

Introduction à la Programmation Orientée Objet

10 - Conception et développement de programmes orientés objet

Valentin Honoré

`valentin.honore@ensiie.fr`

FISA 1A

Quels objets et quelles sont leurs relations ?

- ▶ Besoins d'avoir une vision globale d'un projet
 - ☐ Quels sont les acteurs et les besoins (expression des besoins)
 - ☐ Quels objets doivent être créés (conception), quelles sont leurs relations
 - ☐ Implantation des objets : héritage, multiplicité, les attributs et les méthodes nécessaires etc

- ▶ Besoin d'un véritable cahier des charges
 - ☐ Vision avec des plans : les diagrammes
 - ☐ Modéliser, concevoir puis programmer quand tout est clair

- ▶ *Unified Modelling Language* : langage de référence pour cela

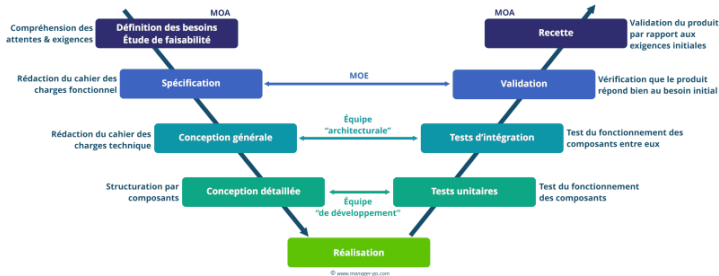
Le langage UML (1/2)

► Langage standardisé

- langage graphique de modélisation objet, 13 types de diagramme
- représenter la succession de phases, de l'analyse à la mise en oeuvre concrète sur site

► Outil de gestion de projet

- Donner une perception unifiée à tous les acteurs
- Vue plus globale, intuitive, malléable et mieux communicable
- "Ce qui se conçoit et se programme bien s'énonce clairement"



- ▶ UML permet de "penser objet" et rester indépendant du langage
 - ☐ normaliser les concepts de l'objet (énumération et définition des concepts)
 - ☐ normalisation des représentations graphiques
- ▶ Quels bénéfices à la modélisation ?
 - ☐ Organiser les fichiers qui constituent le projet
 - ☐ Scruter le programme : messages, traçage d'objets etc
- ▶ Qu'est ce qu'un modèle ?
 - ☐ Un modèle est une représentation simplifiée de la réalité en vue de réaliser quelque chose.
 - ☐ Exemple : une carte est une représentation simplifiée d'un territoire destiné à un usage particulier : randonnée, se diriger en voiture etc

- ▶ Phase d'expression des besoins (établissement cahier des charges)
 - △ diagrammes de cas d'utilisation
 - △ diagrammes de séquence

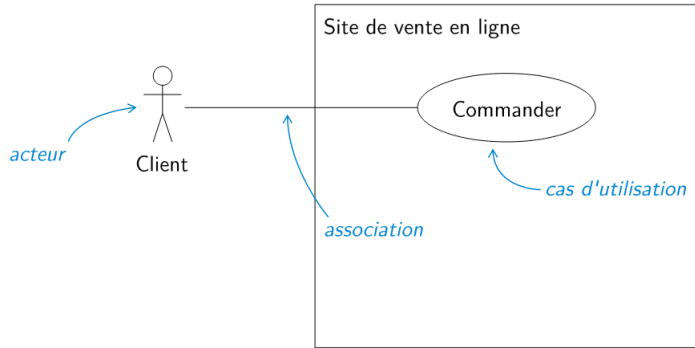
- ▶ Phase de conception
 - △ **Diagrammes de classes**
 - Types abstrait et classes, méthodes, attributs etc
 - Objectif : savoir exprimer un diagramme de classe en fonction d'un énoncé
 - △ Diagrammes d'états-transitions

- 1 Diagrammes de cas d'utilisation
- 2 Diagrammes de séquence
- 3 Diagrammes de classe
 - Structure générale
 - Relations entre classes
 - Relations d'association
 - Relations de généralisation/spécialisation
 - Mise en œuvre d'interfaces
 - Retour sur les classes-association
- 4 Diagrammes d'états-transitions
 - États dans un diagramme d'états-transitions
 - Événements dans un diagramme d'états-transitions
 - Transitions dans un diagramme d'états-transitions
 - États étendus

- ▶ Les diagrammes de cas d'utilisation modélisent à QUOI sert le système, en organisant les interactions possibles avec les acteurs.
- ▶ Comprendre les besoins d'un client pour rédiger le cahier des charges fonctionnel
- ▶ Trois questions principales
 - ☐ Définir les utilisations principales du système : **À quoi sert-il ?**
 - ☐ Définir l'environnement du système : **Qui va l'utiliser ?**
 - ☐ Définir le cadre effectif du système : **Quelles sont ses limites ?**
- ▶ Éléments de description
 - ☐ Diagrammes de cas d'utilisation (scénarios du cahier des charges)
 - ☐ Diagrammes de séquence (cf section suivante) des cas d'utilisation

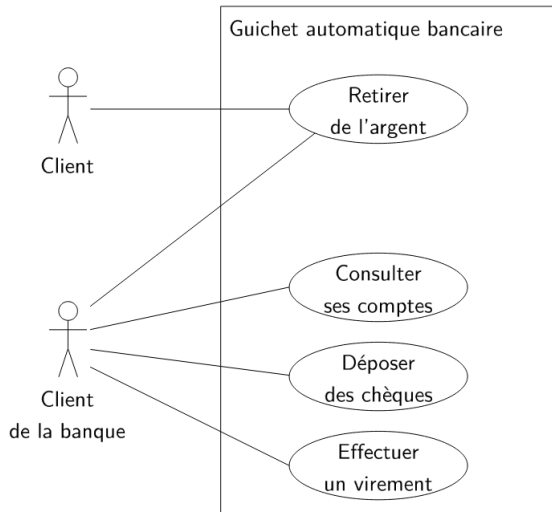
- ▶ Ensemble de scénarios réalisant un objectif de l'utilisateur
- ▶ Vision du système centrée sur l'utilisateur
- ▶ Acteur = entité qui interagit avec le système pour atteindre un but
 - représente un/des rôle(s)
 - identification par le nom du rôle
- ▶ Cas d'utilisation : fonctionnalité visible de l'extérieur
 - fonctionnalité (but) déclenchée par un acteur
 - Nom = Verbe à l'infinitif + Groupe nominal

Diagramme de cas d'utilisation : structure



Source : Delphine Longuet (LRI)

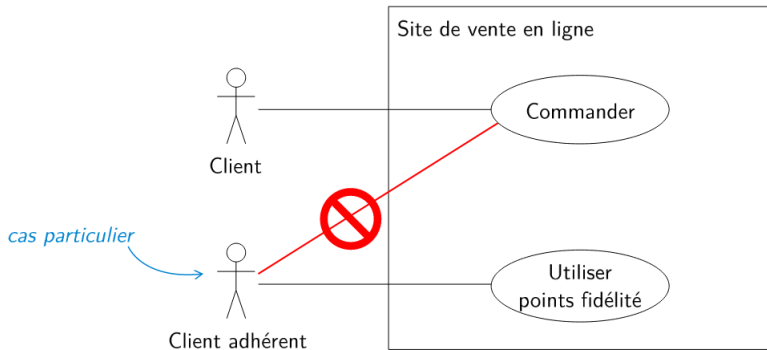
Diagramme de cas d'utilisation : exemple



Source : Delphine Longuet (LRI)

Diagramme de cas d'utilisation : généralisation de rôles

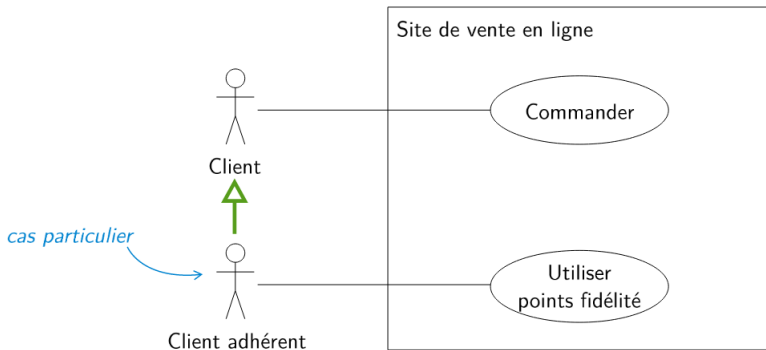
- Si B peut faire tout ce que fait A



Source : Delphine Longuet (LRI)

