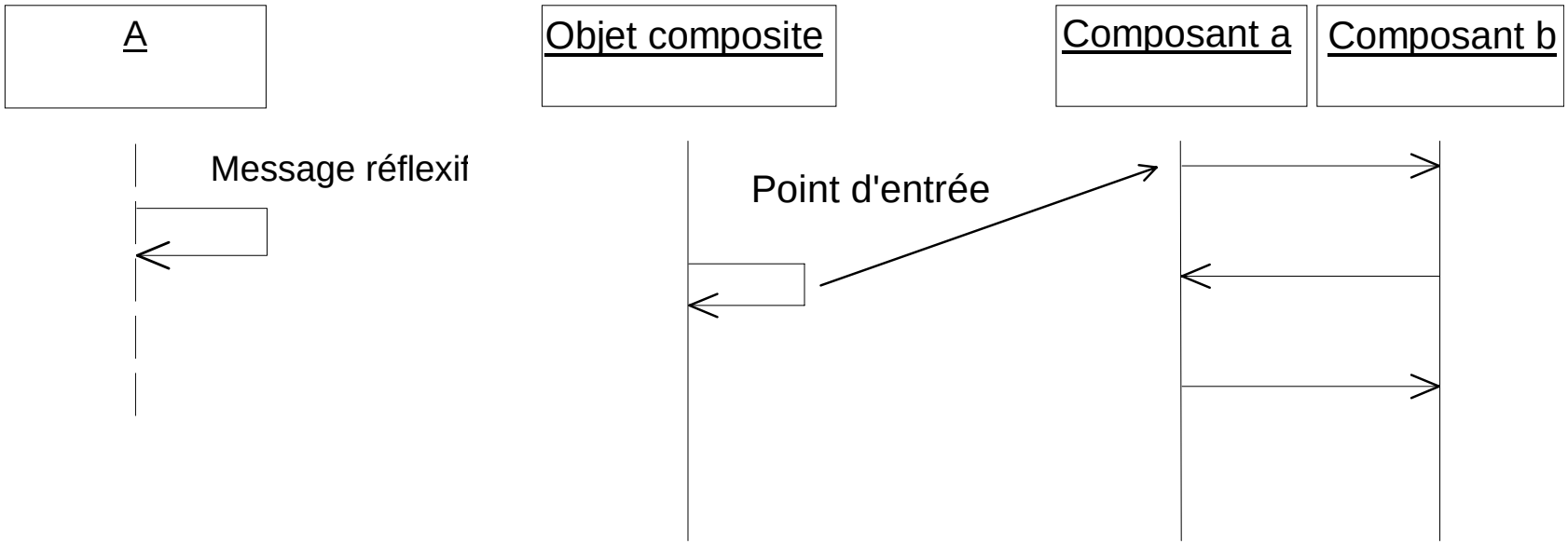
- Représentation temporelle



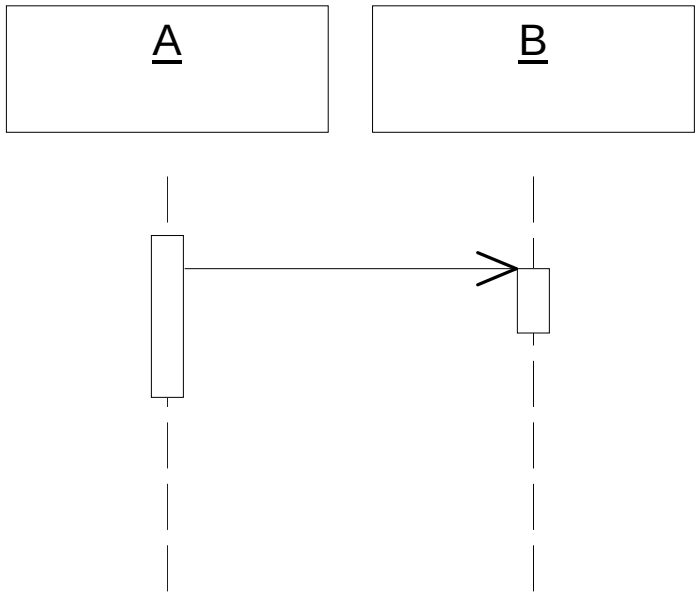
Diagrammes de séquence

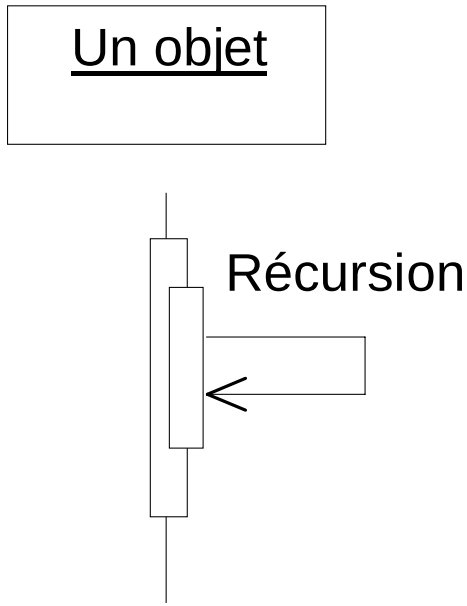


Diagrammes de séquence

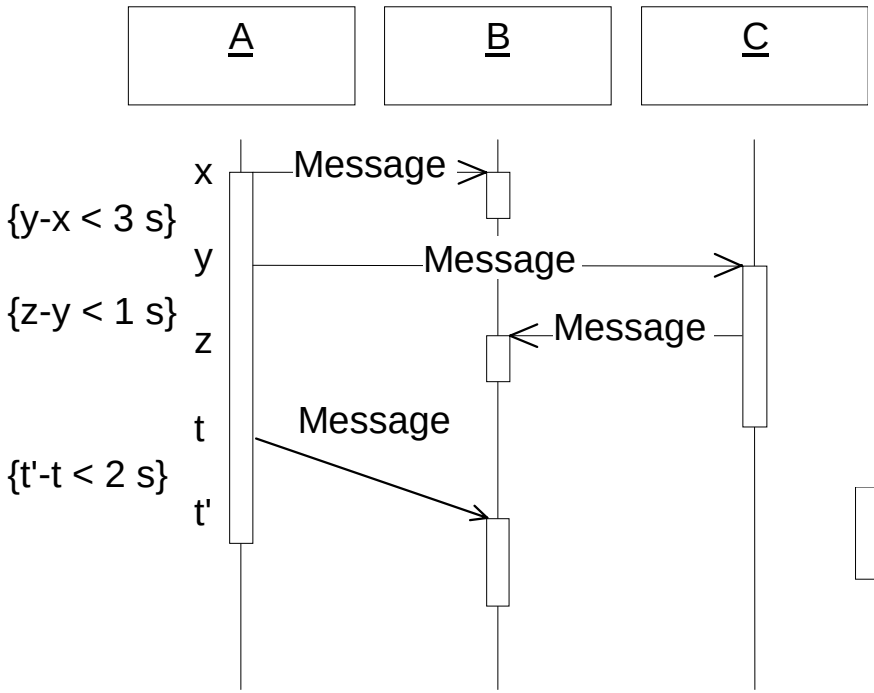


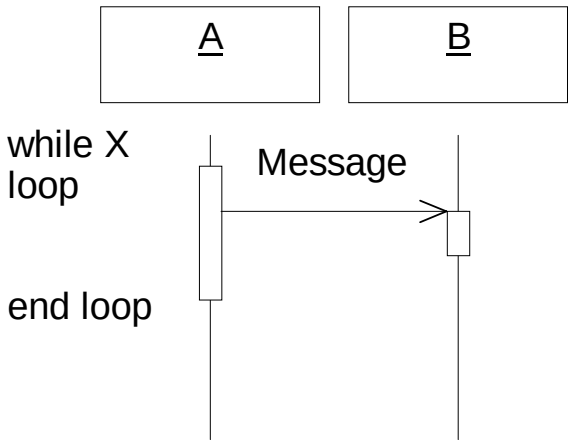
Diagrammes de séquence

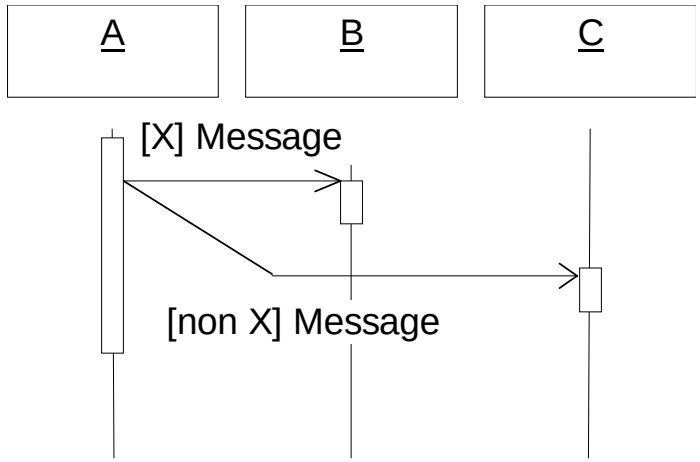




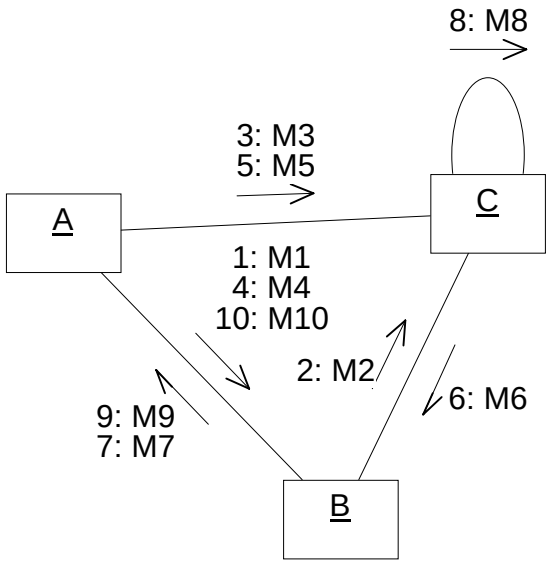
Diagrammes de séquence

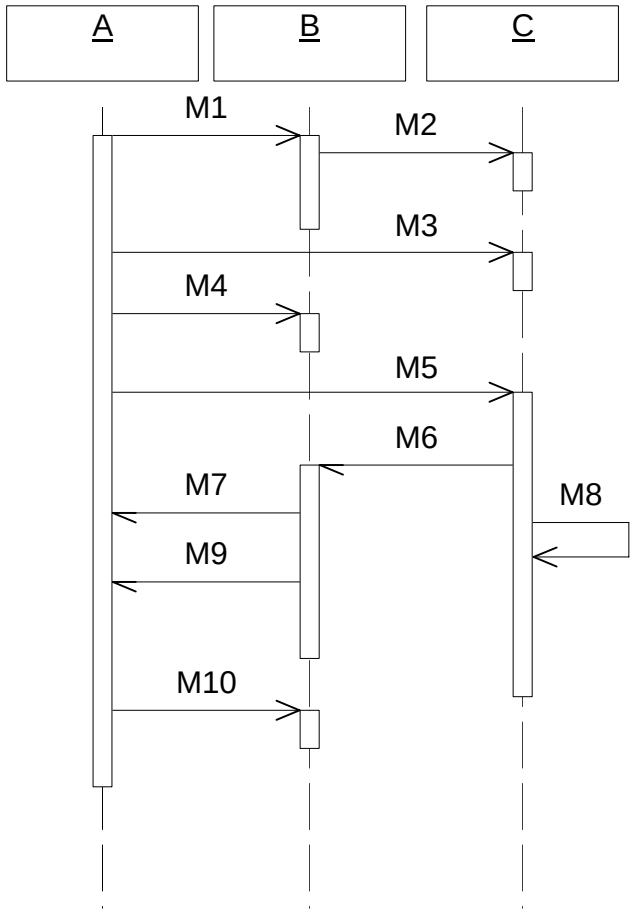






Comparaison

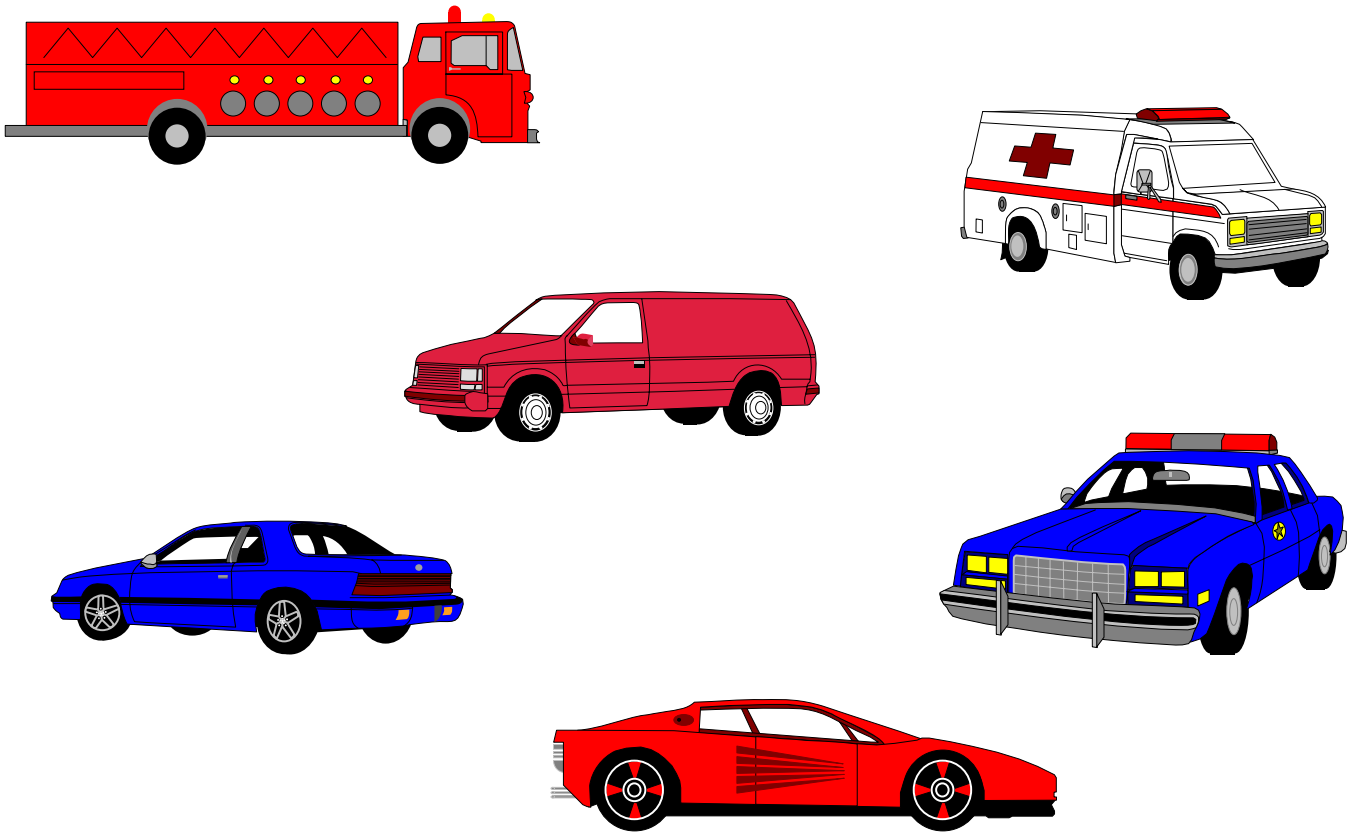




Expression de la structure statique

- Diagrammes des classes
- Les relations
  - Association
  - Agrégation
  - Généralisation
  - Dépendance

Le chaos des objets



Le chaos des objets

- Le monde qui nous entoure est constitué de très nombreux objets
- Pour comprendre le monde, l'être humain a tendance à regrouper les éléments qui se ressemblent
- Regrouper des objets suivants des critères de ressemblance s'appelle classer
- Les humains ont classé les animaux, les plantes, les champignons, les atomes, ...

Les classes

- La classe est une description abstraite d’un ensemble d’objets
- La classe peut être vue comme la factorisation des éléments communs à un ensemble d’objets
- La classe décrit le domaine de définition d’un ensemble d’objets

Représentation graphique des classes

