UML
Unified Method Language
-
Langage unifié pour la modélisation objet

-

Frédéric Julliard
Université de Bretagne Sud
UFR SSI - IUP Vannes

-

année 2001-2002

UML : *Unified Method Language*

- Fondations issues de diverses méthodes OO :
  - Rumbaugh (*OMT*)
  - Booch (*OOD*)
  - Jacobson (OOSE)
- Il s'agit d'un **Langage** de Modélisation

Bibliographie

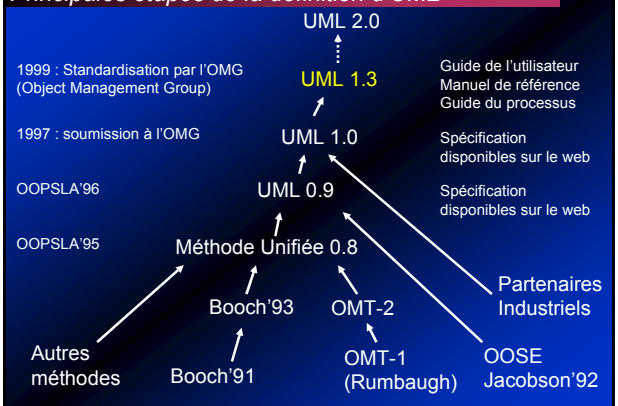
- N. Lopez, J. Migueis et E. Pichon. (1997). Intégrer UML dans vos projets. Eyrolles.
- M. Bouzeghoub, G. Gardarin, P. Valduriez (1997) Les objets. Eyrolles.
- N. Kettani *et al.* (1999). De Merise à UML. Eyrolles
- I. Jacobson (1993) Le génie logiciel orienté objet : une approche fondée sur les cas d'utilisation. ACM Press, Addison-Weslry
- P-A Muller et N. Gaertner (2000) Modélisation objet avec UML. Eyrolles.
- M. Lai (2000). UML la notation unifiée de modélisation objet. Dunod Informatiques.

- Cours sur le web : http://uml.free.fr
- Site : www.uml.org (OMG)

Chapitre I

Introduction

Principales étapes de la définition d'UML



Principales influences

- **Booch** : catégories et sous-systèmes
- Embley : classes singletons et objets composites
- **Fusion** : description des opérations, numérotation des messages
- Gamma : frameworks, patterns et notes
- Harel : machines à états finis (statecharts)
- Jacobson : cas d'utilisation (use cases)
- Meyer : pré et post-conditions
- Odell : classification dynamique, évenements
- **OMT** : associations
- Shlear-Mellor : cycle de vie des objets
- Wirfs-Brock : responsabilités (CRC)

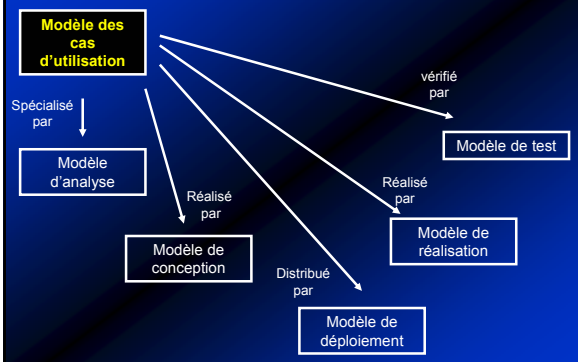
Objectifs du langage UML

- Langage visuel de modélisation
  - Exploitable par des méthodes A/C différentes
  - Adapté à toutes les phases du développement
  - Compatible avec toutes les techniques de réalisation
- Mécanismes d'extension et de spécialisation en vue d'étendre les concepts de base
- Indépendant des langages de programmation
- Base formelle pour la compréhension du langage
- Encourage l'utilisation d'outils OO
- Supporte les concepts de développement de haut niveaux : patterns, composants et frameworks

Diagrammes d'UML

- Différentes vues pour représenter un système :
- En UML : 9 principaux diagrammes (en réalité : 12)
  - 5 Diagrammes **structurels** (vue ***statique***)
    - Cas d'utilisation
    - Classes
    - Objets
    - Composants
    - Déploiement
  - 4 diagrammes **comportementaux** (vue ***dynamique***)
    - Séquence
    - Activités
    - Etats-Transitions
    - Collaboration

Processus d'Ingénierie sous-jacent



Relation entre diagrammes et étapes du processus

- ***Découverte des besoins*** :
  - **Diagramme de cas d'utilisation** : décrit les fonctions du système selon le point de vue ses futurs utilisateurs (Jacobson)
  - **Diagramme de séquence** : représentation des interactions temporelles entre objets dans la réalisation d'une interface Homme-Système
- ***Analyse*** :
  - **Diagramme de classes** : structure des données du système définies comme un ensemble de relations entre classes
  - **Diagramme d'objets** : illustration des objets et de leur relations
  - **Diagramme de collaboration** : représentation des interactions entre objets
  - **Diagramme d'états-transitions** : représentation du comportement des objets d'une classe en terme d'états et de transitions d'états
  - **Diagramme d'activités** : structure d'une opération en actions

Relation entre diagrammes et étapes du processus

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

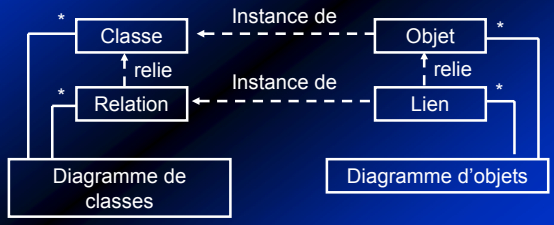
Chapitre II

Diagrammes de classes et diagrammes d'objets

Diagrammes de classes et diagrammes d'objets

- Un **diagramme de classes** représente la structure du système sous la forme de **classes** et de **relations** entre ces classes

- Un **diagramme d'objets** **illustre les objets et les liens** qui les unissent



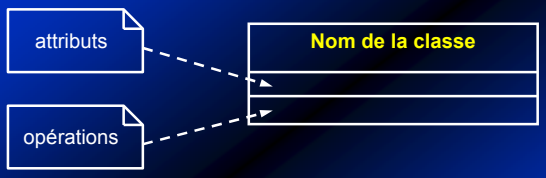
(extrait simplifié du méta-modèle d'UML)

Classe

- Une **classe** est une description abstraite d'un ensemble d'objets ayant :

  - des *propriétés* similaires,
  - un *comportement* commun,
  - des *relations* communes avec d'autres objets
  - des *sémantiques* communes

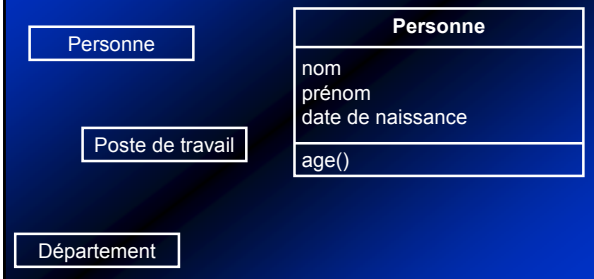
Représentation d'une classe en UML



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

Représentation d'une classe en UML

Exemples :



Attributs de classe

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

Nom de classe
nom d'attribut : type = valeur initiale

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

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

L'identification d'un objet est donc facultative.

