# Introduction au Génie Logiciel

## Séance 2 : UML, Diagramme d'utilisation et d'activité

L. Laversa
laversa@irif.fr

Université Paris Cité

4 février 2025

. Remerciements à E. Bigeon et D. Longuet

# Unified Modeling Language (UML)

"Une image vaut mille mots."

## Objectif

Modéliser des logiciels et systèmes informatiques pour faciliter les échanges lors de la conception et de la maintenance.

## Solution

Des diagrammes, avec un format unifié pour faciliter la lisibilité par les divers acteurs, et qui permettent d'effectuer des représentations partielles du logiciel.

# Différents types de diagrammes

- Structure :
  - diagrammes de classes
  - diagrammes de déploiement
  - ...
- Comportement :
  - diagrammes de cas d'utilisation
  - diagrammes d'activité
  - diagrammes d'états-transitions
  - ...
- Interaction :
  - diagrammes de séquences
  - ...

# Cas d'utilisation

## Objectif

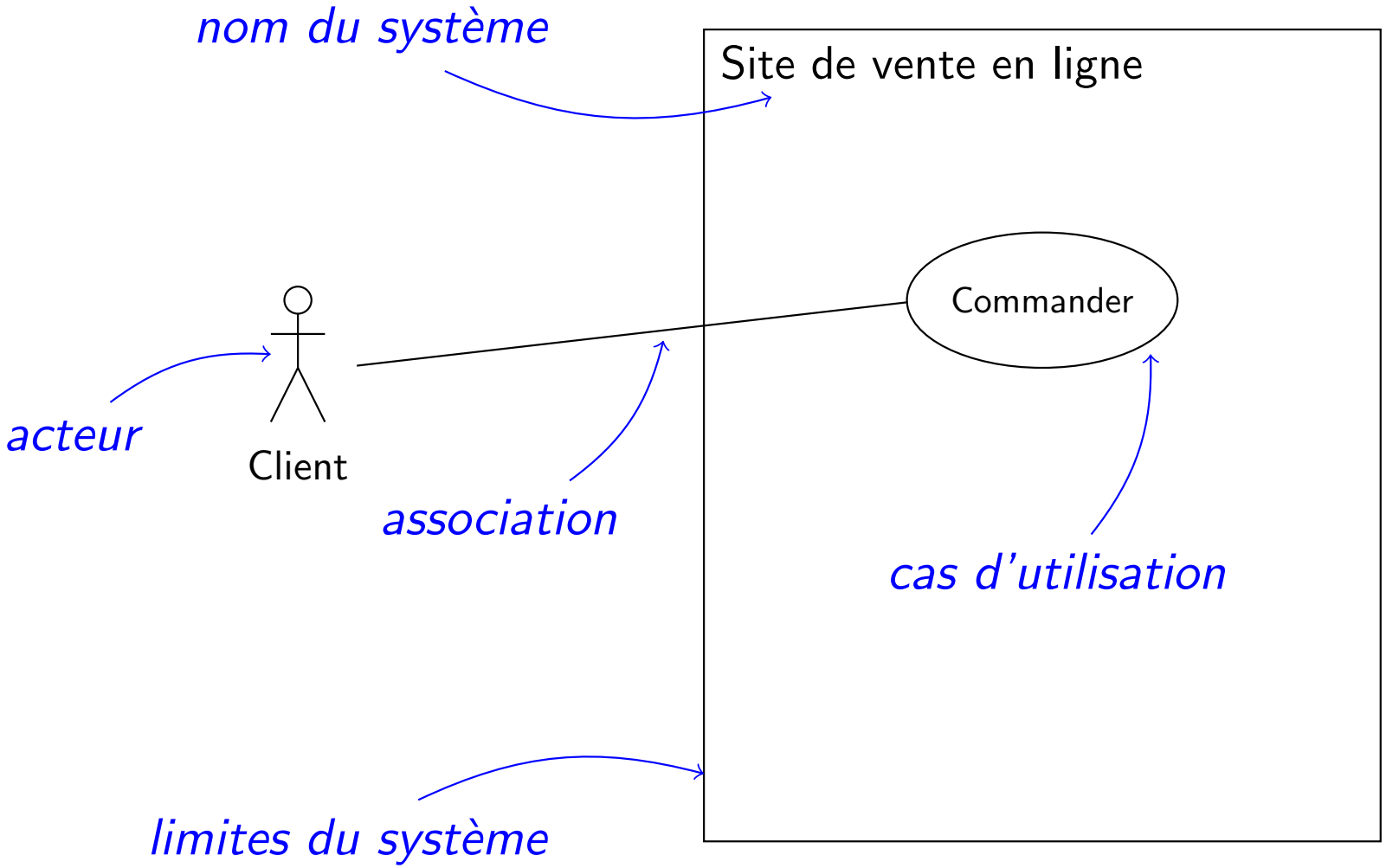
Comprendre les besoins du client pour rédiger le cahier des charges.

1. Définir les utilisations principales du systèmes : à quoi sert-il ?
2. Définir l'environnement du système : qui va l'utiliser ou interagir avec lui ?
3. Définir les limites du système : où s'arrête sa responsabilité ?

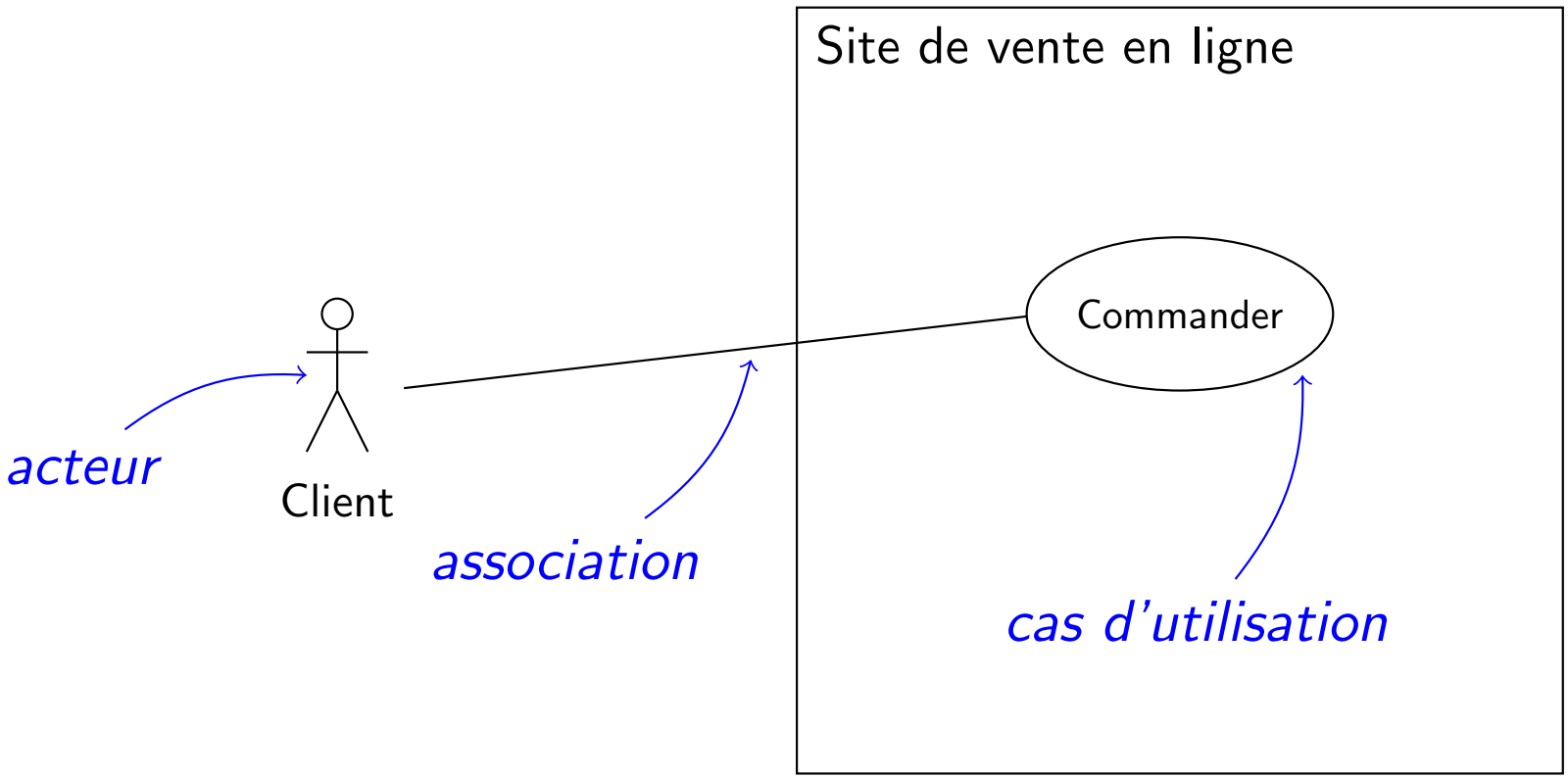
Outils :