Nom de classe
Attributs
Opérations( )

Nombre complexe
Additionner( ) Soustraire( ) Multiplier( ) Diviser( ) Prendre le module( ) Prendre l'argument( ) Prendre la partie réelle( ) Prendre la partie imaginaire( )

Motocyclette
Couleur Cylindrée Vitesse maximale
Démarrer( ) Accélérer( ) Freiner( )

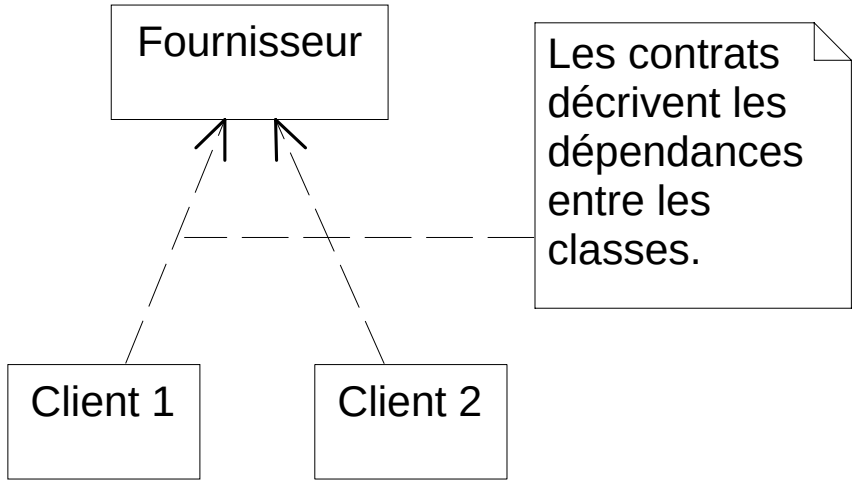
Téléviseur
Allumer( ) Eteindre( ) Changer de programme( ) Régler le volume( )

Description des classes

- Séparée en deux parties
  - La *spécification* d'une classe qui décrit le domaine de définition et les propriétés des instances de cette classe (type de donnée)
  - La *réalisation* qui décrit comment la spécification est réalisée

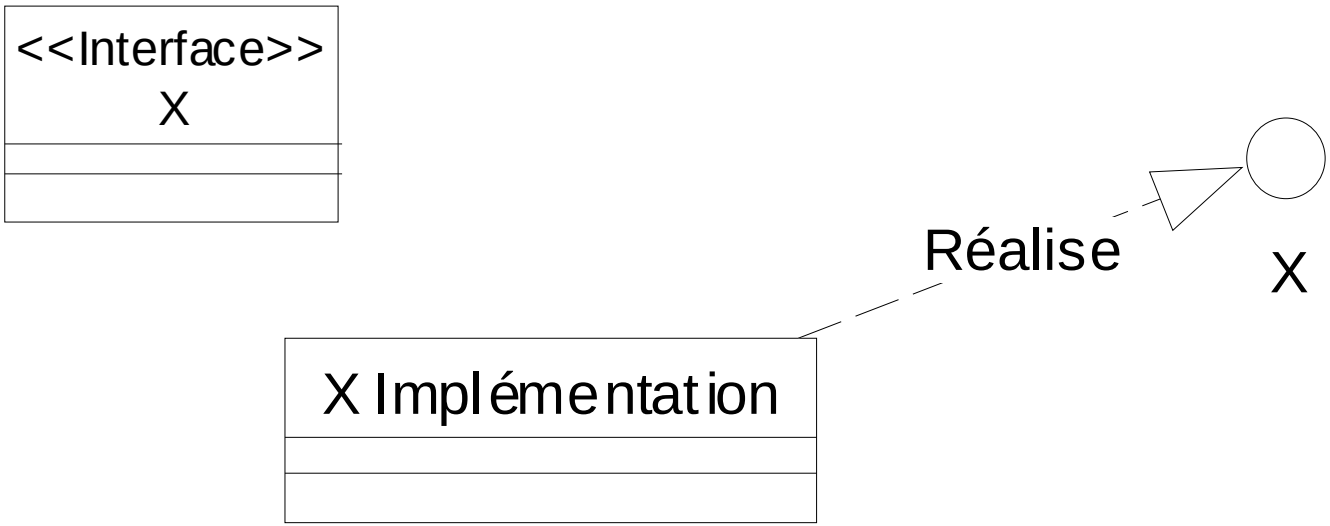
La notion de contrat

- Une classe s’engage à fournir les services publiés dans sa spécification



Les interfaces

- Contrat sans implémentation



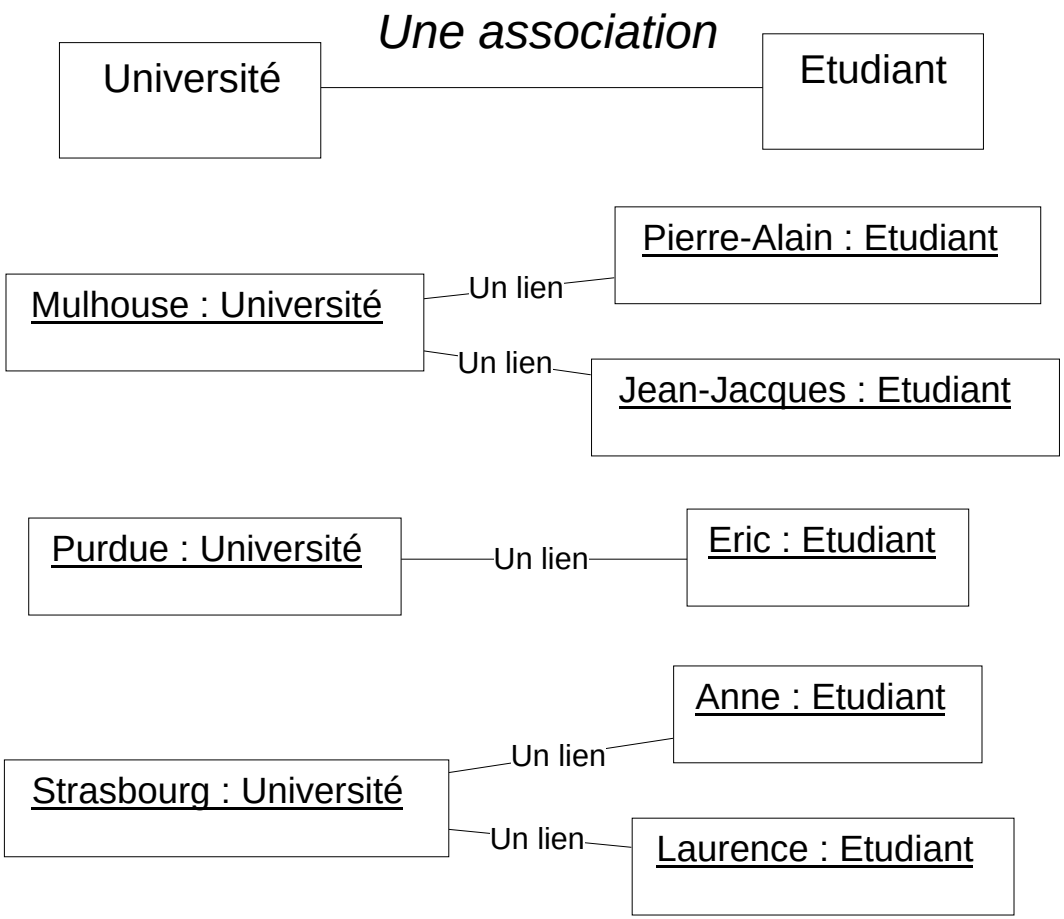
Les relations entre classes

- L’association
- L’agrégation
- La généralisation
- La dépendance

L’association

- L’association exprime une connexion sémantique bidirectionnelle entre classes
- Une association est une abstraction des liens qui existent entre les objets instances des classes associées
- Les associations se représentent de la même manière que les liens.
  - Distinction opérée en fonction du contexte