Attributs de classe

Exemples :

Personne
nom : chaîne prénom : chaîne date de naissance : date

Film
Titre : chaîne réalisateur : chaîne Date de production : date Date de sortie : date

Opération de classe

Exemples :

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

Nom de classe
nom d'opération (liste d'arguments) : type de retour

Opérations de classe

Exemples :

Personne
nom : chaîne prénom : chaîne date de naissance : date
Age() ChangeAdresse()

Objet géométrique
couleur : chaîne position : entier
Déplacer (deltaX : vecteur) Sélectionner (p : Point) : Booléen

Propriétés des attributs et des opérations :

Accessibilité aux attributs et opérations d'une classe

Trois **niveaux** de **protection** :

Public (+) : accès à partir de toute entité interne ou externe à la classe

Protégé (#) : accès à partir de la classe ou des sous-classes

Privé (-) : accès à partir des opérations de la classe

Propriétés des attributs et des opérations :

Accessibilité aux attributs et opérations d'une classe

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

Attributs dérivés

Au niveau de l'analyse des besoins, des propriétés redondantes peuvent être proposées…

Un attribut dérivé permet d'indiquer clairement qu'un attribut découle d'autres propriétés allouées

Les attributs dérivés (noté : **/nom attribut**) ont des valeurs calculées à partir de celles d'autres propriétés :

Rectangle
Longueur Largeur /surface

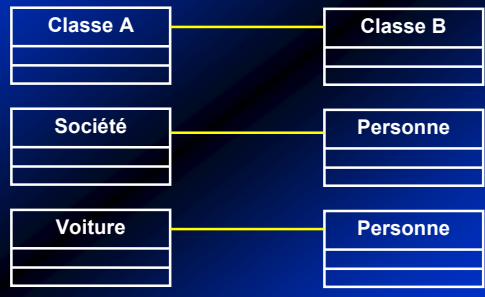
En conception ⇢

Rectangle
Longueur Largeur

Par la suite : Surface()

Association

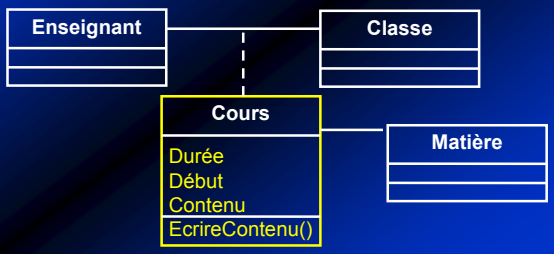
Une **association** représente une classe **d'associations structurelles** entre classes d'objets



Classe association

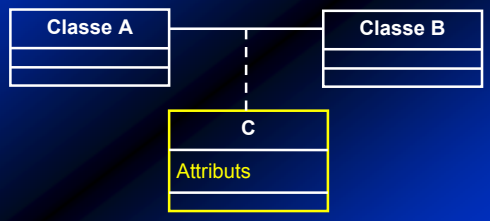
- Une **association** peut être réifiée par une classe appelée **classe associative** ou **classe association**

- Par exemple, lorsque l'association possède des attributs ou des opérations : – – – – – Rattachement de la classe à l'association



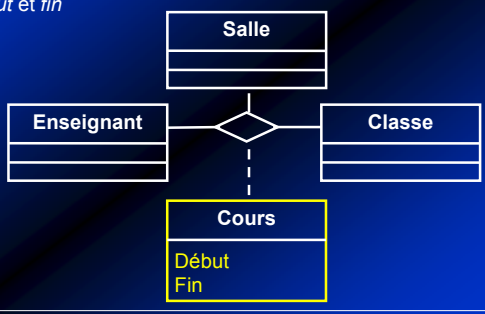
Association

Une association qui contient des attributs et qui ne participe pas à des relations avec d'autres classe est appelée **classe attribuée**.



Association n-aire

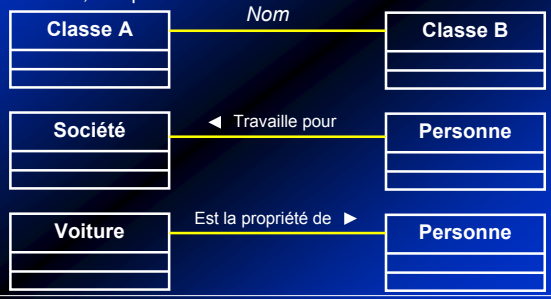
Une association **ternaire** entre salle, étudiant et enseignant est réifiée comme une classe *cours* ayant deux attributs : *début* et *fin*



Nommage des associations

- Nom de l'association en italique au milieu de la ligne

- On note en général les association par une forme verbale, soit active, soit passive

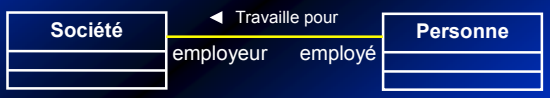


Nommage des rôles

- Toute association binaire possède 2 rôles

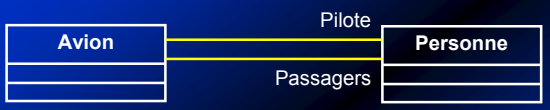
- un rôle définit la manière dont une classe intervient dans une relation

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

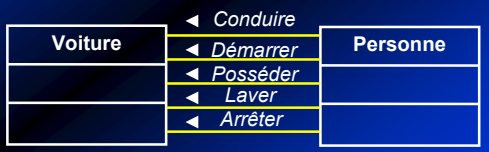


- Intérêt des rôles dans le cas où plusieurs associations lient deux classes : distinction des concepts attachés aux associations

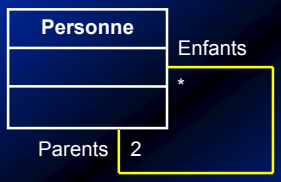
Nommage des rôles



La présence d'un grand nombre d'associations entre deux classes est suspecte :



Association réflexive



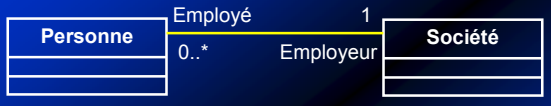
Nommage des rôles indispensable à la clareté du diagramme

Multiplicité des associations

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

1	**un** et **un seul**
0 .. 1	zéro **ou un**
M .. N	**de M à N (entiers naturels)**
*	**de zéro à plusieurs**
0 .. *	**de zéro à plusieurs**
1 .. *	**de un à plusieurs**
N	**exactement N (entier naturel)**

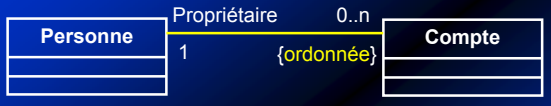
Multiplicité des associations



1 : Chaque personne travaille pour une et une seule société (toute les personnes ont un emploi)

0 .. * : Une société emploie de zéro à plusieurs personnes

Contraintes sur les associations

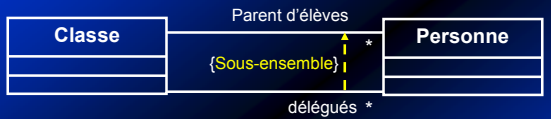


Contrainte d'association : porte sur une relation ou sur un groupe de relations (notée {contrainte })

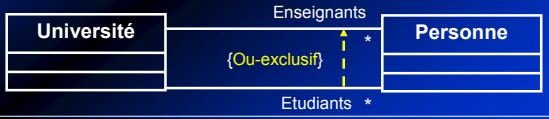
Par exemple, placée sur un rôle, la contrainte **{ordonnée}** définit une relation d'ordre entre les objets de la collection (les comptes) qui sont liés à une personne

Contraintes sur les associations

La contrainte **{sous-ensemble}** indique qu'une collection est incluse dans une autre collection



La contrainte {Ou-exclusif} précise que, pour un objet donné, une seule association parmi un groupe d'associations est valide