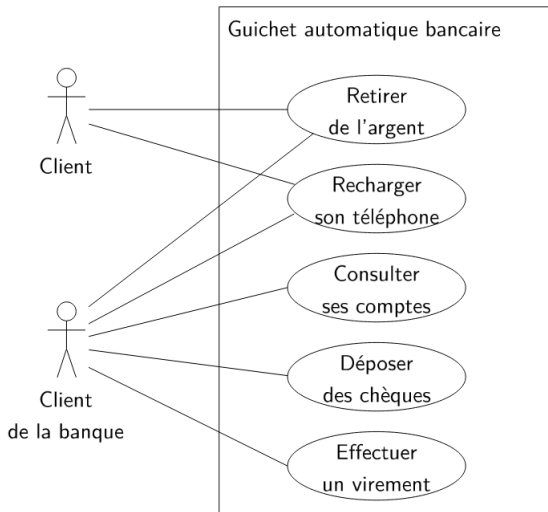
Diagramme de cas d'utilisation : généralisation de rôles

- ▶ Si B peut faire tout ce que fait A
 - Faire apparaître B comme un cas particulier de A (A généralise B)



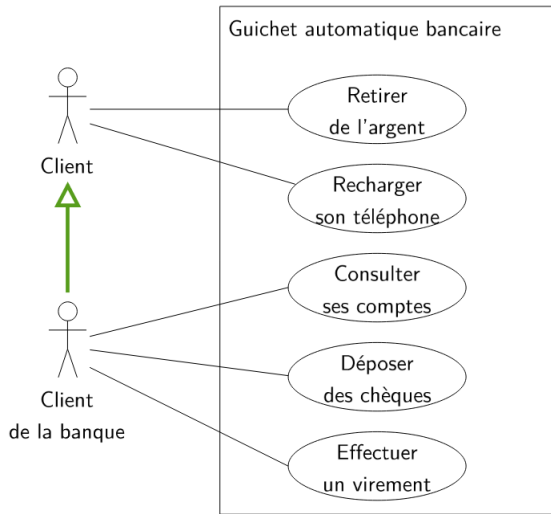
Source : Delphine Longuet (LRI)

Généralisation de rôle : exemples (1/2)



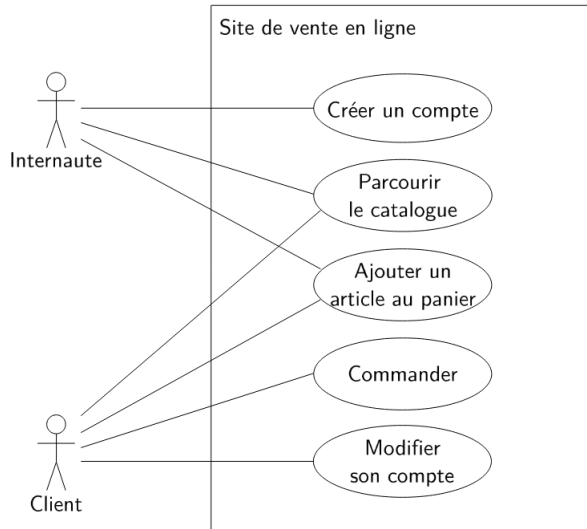
Source : Delphine Longuet (LRI)

Généralisation de rôle : exemples (1/2)



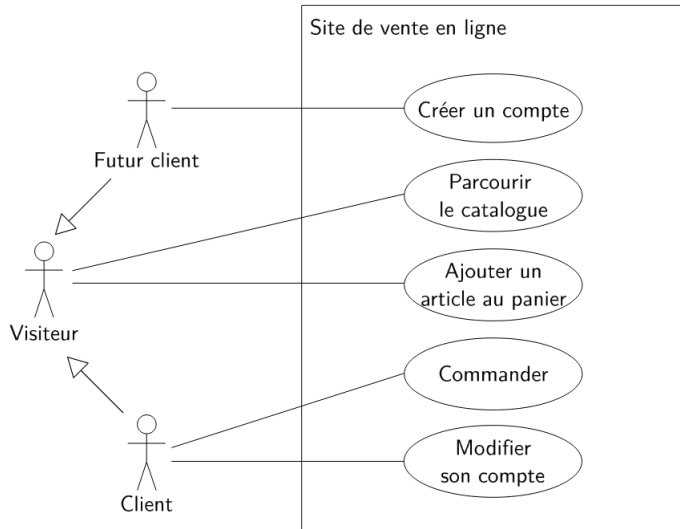
Source : Delphine Longuet (LRI)

Généralisation de rôle : exemples (2/2)



Source : Delphine Longuet (LRI)

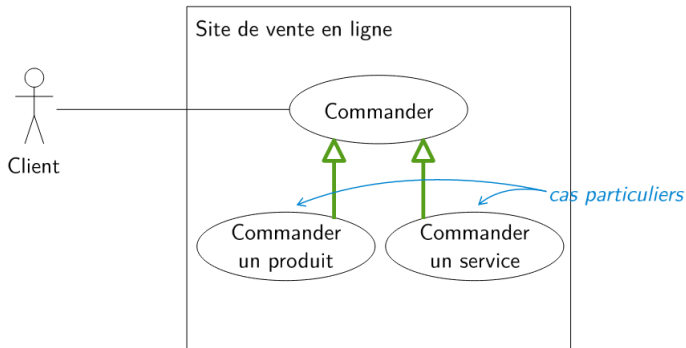
Généralisation de rôle : exemples (2/2)



Source : Delphine Longuet (LRI)

Diagramme de cas d'utilisation : relations entre cas d'utilisation

- Généralisation : X est un cas particulier de Y
 - Tout ou partie du scénario de Y est spécifique à X

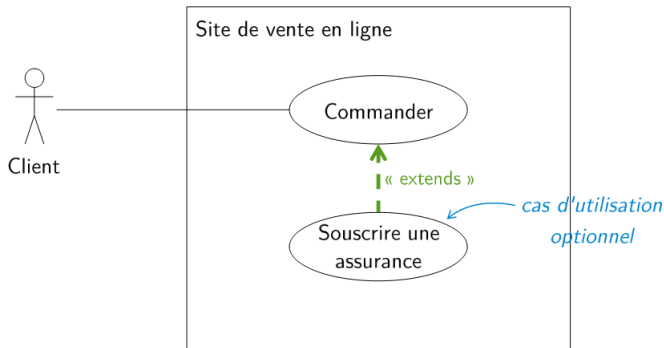


Source : Delphine Longuet (LRI)

Diagramme de cas d'utilisation : relations entre cas d'utilisation

► Extension : X étend Y

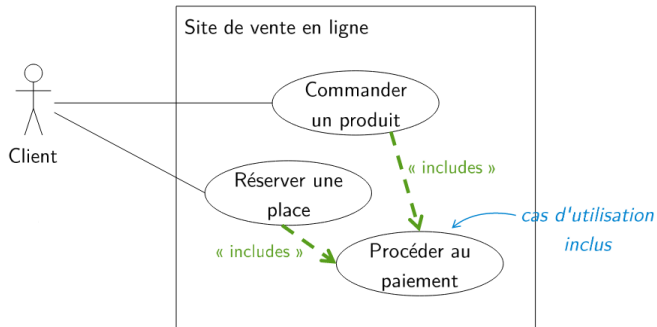
- X peut être déclenché pendant Y, X est **optionnel** pour Y



Source : Delphine Longuet (LRI)

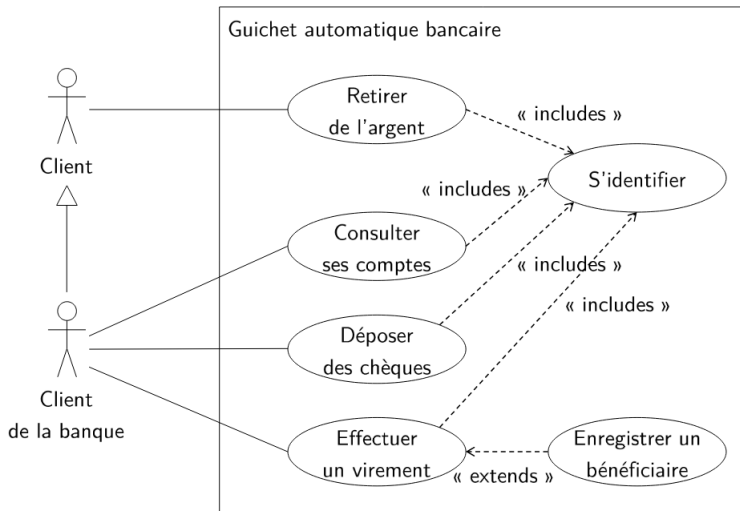
Diagramme de cas d'utilisation : relations entre cas d'utilisation