Exemple

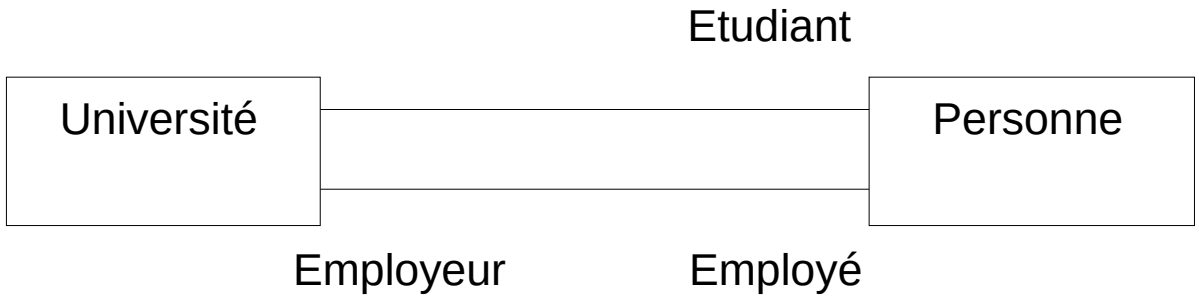

Une association
Université
Etudiant
Pierre-Alain : Etudiant
Mulhouse : Université
Un lien
Un lien
Jean-Jacques : Etudiant
Purdue : Université
Un lien
Eric : Etudiant
Anne : Etudiant
Un lien
Strasbourg : Université
Un lien
Laurence : Etudiant

Nommage des associations

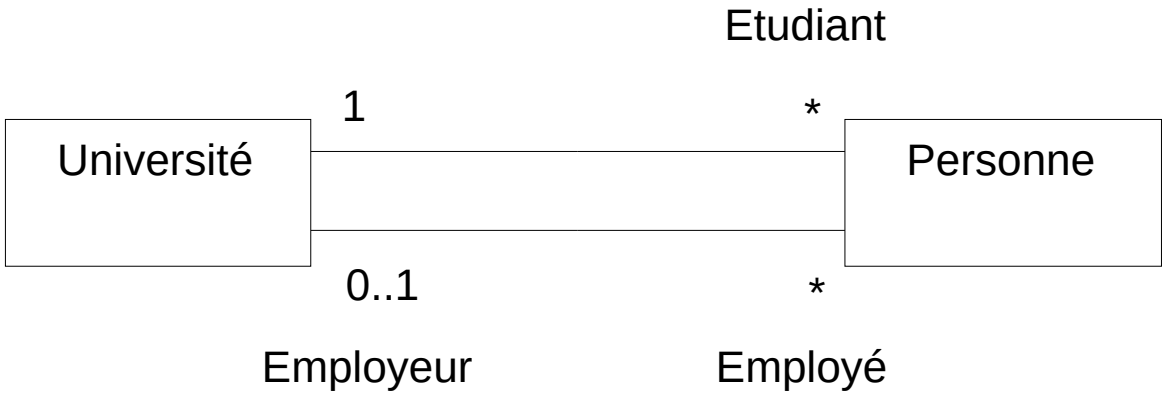
- Indication du sens de lecture

Nommage des rôles

- Le rôle décrit une extrémité d'une association



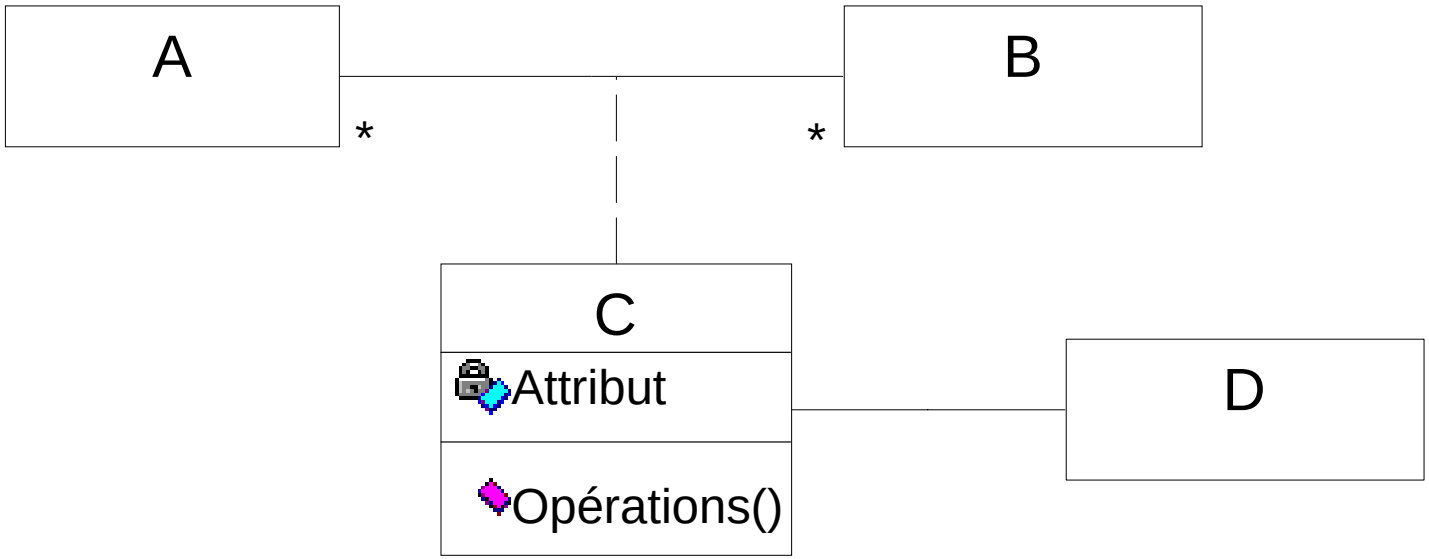
Multiplicité des rôles



1	Un et un seul
0..1	Zéro ou un
M .. N	De M à N (entiers naturels)
*	Plusieurs
0 .. *	De zéro à plusieurs
1 .. *	D'un à plusieurs

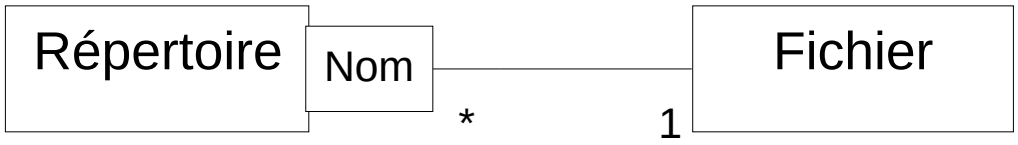
Les classes-associations

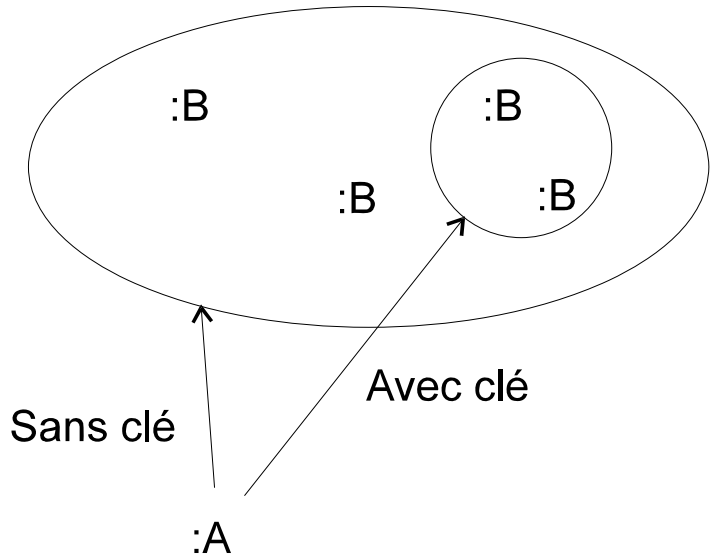
- Ajout d'attributs ou d'opérations dans la relation