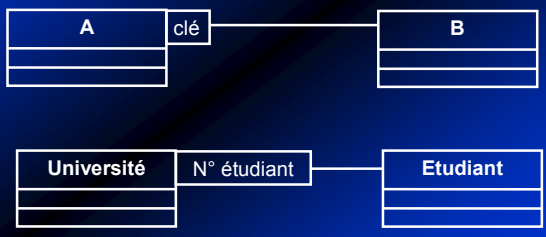
Restriction des associations

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

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

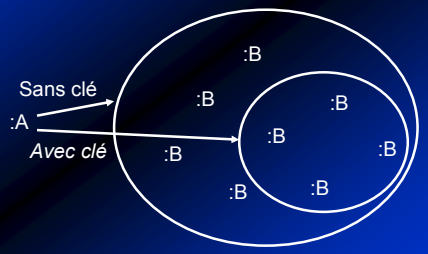
Restriction des associations

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



Restriction des associations

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



Association particulière : Agrégation

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

Elle se justifie dans les cas suivants **:**

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

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

Association particulière : Agrégation

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

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

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

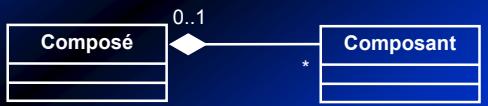
Agrégation particulière : Composition

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

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

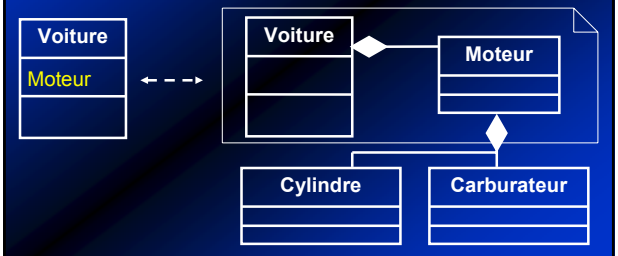
La **composition** implique une **contrainte** sur la valeur de la **multiplicité** du coté de l'agrégat : (0 ou 1)

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

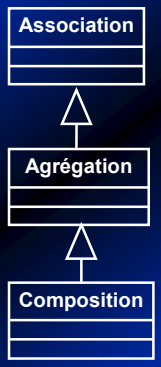


Agrégation particulière : Composition

- La **composition** peut être modélisée au moyen d'attributs
- La notation par **composition** doit être retenue lorsque d'un attribut participe à des relations

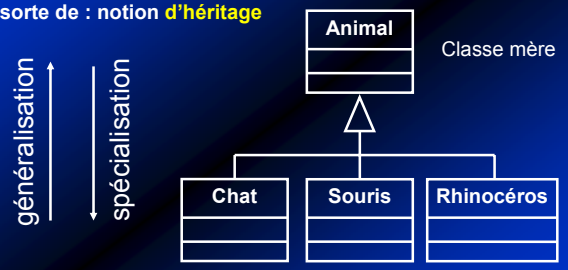


Association, Agrégation et Composition



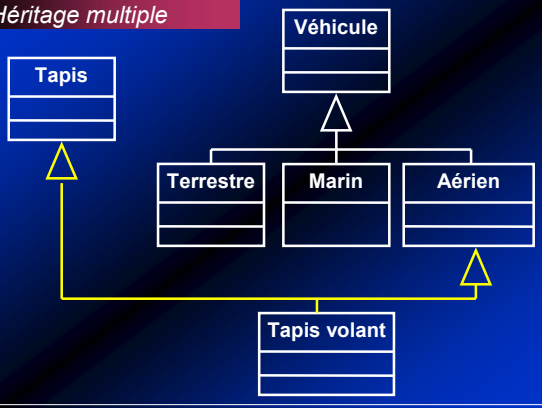
Généralisation - Spécialisation

- La relation de généralisation signifie **est de** ou **est une sorte de : notion d'héritage**



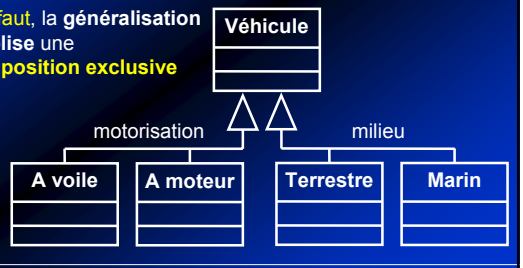
Sous-classes = instances d'une **seule classe Animal** ***(héritage simple)***

Héritage multiple



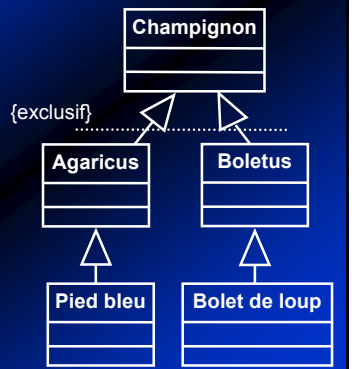
Contraintes de généralisation

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



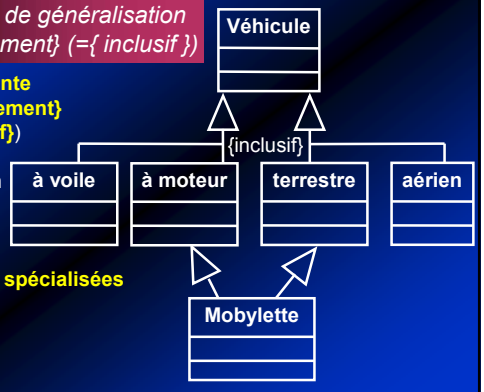
Contraintes de généralisation {disjoint} (= { exclusif })

- La **contrainte {Disjoint}** (ou **{exclusif}**) indique la participation **exclusive d'un objet** à l'une des **collections spécialisées**



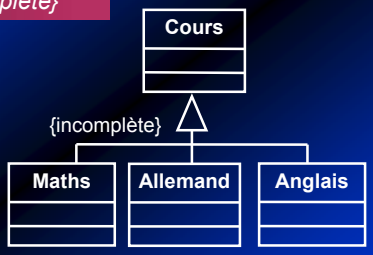
Contraintes de généralisation {chevauchement} (={ inclusif })

- La **contrainte {chevauchement}** (ou **{inclusif}**) indique la participation possible **d'un objet** à plusieurs **collections spécialisées**



Contraintes de généralisation {Complète} ≠ {Incomplète}

- La **contrainte {Complète}** indique la **généralisation** est **terminée** : tout ajout de sous-classe est alors impossible



- à l'inverse, la contrainte **{Incomplète}** indique une **généralisation extensible**