- Inclusion : X est un cas particulier de Y
 - Y inclus dans le scénario de X, Y déclenché au cours du scénario X
 - **À n'utiliser que si scénario commun à plusieurs cas**



Source : Delphine Longuet (LRI)

Relations entre cas d'utilisation : exemple



Source : Delphine Longuet (LRI)

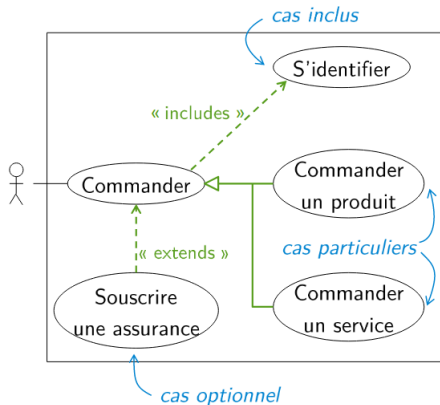
- ▶ Rester lisible dans les diagrammes
 - ☐ Maximum 6 à 8 cas d'utilisation par diagramme
 - ☐ Faire plusieurs diagrammes si nécessaire (ex : cas disjoints entre acteurs)
 - ☐ Relations entre seulement si sémantiquement importantes

- ▶ Par la suite, passage à une description textuelle

- 1 Diagrammes de cas d'utilisation
- 2 Diagrammes de séquence
- 3 Diagrammes de classe
 - Structure générale
 - Relations entre classes
 - Relations d'association
 - Relations de généralisation/spécialisation
 - Mise en œuvre d'interfaces
 - Retour sur les classes-association
- 4 Diagrammes d'états-transitions
 - États dans un diagramme d'états-transitions
 - Événements dans un diagramme d'états-transitions
 - Transitions dans un diagramme d'états-transitions
 - États étendus

- ▶ Utiles pour discuter avec le client et intuitif
- ▶ Mais pas suffisant pour passer directement à la phase développement
- ▶ Étapes suivantes
 - Description détaillée des scénarios (non abordé en détails ici)
 - △ description textuelle en langue naturelle
 - △ explication des notations des diagrammes
 - Diagrammes de séquence

Liens diagrammes de cas d'utilisation & description textuelle



Commander

Acteur : Client

Le cas commence lorsque le client clique sur
« Commander »

Scénario principal

1. Déclenchement du cas S'identifier
2. Fin du cas S'identifier
3. ...
- 4.1 L'objet de la commande est un produit.
Déclenchement du cas Commander un produit.
- 4.2 L'objet de la commande est un service.
Déclenchement du cas Commander un service.

Scénario alternatif

- 3a. Le client choisit de prendre une assurance.

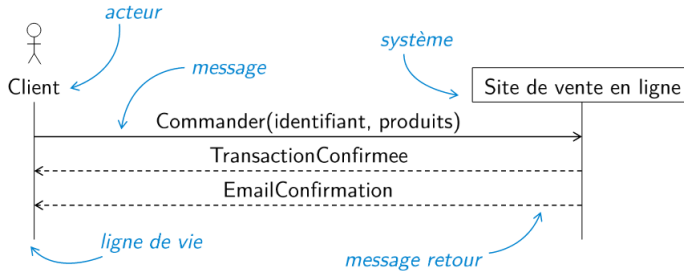
Source : Delphine Longuet (LRI)

Diagrammes de séquence (1/4)

- ▶ Les diagrammes de séquence permettent de décrire COMMENT les éléments du système interagissent entre eux et avec les acteurs
- ▶ Représentation graphique de la chronologie des échanges de messages entre les acteurs et le système
- ▶ Structuration du diagramme en
 - ☐ temps (axe vertical) : lignes de vie
 - ☐ objets (axe horizontal) : messages
- ▶ Point de vue temporel sur les interactions
 - ☐ entre les acteurs et le système (décrire les scénarios des cas d'utilisation)
 - ☐ entre les objets (pendant la conception)

Diagrammes de séquence (2/4) : niveau analyse

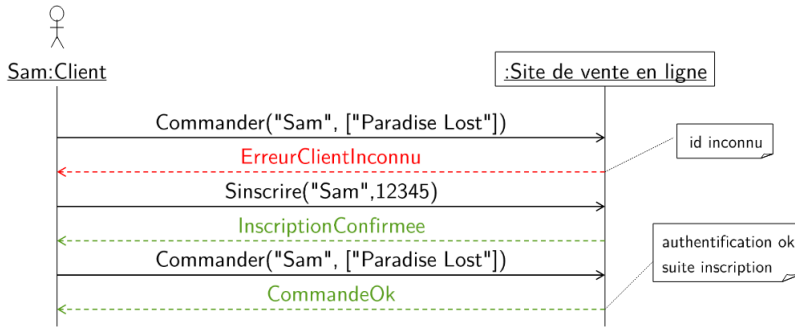
- ▶ Messages informels
- ▶ Noms de messages = cas d'utilisation
- ▶ Utilisation d'arguments utiles au scénario



Source : Delphine Longuet (LRI)

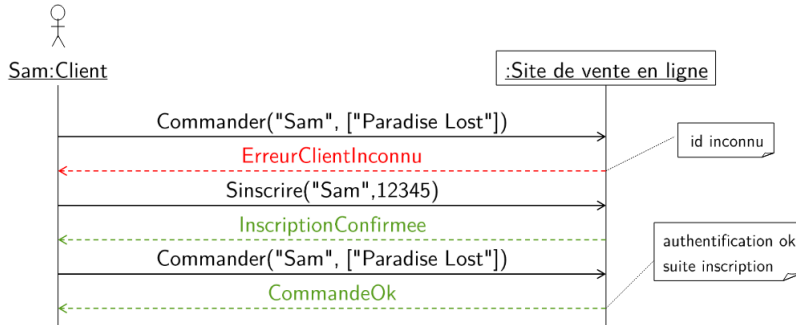
Diagrammes de séquence (3/4) : niveau scénario d'utilisation

- ▶ Variables remplacées par des valeurs concrètes
- ▶ Objectifs
 - Illustrer les différents scénarios de cas d'utilisation
 - Mettre en exergue les relations entre les différents cas
 - Permettre de construire des scénarios plus complexes pour les tests



Source : Delphine Longuet (LRI)

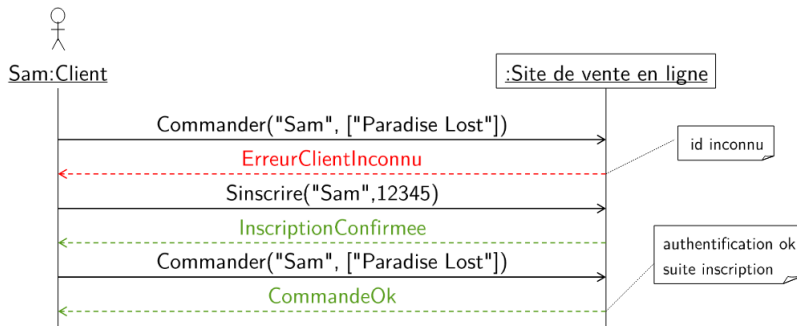
Diagrammes de séquence (3/4) : niveau scénario d'utilisation



Source : Delphine Longuet (LRI)

Que met-on en évidence ici ?

Diagrammes de séquence (3/4) : niveau scénario d'utilisation

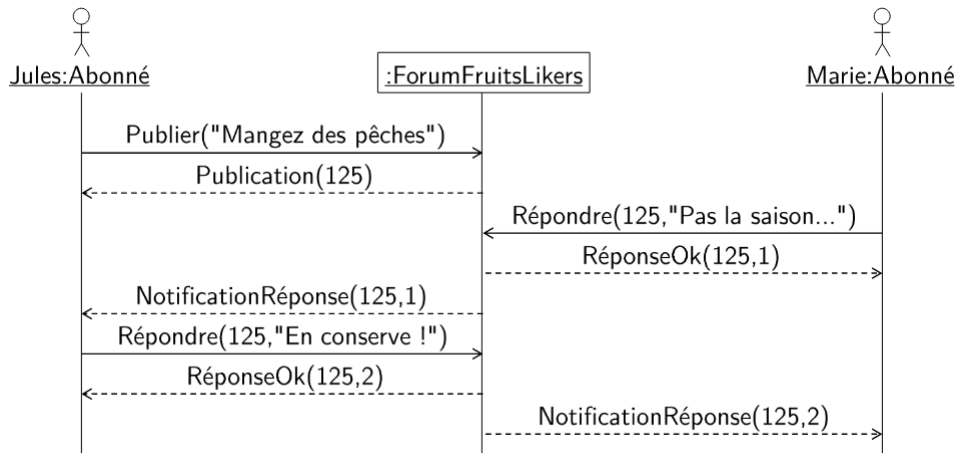


Source : Delphine Longuet (LRI)

Que met-on en évidence ici ? La nécessité d'être inscrit pour pouvoir commander

Diagrammes de séquence (4/4) : exemple avec plusieurs acteurs

- ▶ **ATTENTION** : scénario d'utilisation = interactions acteurs/système
- ▶ Pas de messages entre acteurs



Quelles différences entre diagrammes de séquence et cas d'utilisation ?

