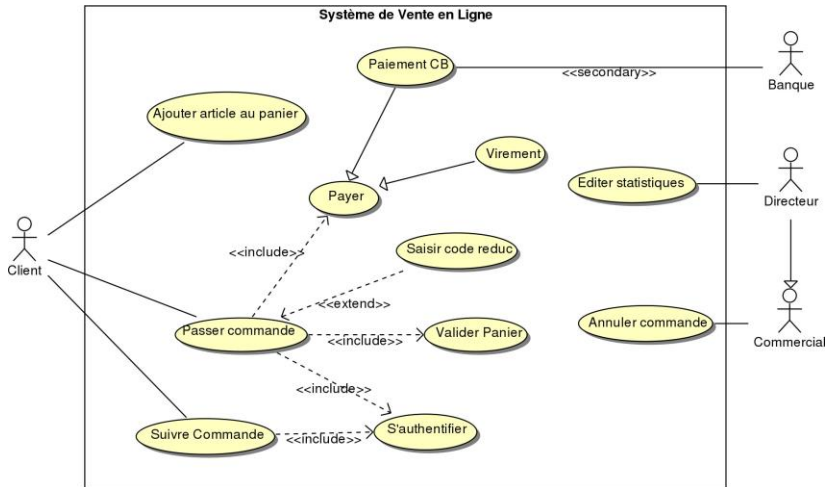


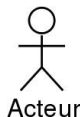
Conception objet élémentaire avec UML Diagrammes de cas d'utilisation

Avant de développer un système, il faut savoir **précisément** à *QUOI* il devra servir, *cad* à quels besoins il devra répondre.


- .. Modéliser les besoins permet de :
 - .. Faire l'inventaire des fonctionnalités attendues ;
 - .. Organiser les besoins entre eux, de manière à faire apparaître des relations (réutilisations possibles, ...).
- .. Avec UML, on modélise les besoins au moyen de **diagrammes de cas d'utilisation**.



- Un **acteur** est une entité extérieure au système modélisé, et qui interagit directement avec lui.

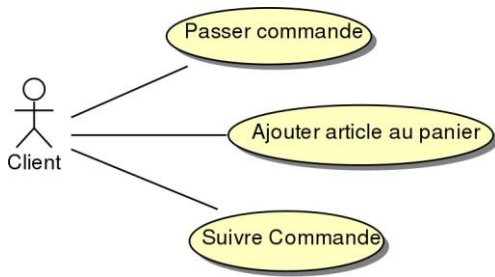


- Un **cas d'utilisation** est un service rendu à un acteur, il nécessite une série d'actions plus élémentaires.

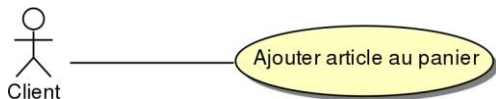
The diagram shows a yellow oval with a black border and a drop shadow, which is the standard UML notation for a use case.

Cas d'utilisation

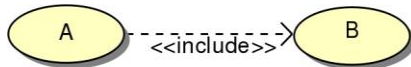
Un cas d'utilisation est l'expression d'un service réalisé de bout en bout, avec un déclenchement, un déroulement et une fin, pour l'acteur qui l'initie.



- ... Les acteurs impliqués dans un cas d'utilisation lui sont liés par une **association**.
- ... Un acteur peut utiliser plusieurs fois le même cas d'utilisation.



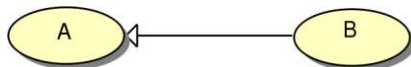
- ... **Inclusion** : le cas A inclut le cas B (B est une partie *obligatoire* de A).



- ... **Extension** : le cas B étend le cas A (B est une partie *optionnelle* de A).




- ... **Généralisation** : le cas A est une généralisation du cas du cas B (B est une sorte de A).

