Université de Msila – Département d'Informatique

Génie logiciel (GL2)

(3ème année SI&ISIL)

Dr BOUNIF M-E

- **3.1UML**
- Le langage de modélisation unifié, de l'anglais Unified Modeling Language (UML), est un langage de modélisation graphique à base de pictogrammes conçu pour fournir une méthode normalisée pour visualiser la conception d'un système. Il est couramment utilisé en développement logiciel et en conception orientée objet.
- **3.2 Utilisation**
- UML est utilisé pour spécifier, visualiser, modifier et construire les documents nécessaires au bon développement d'un logiciel orienté objet. UML offre un standard de modélisation, pour représenter l'architecture logicielle. Les différents éléments représentables sont :
- Activité d'un objet/logiciel
- Acteurs
- Processus
- Schéma de base de données
- Composants logiciels
- Réutilisation de composants

- Grâce aux outils de modélisation UML, il est également possible de générer automatiquement tout ou partie du code d'une application logicielle, par exemple en langage Java, à partir des divers documents réalisés.
- 3.3 Formalisme
- UML se décompose en plusieurs parties :
  - Les vues : ce sont les observables du système. Elles décrivent le système d'un point de vue donné, qui peut être organisationnel, dynamique, temporel, architectural, géographique, logique, etc. En combinant toutes ces vues, il est possible de définir (ou retrouver) le système complet.

- Les diagrammes : ce sont des ensembles d'éléments graphiques. Ils décrivent le contenu des vues, qui sont des notions abstraites. Ils peuvent faire partie de plusieurs vues.
- Les modèles d'élément : ce sont les éléments graphiques des diagrammes.

- **Vues**
- Une façon de mettre en œuvre UML est de considérer différentes vues qui peuvent se superposer pour collaborer à la définition du système :

- Vue des cas d'utilisation (use-case view) : c'est la description du modèle vu par les acteurs du système. Elle correspond aux besoins attendus par chaque acteur (c'est le quoi et le qui).
- Vue logique (logical view): c'est la définition du système vu de l'intérieur. Elle explique comment peuvent être satisfaits les besoins des acteurs (c'est le comment).

- *Vue d'implémentation* (*implementation view*) : cette vue définit les dépendances entre les modules.
- *Vue des processus* (*process view*) : c'est la vue temporelle et technique, qui met en œuvre les notions de tâches concurrentes, stimuli, contrôle, synchronisation…
- *Vue de déploiement* (*deployment view*) : cette vue décrit la position géographique et l'architecture physique de chaque élément du système (c'est le où).
- Le pourquoi n'est pas défini dans UML.
- **Diagrammes**
- Les *diagrammes* sont dépendants hiérarchiquement et se complètent, de façon à permettre la modélisation d'un projet tout au long de son cycle de vie. Il en existe quatorze type de diagramme.

- **Diagrammes de structure ou diagrammes statiques**

- ***Diagramme de classes (class diagram)*** : représentation des classes intervenant dans le système.
- ***Diagramme d'objets (object diagram)*** : représentation des instances de classes (objets) utilisées dans le système.
- ***Diagramme de composants (component diagram)*** : représentation des composants du système d'un point de vue physique, tels qu'ils sont mis en œuvre (fichiers, bibliothèques, bases de données…)
- ***Diagramme de déploiement (deployment diagram)*** : représentation des éléments matériels (ordinateurs, périphériques, réseaux, systèmes de stockage…) et la manière dont les composants du système sont répartis sur ces éléments matériels et interagissent entre eux.
- ***Diagramme des paquets (package diagram)*** : représentation des dépendances entre les paquets (un paquet étant un conteneur logique permettant de regrouper et d'organiser les éléments dans le modèle UML), c'est-à-dire entre les ensembles de définitions.
- ***Diagramme de structure composite (composite structure diagram)*** : représentation sous forme de boîte blanche les relations entre composants d'une classe .
- ***Diagramme de profils (profile diagram)*** : spécialisation et personnalisation pour un domaine particulier d'un meta-modèle de référence d'UML.