▶ Diagramme de cas d'utilisations

- ☐ Récapitulatif graphique des interactions entre acteurs et système

▶ Diagramme de séquence

- ☐ Description de chaque scénario
- ☐ Séquences des interactions entre les acteurs et le système
- ☐ Système vu comme une boîte noire

▶ Spécification des cas d'utilisation

- ☐ Diagrammes de cas d'utilisation +
- ☐ Description textuelle +
- ☐ Scénarios d'utilisation (diagrammes de séquence)

- 1 Diagrammes de cas d'utilisation
- 2 Diagrammes de séquence
- 3 Diagrammes de classe
 - Structure générale
 - Relations entre classes
 - Relations d'association
 - Relations de généralisation/spécialisation
 - Mise en œuvre d'interfaces
 - Retour sur les classes-association
- 4 Diagrammes d'états-transitions
 - États dans un diagramme d'états-transitions
 - Événements dans un diagramme d'états-transitions
 - Transitions dans un diagramme d'états-transitions
 - États étendus

3 Diagrammes de classe

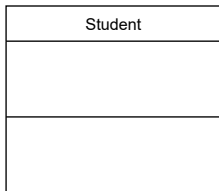
- Structure générale
- Relations entre classes
 - Relations d'association
 - Relations de généralisation/spécialisation
 - Mise en œuvre d'interfaces
 - Retour sur les classes-association

- États dans un diagramme d'états-transitions
- Événements dans un diagramme d'états-transitions
- Transitions dans un diagramme d'états-transitions
- États étendus

- ▶ Diagramme de classes = une modélisation interne et statique du système
 - Mettre en exergue la structure du cahier des charges établi précédemment
 - Ne tient pas compte de l'évolution temporelle du système
- ▶ Diagrammes de classes = spécifier la structure et les liens entre les objets du système
 - description des objets qui interagissent et s'échangent des messages (montrer le contenu de chaque classe)
 - description des relations qui peuvent exister entre les différentes classes
- ▶ Passage à la phase de conception
 - Non abordés ici : diagrammes d'objets
 - = représentation de l'état du logiciel (création/suppression d'objets, modifications des objets etc)

Structure d'un diagramme de classe

- ▶ Une classe est représentée par un rectangle séparé en trois parties :
 - la première partie contient le nom de la classe
 - la seconde contient les attributs de la classe
 - la dernière contient les méthodes de la classe



```
public class Student{  
  
}
```

Définition d'un attribut (1/2)

- ▶ Pour définir un attribut, il faut préciser son nom suivi du caractère " :" et du type de l'attribut
- ▶ Les droits sur l'attribut doit précéder son nom et peut prendre les valeurs suivantes :

Caractère	Visibilité
+	public
#	protected
-	private

- ▶ Une valeur d'initialisation peut être précisée juste après le type en utilisant le signe "=" suivi de la valeur.
- ▶ Attributs statiques = soulignés

Définition d'un attribut (2/2)

Student
+ name : String = ""
schoolID : int = 0
- idNum : int = 0

```
public class Student {  
  
    public String name = "";  
    protected int schoolID = 0;  
    private int idNum = 0;  
  
}
```

Définition d'une méthode (1/2)

- ▶ La visibilité est définie comme pour les attributs (méthodes statiques = soulignées)
- ▶ Les paramètres sont précisés entre parenthèses sous la forme `nom_methode(nom_param : type)`
- ▶ Si la méthode renvoie une valeur, son type est précisé après un signe " :"
- ▶ Pour les constructeur, on peut rajouter «Constructor» avant le nom de la classe
- ▶ On peut omettre les attributs/méthodes non significatifs (*getter/setters* etc)

Définition d'une méthode (2/2)

Student
+ name : String = "" # schoolID : int = 0 - idNum : int = 0
+ getName() : String # setSchoolID(num:int) - getIdNum() : Int

```
public class Student {  
  
    public String name = "";  
    protected int schoolID = 0;  
    private int idNum = 0;  
  
    public void getName(){  
    }  
  
    protected void  
        setSchoolID(int num){  
    }  
  
    private int getIdNum(){  
    }  
  
}
```

- 1 Diagrammes de cas d'utilisation
- 2 Diagrammes de séquence
- 3 Diagrammes de classe
 - Structure générale
 - Relations entre classes
 - Relations d'association
 - Relations de généralisation/spécialisation
 - Mise en œuvre d'interfaces
 - Retour sur les classes-association
- 4 Diagrammes d'états-transitions
 - États dans un diagramme d'états-transitions
 - Événements dans un diagramme d'états-transitions
 - Transitions dans un diagramme d'états-transitions
 - États étendus

► Association

- ☐ les instances des classes sont liées
- ☐ communication avec les objets
- ☐ si relation forte : **composition**
- ☐ Association de dépendance : la modification d'une classe peut avoir des conséquences sur une autre

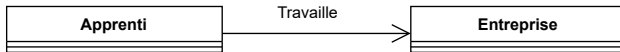
► Généralisation/spécialisation

- ☐ les instances de la sous-classe sont des instances de la super-classe (conceptuel)
- ☐ héritage (implémentation)

► Mise en œuvre

- ☐ mise en œuvre : une classe réalise une interface

- ▶ Une association exprime une connexion sémantique bidirectionnelle entre deux classes.
- ▶ Peut être unidirectionnelle (préciser dans ce cas avec les flèches)
- ▶ Une association possède un nom constitué classiquement d'un verbe conjugué (la plupart du temps au présent)
- ▶ Diagramme : rajouter un lien entre deux classes et exprimer avec du texte la relation

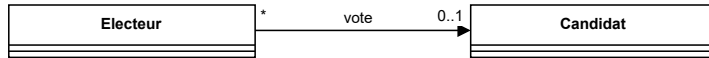


Relation d'association (1/2)

- ▶ Nom = forme verbale, éventuellement en sens de lecture avec la flèche sur unidirectionnelle
- ▶ Rôles
 - ☐ à indiquer si besoin à chaque extrémité
 - ☐ forme nominale
- ▶ Multiplicité
 - ☐ notée à côté de la classe, du côté de l'association
 - ☐ 1, 0..1, 0..* etc

_____	1 (par défaut)
_____1	1 (explicite)
_____3	3
_____0..1	0 ou 1
_____*	0 ou plusieurs
_____1..*	n-aire
_____2..8	intervalle

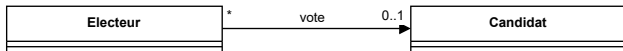
- ▶ Ajout éventuel de mots clés (*ordered*, *set* (uniques); *bag*, *list* (doublons))



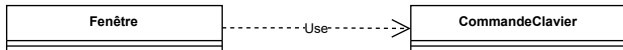
► Les associations

- ☐ ont une durée de vie
- ☐ sont héritées
- ☐ sont indépendantes les unes des autres (comme les attributs)

- Association à navigabilité restreinte (les instances d'une classe de "connaissent" pas les instances d'une autre)



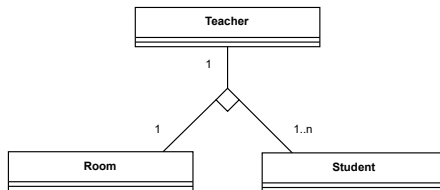
- Relation de dépendance (relation unidirectionnelle et d'obsolescence)



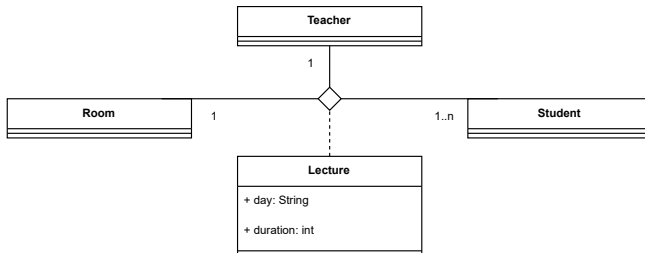
- Bidirectionnalité par défaut !