- Diagramme de cas d'utilisation
- Description textuelle des cas d'utilisation

# Éléments du diagramme

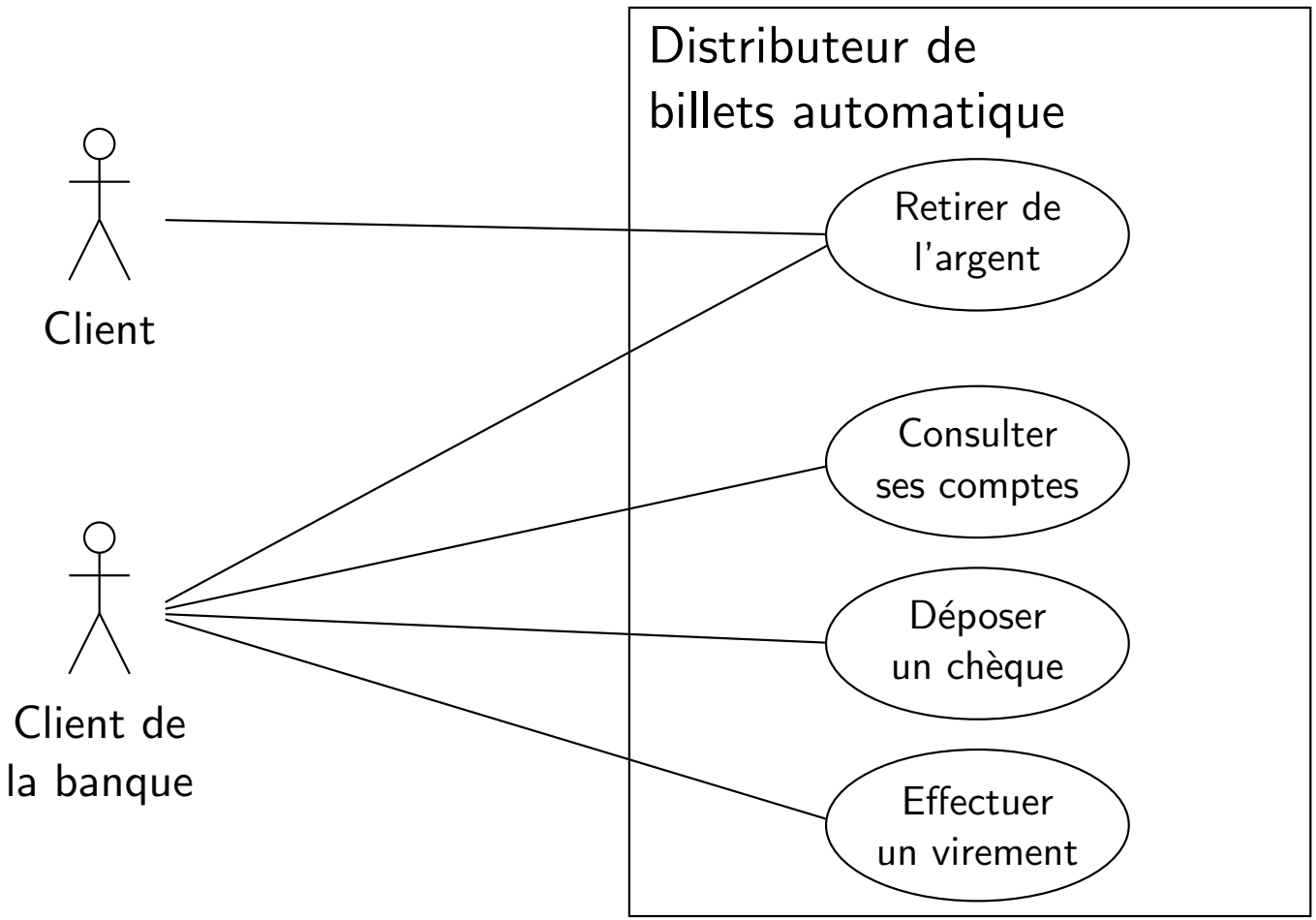


# Association

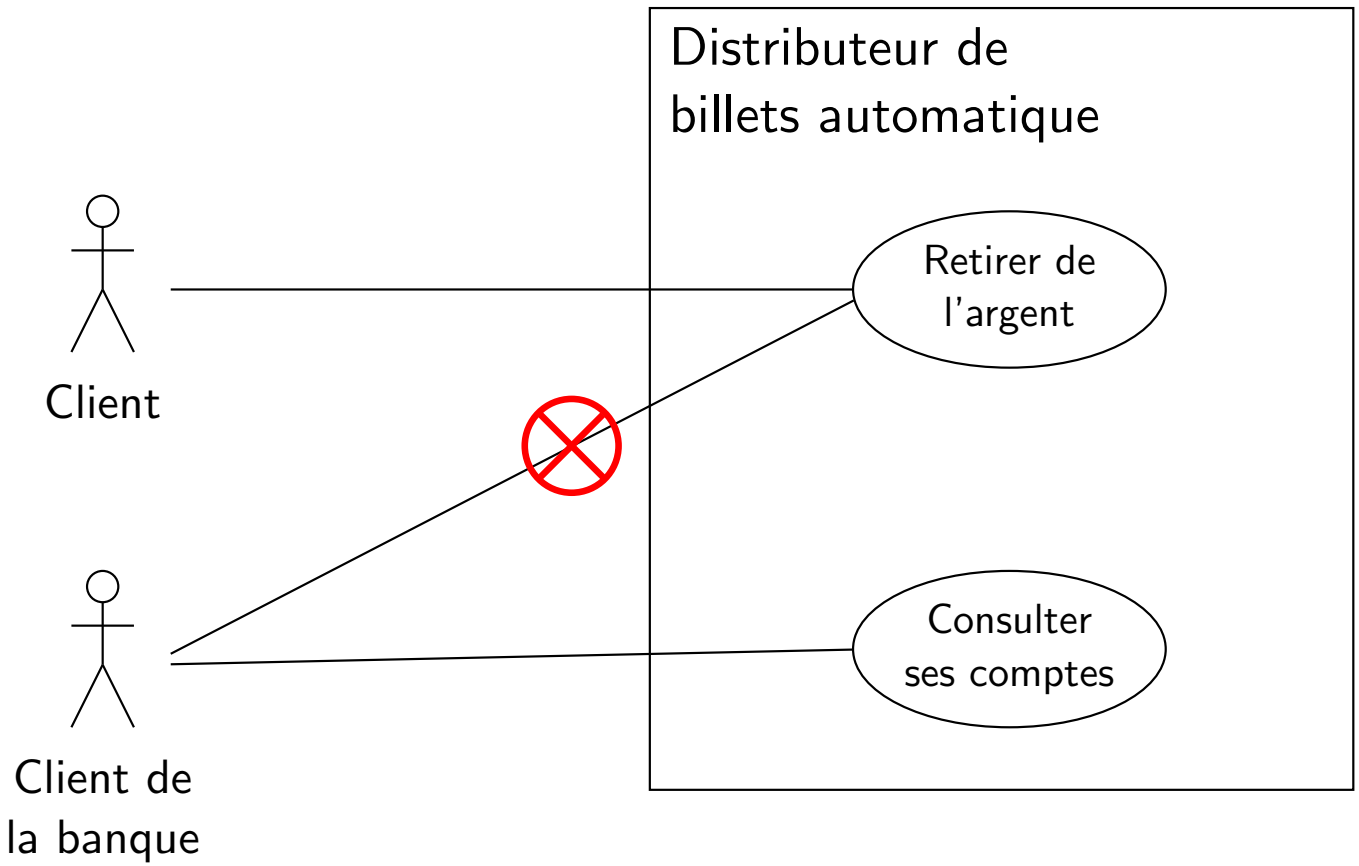



- Relation entre acteurs et cas d'utilisation
- Représente la possibilité pour l'acteur de déclencher le cas