- **Diagrammes de comportement**
- Les diagrammes de comportement rassemblent :
- **Diagramme des cas d'utilisation (use-case diagram) :** représentation des possibilités d'interaction entre le système et les acteurs (intervenants extérieurs au système), c'est-à-dire de toutes les fonctionnalités que doit fournir le système.
- **Diagramme états-transitions (state machine diagram) :** représentation sous forme de machine à états finis le comportement du système ou de ses composants.
- **Diagramme d'activité (activity diagram) :**
- représentation sous forme de flux ou d'enchaînement d'activités le comportement du système ou de ses composants

- **Diagrammes d'interaction ou diagrammes dynamiques**
- Les diagrammes d'interaction (interaction diagrams) ou diagrammes dynamiques (dynamic diagrams) rassemblent :

- **Diagramme de séquence (sequence diagram) :** représentation de façon séquentielle du déroulement des traitements et des interactions entre les éléments du système et/ou de ses acteurs.
- **Diagramme de communication (communication diagram) :** représentation de façon simplifiée d'un diagramme de séquence se concentrant sur les échanges de messages entre les objets
- **Diagramme global d'interaction (interaction overview diagram) :** représentation des enchaînements possibles entre les scénarios préalablement identifiés sous forme de diagrammes de séquences (variante du diagramme d'activité)
- **Diagramme de temps (timing diagram) :** représentation des variations d'une donnée au cours du temps.

- **4. Le diagramme des cas d'utilisation**

Le diagramme des cas d'utilisation montre les cas d'utilisation représentés sous la forme d'ovales et les acteurs sous la forme de personnages. Il indique également les relations de communication qui les relient.
Exemple : Le cas d'utilisation de l'achat d'un produit est représenté par la figure 4.1 :

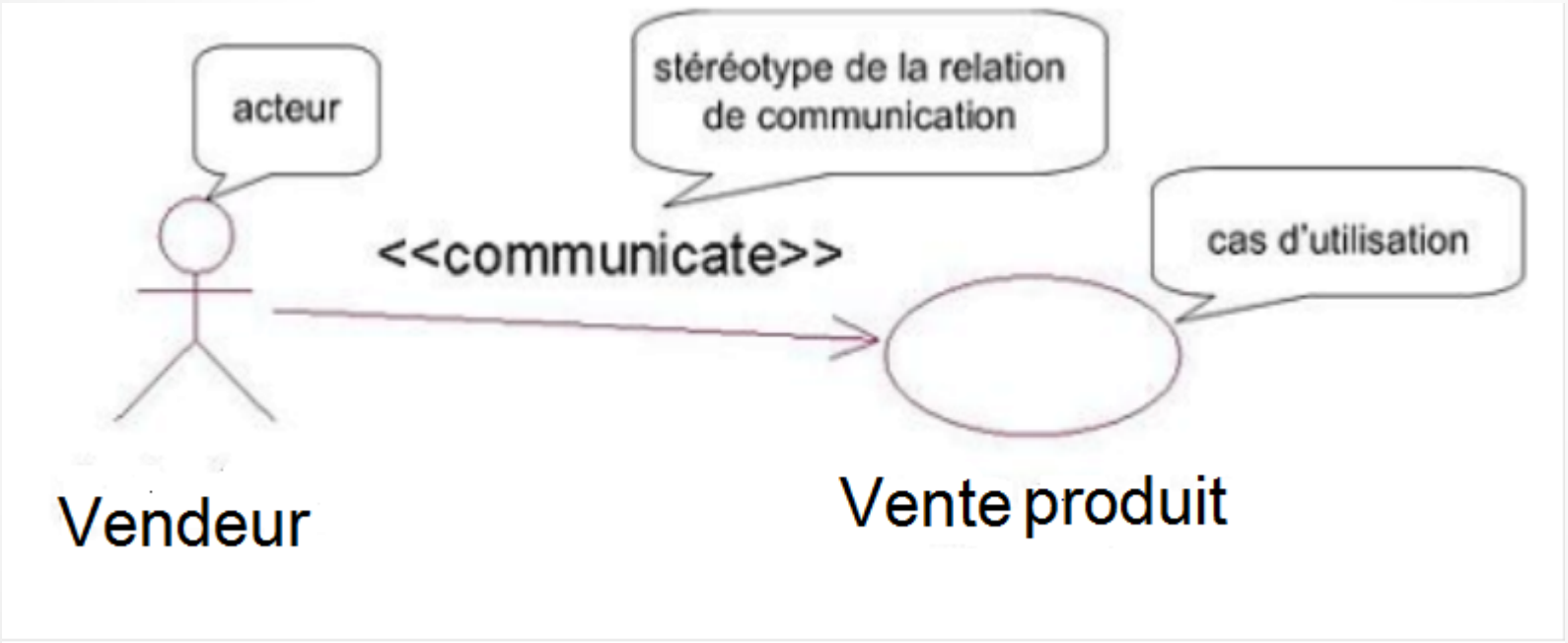


Figure 4.1 Le cas d'utilisation d'achat d'un cheval

Il est possible de représenter le système qui répond au cas d'utilisation sous la forme d'un rectangle englobant le cas.