Exemples d'association

- Association n-aire (relier plus de deux classes, essayer de limiter leurs utilisations)



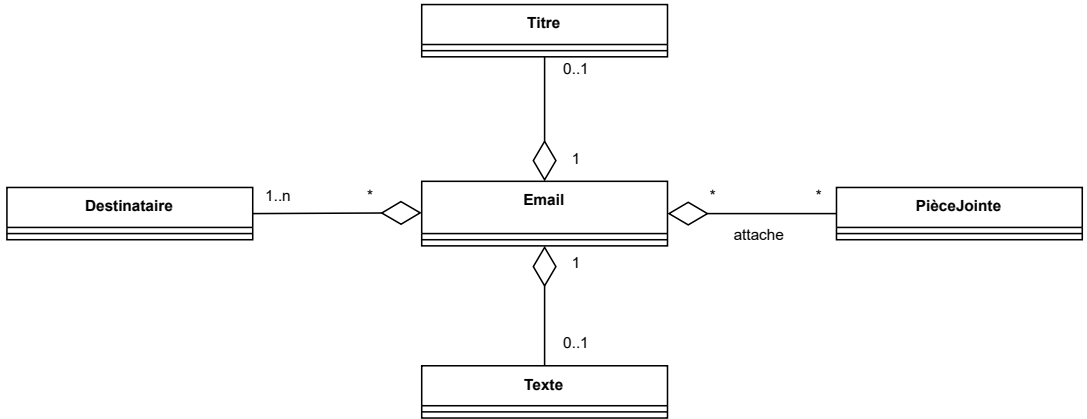
- Classe d'association : réalise la navigation entre les instances d'autres classes (cf plus loin)



Relation d'association : agrégation (1/2)

- ▶ Exprime un couplage **fort**, **relation de subordination** : **EST-UNE-PARTIE-DE, A-UN**
 - ☐ association asymétrique et transitive (contrairement à association simple)
 - ☐ représentation des "ensemble/élément" ou "contenant/contenu" en vue subjective mais explicite
- ▶ Une agrégation peut notamment (mais pas nécessairement) exprimer :
 - ☐ qu'une classe (un "élément") fait partie d'une autre ("l'agrégat"), **est-ce une partie de ?**
 - ☐ qu'un changement d'état d'une classe, entraîne un changement d'état d'une autre,
 - ☐ qu'une action sur une classe, entraîne une action sur une autre.
- ▶ Un élément agrégé peut être partagé entre plusieurs classes
- ▶ Une instance d'un élément agrégé peut exister sans agrégat (et inversement)
 - ☐ la suppression de l'ensemble n'entraîne pas celle de l'élément

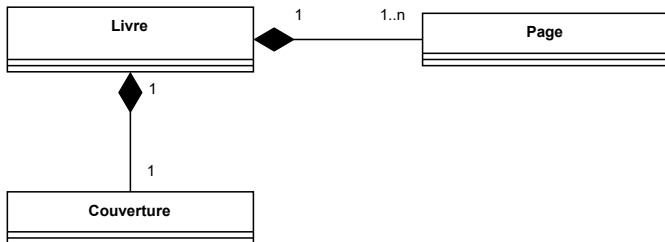
Relation d'association : agrégation (2/2)



- ▶ Graphiquement, on ajoute un losange vide du côté de l'agregat
- ▶ Ici, un fichier peut être attaché à un email (ou plusieurs ou aucun),
- ▶ et un email peut (ou non) attacher un ou plusieurs fichier(s)

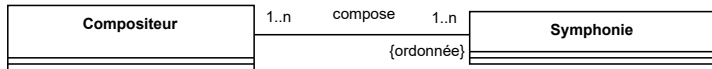
Relation d'association : composition

- ▶ Composition = agrégation forte (agrégation par valeur)
 - Cycles de vie composants/agrégat sont liés : si l'agrégat est détruit/copié, ses composants le sont aussi
 - ex : les pages qui composent un livre
 - La multiplicité du côté composite ne doit pas être supérieure à 1 (cad 1 ou 0..1).
 - Graphiquement, on ajoute un losange plein du côté de l'agrégat.
- ▶ Une instance de composant ne peut être liée qu'à un seul agrégat de manière simultanée



- ▶ Vues subjectives
- ▶ Lorsqu'on représente (avec UML) qu'une Molécule est "composée" d'Atomes
 - destruction d'une instance de Molécule → destruction des instances d'Atomes
- ▶ Agrégation et composition servent à ajouter de la sémantique aux modèles
 - l'exemple ci-dessus n'est pas vrai dans la réalité
 - mais abstraction satisfaisante pour modéliser une molécule

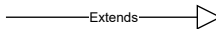
- ▶ Expressions qui précisent le rôle ou la portée d'un élément de modélisation
- ▶ Elles permettent d'étendre ou préciser sa sémantique
 - Peuvent par exemple restreindre le nombre d'instances visées (ce sont alors des "expressions de navigation")
- ▶ Peuvent s'exprimer en langage naturel. Graphiquement, il s'agit d'un texte encadré d'accolades.



- ▶ Généralisation : factorisation de caractéristiques (attributs et opérations)
 - **EST-UN, EST-UNE-SORTE-DE**
 - toutes les instances de la sous-classe sont des instances de la super-classe (définition ensembliste)
- ▶ Une généralisation (ou agrégation) est un ensemble de sous concepts qui collectivement forment un nouveau concept.
- ▶ Une spécialisation (ou décomposition) divise un concept en sous-concepts.

- ▶ Mise en oeuvre à travers l'héritage de méthodes/attributs
- ▶ Partitioner une classe en sous-classes
 - ☐ la sous-classe a des attributs et/ou des associations supplémentaires pertinents
 - ☐ par rapport à la super-classe ou à d'autres sous-classes, la sous-classe doit être gérée, manipulée, on doit agir sur elle ou elle doit réagir différemment, et cette distinction est pertinente
 - ☐ le concept de la sous-classe représente une entité (humain, animal, robot) qui a un comportement différent de celui de la super-classe, et cette distinction est pertinente

- ▶ Relation avec une flèche à tête creuse



- ▶ Possible d'ajouter aux méthodes de la sous-classe celles de la classe mère qui seront redéfinies
- ▶ Permet de factoriser du code (héritage des attributs & méthodes)
 - ☐ Classe abstraites : ajouter {abstract} en dessous du nom
 - ☐ Méthodes abstraites : pareil à côté de la définition de la méthode

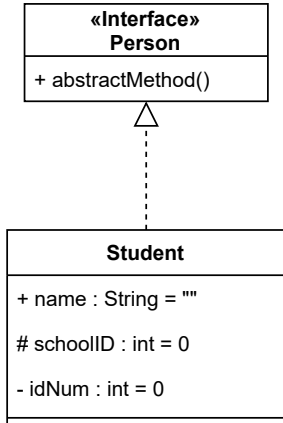
- ▶ Classe Forme (abstraite) avec périmètre et surface
- ▶ Classe carré qui étend forme
- ▶ Classe Cercle qui étend forme
- ▶ **Exercice** : dessinez le diagramme de classe

Mise en œuvre d'interface (1/2)

- ▶ Une boîte pour l'interface avec 2 cadres : nom & méthode(s) abstraite(s)
- ▶ Le nom de l'interface est précédé de «Interface»
- ▶ Méthodes décrites pareil qu'avant
- ▶ Une classe met en œuvre : relier la boîte de la classe vers celle de l'interface avec



Mise en œuvre d'interface (2/2)



```
public interface Person { void
    abstractMethod(); }

public class Student
    implements Person {

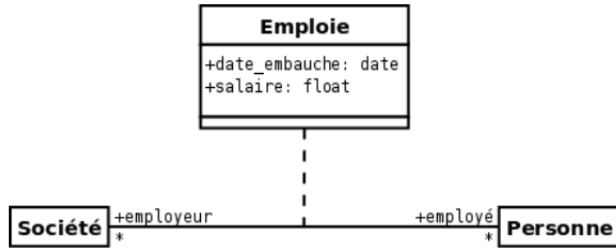
    public String name = "";
    protected int schoolID = 0;
    private int idNum = 0;

    void abstractMethod(){
        ...
    }
}
```