Diagramme d'objets

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

Nom de l'objet

Nom de l'objet:NomClasse

:NomClasse

Diagramme d'objets

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

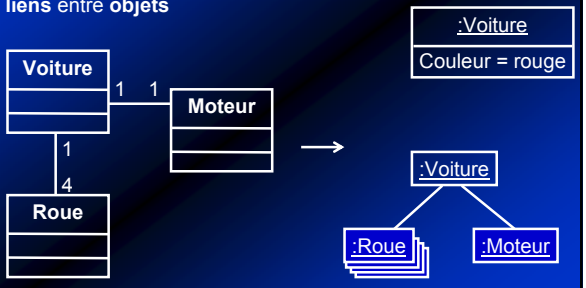


Diagramme d'objets

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

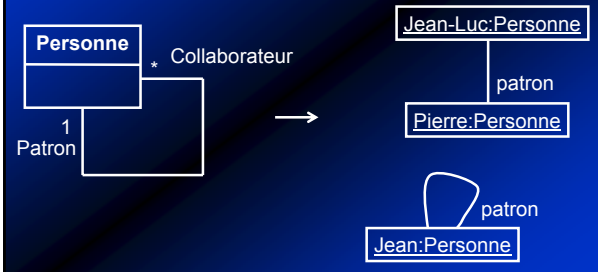


Diagramme d'objets : liens entre objets

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

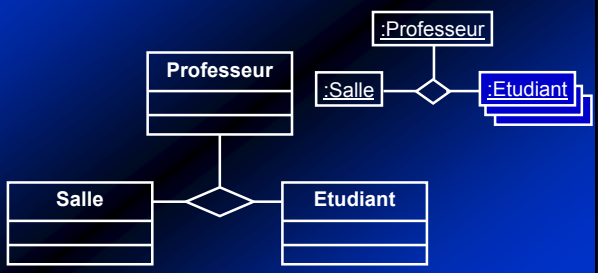


Diagramme d'objets : liens entre objets

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

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

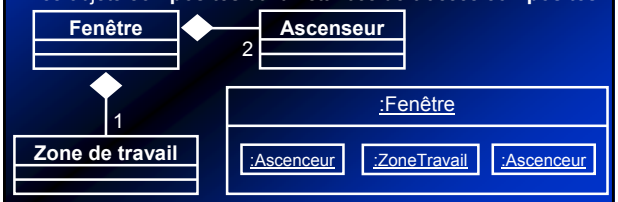


Diagramme d'objets : structures complexes

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

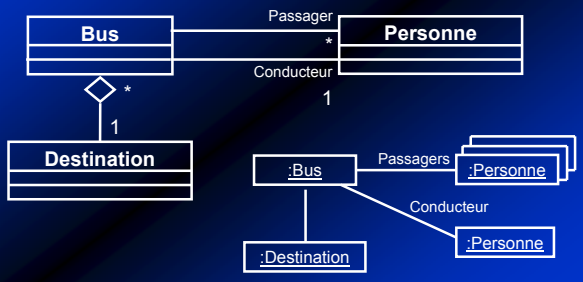
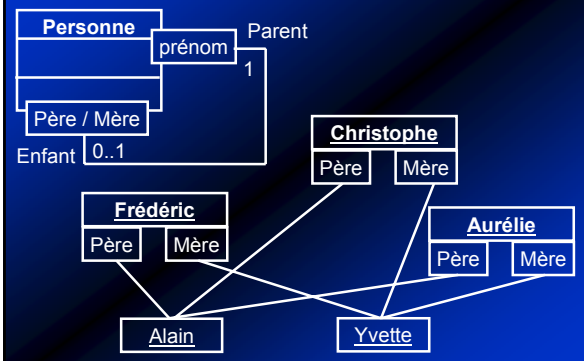


Diagramme d'objets : structures complexes



Chapitre III

Diagrammes de Cas d'utilisation (use cases)

Représenter les besoins

- La phase **d'analyse des besoins** nécessite :
  - de **comprendre** les **besoins** à **couvrir**
  - **d'exprimer** et de **formaliser** les **besoins**
- **Moyens** pour **représenter les besoins en UML** →
  - **Diagramme de cas d'utilisation** : organisation générale utilisation système par ses acteurs
  - **Diagramme de séquence** : pour chaque cas d'utilisation : description temporelle de l'interaction d'un acteur sur le système = scenario
  - **Diagrammes objets/classes** : informations échangées entre système et acteurs
  - **Diagramme de collaboration** : interactions entre objets métier du système

Cas d'utilisation

Constat : Le système **existe** pour **servir** ses **utilisateurs**

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

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

Comportement = {**Actions**}+{**Réactions**}

Les Cas d'utilisation :

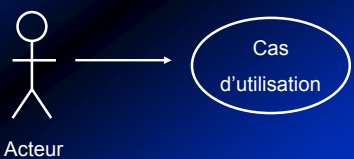
→ **facilitent** la **structuration** des **besoins** des utilisateurs

→ **représentation simple** et **expressive**

→ **expriment** les **limites** et les **objectifs** du système

Cas d'utilisation

- Un **cas d'utilisation** correspond à une manière **spécifique** d'utiliser le système
- C'est la **représentation** d'une **fonctionnalité**, **déclenchée** en réponse à une **stimulation** du système



Cas d'utilisation : définitions

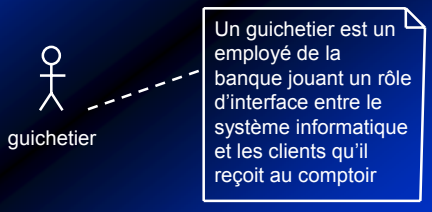
- **Acteur** : entité externe qui agit sur le système
  - prend les décisions contrairement à un élément logiciel
  - possède un rôle par rapport au système
  - soit utilisateur
  - soit un autre système

Acteurs vs utilisateurs

- **Ne pas confondre acteur et personne utilisant le système :**
  - une même personne peut jouer plusieurs rôles
  - plusieurs personnes peuvent jouer un même rôle
  - un acteur n'est pas forcément une personne physique…
- **Types d'acteurs** :
  - Utilisateur **principaux**
  - Utilisateurs **secondaires**
  - **Périphériques externes**
  - **Systèmes externes**

Définition des Acteurs

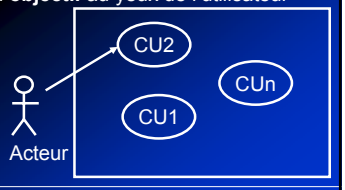
- Pour chaque acteur :
  - **choisir un identificateur représentatif du rôle**
  - **éventuellement accompagné** d'une brève description textuelle :



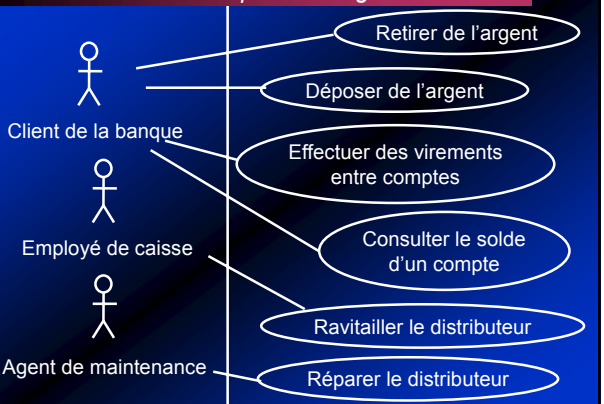
Cas d'utilisation : définitions

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

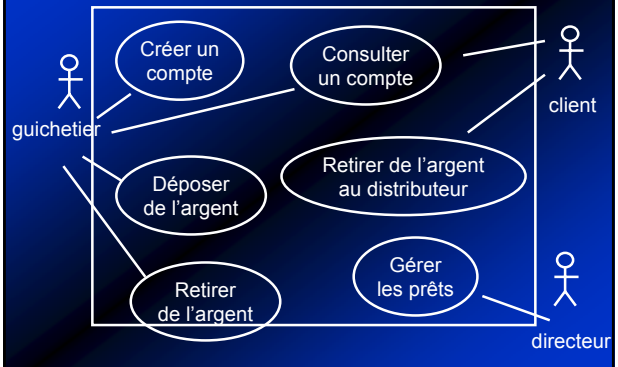
- suite **d'interactions** entre un **acteur** et le **système**
- correspond à une fonction **visible** par **l'utilisateur**
- permet d'atteindre un **objectif** au yeux de l'utilisateur
- doit être **utile**
- **les cas d'utilisation ne doivent pas se chevaucher**



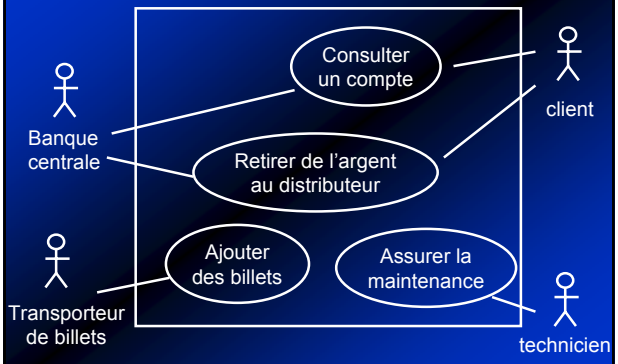
Cas d'utilisation : exemples de diagrammes de CU



