# *Digramme de cas d'utilisation*

**Introduction**

Formalisés par Ivar Jacobson dans Object-Oriented Software Engineering (Addison-Wesley, 1992), les cas d'utilisation ('use cases') servent à exprimer le comportement du système en termes d'actions et de réactions, selon le point de vue de chaque utilisateur (« approche centrée utilisateur »).

**Rôle**

Les cas d'utilisation délimitent le système, ses fonctions (ses cas), et ses relations avec son environnement. Ils constituent un moyen de déterminer les besoins du système. Ils permettent d'impliquer les utilisateurs dès les premiers stades du développement pour exprimer leurs attentes et leurs besoins (analyse des besoins). Ils constituent un fil conducteur pour le projet et la base pour les tests fonctionnels

**Les cas d'utilisations décrivent les fonctionnalités fournies par le système à un acteur.**

Les cas d'utilisation sont utilisés par les clients, les concepteurs, les développeurs, et les testeurs. Les fonctionnalités d'un système sont décrites dans le modèle des besoins (use case view) par un ensemble de cas d'utilisation. On un cas d'utilisation est une description générique d'une utilisation du système (les scénarios en merise). Il doit fournir un service à l'acteur.

Un cas d'utilisation modélise un **dialogue** entre un acteur et le système. Il représente une fonctionnalité importante proposée par le système décrite de son initialisation à sa terminaison

Définition 1: Un cas d'utilisation est une manière d'utiliser le système «un besoin utilisateur». Une façon générale d'utiliser une partie de la fonctionnalité du système.

Définition 2: Un cas d'utilisation est une séquence de transactions offertes par le système. Description d'un ensemble de séquences d'actions, comportant éventuellement des variantes, que le système exécute pour produire un résultat tangible et qui a de la valeur pour l'utilisateur.

**Représentation**

Un cas d'utilisation se représente par une ellipse contenant le nom du cas (un verbe à l'infinitif), et optionnellement, au-dessus du nom.

**Flots d'évènements**

Chaque cas d'Utilisation est décrit par un flot d'évènements.

Les Flots d'évènements décrit ce que le système **doit** faire et pas comment il le fait.

Les flots d'évènements doivent contenir :

- Quand et comment le cas d'utilisation démarre et se termine
- Les interactions du cas d'utilisation et des acteurs
- Quelles sont les données nécessaires au cas d'utilisation
- la séquence d'évènement normale
- la description de tous les flots d'évènements alternatifs ou exceptionnels (erreurs).

La description des flots d'évènements est une opération importante qui doit être mené de façon itérative avec le client.

**Acteur**

Définition : Un acteur est une personne ou un système extérieur au système en cours de modélisation qui interagit avec notre système.

Un acteur est une personne ou un système qui interagit avec le système étudié, en échangeant de l'information (en entrée et en sortie). On trouve les acteurs en observant les utilisateurs directs du système, les responsables de sa maintenance, ainsi que les autres systèmes qui interagissent avec lui. Un acteur représente un rôle joué par un utilisateur qui interagit avec le système. La même personne physique peut jouer le rôle de plusieurs acteurs. D'autre part, plusieurs personnes peuvent jouer le même rôle, et donc agir comme un même acteur.

**Rôle**

On utilise aussi les acteurs pour modéliser des systèmes externes comme une imprimante, un autre logiciel, le système d'exploitation, le système de gestion de fichier, un réveil matin avec un branchement tcp/ip, etc. ...

**Représentation**

Il se représente par un petit bonhomme avec son nom (*i.e.* son rôle) inscrit dessous.

**Concepts de base : relations entre cas d'utilisation**

Association :

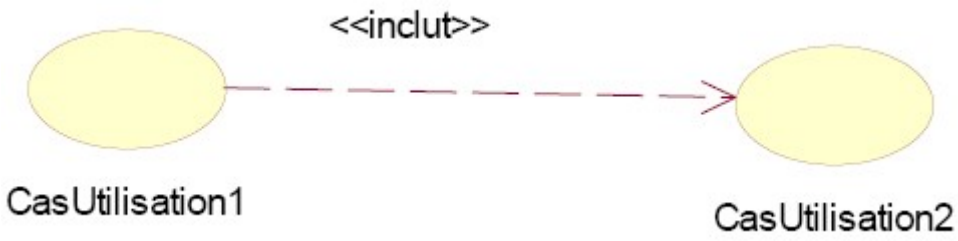
Relation entre un acteur et un cas d'utilisation

Un acteur déclenche un cas d'utilisation :


**Relation d'inclusion** :

- Entre cas d'utilisation

- L'instance du CU source contient aussi le comportement décrit par le CU destination.