Filtrage des associations

- Réduction de la multiplicité

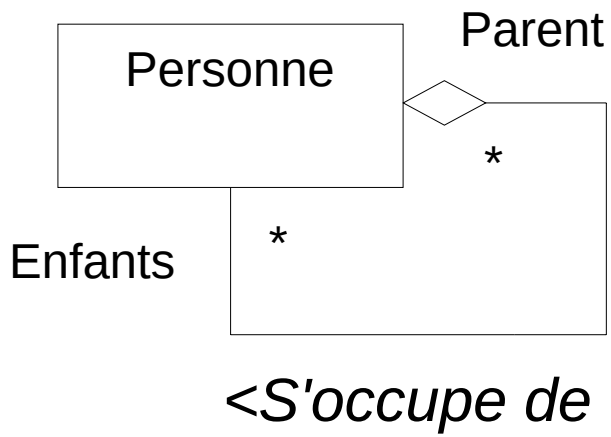




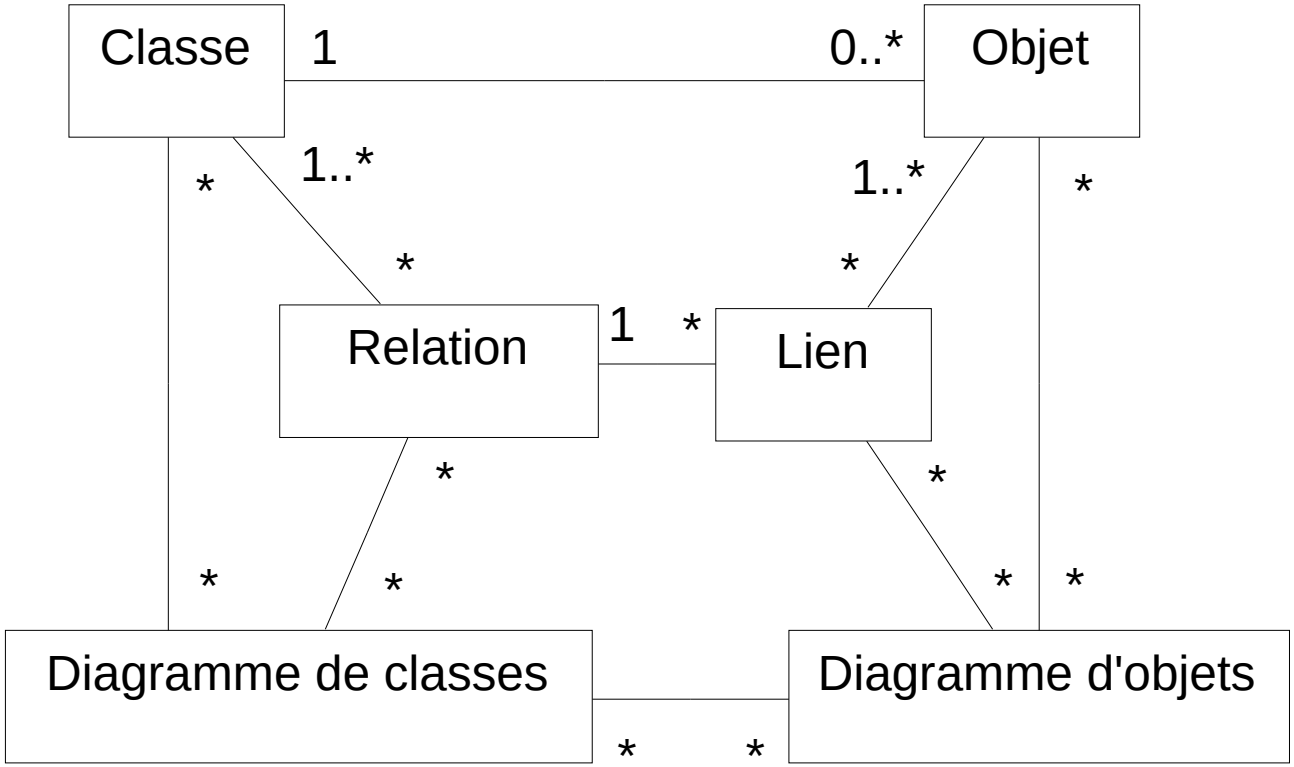
L'agrégation

- Connexions sémantiques bidirectionnelles antisymétriques
- Forme d’association qui exprime un couplage plus fort entre classes
- Représentation des relations
  - maître et esclaves
  - tout et parties
  - composé et composant.

Exemples



Correspondances entre diagrammes

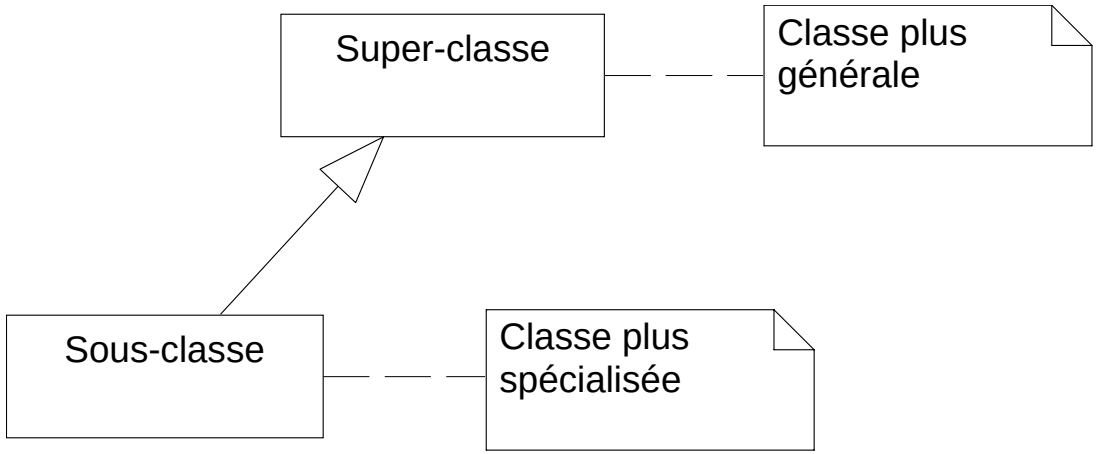


Correspondances (suite)

- Chaque objet est instance d'une classe
- Chaque lien est instance d'une relation
- Les liens relient les objets, les relations relient les classes
- Un lien entre deux objets implique une relation entre les classes des deux objets

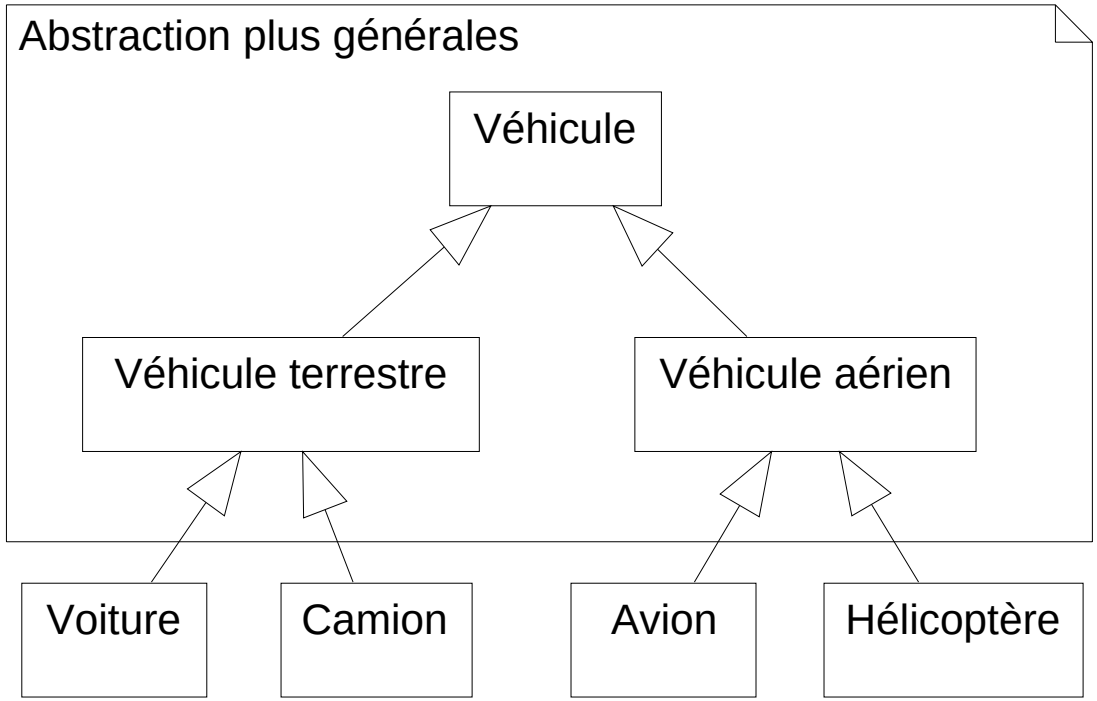
Hiérarchies de classes

- Gérer la complexité
  - Arborescences de classes d'abstraction croissante
- Généralisation
  - Super-classes
- Spécialisation
  - Sous-classes