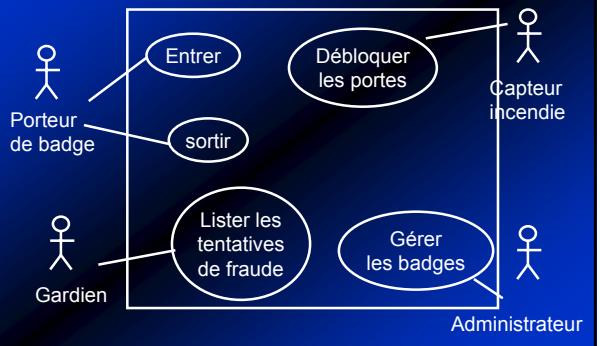
Cas d'utilisation : exemples de diagrammes de CU



Cas d'utilisation : exemples de diagrammes de CU

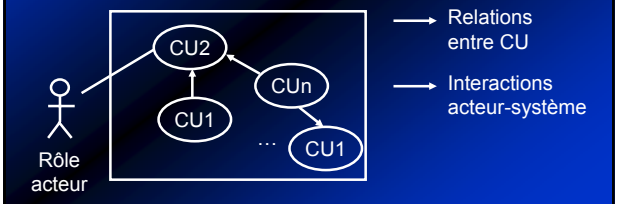


Cas d'utilisation : exemples de diagrammes de CU



Cas d'utilisation : définitions

- **Pas de communications entre les CU d'un système, mais simplement des relations d'utilisation (*uses* ou *include*) ou d'extension (*extends*)**
- Les **communications** entre les **acteurs ne sont pas représentées**



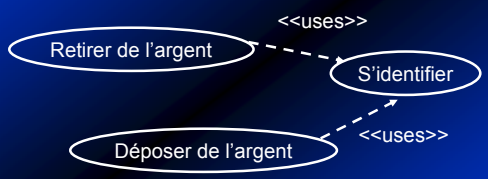
Relations entre cas d'utilisation

Relations entre cas d'utilisation : structuration des cas d'utilisation

- **l'utilisation** (*uses* ou *include*) : association entre un cas d'utilisation client et un cas d'utilisation fournisseur d'un comportement : extraction d'une séquence d'interactions communes présentes dans le scénario de plusieurs cas d'utilisation
- **l'extension** (*extends*) : extraction de **scénarii communs** ou **non** et **optionnels** (déclenchés **sous certaines conditions**)

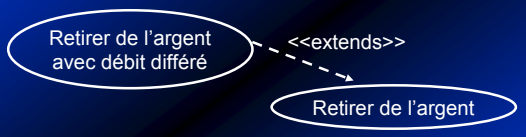
Relations entre cas d'utilisation

Relation ***uses*** entre cas d'utilisation

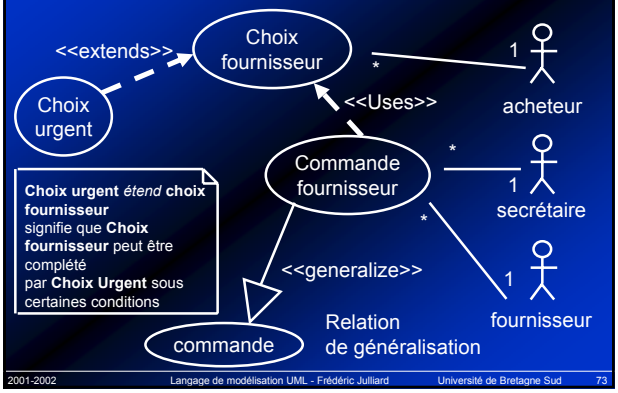


Relations entre cas d'utilisation

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



Exemple de relations



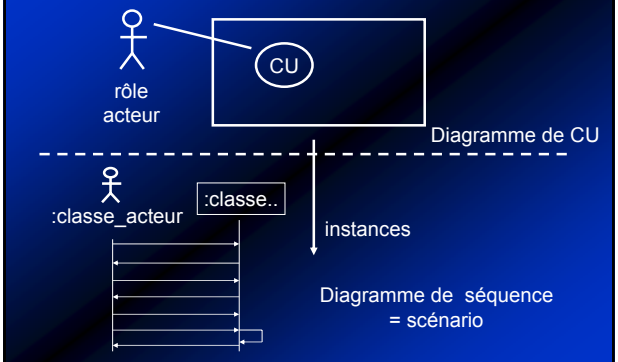
Cas d'utilisation et scénario

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

→ **Un scénario = Instance** de cas d'utilisation

→ Une **instance d'acteur créer** un **scénario**

Cas d'utilisation vs Scénario



Cas d'utilisation et scénario

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

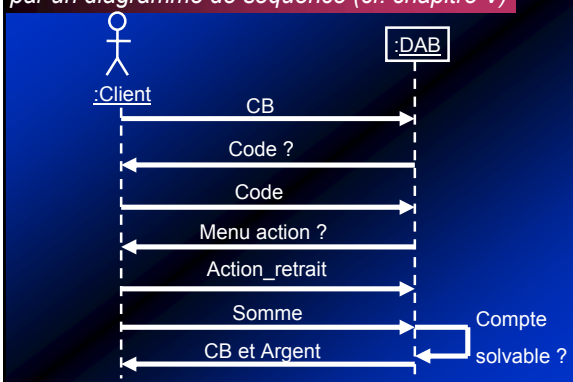
Cas d'utilisation et scénario

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

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

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

Exemple de scénario décrit par un diagramme de séquence (cf. chapitre V)



Chapitre IV

Diagrammes de collaboration

Introduction

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

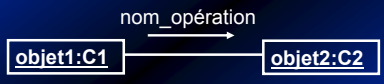
→ Décrit le **comportement collectif** d'un **ensemble d'objets**,

→ en vue de **réaliser** une **opération**

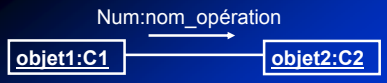
→ en **décrivant** leurs **interactions modélisées** par des **envois** (éventuellement **numérotés**) de **messages**

Envois de messages entre objets

Message = ***nom opération***



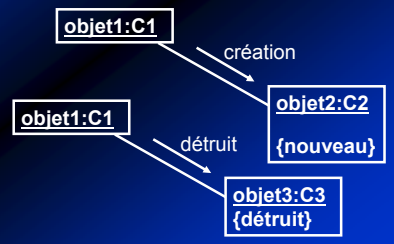
Envois éventuellement numérotés :
ordre des envois de message au cours d'une opération



Contraintes associés aux envois de message

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

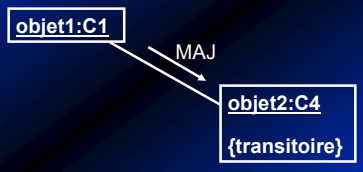
- **{Nouveau}**
- **{Détruit}**



Contraintes associés aux envois de message

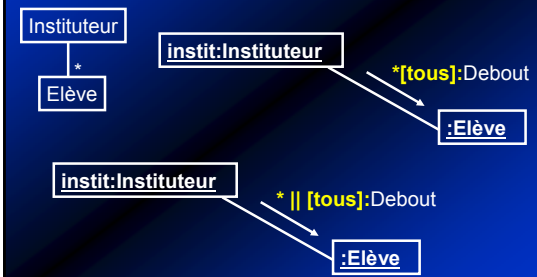
Les **objets (et les liens) crées** et **détruit** au cours de la même interaction porte la contraintes :

- **{transitoire}**



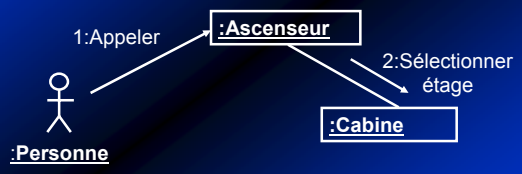
Itérations dans un diagramme de collaboration

Possibilité d'exprimer l'envoi répétitifs de messages (éventuellement en parallèle) sur une collection d'objets