# Exemple du distributeur

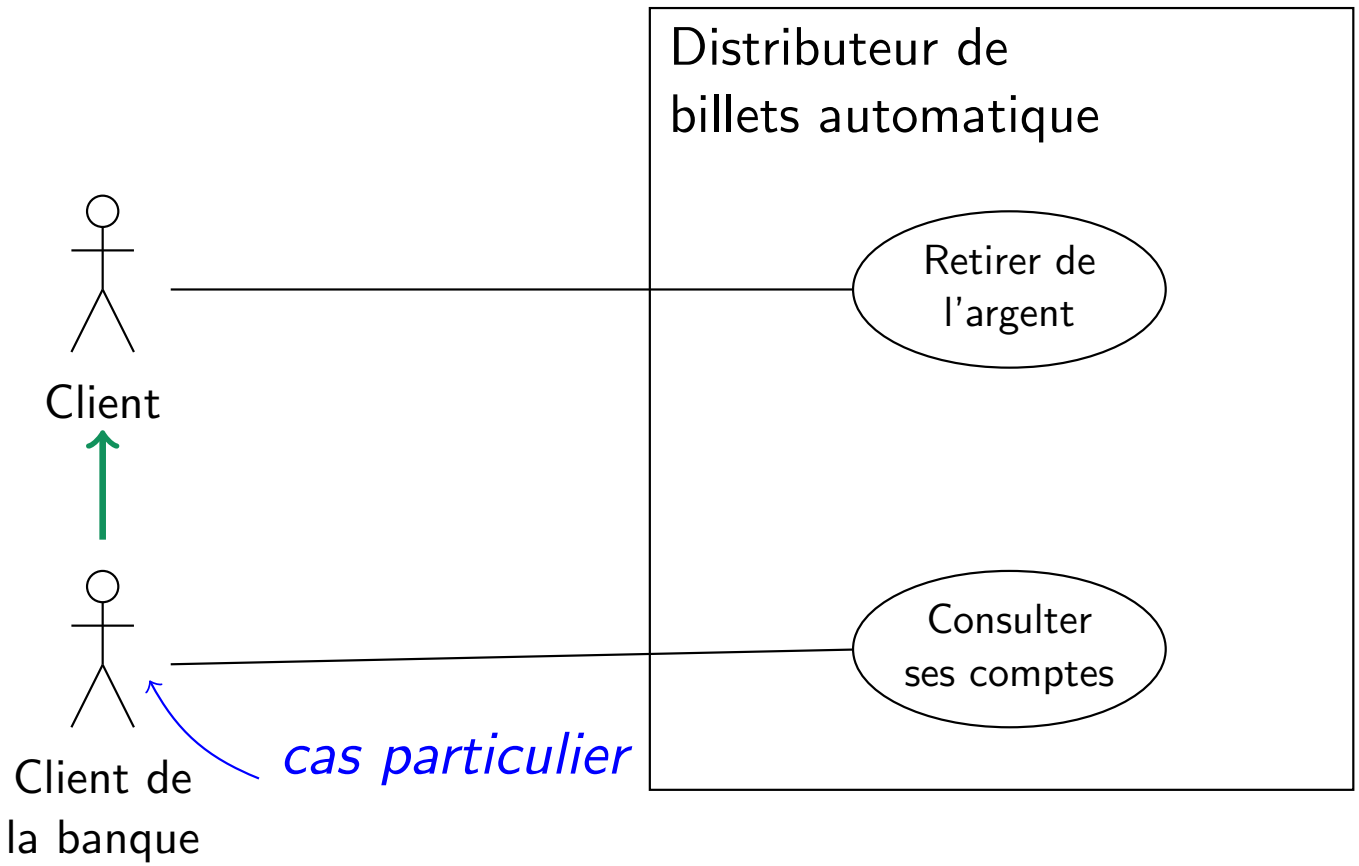


# Généralisation



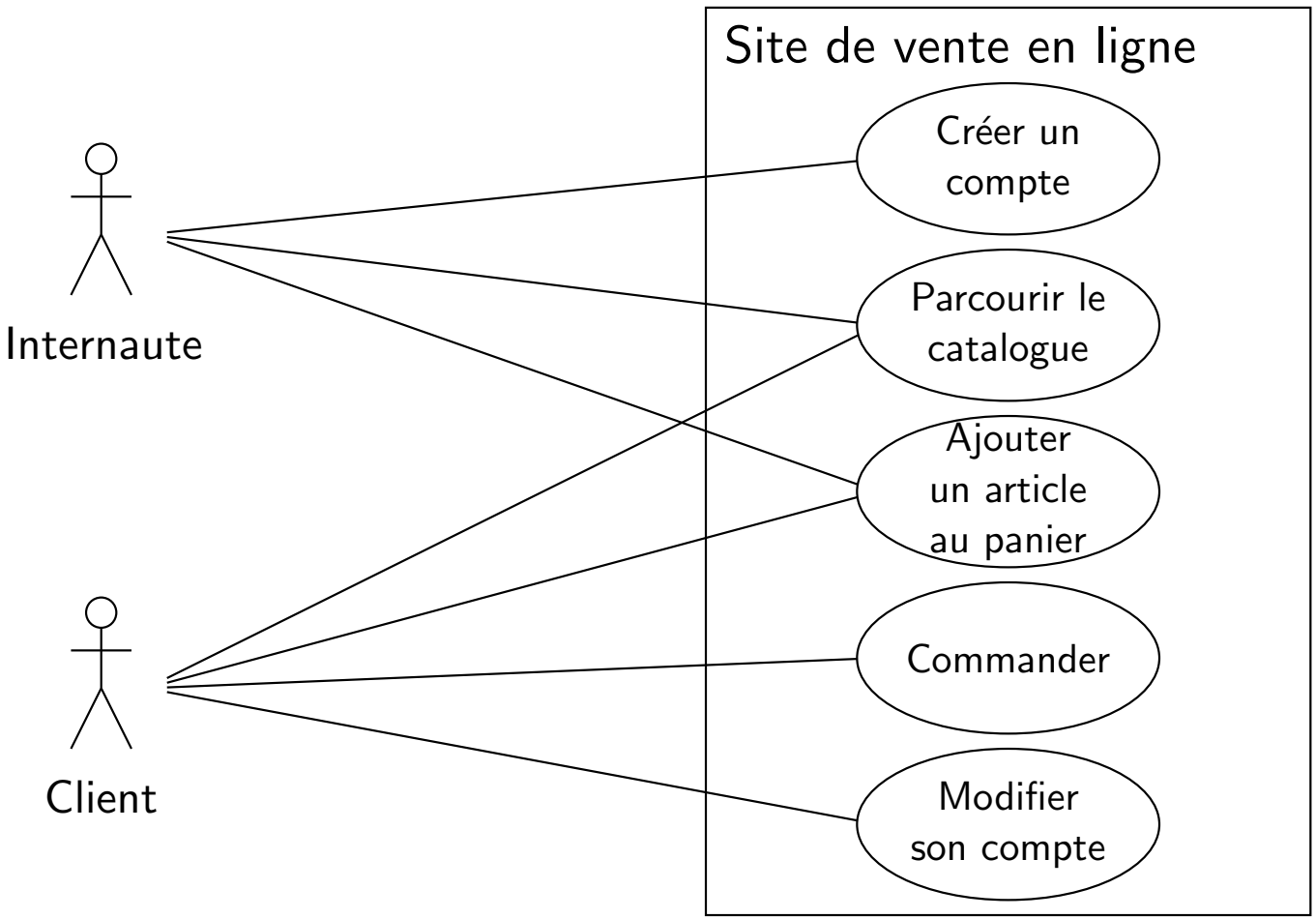

- *Client de la banque* peut faire tout ce que fait *Client* (+ d'autres choses)