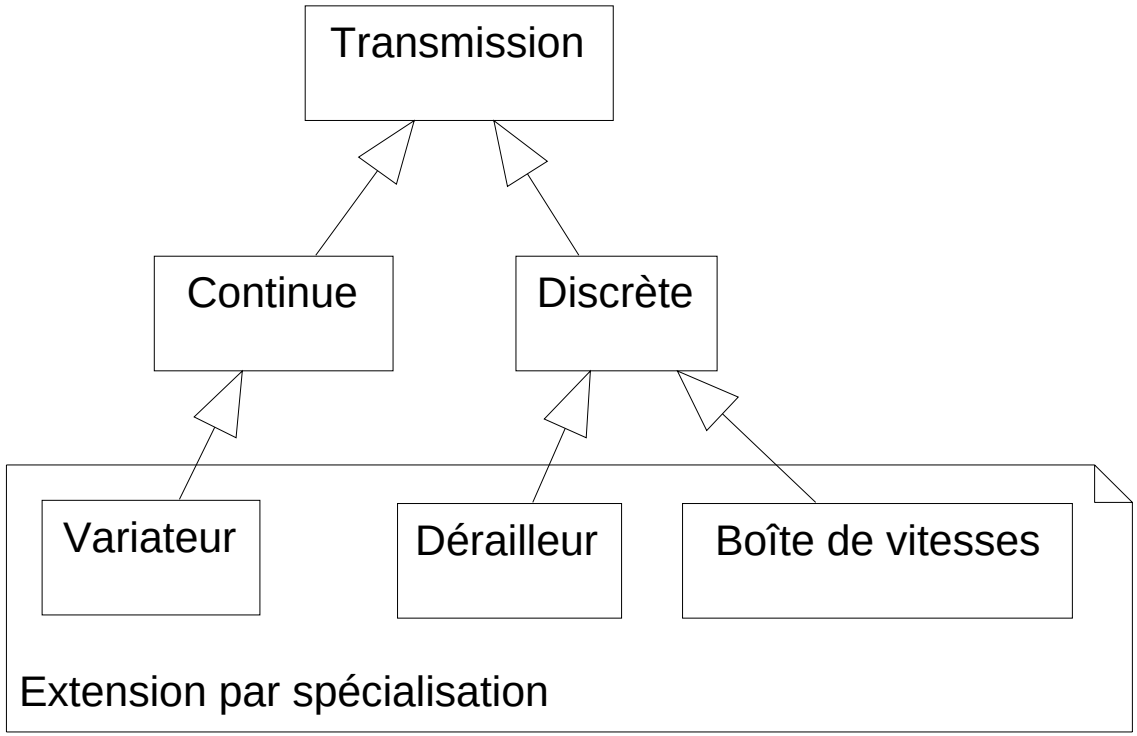
Généralisation

- Factoriser les éléments communs
  - attributs, opérations et contraintes



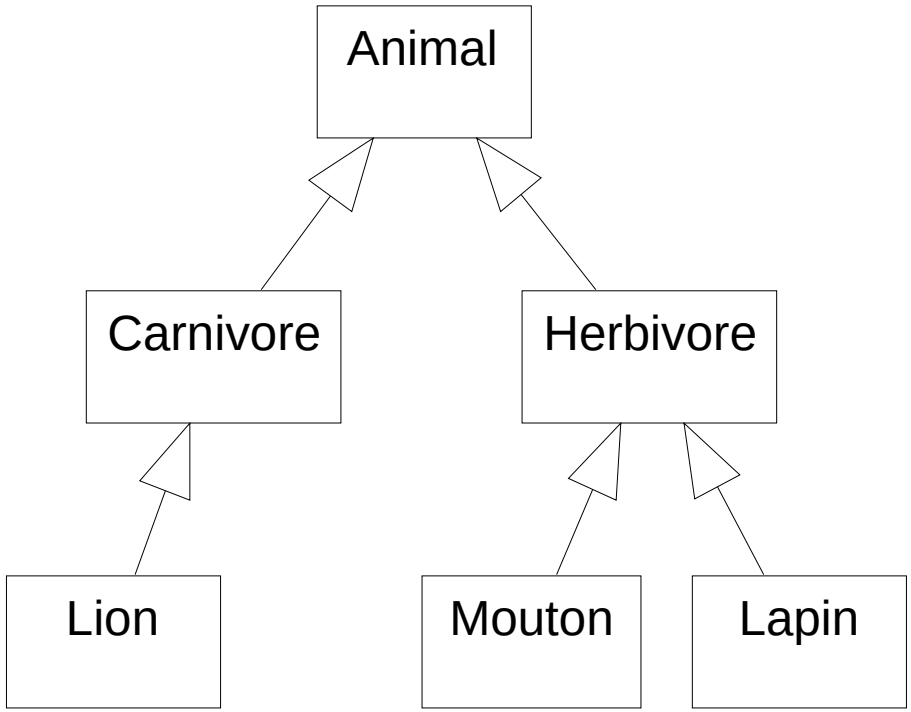
Spécialisation

- Extension cohérente d'un ensemble de classes



Propriétés de la généralisation

- Signifie toujours : *est un* ou *est une sorte de*

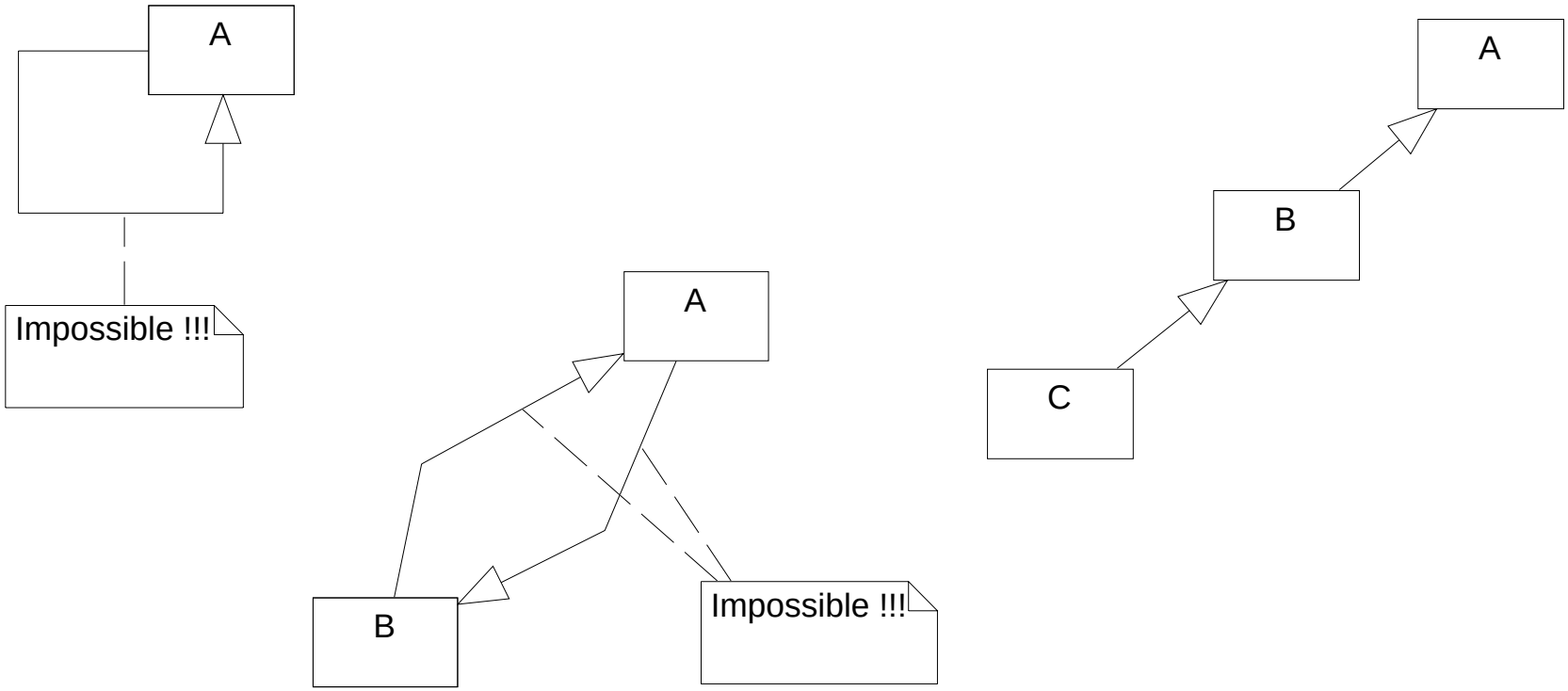


Principe de substitution

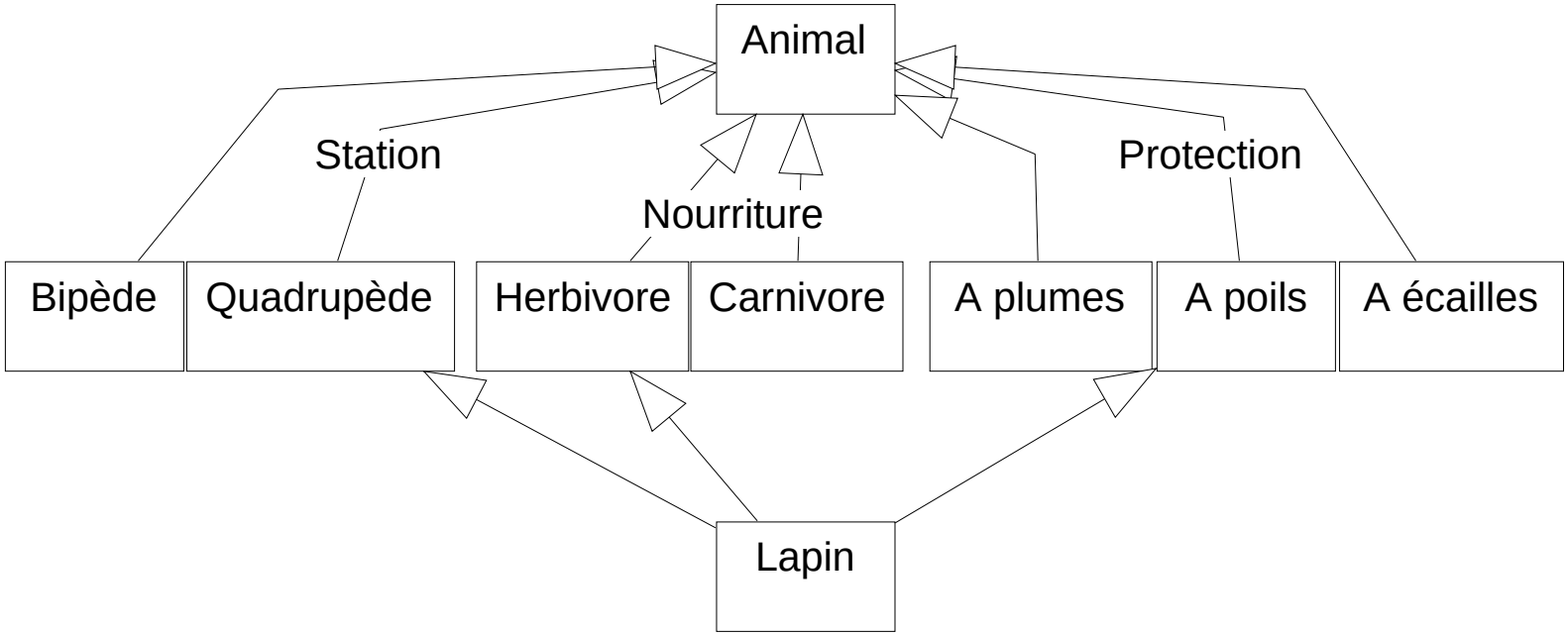
- *Il doit être possible de substituer n'importe quel objet instance d'une sous-classe à n'importe quel objet instance d'une super-classe sans que la sémantique du programme (du modèle) écrit dans les termes de la super-classe ne soit affectée.*

Propriétés de la généralisation

- Non-réflexive, non-symétrique, transitive



Généralisation multiple

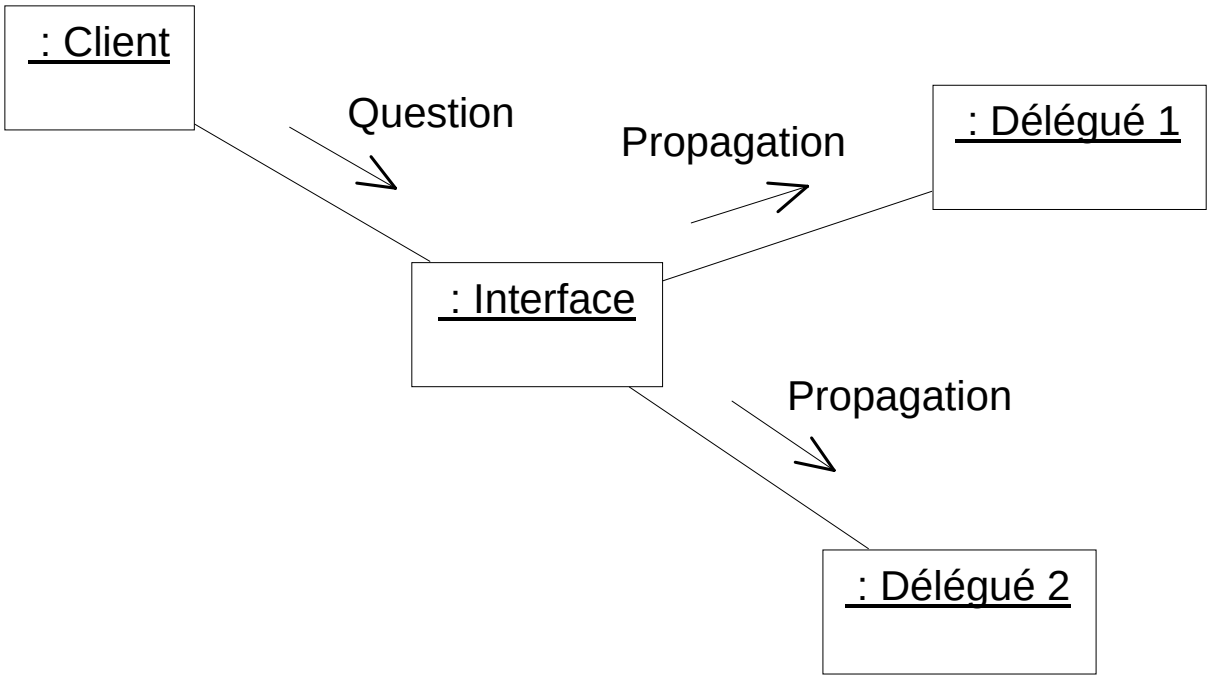


L’héritage

- Technique la plus utilisée pour réaliser la généralisation
- Construire une classe à partir d’une ou plusieurs autres classes, en partageant des attributs, des opérations et des contraintes, au sein d'une hiérarchie de classes.