Définition d'une classe-association

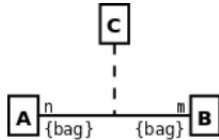
- ▶ Parfois, une association doit posséder des propriétés.
 - **Exemple** : l'association "Emploie" entre une société et une personne possède comme propriétés le salaire et la date d'embauche
 - ces deux propriétés n'appartiennent ni à la société, qui peut employer plusieurs personnes, ni aux personnes, qui peuvent avoir plusieurs emplois.
 - il s'agit donc bien de propriétés de l'association Emploie.
- ▶ Utilité : introduire un nouveau concept pour modéliser cette situation : celui de classe-association.
- ▶ Une classe-association possède les caractéristiques des associations et des classes
 - elle se connecte à deux ou plusieurs classes et possède également des attributs et des opérations.
 - Une classe-association est caractérisée par un trait discontinu entre la classe et l'association qu'elle représente

Classe-association : exemple

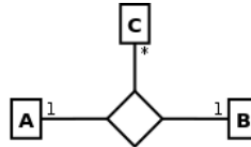
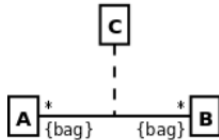


Equivalence classe-association / associations n-aires

Parfois, la sémantique d'une classe-association peut être représentée de façon identique avec des associations

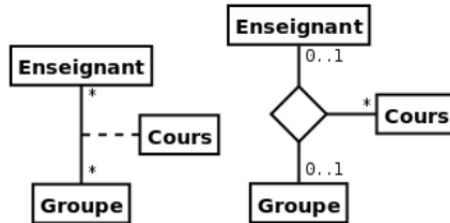


ou encore



Classe-association, association n-aire ou association avec rôles ?

- ▶ Pas souvent simple de trancher entre classe-association, association n-aire ou autre
 - garder à l'esprit qu'une classe-association, ça n'est qu'une association
 - et que la classe-association est indissociable de cette association
- ▶ Exemple : classe-association cours entre un groupe d'étudiants et un enseignant
 - A gauche : Un cours ne peut exister que s'il existe un lien entre un objet Enseignant et un objet Groupe (si le lien disparaît, le cours aussi)
 - Si un cours doit pouvoir exister indépendamment de l'existence d'un lien (ex : enseignant en cours d'affectation), il vaut mieux l'association ternaire de droite

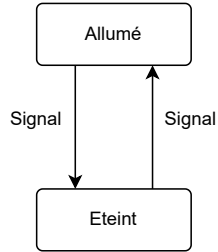


- 1 Diagrammes de cas d'utilisation
- 2 Diagrammes de séquence
- 3 Diagrammes de classe
 - Structure générale
 - Relations entre classes
 - Relations d'association
 - Relations de généralisation/spécialisation
 - Mise en œuvre d'interfaces
 - Retour sur les classes-association
- 4 Diagrammes d'états-transitions
 - États dans un diagramme d'états-transitions
 - Événements dans un diagramme d'états-transitions
 - Transitions dans un diagramme d'états-transitions
 - États étendus

- ▶ Décrivent le comportement dynamique d'un objet à l'aide d'un automate à états finis.
 - présentent les séquences possibles d'états et d'actions qu'une instance de classe peut traiter au cours de son cycle de vie en réaction à des événements discrets (signaux, méthodes etc)
- ▶ La vision globale du système n'apparaît pas sur ce type de diagramme
 - ils ne s'intéressent qu'à un seul élément du système (classe/composant), indépendamment de son environnement
- ▶ Un diagramme d'états-transitions est un graphe représenté par un automate à états finis
 - une machine dont le comportement des sorties ne dépend pas seulement de l'état de ses entrées, mais aussi d'un historique des sollicitations passées
 - rassemble et organise les états et les transitions d'un objet donné

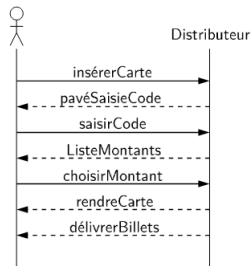
- ▶ Automate à états finis = automate dont le comportement des sorties dépend l'état de ses entrées + historique des sollicitations passées
 - Cet historique est caractérisé par un **état global**
- ▶ Un état global est un jeu de valeurs d'objet, pour une classe donnée, produisant la même réponse face aux événements.
 - Toutes les instances d'une même classe ayant le même état global réagissent de la même manière à un événement
 - ATTENTION : ne pas confondre les notions d'état global et d'état (on va en reparler juste après)
- ▶ Un automate à états finis est graphiquement représenté par un graphe comportant :
 - des **états** : abstraction de la vie d'une entité pendant lequel elle satisfait un ensemble de conditions
 - des **transitions** : changement d'états d'une entité

Exemple simple de diagramme à états-transitions

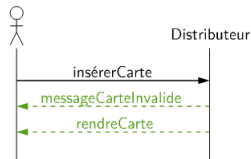


Exemple : distributeur automatique

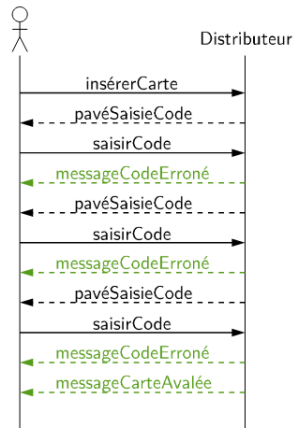
- **Intérêt** : regrouper des scénarios d'utilisation d'un même acteur



Scénario principal



Carte invalide



Trois erreurs de code

- 1 Diagrammes de cas d'utilisation
- 2 Diagrammes de séquence
- 3 Diagrammes de classe
 - Structure générale
 - Relations entre classes
 - Relations d'association
 - Relations de généralisation/spécialisation
 - Mise en œuvre d'interfaces
 - Retour sur les classes-association
- 4 Diagrammes d'états-transitions
 - États dans un diagramme d'états-transitions
 - Événements dans un diagramme d'états-transitions
 - Transitions dans un diagramme d'états-transitions
 - États étendus

- ▶ Représentation par un rectangle à bords arrondis
 - possibilités d'états composites (qui enveloppe des sous-états) [cf dernière sous-section]
 - nom de l'état doit être **unique** dans le diagramme

- ▶ Un état peut être partitionné en plusieurs compartiments si nécessaire
 - séparation par ligne horizontale
 - le premier contient le nom de l'état
 - les autres : transitions internes (cf plus loin) ou sous-états pour les états composites

État d'un objet, ou du diagramme d'états-transitions (état global)

- ▶ Un objet peut passer par une série d'états pendant sa durée de vie
 - Un état = période dans la vie d'un objet pendant laquelle ce dernier attend un événement ou accomplit une activité
 - Configuration de l'état global de l'objet = le jeu des états (élémentaires) qui sont actifs à un instant donné.
- ▶ Si diagramme d'états-transitions simple (sans transition concurrente)
 - un seul état actif à la fois
 - dans ce cas, les notions d'état actif et d'état global se rejoignent !
- ▶ Un état global peut contenir plusieurs états actifs à un instant donné
 - états concurrents = plusieurs états sont actifs en même temps (*objet concurrent*)
 - dans ce cas, les notions d'état actif et d'état global se rejoignent (cf Section États étendus)
- ▶ On va voir par la suite les notions d'évènements, de transitions et de points de choix

► État initial

- ☐ pseudo-état qui indique l'état de départ, par défaut, lorsque le diagramme d'états-transitions est invoqué
- ☐ un objet créé entre dans l'état initial



► État final

- ☐ pseudo-état qui termine le diagramme d'états-transitions



- 1 Diagrammes de cas d'utilisation
- 2 Diagrammes de séquence
- 3 Diagrammes de classe
 - Structure générale
 - Relations entre classes
 - Relations d'association
 - Relations de généralisation/spécialisation
 - Mise en œuvre d'interfaces
 - Retour sur les classes-association
- 4 Diagrammes d'états-transitions
 - États dans un diagramme d'états-transitions
 - Événements dans un diagramme d'états-transitions
 - Transitions dans un diagramme d'états-transitions
 - États étendus

- ▶ Fait instantané qui se produit pendant l'exécution d'un système et dont il est intéressant de modéliser l'impact
 - but du diagramme à états-transitions : modéliser les réactions d'un objet à ces événements
 - événements = pas de durée, se produit à un instant précis
- ▶ Quand un événement se produit
 - peut déclencher la transition vers un nouvel état
 - plusieurs types d'événements : signal, appel, changement et temporel

- ▶ Véhicule une communication asynchrone à sens unique entre deux objets
 - ☐ créé et initialisé explicitement par un objet expéditeur
 - ☐ envoi à un objet (ou à un groupe d'objets)
 - ☐ pas d'attente de traitement du signal par le receveur

- ▶ Réception d'un signal = événement pour le destinataire
 - ☐ L'expéditeur peut être le receveur
 - ☐ Les signaux supportent l'héritage (ex : E/S -> souris, clavier etc)

Évènement temporel (after ou when)

- ▶ Générés avec le temps qui s'écoule
 - Spécification absolue (date)
 - Spécification relative (temps écoulé)

- ▶ Évènement temporel relatif
`after ( <durée> )`

- ▶ Évènement temporel absolu
`when ( date = <date> )`

- ▶ Événement d'appel (*call*) = appel à une méthode du diagramme de classe par un objet
 - ☐ Paramètres de l'opération = ceux de l'événement
 - ☐ Même syntaxe que pour un signal.