Itérations dans un diagramme de collaboration

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



Itérations dans un diagramme de collaboration

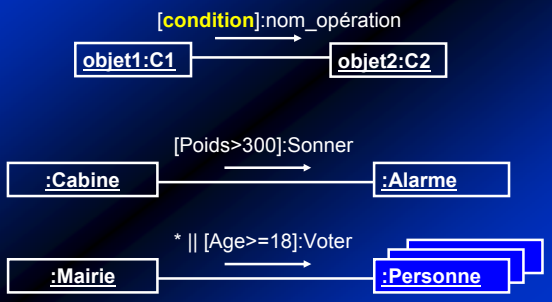
Les **objets** qui contrôlent le flot sont dits **actifs**

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

Ex : photocopieuse

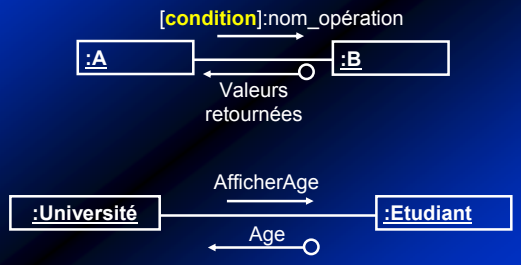
Conditions sur les envoi de message

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



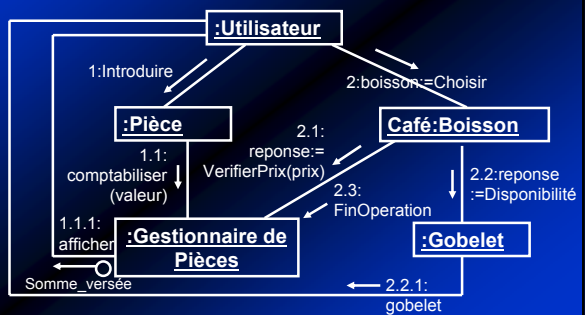
Retour d'une liste de valeurs à l'issue d'un envoi de message

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



Exemple : distributeur de boisson

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



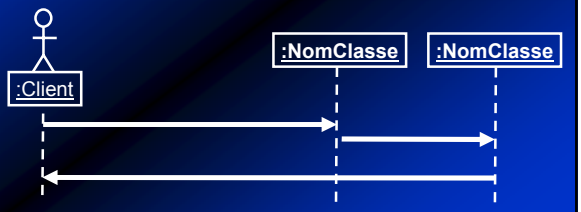
Chapitre V

Diagrammes de séquence

Introduction

Diagramme de séquence :

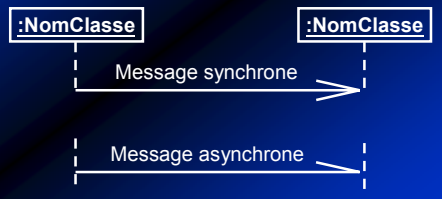
- Modélisation des **interactions entre objets** suite à un **événement externe**
- **Aspect temporel : messages asynchrones ou synchrones**



Catégories de messages

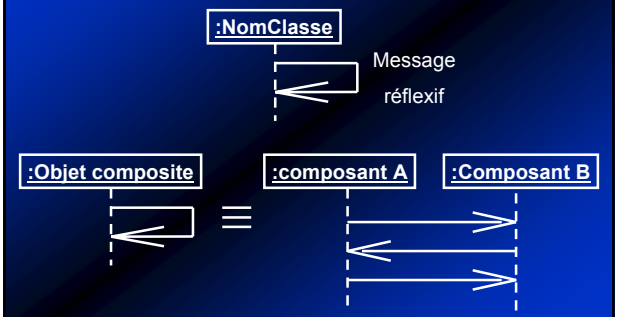
2 catégories de messages :

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

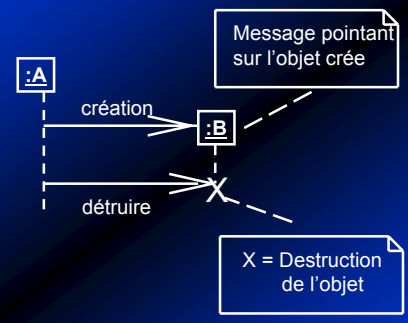


Envoi de messages d'un objet sur lui même

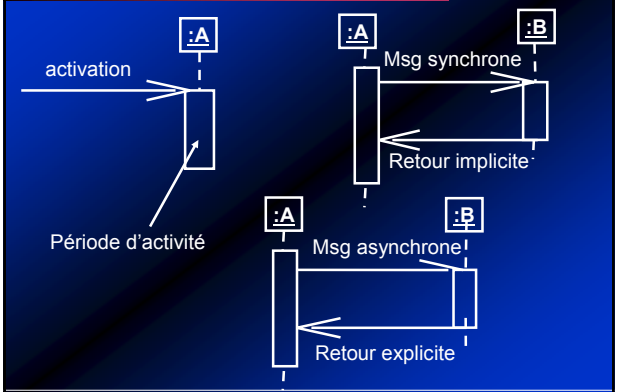
Un objet peut s'envoyer des messages à lui-même :



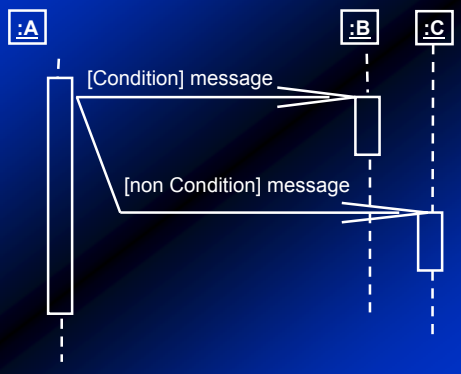
Création et destruction d'un objet par message



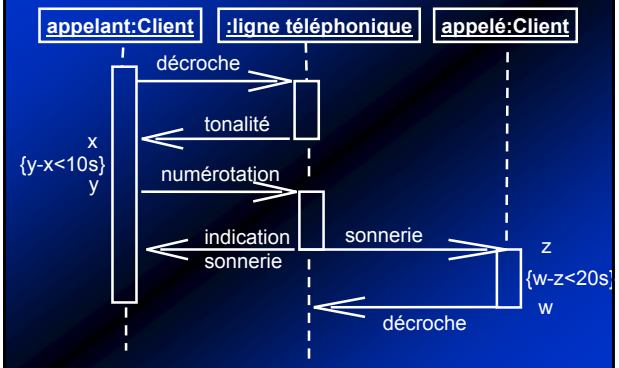
Période d'activité des objets



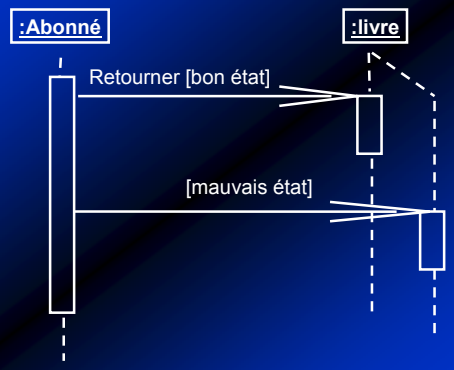
Envoi conditionnel de message



Contraintes temporelles



Traitement conditionnel d'un message reçu



Chapitre VI

Diagrammes d'états-transitions

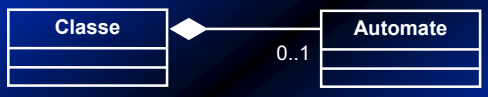
Diagramme d'états-transitions

- décrit le **comportement** des objets d'une classe au moyen d'un automate d'états associé à la classe
- Le comportement est modélisé par un graphe :
  - **Nœuds** = **états possibles** des objets
  - **Arcs** = **transitions** d'état à état.
- Une **transition** :