# Généralisation

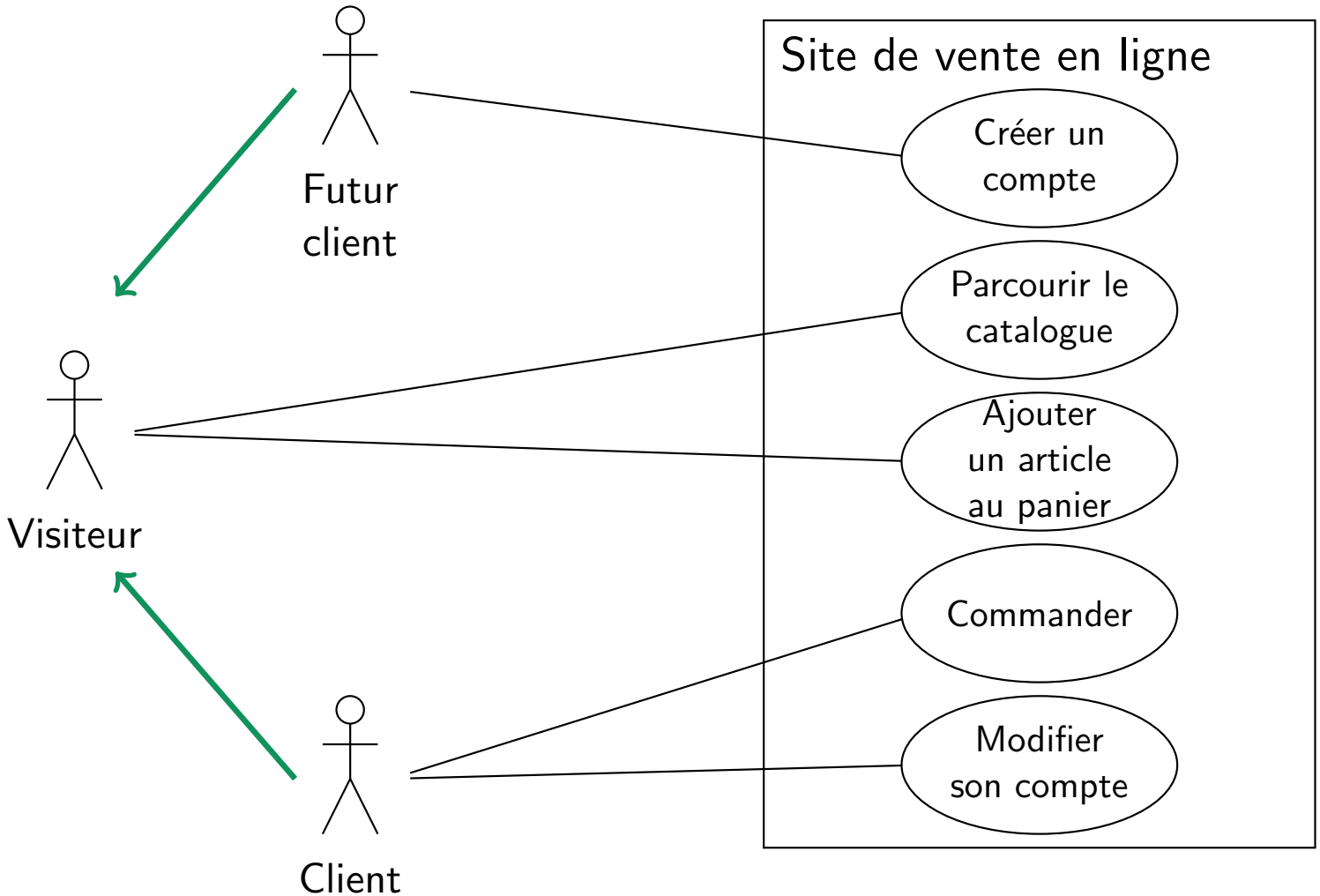



- Alors *Client de la banque* est un cas particulier de *Client* (ou *Client* une généralisation de *Client de la banque*)

