Conception Orientée Objets

Cas d’utilisations / Use Cases

Objectifs

- Montrer comment capturer les exigences des utilisateurs
- Lire et interpréter les *Use Case*s.

Le comportement du systeme

- C’est l’ensemble des actions et réactions du système
  - L’ensemble de la fonctionnalité et des responsabilités du **système** et de son **environnement** sont capturés par un diagramme de *UseCase*.
  - L’environnement est l’ensemble des **acteurs** (personnes, logicielles, machines) qui interagissent avec le système et ne sont **PAS** à concevoir !
- Les diagrammes d’activités
  - Capture l’ensemble des actions à réaliser, l’algorithme à réaliser

Qu’est-ce que le diagramme des cas d’utilisation?

Le diagramme de cas d’utilisation représente la structure des grandes fonctionnalités nécessaires aux utilisateurs du système

Le diagramme des cas d'utilisation (***Use Case Diagram***) constitue la première étape de l’analyse UML en :

-Modélisant les besoins des utilisateurs.
-Identifiant les grandes fonctionnalités et les limites du système.
-Représentant les interactions entre le système et ses utilisateurs.

\+ Le diagramme des cas d’utilisation apporte une vision utilisateur et absolument pas une vision informatique. Il ne nécessite aucun connaissance informatique et l’idéal serait qu’il soit réalisé par le client.

\- Le diagramme des cas d’utilisations n’est pas un inventaire exhaustif de toutes les fonctions du système. Il ne liste que des fonctions générales essentielles et principales sans rentrer dans les détails.

Qu’est-ce que UML?

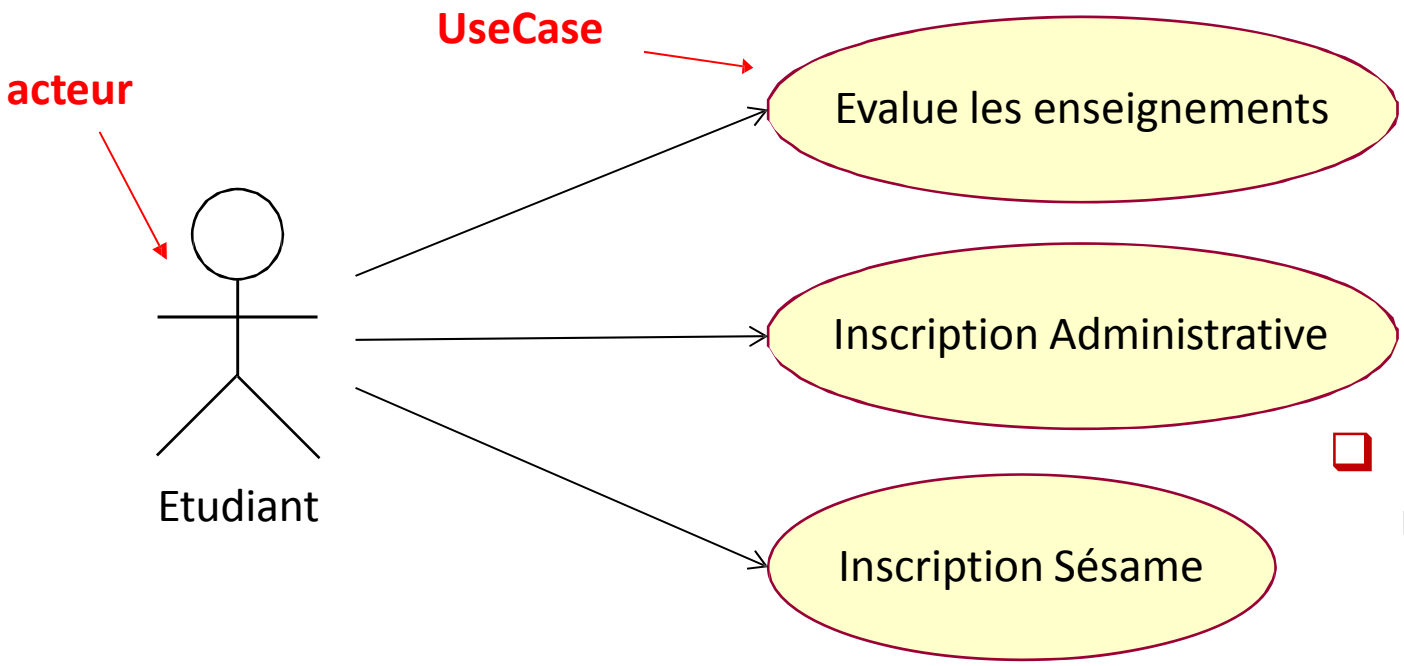
UML (Unified Modeling Language) :

- a été créé pour être le langage standard de modélisation orienté- objet.