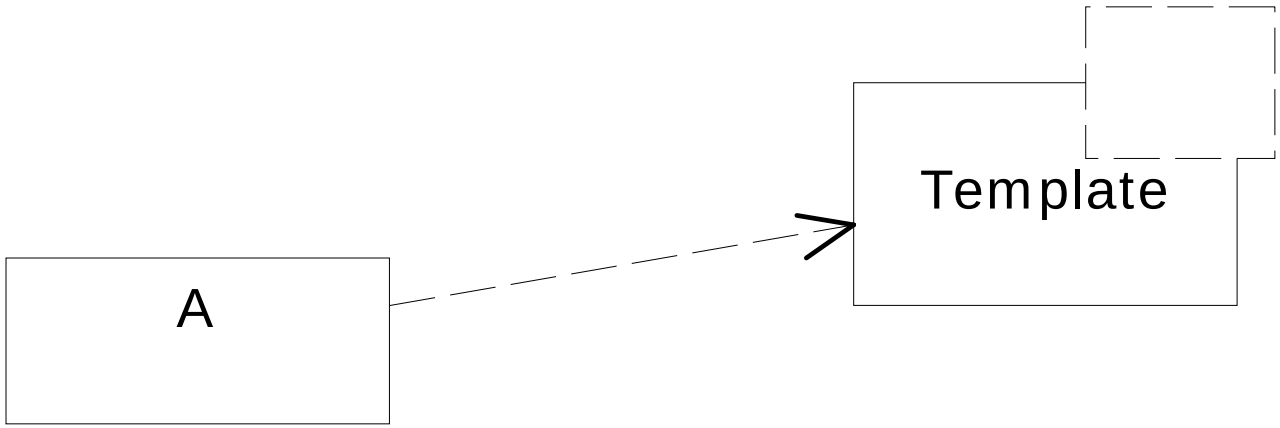
La délégation

- Remplacement de l'héritage



La dépendance

- Relation à faible contenu sémantique
- Exprime des relations d ’obsolescence
  - `Include, With, Instanciation`



Résumé

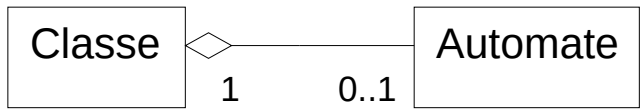
- Les classes sont connectées par des relations
  - L’association exprime une connexion sémantique bidirectionnelle
  - L’agrégation est une forme d’association plus forte
  - La généralisation permet d’ordonner les objets au sein de hiérarchies de classes
  - La dépendance exprime une relation d ’obsolescence

Expression abstraite du comportement

- Les automates à états

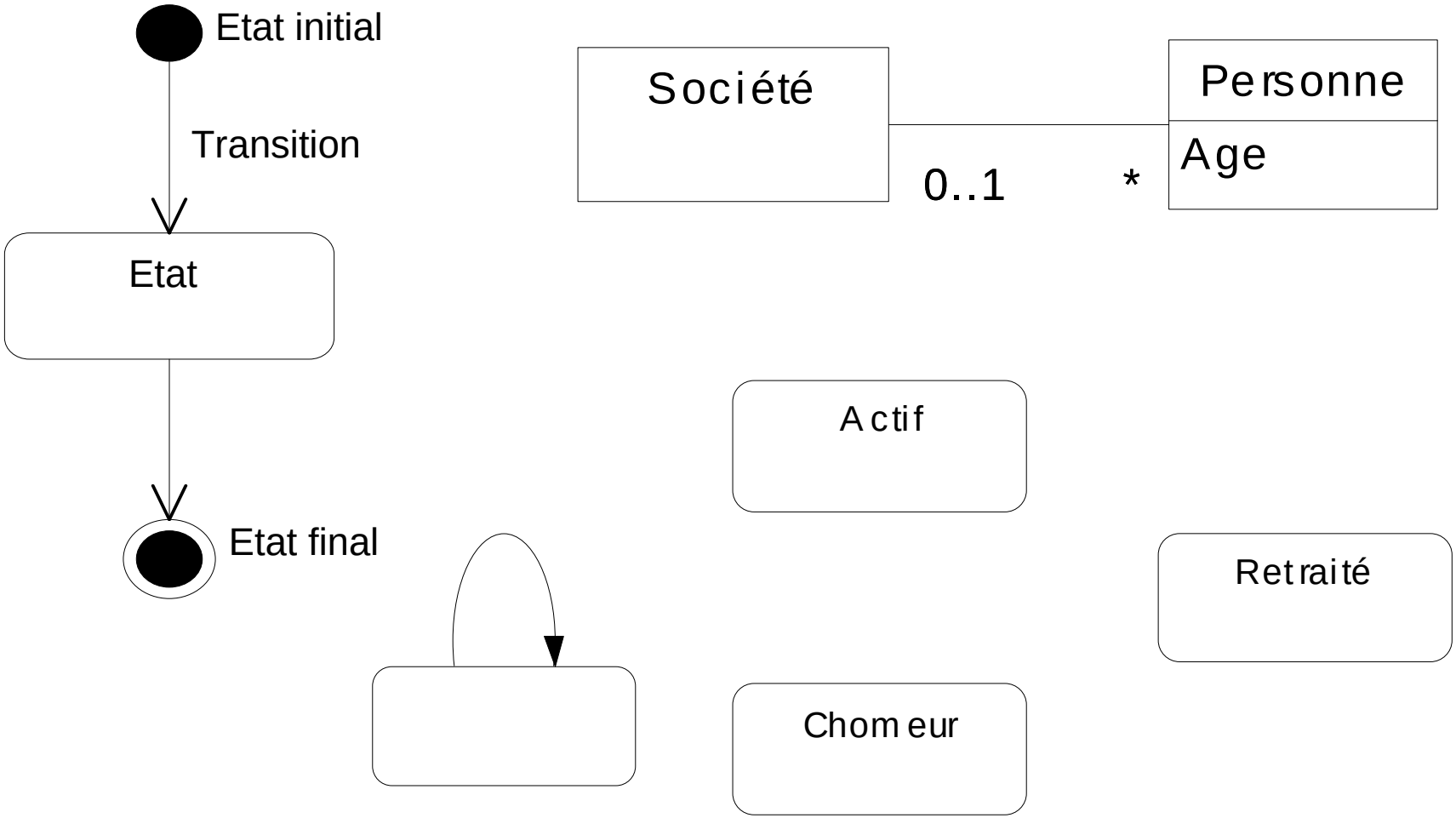
Automates d'états-finis

- Abstraction des comportements possible
- Représentation formelle du comportement d'un groupe d'objets
  - Classe
  - Collaboration
  - Cas d'utilisation