- La relation d'inclusion a un caractère obligatoire:

    -La source doit indiquer à quel endroit le CU cible doit être inclus

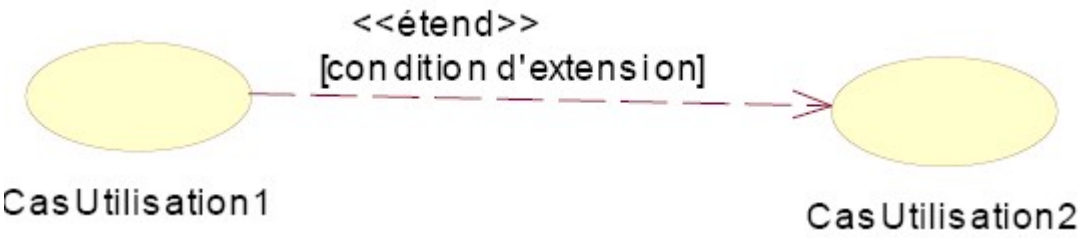
    -Permet de décomposer les comportements et définir des comportements partageables entre plusieurs CU.

Relation d'extension :

-Entre cas d'utilisation

-L'instance du CU source ajoute son comportement au CU destination (si la condition est réalisée).

-Le CU destination étend son comportement par l'ajout de celui du CU source, si la condition est vérifiée.



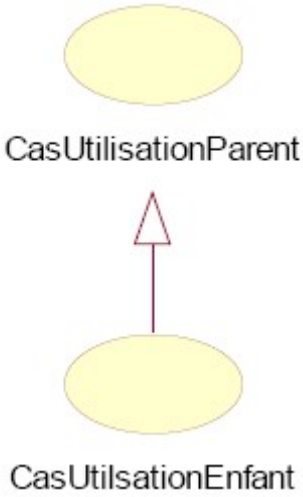

-Peut être soumise à une condition :

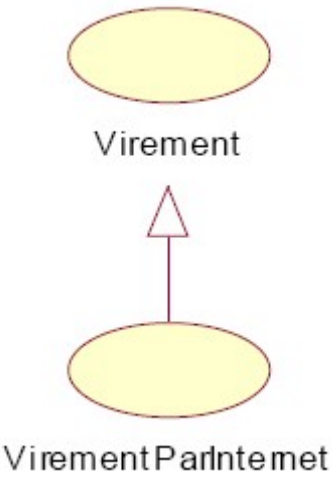
- Le comportement ajouté est inséré au niveau d'un point d'extension
- Ce point d'extension est défini dans le CU destination
- Permet de modéliser des variantes de comportement d'un CU
- Selon les interactions des acteurs et l'environnement du système

-la condition d'extension peut être spécifiée à côté du mot-clés <<étend>>

Relation de généralisation :-Entre cas d'utilisation

-Le cas d'utilisation Fils est une spécialisation du cas d'utilisation Parent




Il n'existe pas de norme (UML) établie pour la description textuelle des cas d'utilisation. Généralement, on y trouve pour chaque cas d'utilisation : son nom, un bref résumé de son déroulement, le contexte dans lequel il s'applique, les acteurs qu'il met en jeu, une description détaillée : le déroulement nominal de toutes les interactions, les cas nécessitant des traitements d'exception, les effets du déroulement sur l'ensemble du système,etc.

**Quelques conseils** :

- Un cas d'utilisation doit être :
  - Simple
  - Décrit de manière claire et concise
  - Le nombre d'acteurs interagissant avec le CU doit être limité
- Lors de la construction d'un CU, il faut se demander :
  - Quelles sont les tâches de l'acteur ?
  - Quelles informations l'acteur doit-il créer, sauvegarder, modifier, détruire ou simplement lire ?
  - L'acteur devra-t-il informer le système des changements externes ?
  - Le système devra-t-il informer l'acteur des conditions internes ?