# Autre exemple de généralisation



# Autre exemple de généralisation

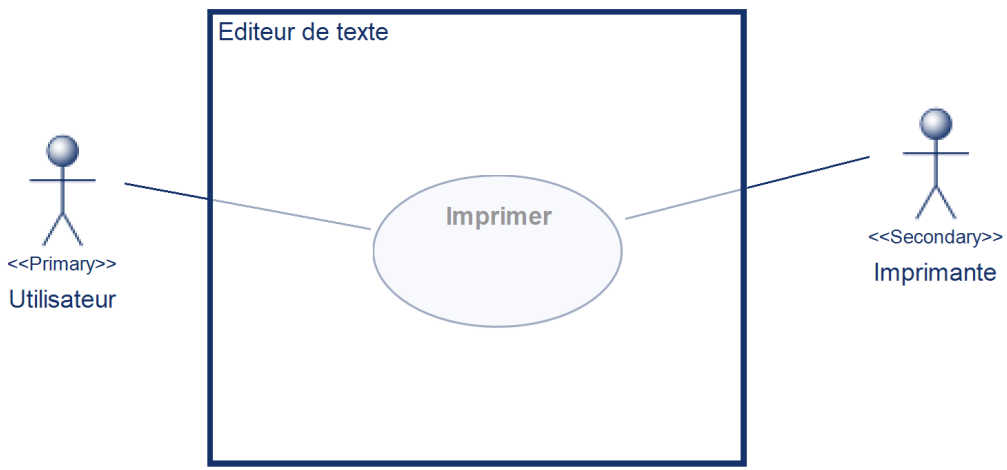


# Positionnement des acteurs

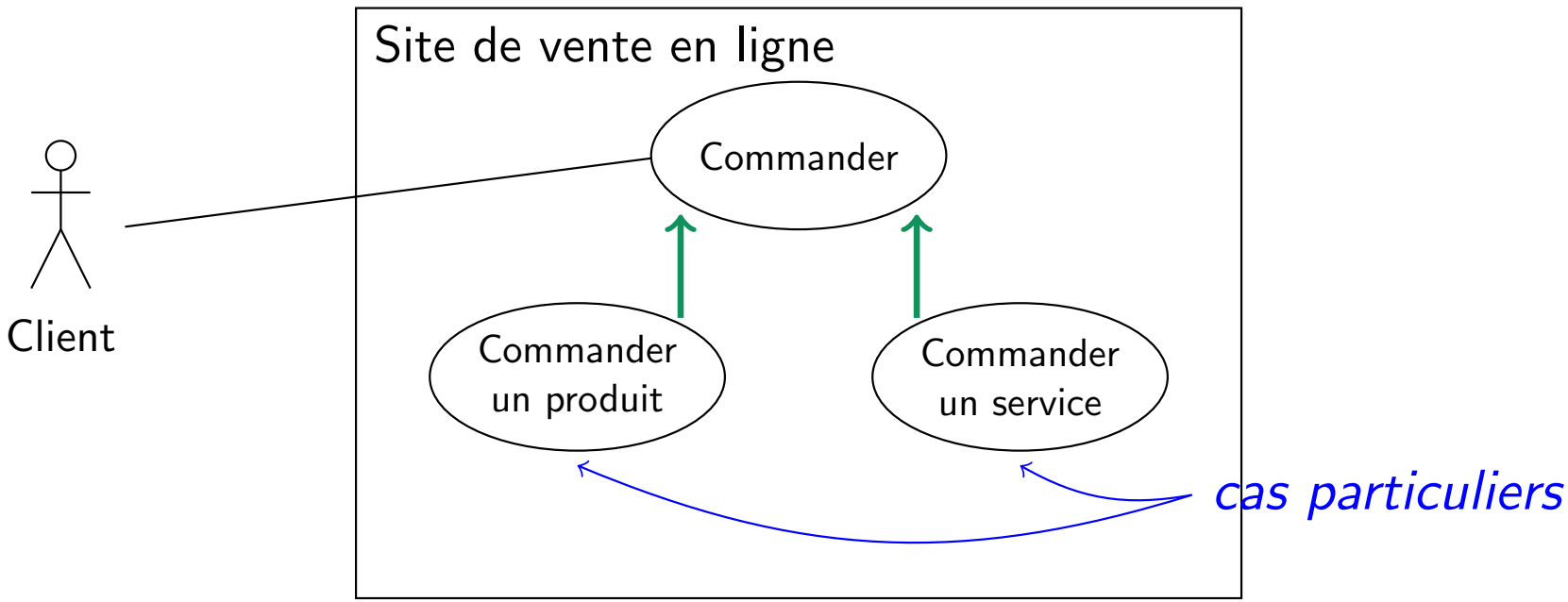
## Convention

On distingue

- acteurs consommateurs (*primary*), positionnés à gauche du système, et
- acteurs fournisseurs (*secondary* ou *supporting*), positionnés à droite du système.



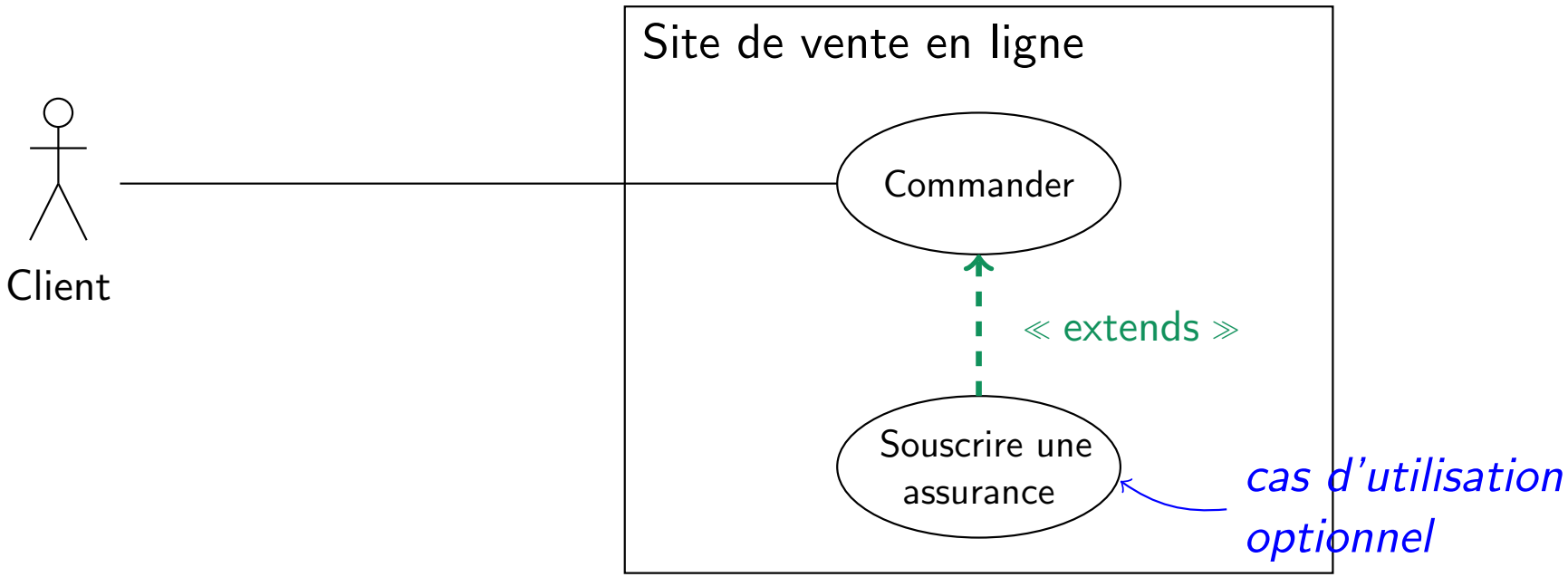
# Généralisation




## Généralisation

L'action qui pointe est un cas plus spécifique que l'action pointé.