- ▶ Événement de changement (*change*)
 - ☐ changement dans une expression booléenne (de faux à vrai) sur des valeurs d'attributs
 - ☐ évaluation continue dans le système
 - ☐ Syntaxe :

```
when ( <condition_booléenne> )
```

- ▶ Réaction du système à un évènement
- ▶ Propriétés
 - ☐ atomique
 - ☐ instantanée
 - ☐ non interruptible
- ▶ Exemple d'actions
 - ☐ création/destruction d'objet
 - ☐ affectation
 - ☐ envoi d'un signal
- ▶ Spécification de l'action en langage naturel ou pseudocode (description libre)

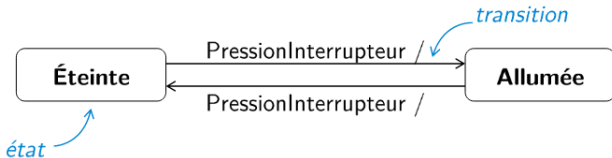
- 1 Diagrammes de cas d'utilisation
- 2 Diagrammes de séquence
- 3 Diagrammes de classe
 - Structure générale
 - Relations entre classes
 - Relations d'association
 - Relations de généralisation/spécialisation
 - Mise en œuvre d'interfaces
 - Retour sur les classes-association
- 4 Diagrammes d'états-transitions
 - États dans un diagramme d'états-transitions
 - Événements dans un diagramme d'états-transitions
 - **Transitions dans un diagramme d'états-transitions**
 - États étendus

Transitions (1/2)

- ▶ Transition = réponse d'un objet à l'occurrence d'un événement
 - liaison entre deux états
 - indique qu'un objet dans un état E peut passer dans un état E'
 - indique quelles actions seront réalisées quand la condition de garde est vérifiée
- ▶ Syntaxe d'une transition (condition uniquement évaluée si événement déclencheur se produit)

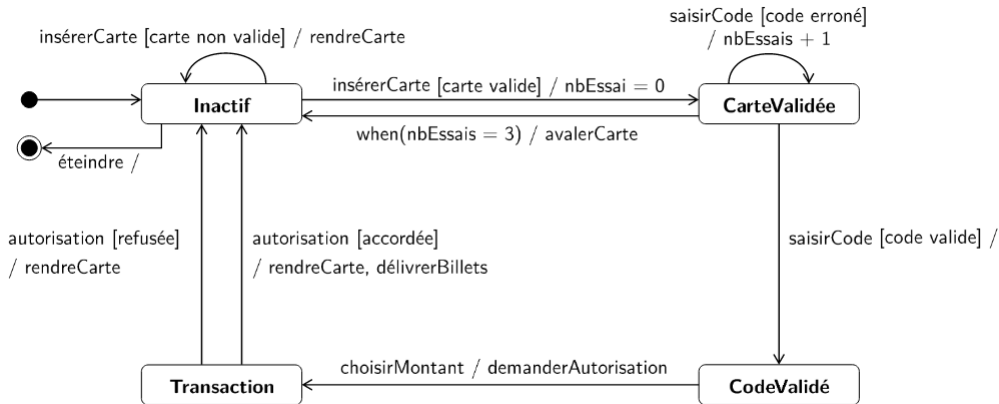
`[ <événement> ] [ <condition> ] [ / <action> ]`
- ▶ Le même événement peut être le déclencheur de plusieurs transitions quittant un même état
 - une seule transition peut se déclencher dans un même flot d'exécution (chaque transition avec le même événement doit avoir une condition de garde différente)
 - comportement non déterministe si deux transitions activées en même temps

- ▶ Au déclenchement d'une transition, l'action décrit l'effet produit
 - une opération primitive (assignation, création d'objet etc)
 - envoi d'un signal
 - appel d'une méthode etc



Source : Delphine Longuet (LRI)

Exemple complet

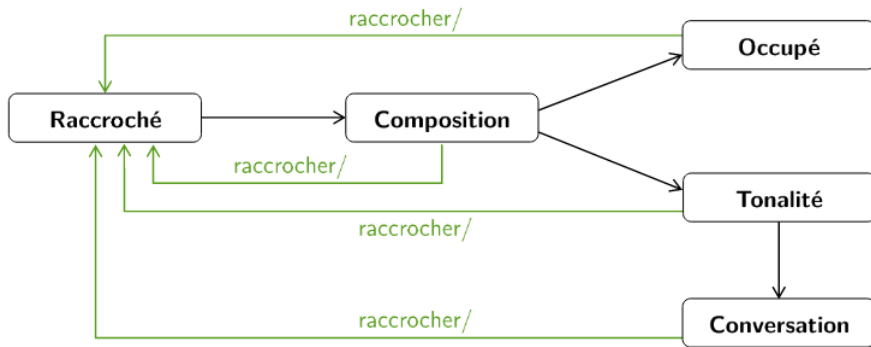


Source : Delphine Longuet (LRI)

- 1 Diagrammes de cas d'utilisation
- 2 Diagrammes de séquence
- 3 Diagrammes de classe
 - Structure générale
 - Relations entre classes
 - Relations d'association
 - Relations de généralisation/spécialisation
 - Mise en œuvre d'interfaces
 - Retour sur les classes-association
- 4 Diagrammes d'états-transitions
 - États dans un diagramme d'états-transitions
 - Événements dans un diagramme d'états-transitions
 - Transitions dans un diagramme d'états-transitions
 - États étendus

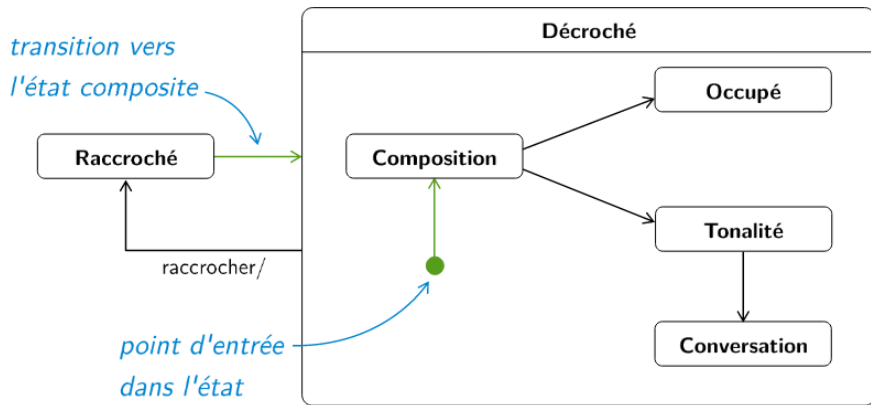
- ▶ État regroupant un ensemble d'états
- ▶ Objectifs
 - ☐ Factoriser les actions
 - ☐ Hiérarchiser les états
 - ☐ Structurer les comportements complexes

Exemple



Source : Delphine Longuet (LRI)

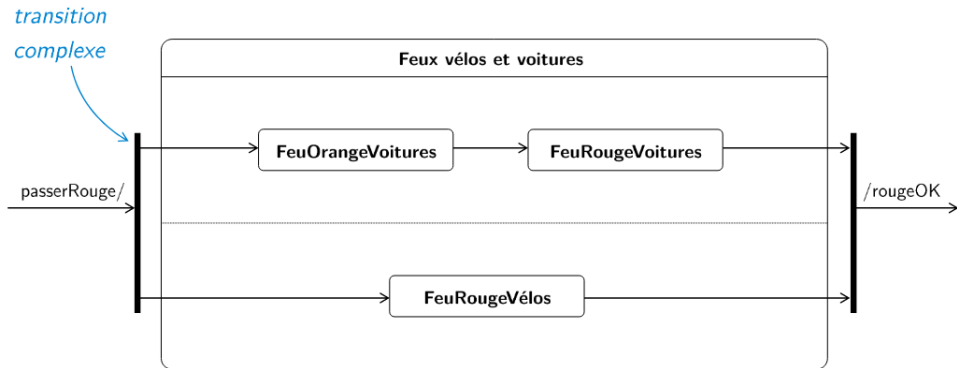
Exemple



Source : Delphine Longuet (LRI)