Exemple :

Dans l'exemple précédent, le système, c'est à dire la gestion de stock, est illustré à la figure 4.2.

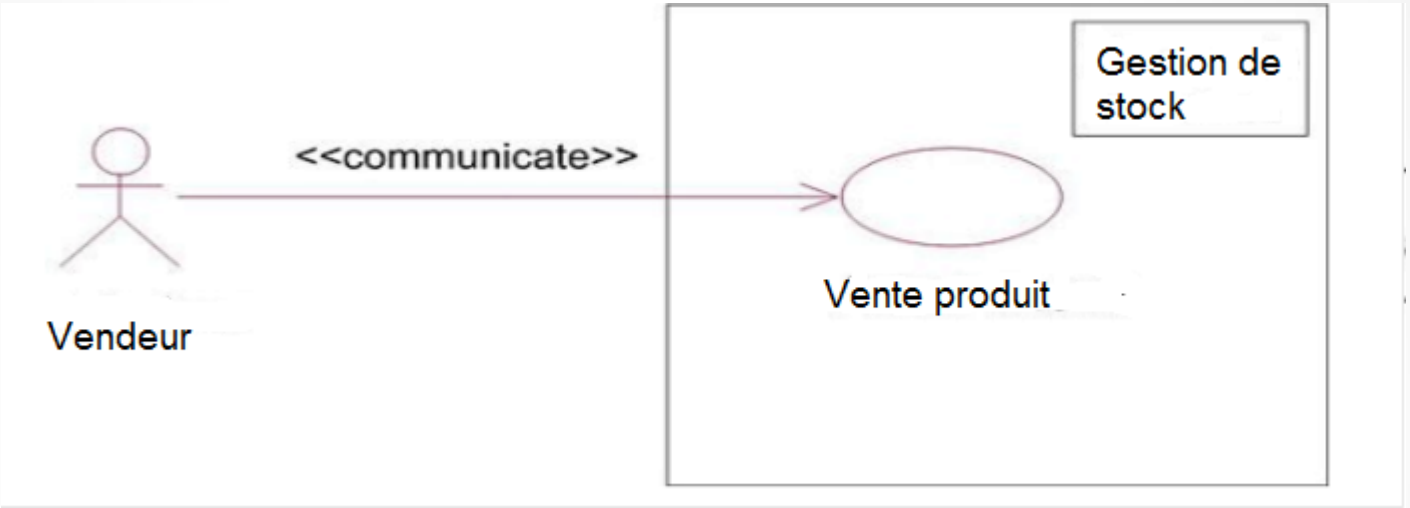


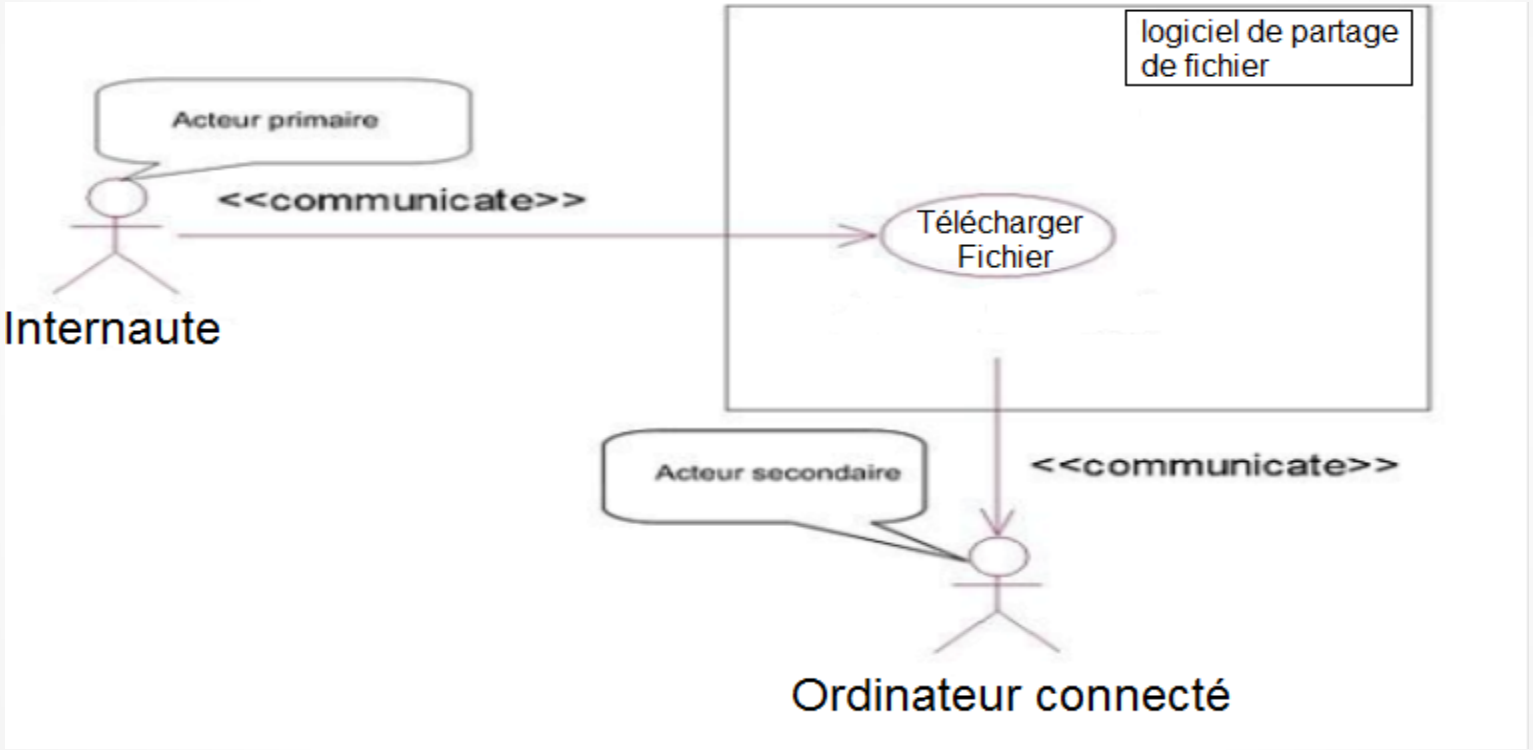
Figure 4.2 Système d'un cas d'utilisation