- contient différents diagrammes utilisés pour décrire de nombreux aspects du logiciel.

Qu'est-ce qu'un modèle *UseCase*?

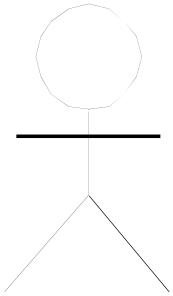
- C'est un modèle qui décrit les exigences fonctionnelles du système



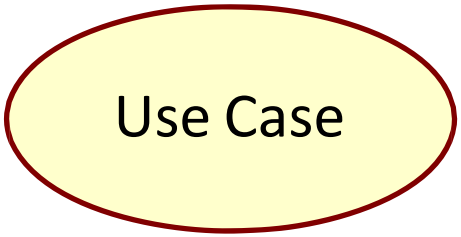
- Exemples d'exigences **non**-fonctionnelles
  - Coût du projet, contraintes de portabilité, etc

Acteurs et *Use Cases*

- Un acteur représente tout ce qui peut réagir avec le système
  - On peut adapter l'icône à la nature de l'acteur
- Un *use case* décrit une séquence d'événements réalisés par le système et qui conduit à un résultat observable par les acteurs concernés.
  - Ce que le système fait
  - **Pas** comment il le fait
- Un ensemble de relations
  - Interactions entre les use cases et les acteurs

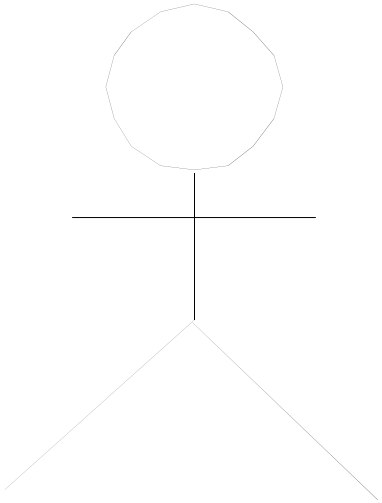


Acteur



Qu’est-ce qu’un acteur?

- C’est le **rôle** que peut jouer l’utilisateur d’un système.
  - 1 entité peut avoir plusieurs rôles
- Un humain, une machine ou un autre système.
- Interagit activement avec le système.
  - Il peut fournir des informations
  - Il peut recevoir « passivement » des informations
- Les acteurs sont **externes** au système.



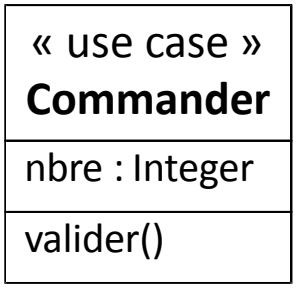
Acteur

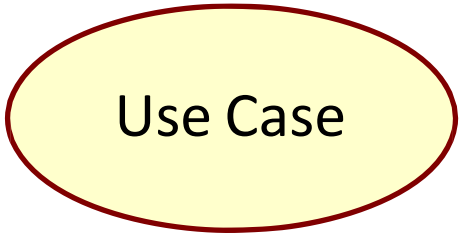
« actor »
Client

Client

Qu'est-ce qu'un *Use Case*?

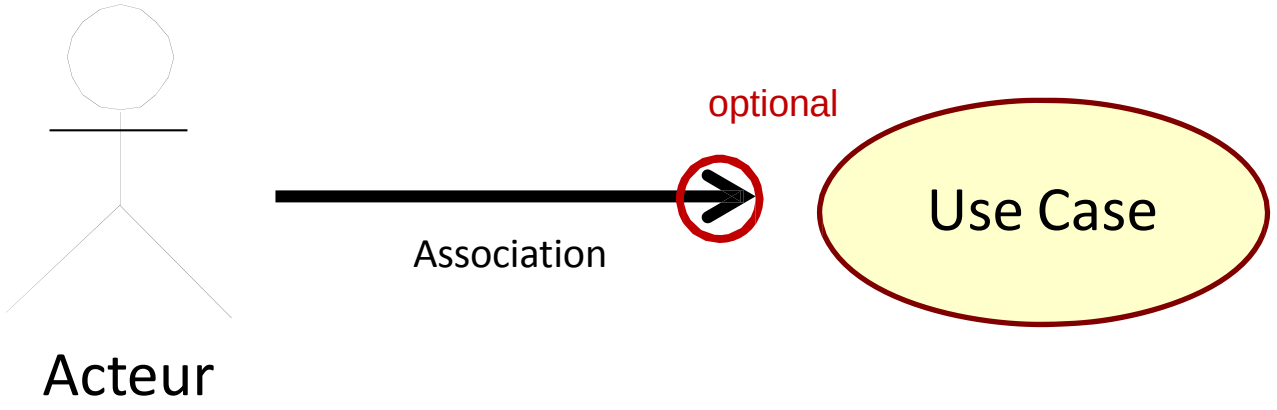
- Un ensemble d'instance de *Use Case*
  - Chaque instance est une séquence d'actions qu'un système réalise et qui produit un résultat visible par au moins un acteur.
  - Un *Use Case* représente un dialogue entre un ou plusieurs acteurs et le système.
  - Un *Use Case* décrit les actions prises par le système pour délivrer un résultat à un acteur.
  - Ni trop petit, ni trop grand.