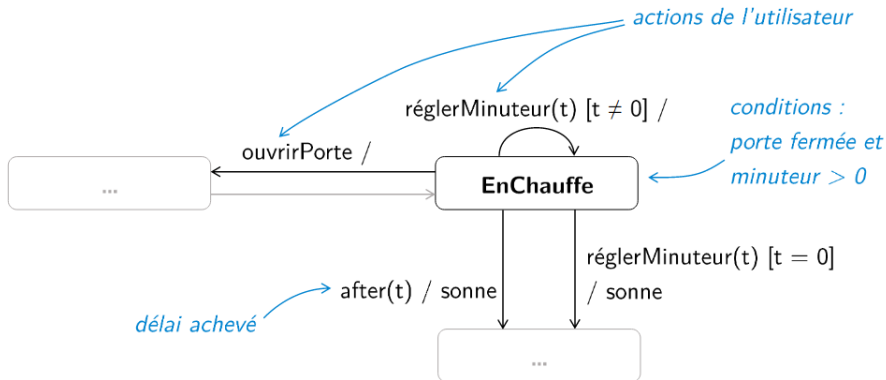
États composites : état horthogonal

- ▶ État composite dans lequel plusieurs états sont actifs simultanément (concurrence/parallélisme)
- ▶ État actif global = un état actif par région



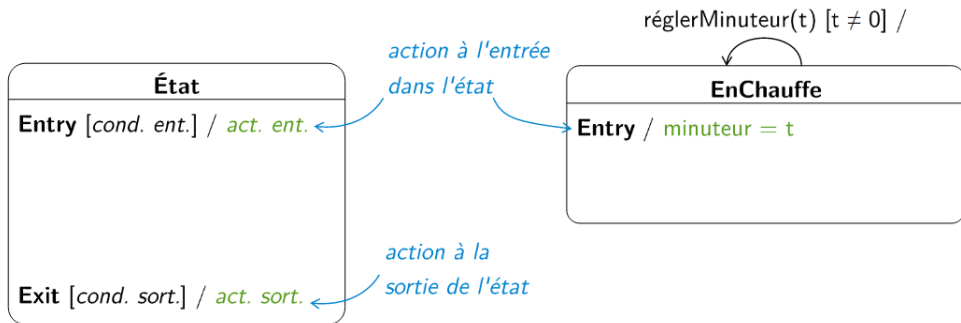
Activités et événements internes (1/2)

- Dans un état simple :
 - Conditions vérifiées qui amènent des transitions
 - Événements attendus



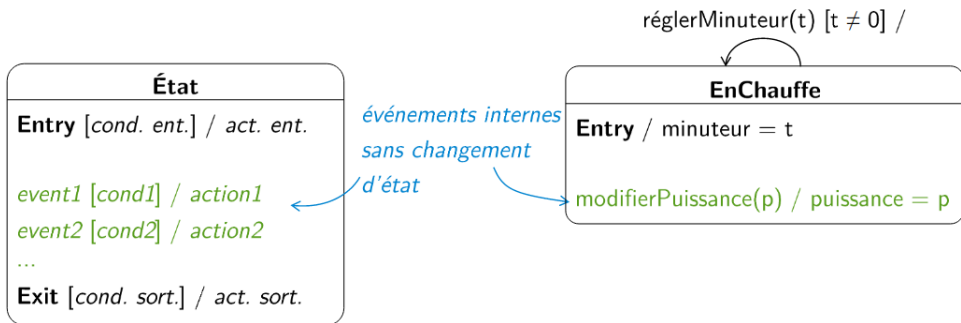
Activités et événements internes (2/2)

- ▶ Événements internes : caractéristiques supplémentaires d'un état
 - à l'entrée, à la sortie



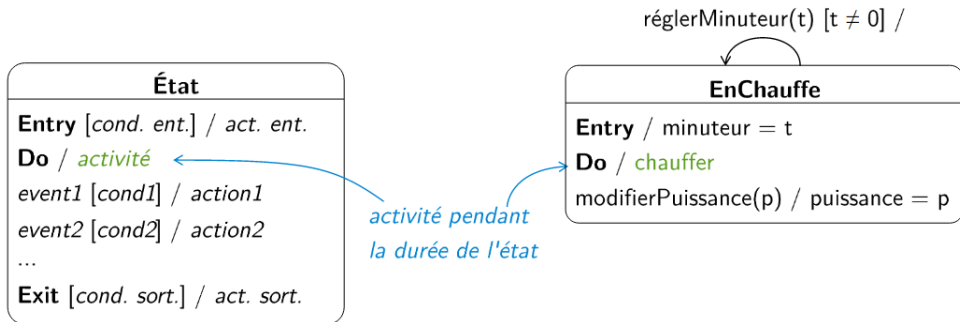
Activités et événements internes (2/2)

- ▶ Événements internes : caractéristiques supplémentaires d'un état
 - à l'entrée, à la sortie
 - pendant l'état



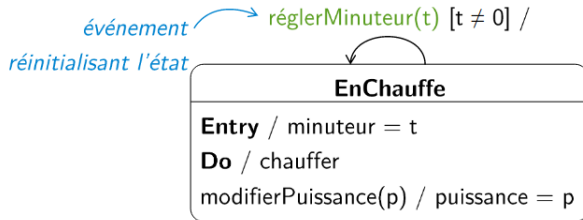
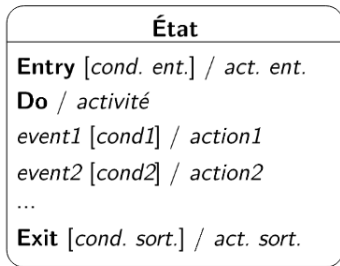
Activités et événements internes (2/2)

- ▶ Événements internes : caractéristiques supplémentaires d'un état
 - à l'entrée, à la sortie
 - pendant l'état
- ▶ Activité

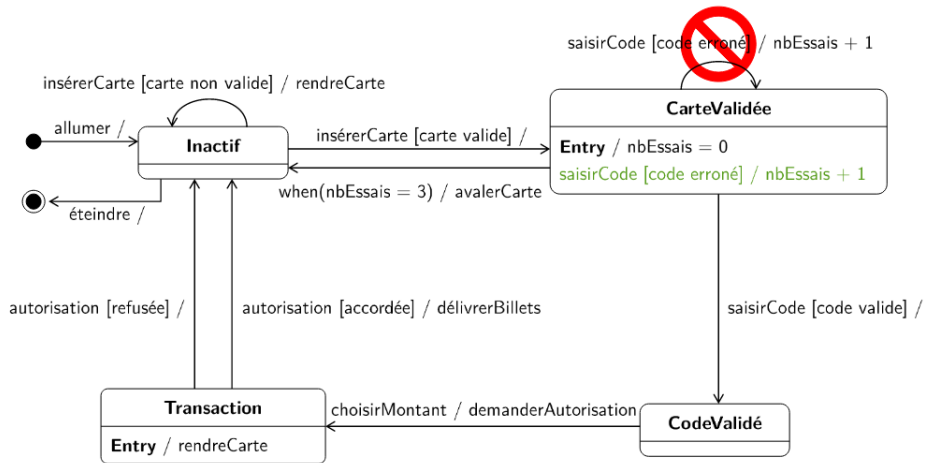


Activités et événements internes (2/2)

- ▶ Événements internes : caractéristiques supplémentaires d'un état
 - à l'entrée, à la sortie
 - pendant l'état
- ▶ Activité
- ▶ Réinitialisation de l'état par des événements externes



Exemple de l'utilité des événements internes



Source : Delphine Longuet (LRI)

- ▶ En phase d'analyse
 - ☐ Description de la dynamique d'un système vu de l'extérieur
 - ☐ Synthèse des scénarios liés aux cas d'utilisation
 - ☐ Événements = actions des utilisateurs

- ▶ En phase de conception
 - ☐ Description de la dynamique d'un objet donné
 - ☐ Événements = appels d'opérations