Un acteur secondaire est représenté comme un acteur primaire. Souvent, le sens de la relation de communication entre un acteur secondaire et le système est inversé par rapport au sens de la relation entre un acteur primaire et le système.

En effet, la communication est initiée par le système et non par l'acteur.

- Exemple :

Dans l'exemple suivant, un diagramme de cas d'utilisation représentant un logiciel de partage de fichiers.. Un ordinateur connecté constitue un acteur secondaire (voir figure 4.3).



- Figure 4.3 Acteurs primaire et secondaire d'un cas d'utilisation

- **Les relations entre les cas d'utilisation**
- **1. La relation d'inclusion**

La relation d'inclusion sert à enrichir un cas d'utilisation par un autre cas d'utilisation. Cet enrichissement est réalisé par une inclusion impérative, il est donc systématique. Le cas d'utilisation inclus existe uniquement dans ce but. En effet, il ne répond pas à un objectif d'un acteur primaire. Un tel cas d'utilisation est une sous fonction.

L'inclusion sert à partager une fonctionnalité commune entre plusieurs cas d'utilisation. Elle peut également être employée pour structurer un cas d'utilisation en décrivant ses sous-fonctions. Dans le diagramme des cas d'utilisation, cette relation est représentée par une flèche pointillée munie du stéréotype «include».

Exemple

Lors de la vente d'un produit, un vendeur va vérifier la disponibilité de produit .

Par conséquent, le cas d'utilisation Vente produit inclut cette vérification (voir figure 4.4).