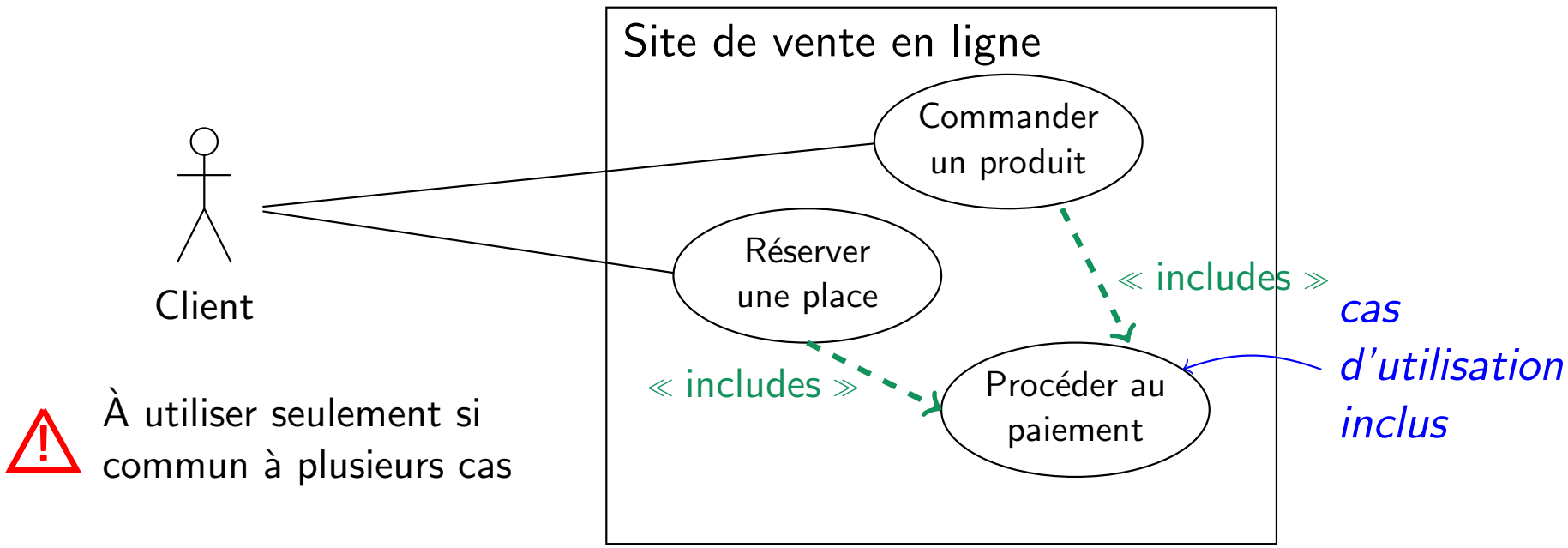
# Extends




## ≪ extends ≫

L'action qui pointe *peut* être déclenchée au cours de l'action pointée

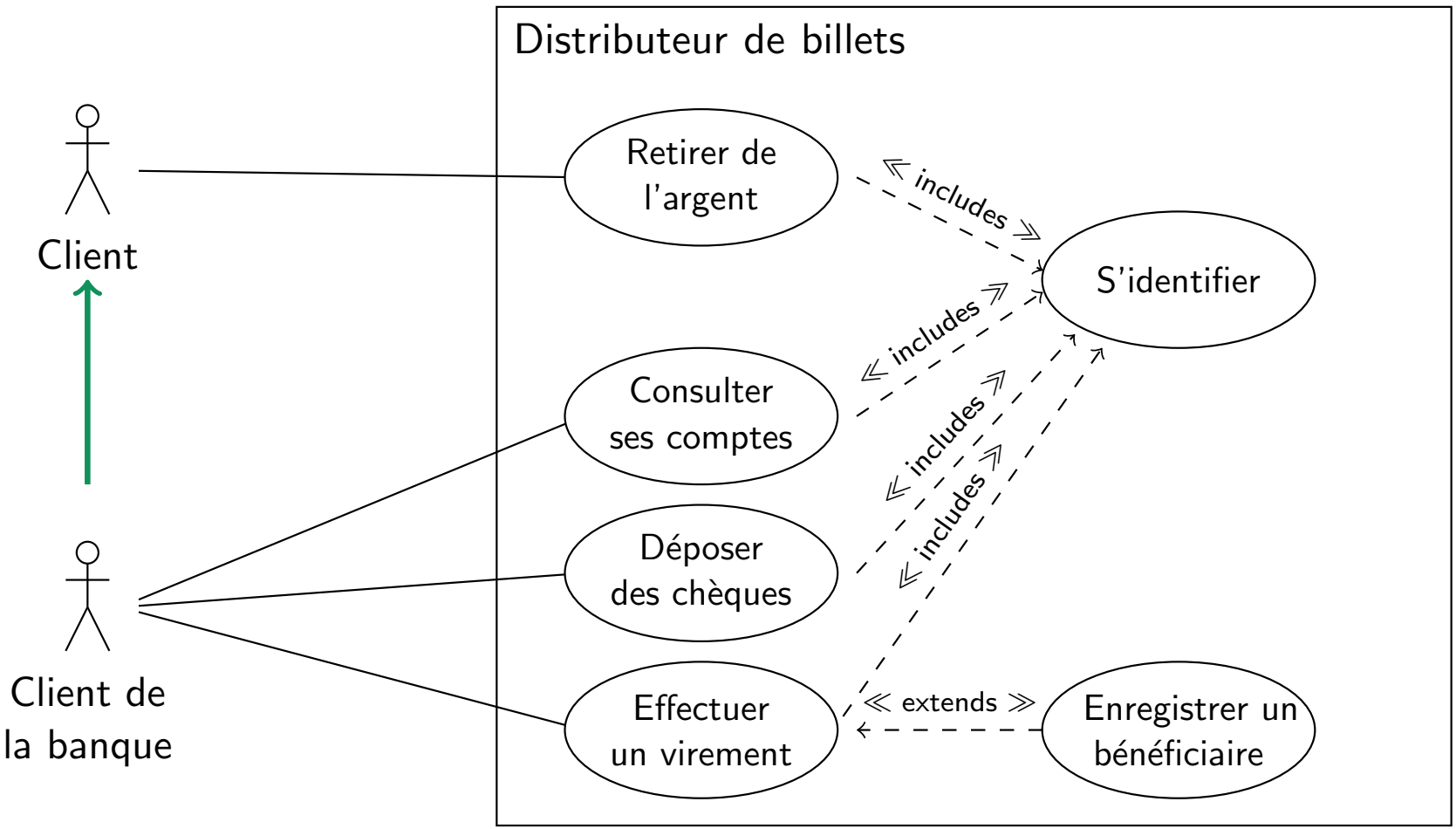
# Includes




## « includes »

L'action pointée est toujours faite quand l'action qui pointe est effectuée.