  = **exécution** d'une **action**

  = **réaction** de **l'objet** sous l'effet d'une occurrence d'evt

Classe et automate

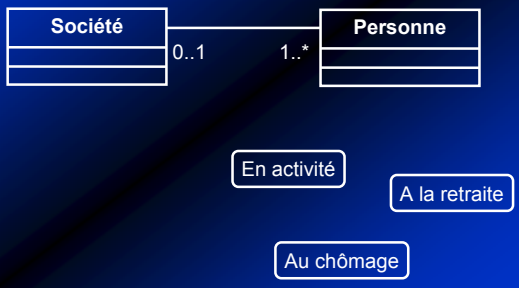


Certains objets ne possèdent pas de comportement réactif : leur classe ne possède pas d'automate

Notion d'état

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

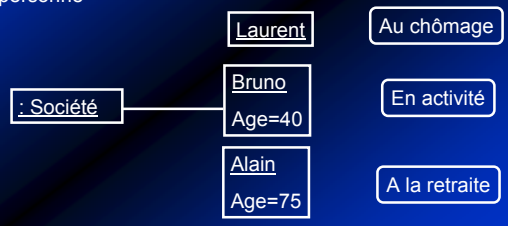
Notion d'état



Notion d'état

• un **état** = **image** de le **conjonction instantanée** des valeurs **des attributs d'un objet + présence** ou non de ses liens à d'autres objets

• Exemple : présence d'un lien vers société ou âge d'une personne



Notion d'état

• Chaque diagramme d'états-transitions comprend **un état initial**.

• Pour **un niveau hiérarchique donné**, il y a **un et un seul état initial,** mais **plusieurs états finaux** correspondant chacun à une fin de vie de l'objet différente.

• Il est possible de n'avoir **aucun état final :** ex : un système que ne s'arrête jamais.

Notion de transition

• Lorsque les événements se produisent, les objets changent d'état en respectant les règles décrites par l'automate associé à leur classe

• Les diagrammes d'états-transitions sont des graphes orientés

• Les états sont reliés par des connexions unidirectionnelles appelées transitions

Ex : classe « Place de parking »

Notion d'événement

• un événement correspond à **l'occurrence** d'une **situation donnée** dans le domaine étudié

• un **événement** est une **information instantanée** qui doit être **traitée** dans **l'instant où il se produit**

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

Notion d'événement

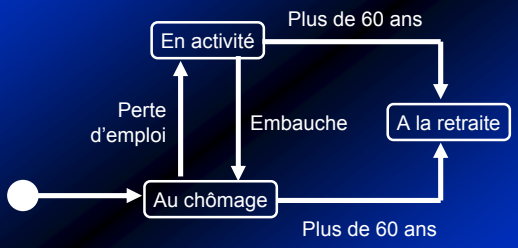
• syntaxe d'un événement :

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

La description complète d'un évt est donnée par :

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

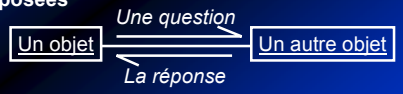
Notion d'événement



Communication entre objets par évenements

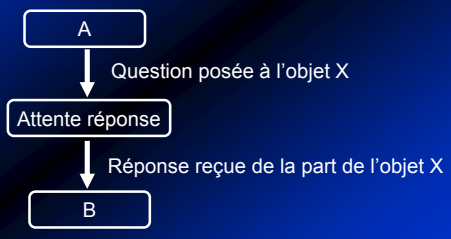
• La communication est de type **asynchrone**, **atomique** et **unidirectionnelle**. Un objet peut envoyer un événement à un autre objet qui doit toujours être à même de l'interpréter

• Les **besoins** de communication par **événements synchrones** ou les **échanges bidirectionnels** peuvent se représenter au moyen de deux échanges **asynchrones** de **directions opposées**



Communication entre objets par évenements

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



Notion de garde

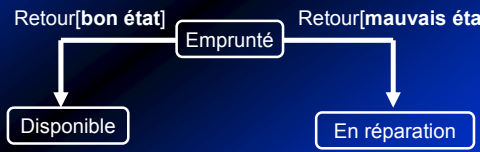
• Une **garde** est une **condition booléenne** qui **permet ou non** le **déclenchement** d'une **transition** lors **de l'occurrence** d'un **événement**

Communication entre objets par évenements

• Les **gardes** permettent de **conserver** la **propriété** de **déterminisme** d'un automate d'états finis.

• Lorsqu'un occurrence d'**événement survient**, les **gardes,** qui doivent être **mutuellement exclusives**, sont **évaluées**.

• Le **résultat** de cette **évaluation permet** de **valider** puis de **déclencher** une **transition** possible



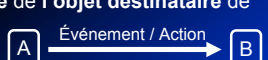
Notion d'opération et d'action

• **Action et activités =** le **lien** entre les **opérations** définies dans la **spécification** d'une **classe** et les **événements apparaissant** dans le diagramme d'états-transitions

• Chaque transition peut avoir une **action** à **exécuter** lorsqu'elle est **déclenchée**

• **L'action** est considérée comme **instantanée** et **atomique**

• Une action correspond à **l'exécution d'une** des **opérations déclarées** dans la **classe** de **l'objet destinataire** de **l'événement**.

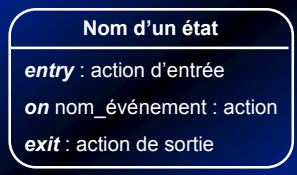


• **L'action** a **accès** aux **paramètres** de **l'événement** ainsi qu'aux **attributs** de **l'objet** sur **lequel elle s'applique**

Actions dans un état

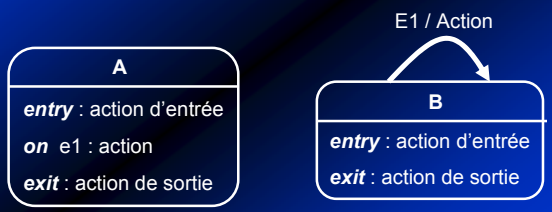
- **Les états** peuvent également contenir des actions :
  - elles sont exécutée
    - à **l'entrée** ou à la **sortie** de l'état ou
      - l'action **d'entrée** (***entry***) est exécutée de manière **instantanée** et **atomique**
      - l'action de **sortie** (***exit***) est exécutée à la sortie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

Actions dans un état



Opérations, actions et activités

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



Opérations, actions et activités

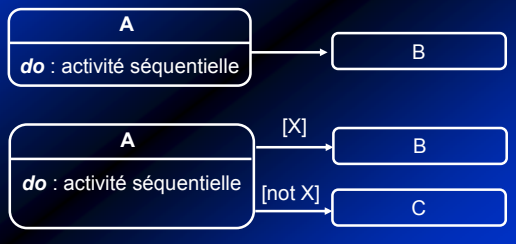
- Les **actions** correspondent à des **opérations** dont la **durée d'exécution** est **négligeable**.
- Une **opération** qui **prend un certain temps** doit être **modélisée à travers un état plutôt** que par une **action**… Une telle opération est appelée **activité.**
- Le mot clé ***do:*** indique une activité
- Contrairement aux actions, les activités peuvent être interrompues à tout moment, dès qu'une transition de sortie est déclenchée

Opérations, actions et activités

- Il existe **deux formes d'activités :**
  - activité **cyclique** : qui ne **s'arrête** que par le biais d'une **transition de sortie**
  - activité **séquentielle** : qui démarre à **l'entrée** de **l'état**. Lorsqu'elle parvient à son terme (**auto-terminaison**), **l'état** peut être **quitté** si l'une des **transition** de sortie est **franchissable**. C'est une transition **automatique**, **éventuellement protégée** par une **garde**