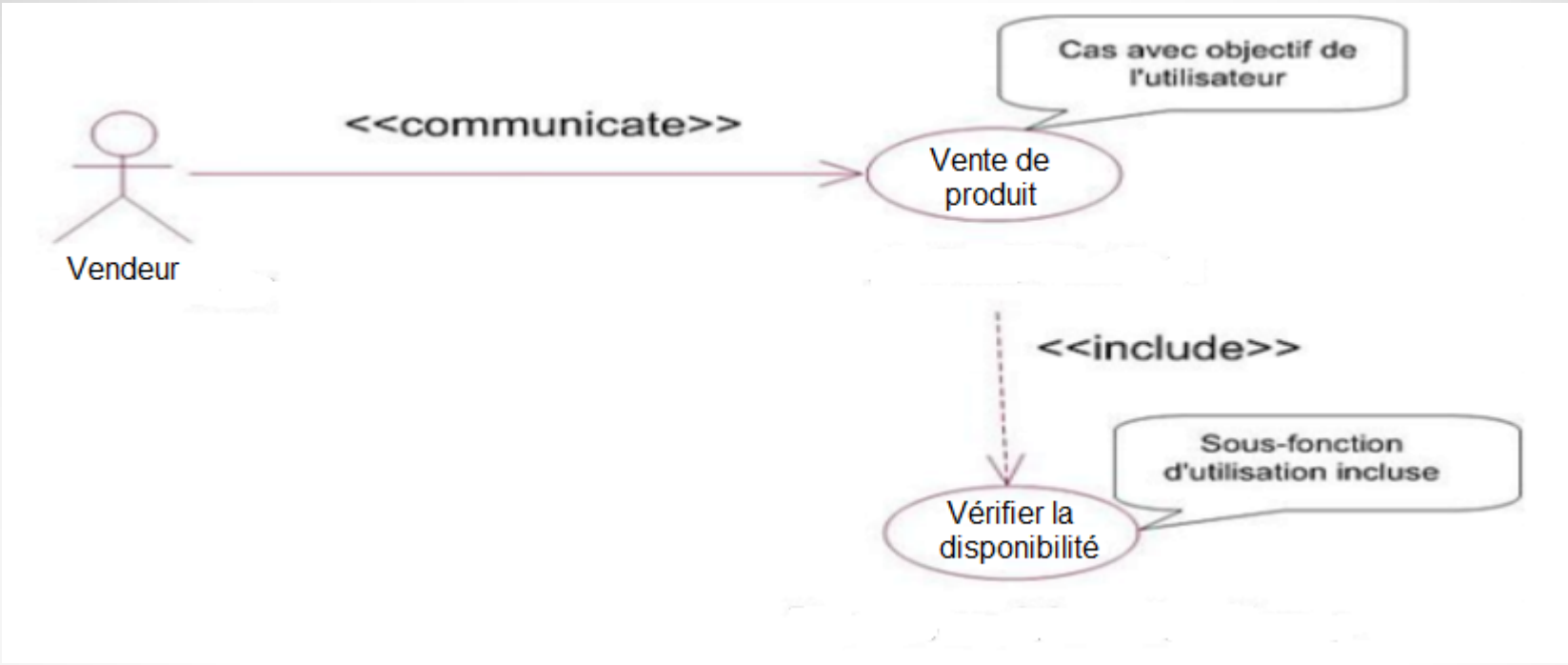


Figure 4.4 Inclusion d'un cas d'utilisation

La mise en commun du cas d'utilisation Verifier la disponibilité est illustrée à la figure 4.5