# Exemple complet



# Conseils

Rester lisible :

- Pas plus de 6 ou 8 cas dans un diagramme
- Au besoin, faire plusieurs diagrammes (si des cas sont disjoints entre acteurs, pour détailler un cas, etc.)
- Relations entre cas seulement si nécessaires et pas trop lourdes

Pour les détails : description textuelle.

# Scénario d'utilisation

Séquence d'étapes

- décrivant une interaction entre utilisateur et système
- permettant à l'utilisateur de réaliser un objectif

**Scénario : Commander**
Système : Site de vente en ligne

Le client s'authentifie dans le système puis choisit une adresse et un mode de livraison. Le système indique le montant total de sa commande au client. Le client donne ses informations de paiement. La transaction est effectuée et le système en informe le client par e-mail.

# Conclusion : Diagramme de cas d'utilisation

- Utiles pour la discussion avec le client car intuitifs et concis
- Pas suffisants pour l'équipe de développement

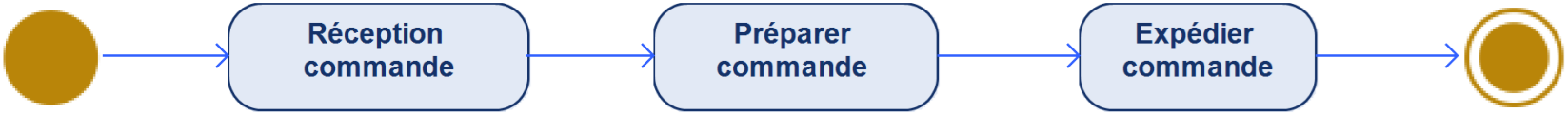
# Objectifs

- Décrire le déroulement d'un cas d'utilisation
- Correspond à un enchaînement d'actions, qui peuvent avoir lieu en parallèle
- Flux d'un point de départ à un point d'arrivée

# Catégories de nœuds d'activité

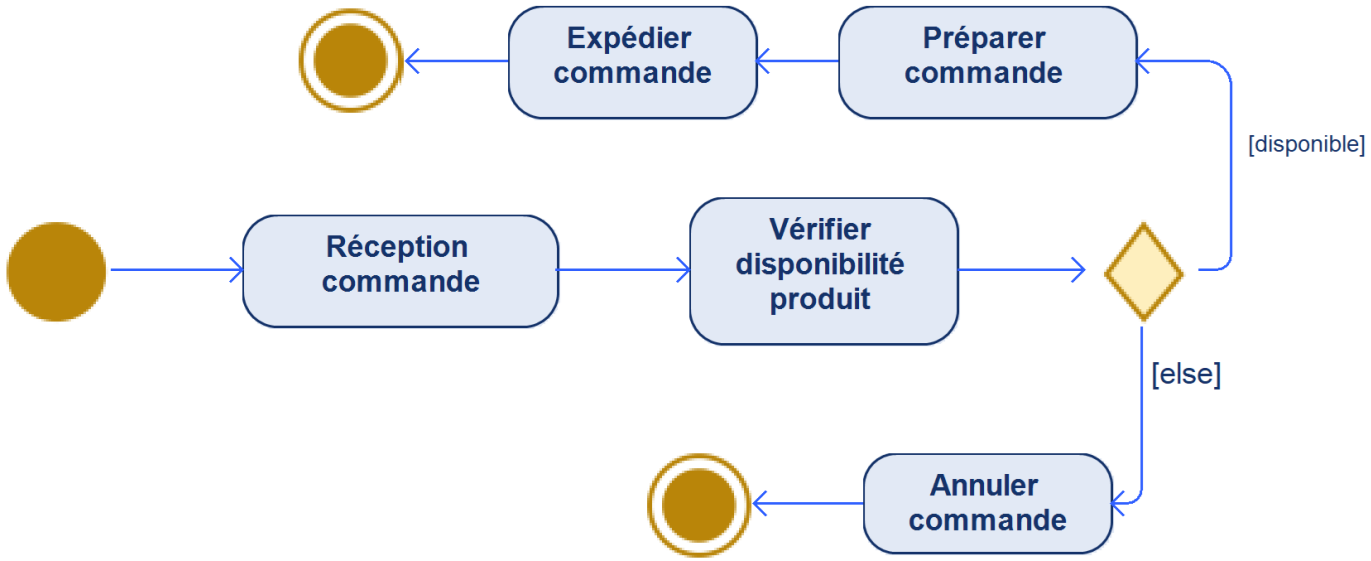
1. Les nœuds d'exécution
2. Les nœuds de contrôle
3. Les nœuds objets