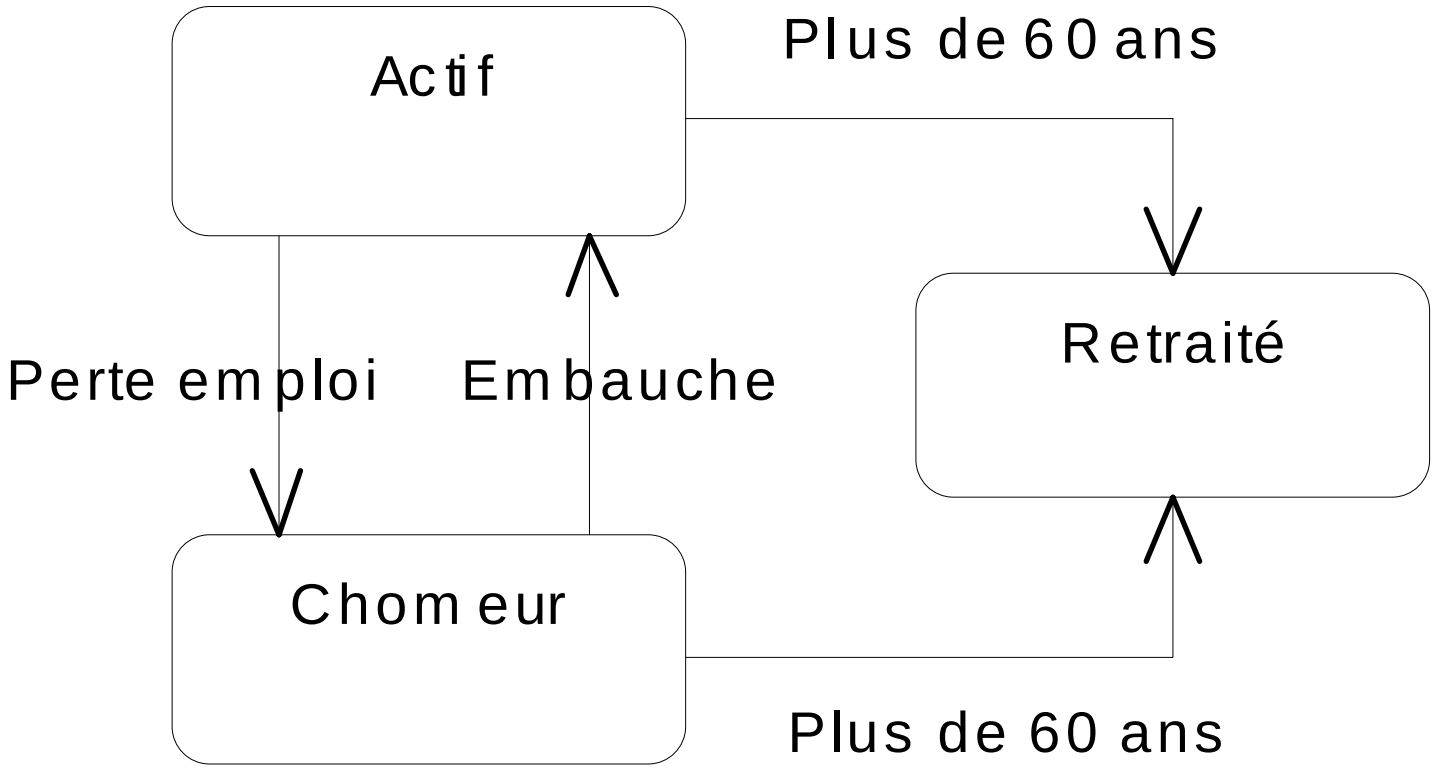


- Formalisme des `Statecharts` de David Harrel

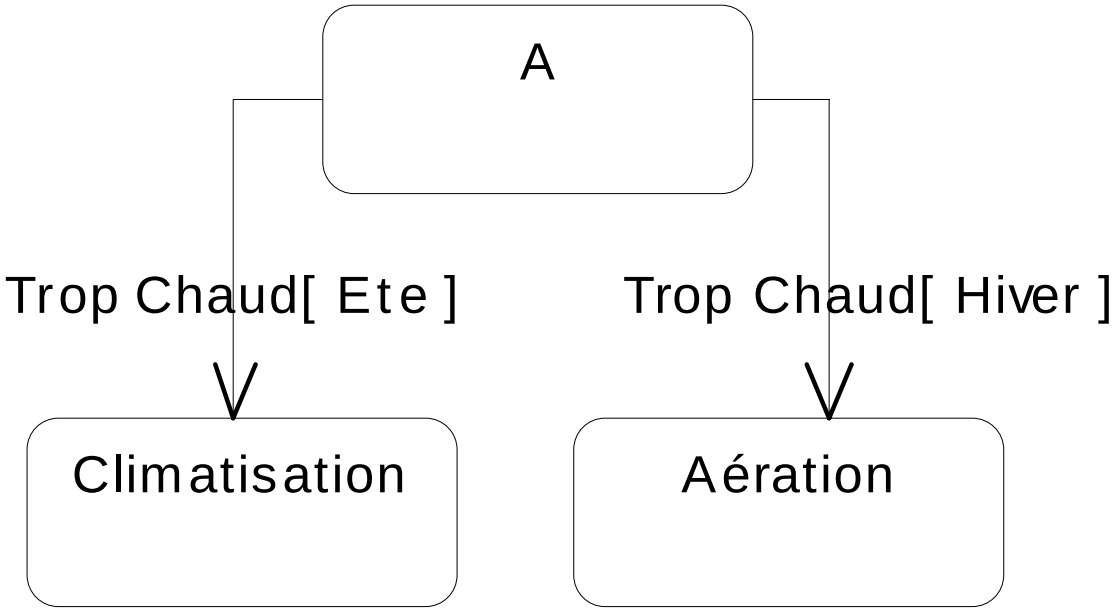
Les états et les transitions



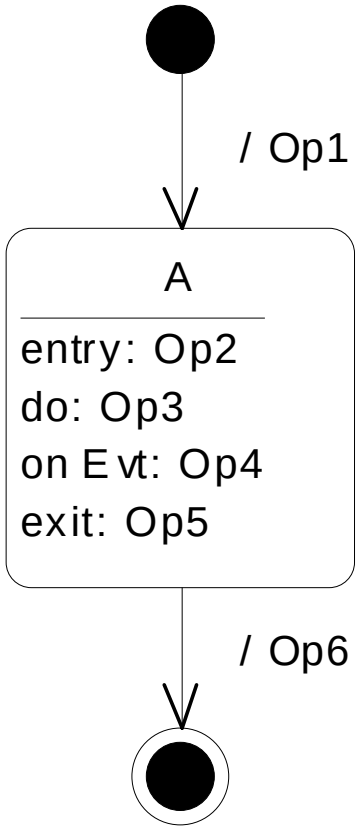
Les événements

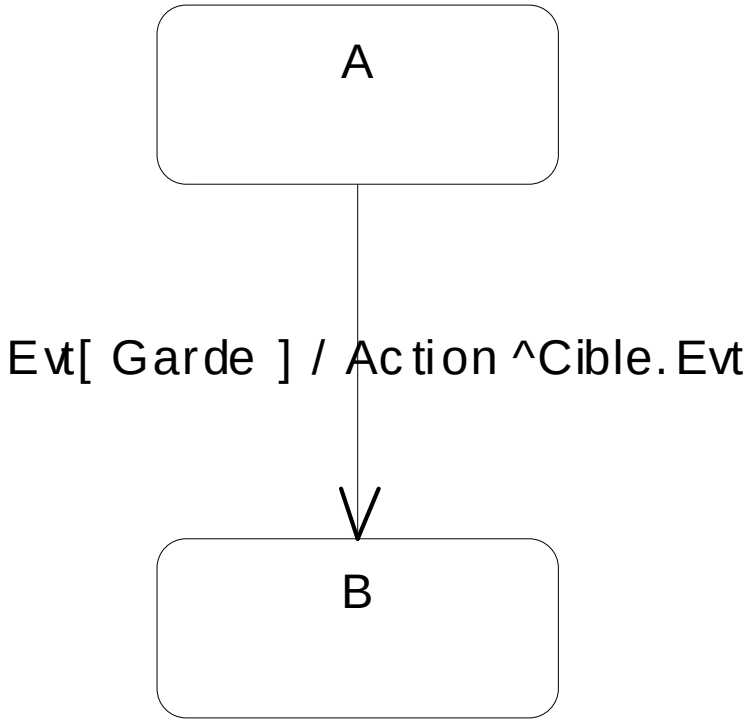


Les gardes



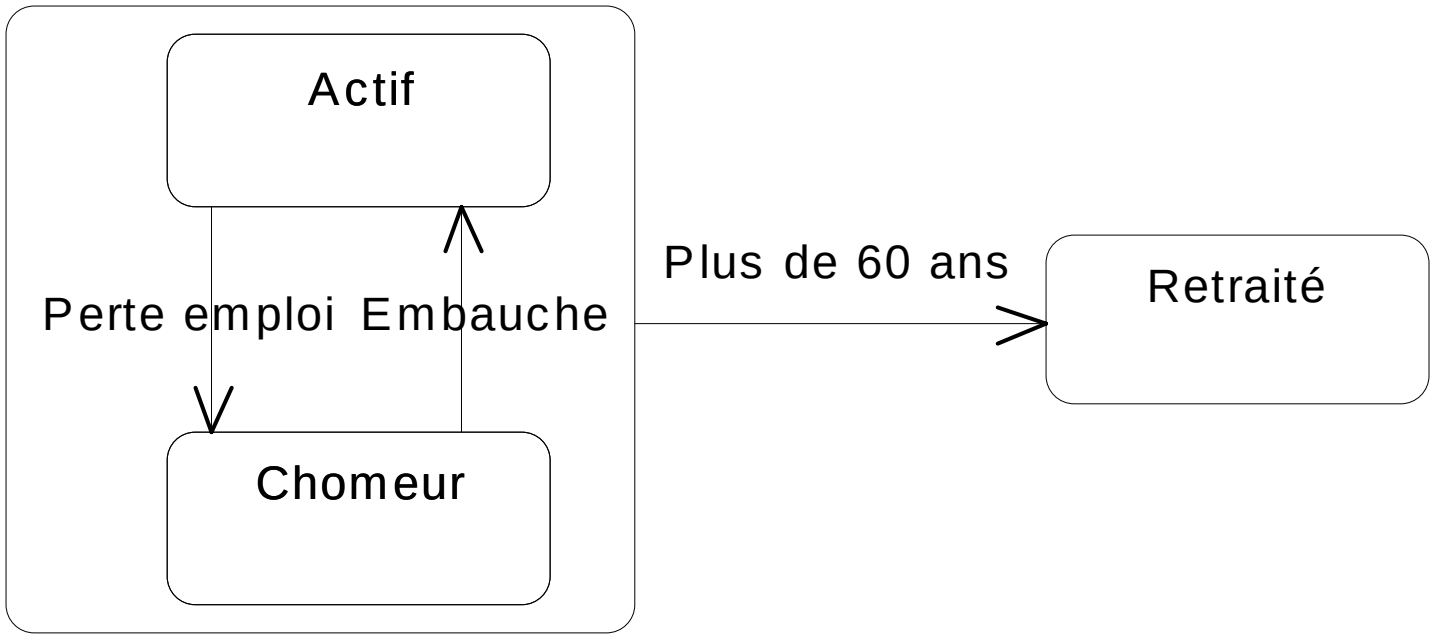
Les actions et les activités



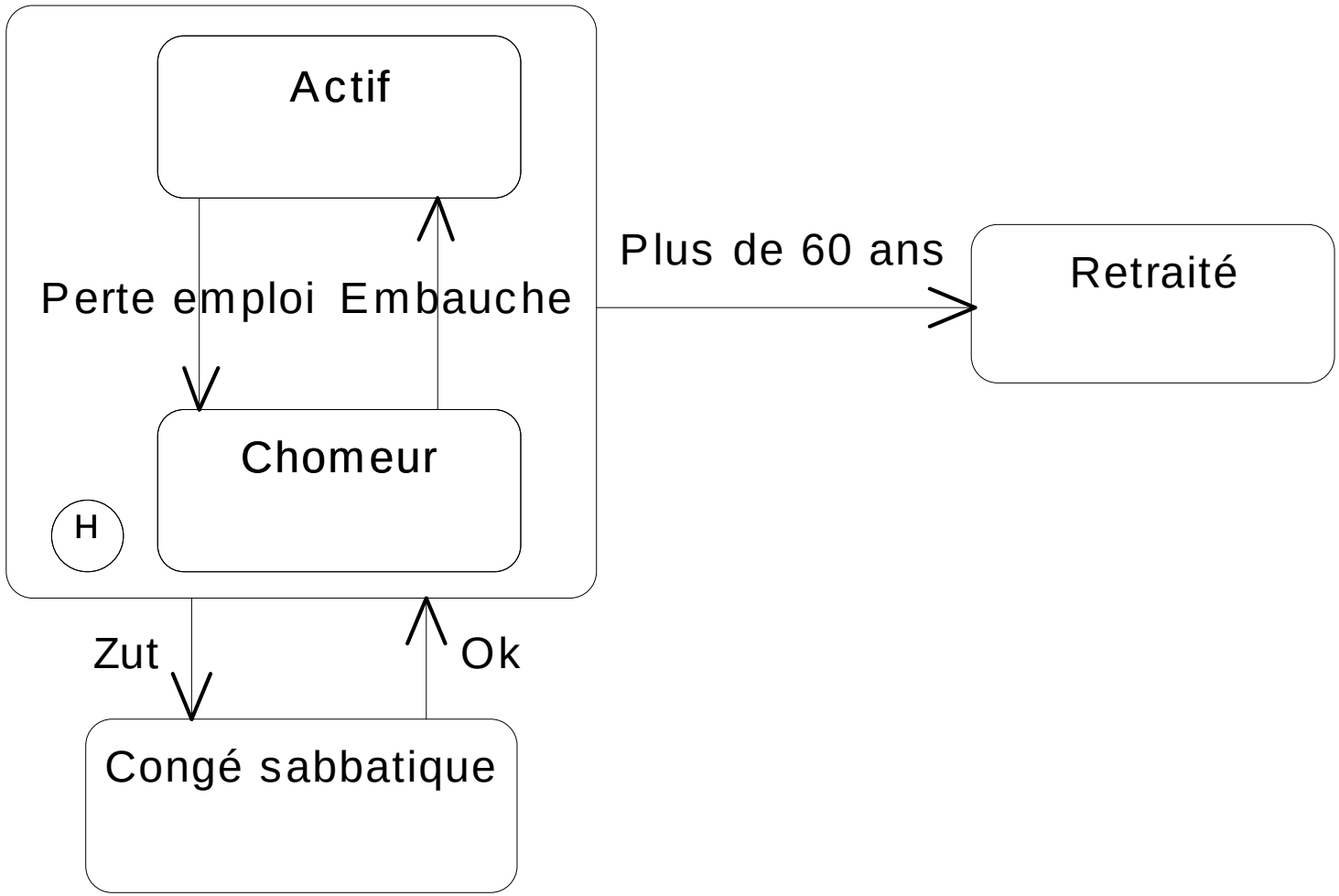


Automates hiérarchiques

- Généralisation d'état



Automate à mémoire



Expression de la réalisation et du déploiement

- Les composants
- Les nœuds

Les diagrammes de composants

- Description des éléments physiques et de leurs relations dans l'environnement de réalisation