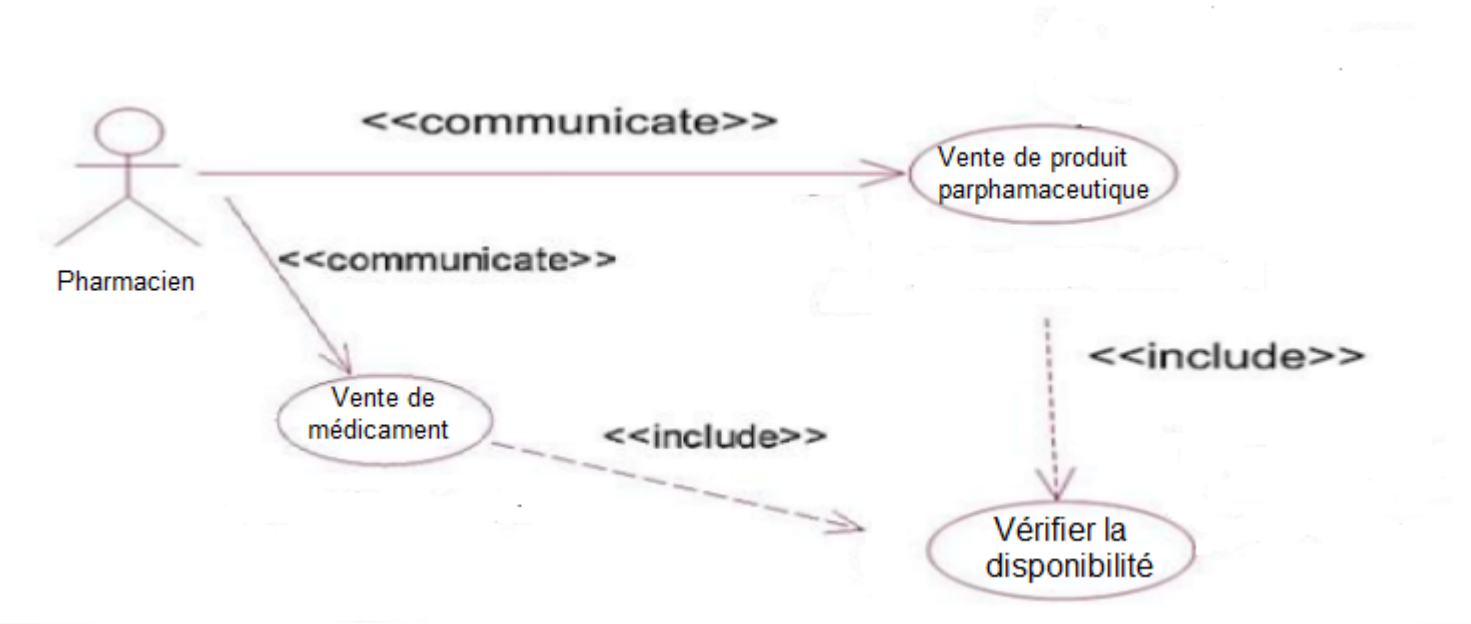


Figure 4.5 Mise en commun d'un cas d'utilisation inclus

L'inclusion peut également être employée pour décomposer l'intérieur d'un cas d'utilisation sans que le cas inclus soit partagé À la figure 4.6, Demande Ordonnance n'est pas partagé mais sa présence illustre bien que cet demande fait partie des points étudiés lors de la vente des médicaments .

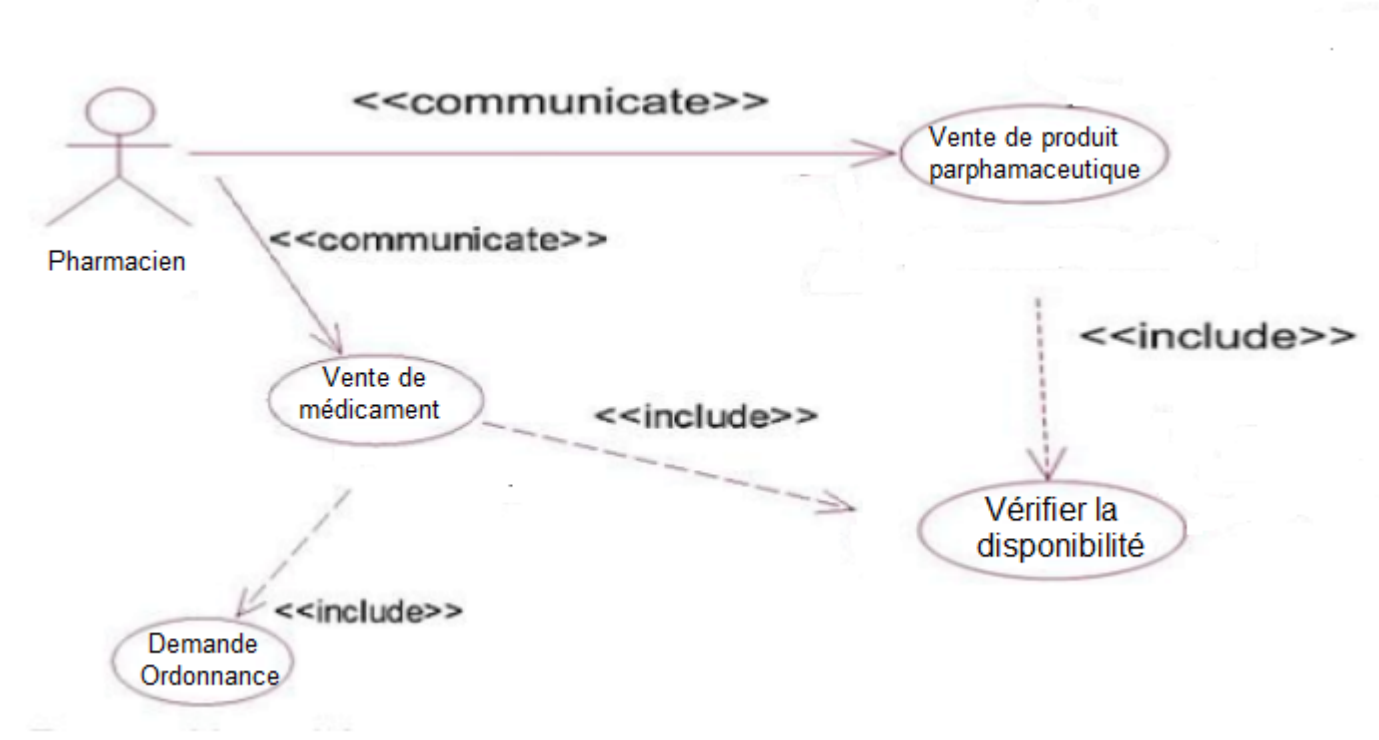


Figure 4.6 Décomposition d'un cas d'utilisation par inclusion

- **La relation d'extension**

Comme la relation d'inclusion, la relation d'extension enrichit un cas d'utilisation par un cas d'utilisation de sous fonction.
Cet enrichissement est analogue à celui de la relation d'inclusion mais il est optionnel.
L'extension se fait dans le cas d'utilisation de base, en des points précis et prévus lors de la conception, appelés points d'extension.