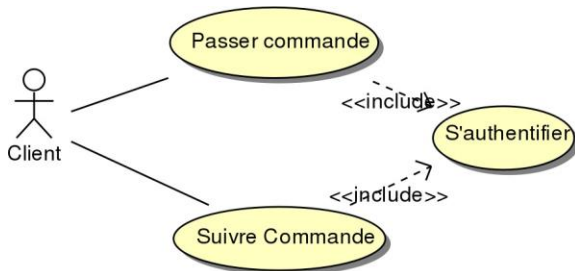


Attention

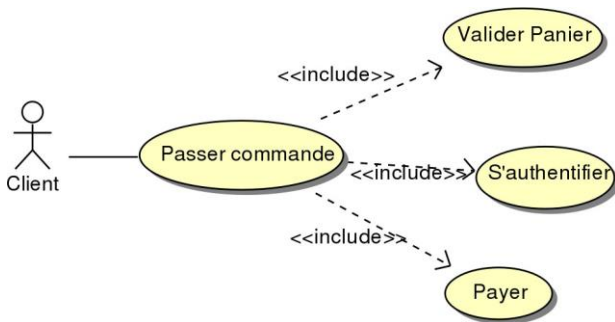
Le sens des flèches pointillées indique une dépendance, pas le sens de la relation d'inclusion.

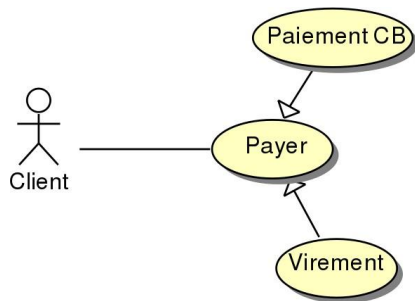
- .. Les inclusions et les extensions sont toutes deux des **dépendances**.
 - .. Lorsqu'un cas B inclut un cas A, B dépend de A.
 - .. Lorsqu'un cas B étend un cas A, B dépend aussi de A.
 - .. On note toujours la dépendance par une flèche pointillée B  A qui se lit "B dépend de A".
- .. Lorsqu'un élément B dépend d'un élément A, toute modification de A sera susceptible d'avoir un impact sur B.
- .. Les "include" et les "extend" sont des **stéréotypes** (entre guillemets) des relations de dépendance.

- ... Les relations entre cas permettent la **ré-utilisabilité** du cas "s'authentifier" : il sera inutile de développer plusieurs fois un module d'authentification.



- ... Quand un cas est trop complexe (faisant intervenir un trop grand nombre d'actions élémentaires), on peut procéder à sa **décomposition** en cas plus simples.



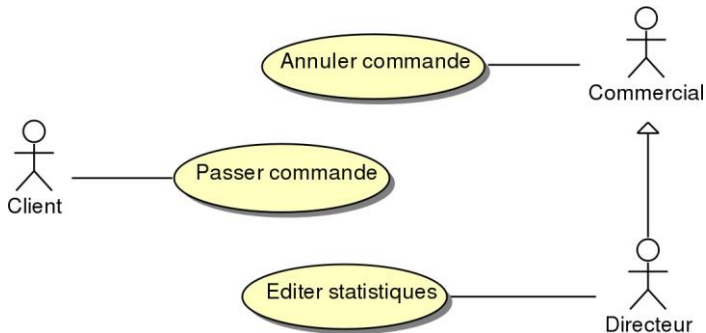


.. Un virement est un cas particulier de paiement.

Un virement est une sorte de paiement.

- .. La flèche pointe vers l'élément général.
- .. Cette relation de généralisation/spécialisation est présente dans la plupart des diagrammes UML et se traduit par le concept d'**héritage** dans les langages orientés objet.

.. Une seule relation possible : la **généralisation**.



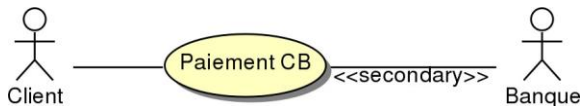
- .. Les principaux acteurs sont les utilisateurs du système.