# Les nœuds d'activité (1)



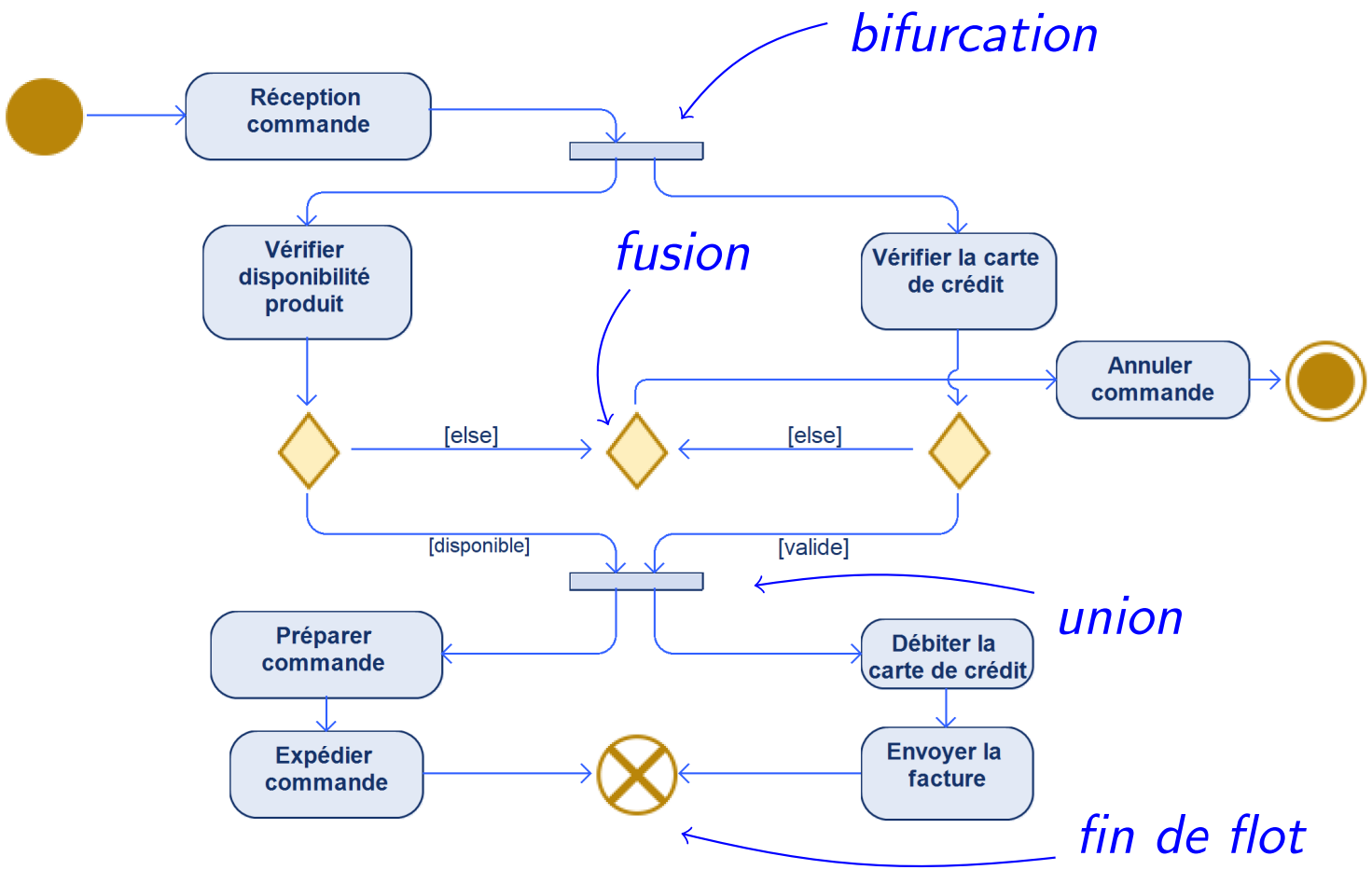

- **Nœud initial** (contrôle) : début du flot, peut en exister plusieurs, pas d'arc entrant
- **Nœud d'action** (exécution) : activité exécutable, représente une transformation ou un calcul dans le système, a au moins un arc entrant
- **Nœud de fin d'activité** (contrôle) : pas d'arc sortant, toute activité est arrêtée

# Les nœuds d'activité (2)




- **Nœud de décision** (contrôle) : permet de faire un choix entre plusieurs flots sortants (accompagné de conditions de garde)

# Les nœuds d'activité (3)

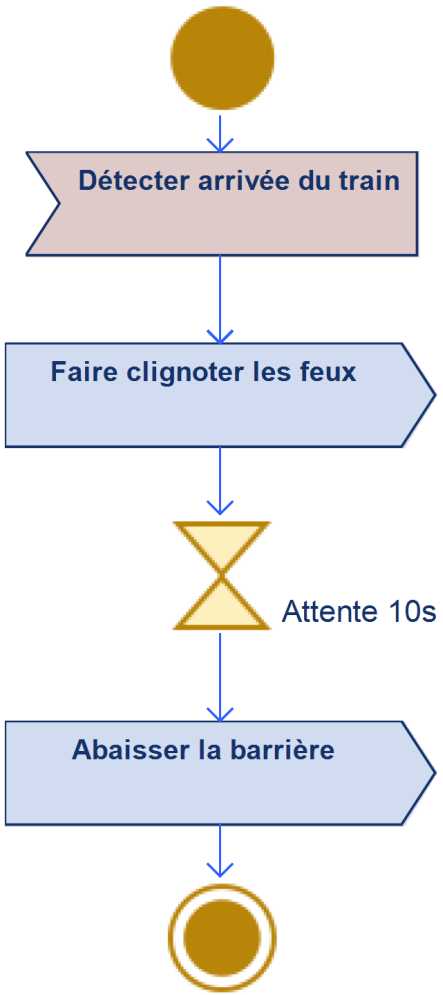


# Les nœuds d'activité (3)

- **Nœud de fusion** (contrôle) : rassemble plusieurs flots entrants en un seul flot sortant, il n'accepte qu'un seul des flots entrants (il ne synchronise pas).
- **Nœud de bifurcation** (contrôle) : sépare un flot en plusieurs flots concurrents.
- **Nœud d'union** (contrôle) : synchronise plusieurs flots entrants.
- **Nœud de fin de flot** (contrôle) : pas d'arc sortant, une exécution s'arrête mais n'a pas n'impact sur les autres flots actifs

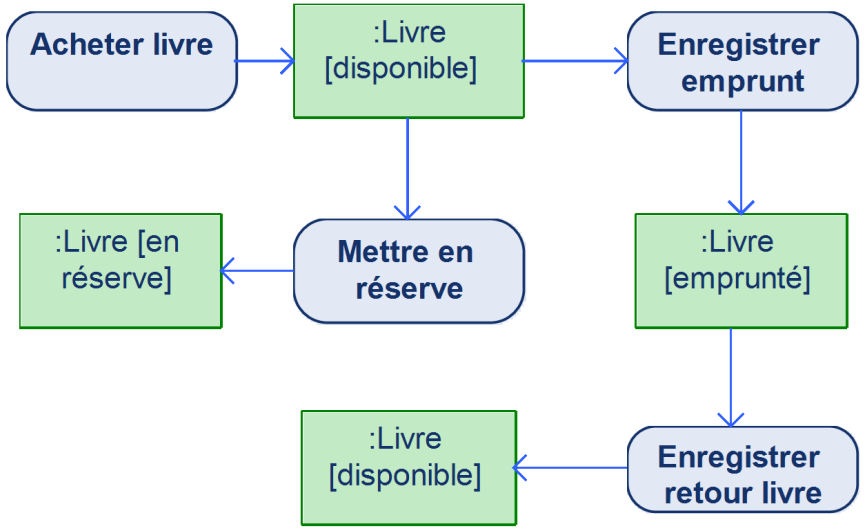
# Les nœuds d'activité (4)

- « **accept event** » (exécution) : recevoir un signal
- « **signal sending** » (exécution) : envoyer un signal
- « **accept time event** » (exécution) : attend la durée précisée avant de continuer le flot



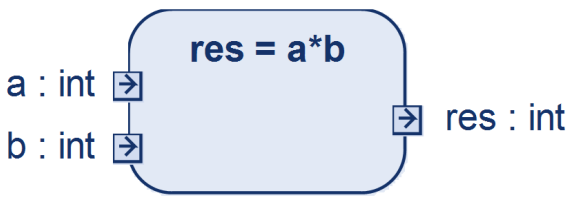
# Les nœuds d'activité (5)

- **Nœuds objets** (objet) : objet généré par une action et utilisé par d'autres, précise le type de l'objet, peut préciser l'état de l'objet (entre crochets).



# Les nœuds d'activité (6)

- **Les pins d'entrée/sortie** (objet) :
  - spécifie les valeurs passées en argument à une activité,
  - l'activité ne peut débuter que si une valeur est affectée à chaque pin d'entrée,
  - une fois l'activité terminée, une valeur doit être affectée à chacun des pins de sortie.



# Les partitions

Pour préciser qui fait quoi.