L'application de chaque extension est décidée lors du déroulement d'un scénario. Par conséquent, le cas d'utilisation de base peut être employé sans être étendu.
Comme pour l'inclusion, l'extension sert à structurer un cas d'utilisation ou à partager un cas d'utilisation de sous fonction.
Dans le diagramme des cas d'utilisation, cette relation est représentée par une flèche pointillée munie du stéréotype «extend».
Exemple Lors de vente d'un médicament, un acheteur peut acheter des médicaments par son carte de CHIFA.(voir figure 4.7).

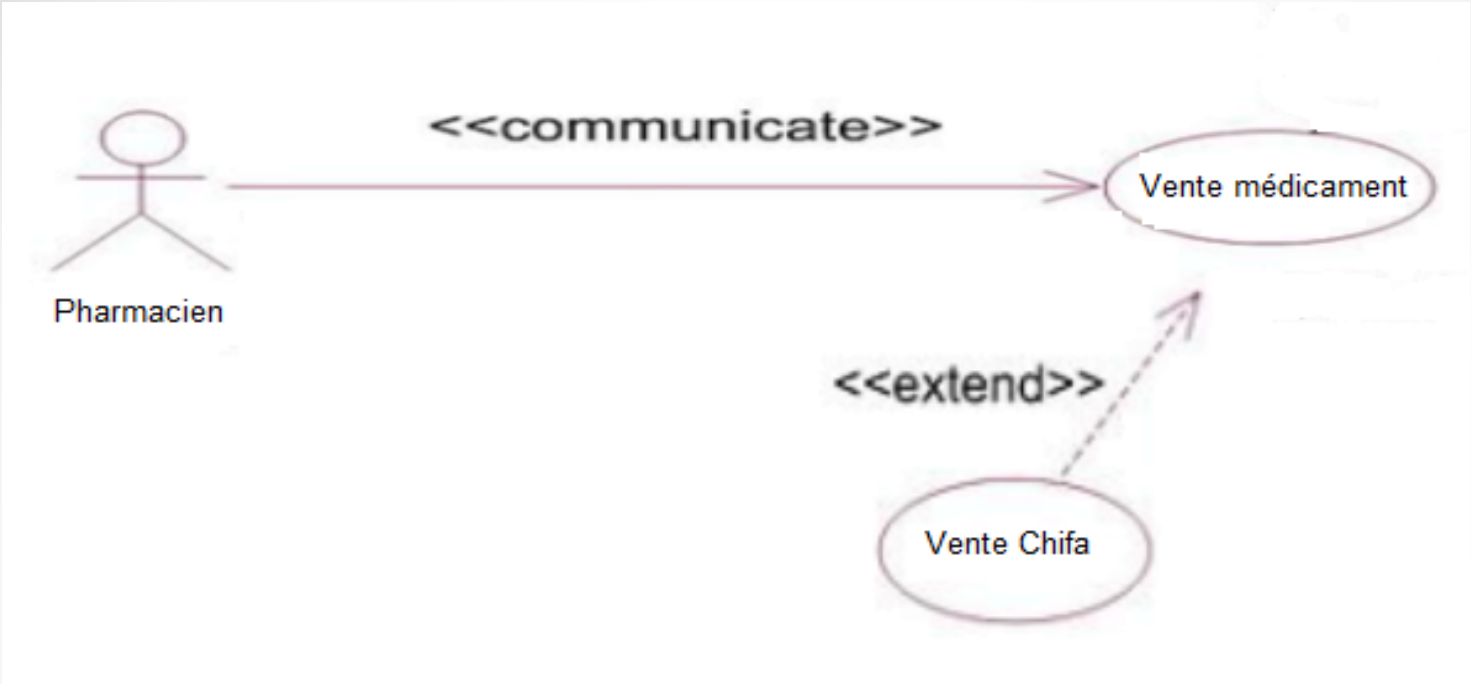


Figure 4.7 Extension d'un cas d'utilisation

- **La spécialisation et la généralisation des cas d'utilisation**

Il est possible de spécialiser un cas d'utilisation en un autre. On obtient ainsi un sous cas d'utilisation. Comme pour les classes, le sous cas hérite du comportement du sur-cas d'utilisation. Un sous cas d'utilisation hérite également des relations de communication, d'inclusion et d'extension du sur cas.
Souvent, le sur-cas d'utilisation est abstrait, c'est à dire qu'il correspond à un comportement partiel complété dans les sous cas d'utilisation.