Attention

Un acteur correspond à un **rôle**, pas à une personne physique.

- .. Une même personne physique peut être représentée par plusieurs acteurs si elle a plusieurs rôles.
- .. Si plusieurs personnes jouent le même rôle vis-à-vis du système, elles seront représentées par un seul acteur.
- .. En plus des utilisateurs, les acteurs peuvent être :
 - .. Des périphériques manipulés par le système (imprimantes...) ;
 - .. Des logiciels déjà disponibles à intégrer dans le projet ;
 - .. Des systèmes informatiques externes au système mais qui interagissent avec lui, etc.
- .. Pour faciliter la recherche des acteurs, on se fonde sur les **frontières** du système.

- L'acteur est dit **principal** pour un cas d'utilisation lorsque l'acteur est à l'initiative des échanges nécessaires pour réaliser le cas d'utilisation.



- Les acteurs **secondaires** sont sollicités par le système alors que le plus souvent, les acteurs principaux ont l'initiative des interactions.
 - Le plus souvent, les acteurs secondaires sont d'autres systèmes informatiques avec lesquels le système développé est inter-connecté.

- .. Il n'y a pas une manière mécanique et totalement objective de repérer les cas d'utilisation.
 - .. Il faut *se placer du point de vue de chaque acteur* et déterminer comment il se sert du système, dans quels cas il l'utilise, et à quelles fonctionnalités il doit avoir accès.
 - .. Il faut *éviter les redondances* et *limiter le nombre de cas* en se situant au bon niveau d'abstraction (par exemple, ne pas réduire un cas à une seule action).
 - .. Il ne faut pas faire apparaître les détails des cas d'utilisation, mais il faut rester au niveau des grandes fonctions du système.

Trouver le bon niveau de détail pour les cas d'utilisation est un problème difficile qui nécessite de l'expérience.

- ... Le diagramme de cas d'utilisation décrit les grandes fonctions d'un système du point de vue des acteurs, mais n'expose pas de façon détaillée le dialogue entre les acteurs et les cas d'utilisation.
- ... **Un simple nom est tout à fait insuffisant pour décrire un cas d'utilisation.**

Chaque cas d'utilisation doit être documenté pour qu'il n'y ait aucune ambiguïté concernant son déroulement et ce qu'il recouvre précisément.

3 Description textuelle des cas d'utilisation

Ce n'est pas obligatoire, mais il est recommandé de rédiger une description textuelle de chaque cas d'utilisation afin de les détailler.

Une description textuelle classique se compose de trois parties :

- Partie 1 : Identification
- Partie 2 : Description des scénarios
- Partie 3 : Exigence non fonctionnelle

3.1 Partie 1 : Identification

- Identification

- o **Titre** : Nom du cas d'utilisation
- o **Résumé** : description du cas d'utilisation.
- o **Acteurs** : descriptions des acteurs principaux et secondaires.
- o **Dates** : Date de création et date de mise à jour.
- o **Responsable** : Noms du ou des responsables.
- o **Version** : Numéro de la version.

3.2 Partie 2 : Description des scénarios

- Description des scénarios.

- o Les **pré-conditions** : Etat du système avant que le cas d'utilisation puisse être déclenché.
- o Les **Scénarios** (*un scénario est une instance d'un cas d'utilisation dans lequel tous les paramètres ont été fixés*). Il y a plusieurs **types de scénarios** :
 - **Nominale** qui correspond à un déroulement normale d'un cas d'utilisation.
 - **Alternatifs** qui sont des variantes du scénario normal.
 - **D'exceptions** qui décrivent ce qui se passe lors d'une erreur.
- o Les **post-conditions** : Elles décrivent l'état du système après l'issue de chaque scénario.

3.3 Partie 3 : Exigence non fonctionnelle

- Exigence non fonctionnelle.

La partie 3 peut être omise, mais si elle est présente, elle permet de préciser des spécifications non fonctionnelles (fréquence, fiabilité, type d'interface homme-machine...).