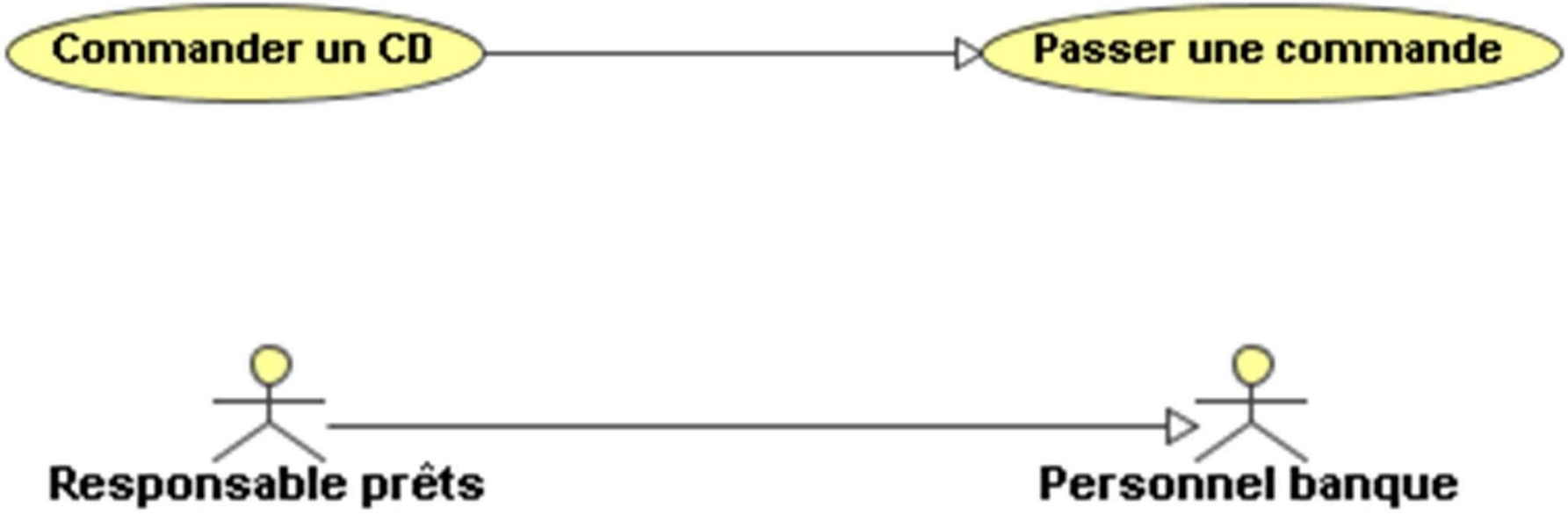
Use Cases et Acteurs

- Un *Use Case* représente un dialogue entre les acteurs et le système.
- Un *Use Case* est initié par un acteur.