Opérations, actions et activités

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



Point d'exécution des opérations

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

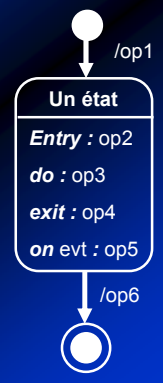
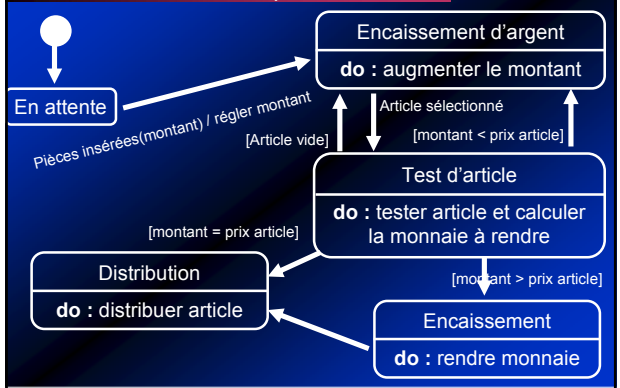
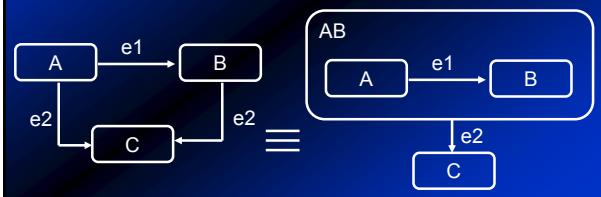


Diagramme d'états-transitions du Distributeur Automatique de Boissons



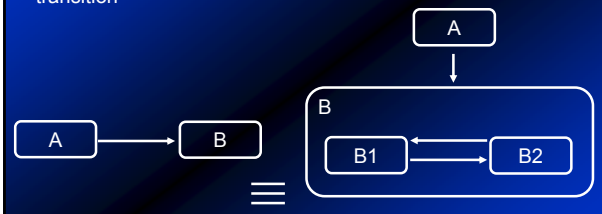
Généralisation d'états

- Un état peut être **décomposé** en **plusieurs sous-états** disjoints; les **sous-états** héritent des **caractéristiques** de leur **super-états**
- **Décomposition disjonctive :** l'objet **doit être** dans **un seul sous-état à la fois**



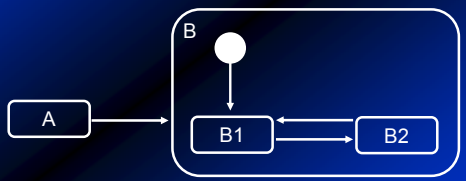
Généralisation d'états

- Les transitions d'entrée ne sont pas héritées par tous les états, seul un état peut être cible de la transition



Généralisation d'états

- Il est préférable de **limiter** les **liens entre niveaux hiérarchiques** d'un automate en définissant **systématiquement** un **état initial** pour chaque niveau

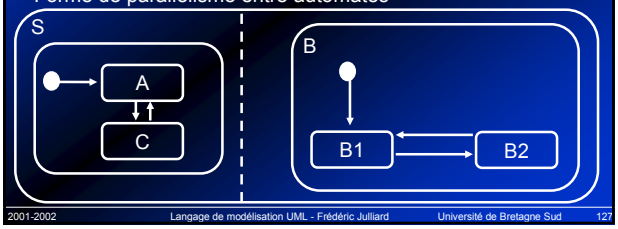


Généralisation d'états

Exercice : Donner le diagramme d'état transition d'une boîte de vitesse à 5 rapports + marche arrière

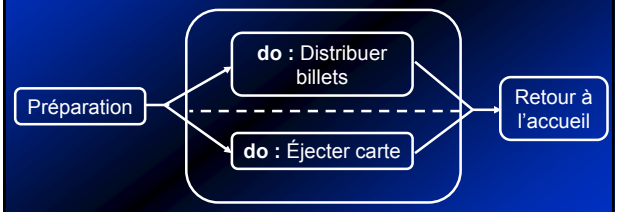
Agrégation d'états

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



Agrégation d'états

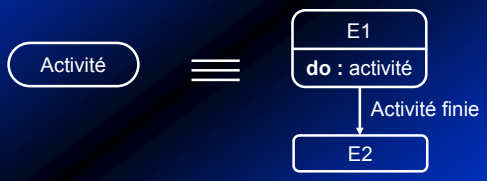
- Exemple : activité d'émission de billets



Chapitre VII
Diagrammes d'activités

Introduction

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

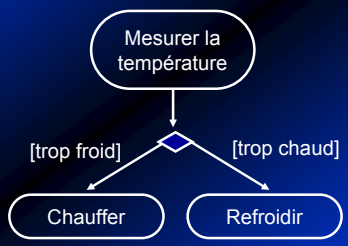


Gardes

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

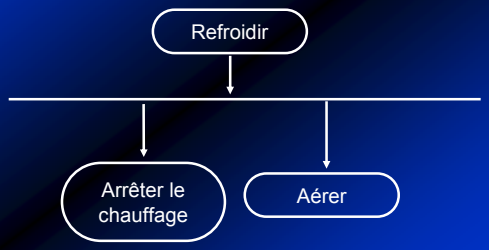
Gardes

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



Synchronisations

• Les **diagrammes d'activités** représentent les synchronisations d'activités au moyen de **barres de synchronisation**



Synchronisations

• Les **diagrammes d'activités** représentent les synchronisations d'activités au moyen de **barres de synchronisation**

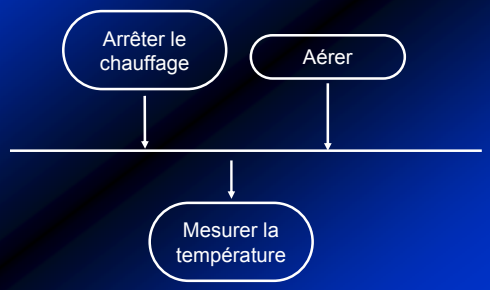
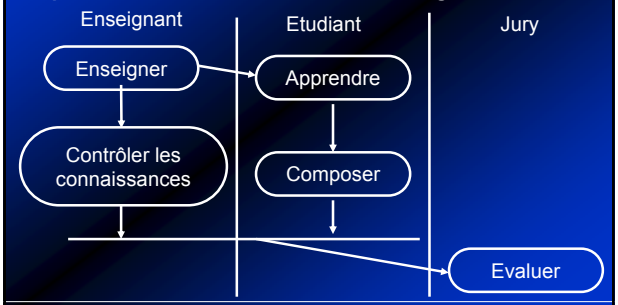


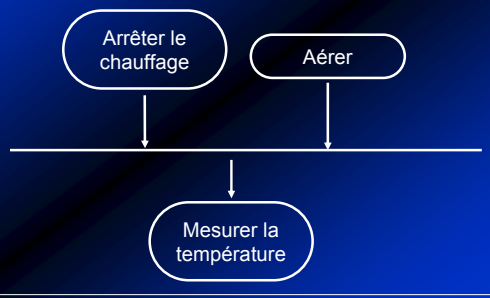
Diagramme d'activités

• Les **diagrammes d'activités** peuvent être découpés en couloirs d'activités : **répartition des responsabilités au sein d'un mécanisme logiciel :**



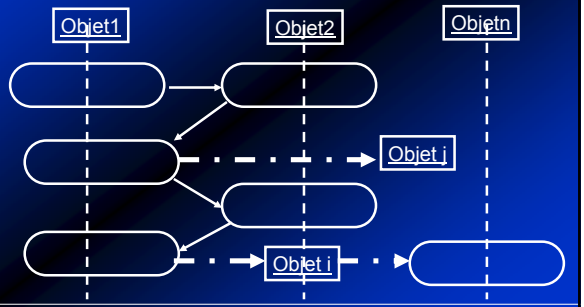
Synchronisations

• Les **diagrammes d'activités** représentent les synchronisations d'activités au moyen de **barres de synchronisation**



Objets dans un diagramme d'activités

• Il est possible de faire apparaître des objets dans un diagramme d'activités, soit au sein d'un couloir d'activité soit en dehors de ces couloirs



Objets dans un diagramme d'activités

• Les diagrammes d'activités peuvent faire référence à des états et à des événements