Sortes de composants

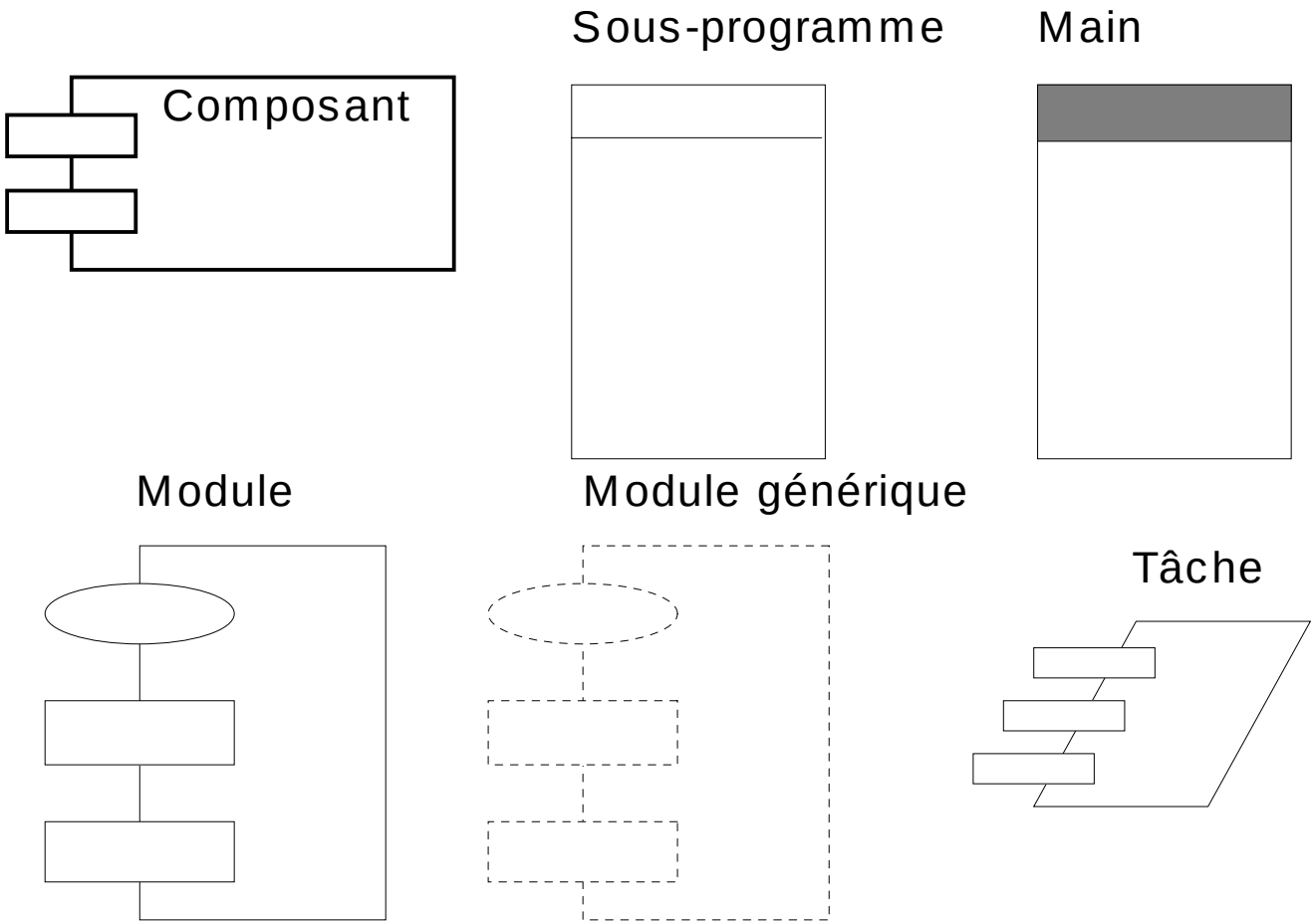


Diagramme de composants

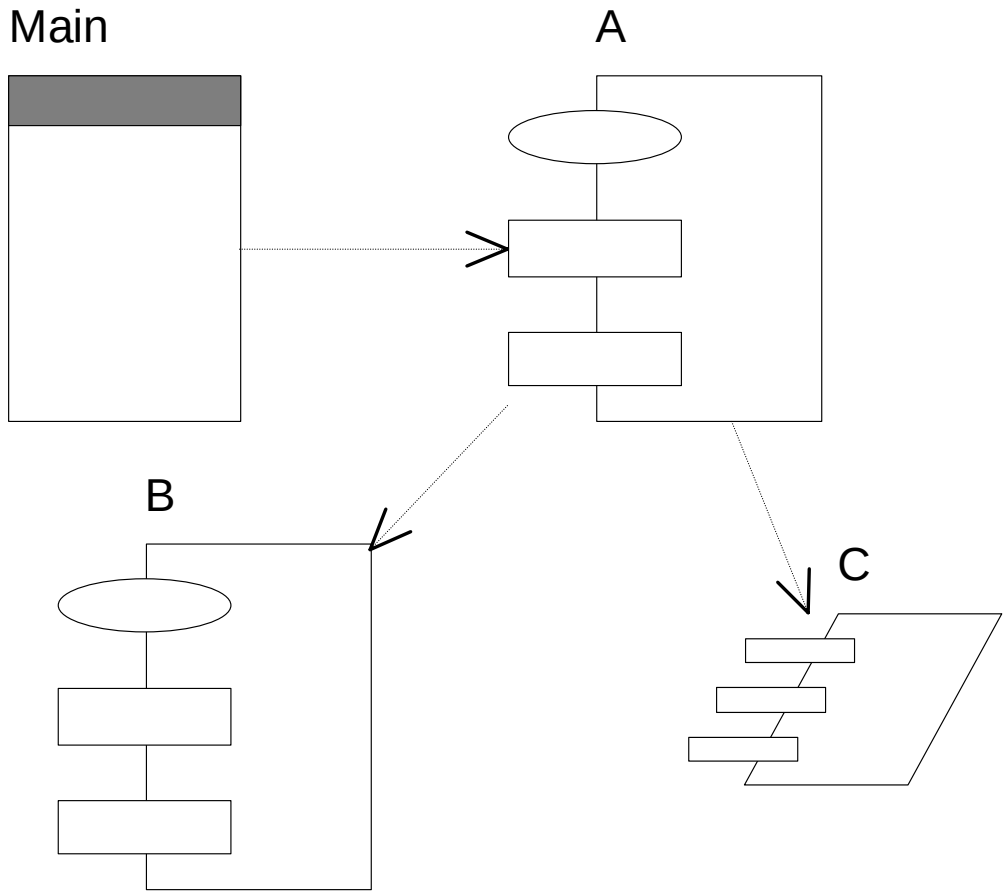
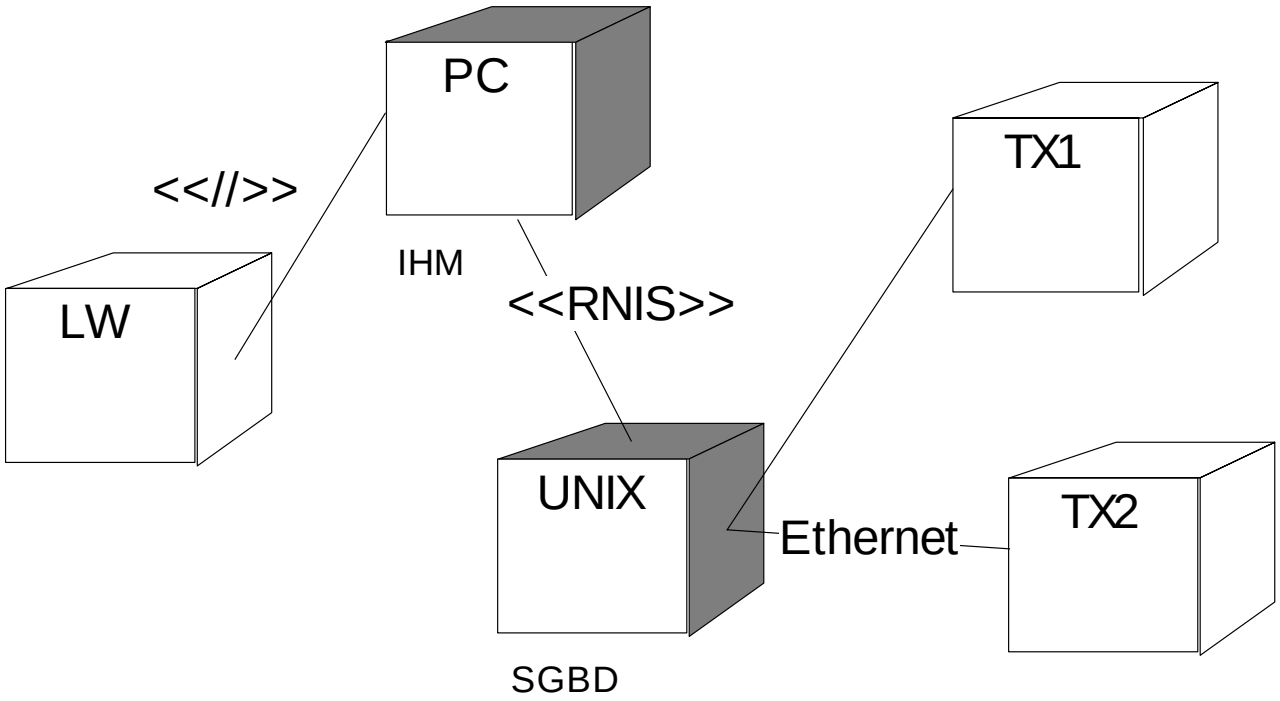


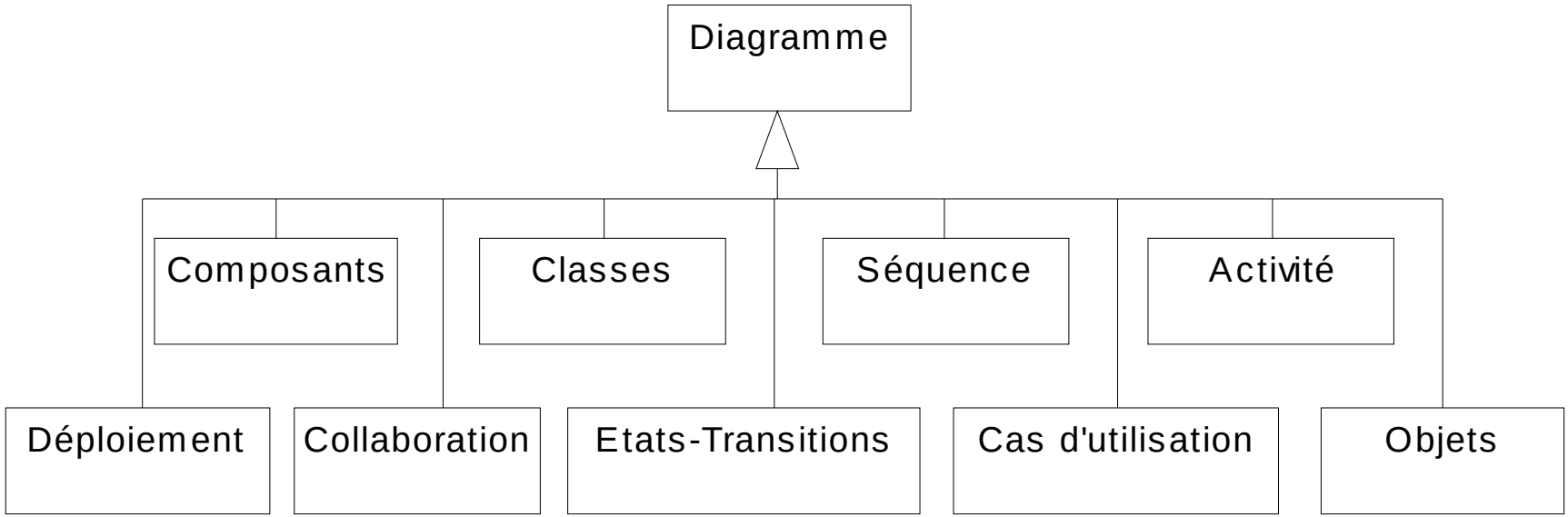
Diagramme de déploiement



Résumé

- Neuf types de diagrammes
- Articulation entre les différents diagrammes

Les diagrammes d ’UML



Articulation des diagrammes

- Plusieurs enchaînements possibles
  - Activités + Cas d ’utilisation
  - Séquence (acteur et système)
  - Collaboration (objets du domaine)
  - Classes + Automates
  - Composants
  - Déploiement

Conclusion

- UML est une notation normalisée, riche, adaptée pour la représentation des modèles objets
- UML ne normalise pas la démarche
- Il y a beaucoup de diagrammes, il faut apprendre à faire le tri