La généralisation

- On peut généraliser des Use Cases ou des acteurs

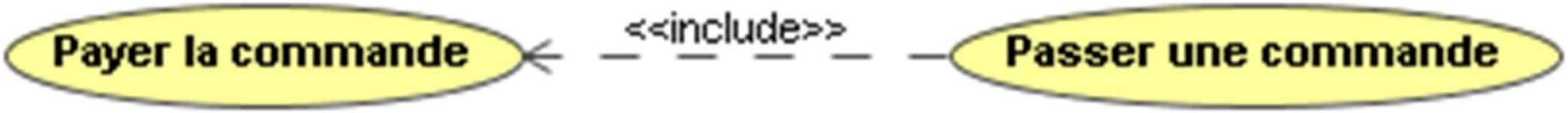


spécifique

général

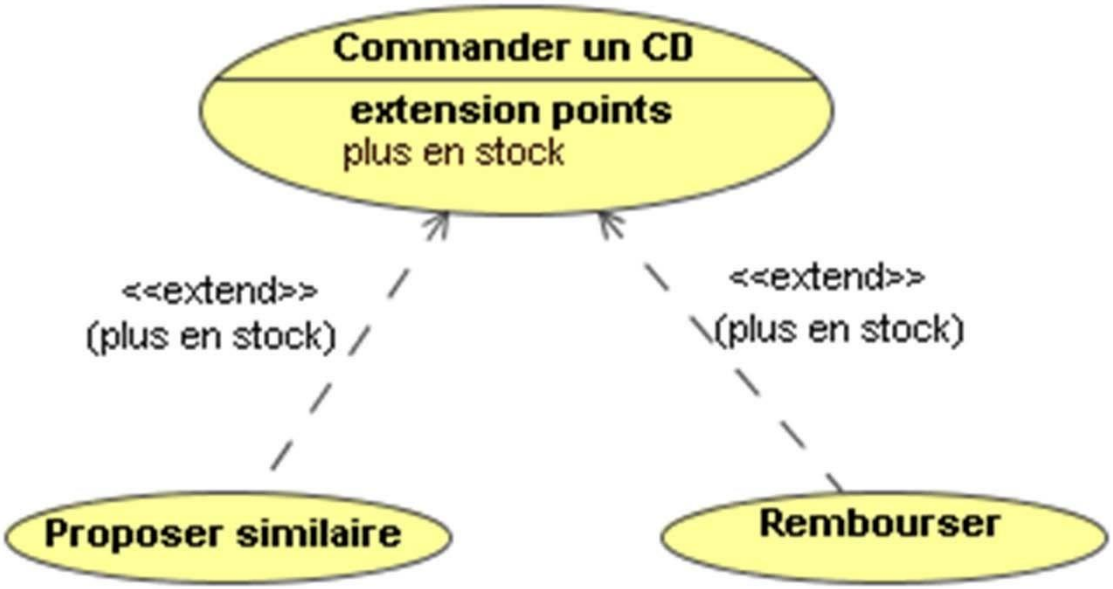
L'inclusion de *Use Case*

- Un *Use Case* inclus est une sous-fonction obligatoire de celui dans lequel il est inclus.
  - On ne peut pas passer une commande sans payer
  - Décompose la complexité d'un *Use Case*.



L'extension de *Use Case*

- Les *Use Case* peuvent définir des points d'extensions optionnels
  - Un *Use Case* peut être exécuté sans que ses points d'extensions ne soient réalisés.
  - Un Use Case peut avoir plusieurs extensions.
  - Plusieurs *Use Case* peuvent étendre un point d'extension donné.



Comment identifier les acteurs ?

- Acteur n'est pas forcément utilisateur
  - Oublier un acteur => se tromper sur l'interface
  - Le nom de l'acteur reflète son **rôle**
- Trouver les différents rôles des utilisateurs
  - Responsable clientèle, responsable d'agence, administrateur…
- Identifier les autres systèmes (imprimantes, logiciel)
- Vérifier que les acteurs interagissent avec le système
  - Les clients d'un magasin ne sont pas des acteurs pour le système de la caisse
  - La caissière est l'acteur

Comment identifier les cas d'utilisation?

- Description exhaustive des exigences fonctionnelles
  - Se placer du point de vue de chaque acteur
  - Comment et pourquoi il se sert du système
- Bon niveau d'abstraction
  - Ni trop détaillé, ni trop grossier
- Règle de nommage
  - Verbe à l'infinitif + complément
  - Du point de vue de l'acteur (pas du système)
  - Ex: retirer de l'argent ou distribuer de l'argent

Description textuelle (1/2)

- Première partie: générale
  - Nom: verbe infinitif + complément
  - Objectif:
    - Description résumée
    - Renseignée au début du projet
  - Acteurs principaux: « primary »
    - Ceux qui réalisent le cas d'utilisation
  - Acteurs secondaires: « secondary »
    - Ceux qui ne font que recevoir de l'information
  - Dates et version: création et mises à jour
  - Responsables

Description textuelle (2/2)

- Deuxième partie: fonctionnement nominal et dégradé
  - Préconditions: état du système pour déclencher
  - Ensemble de scénarios: (cf. diagrammes d'interactions)
    - Séquence d'échanges
    - Scénario nominal (sans erreurs)
    - Autres scénarios potentiels
  - Postconditions:
    - L'état du système à l'issue des différents scénarios
- Troisième partie
  - Spécifications non fonctionnelles (coût, durée des actions)
  - Interface graphique (maquette de l'IHM pour chaque use case)