Un souscas d'utilisation a le même niveau que son surcas. Si le surcas est un cas avec objectif utilisateur, il en va de même pour le souscas. Si le surcas est un cas de sousfonction, le souscas est lui aussi une sous fonction.

Dans le diagramme des cas d'utilisation, la relation de spécialisation est représentée par une flèche pleine de spécialisation identique à celle qui relie les sous-classes aux surclasses.
Le nom d'un cas d'utilisation abstrait est écrit en italique (ou accompagné du stéréotype «abstract»).

- Exemple

Le cas d'utilisation Vente Produit est spécialisé en deux sous-cas :
Vente médicament ou d'un produit parapharmaceutique.
Ce cas est un cas abstrait et son nom apparaît en italiques. La figure 4.9 illustre cette spécialisation.
Les cas d'utilisation Vente médicament ou d'un produit parapharmaceutique sont des cas d'utilisation avec objectif utilisateur et communiquent avec le pharmacien.

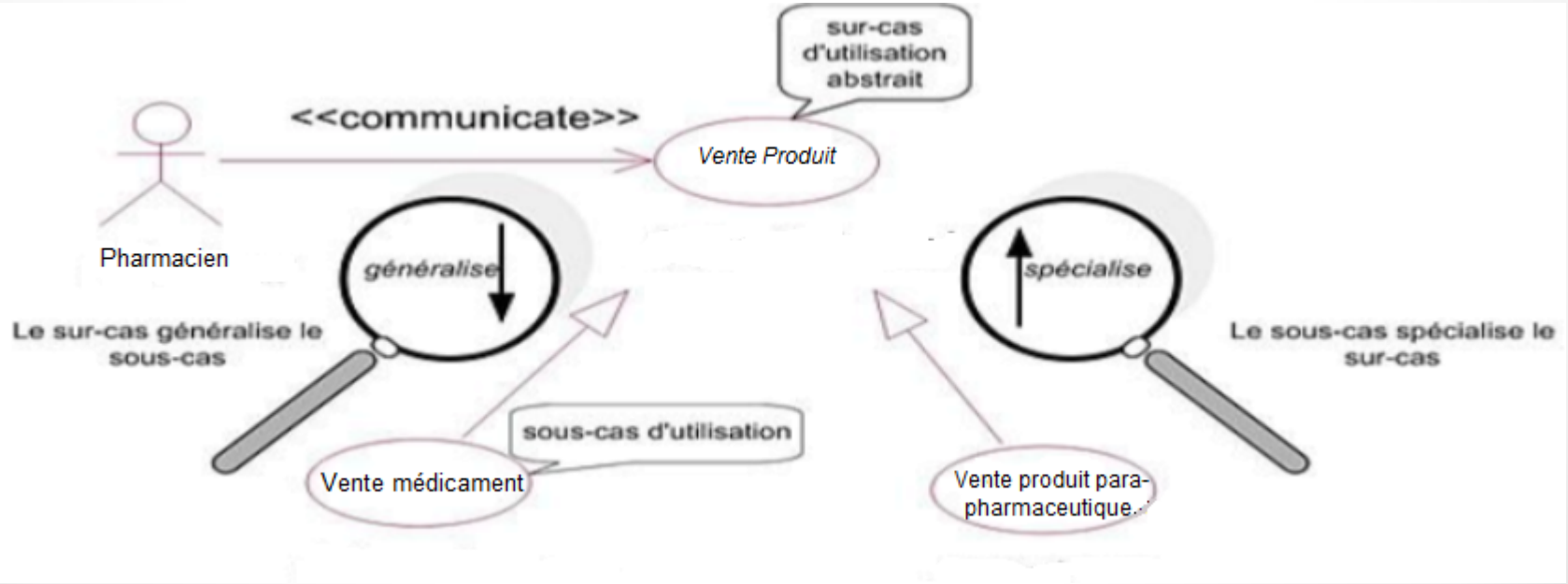


Figure 4.9 Spécialisation d'un cas d'utilisation

Exemple

Les relations d'extension concernant les différentes inclusions et extensions de vérification peuvent être factorisées au niveau du cas abstrait.

Elles sont alors héritées dans les sous-cas à l'image de la relation de communication dans l'exemple précédent (voir figure 4.10).

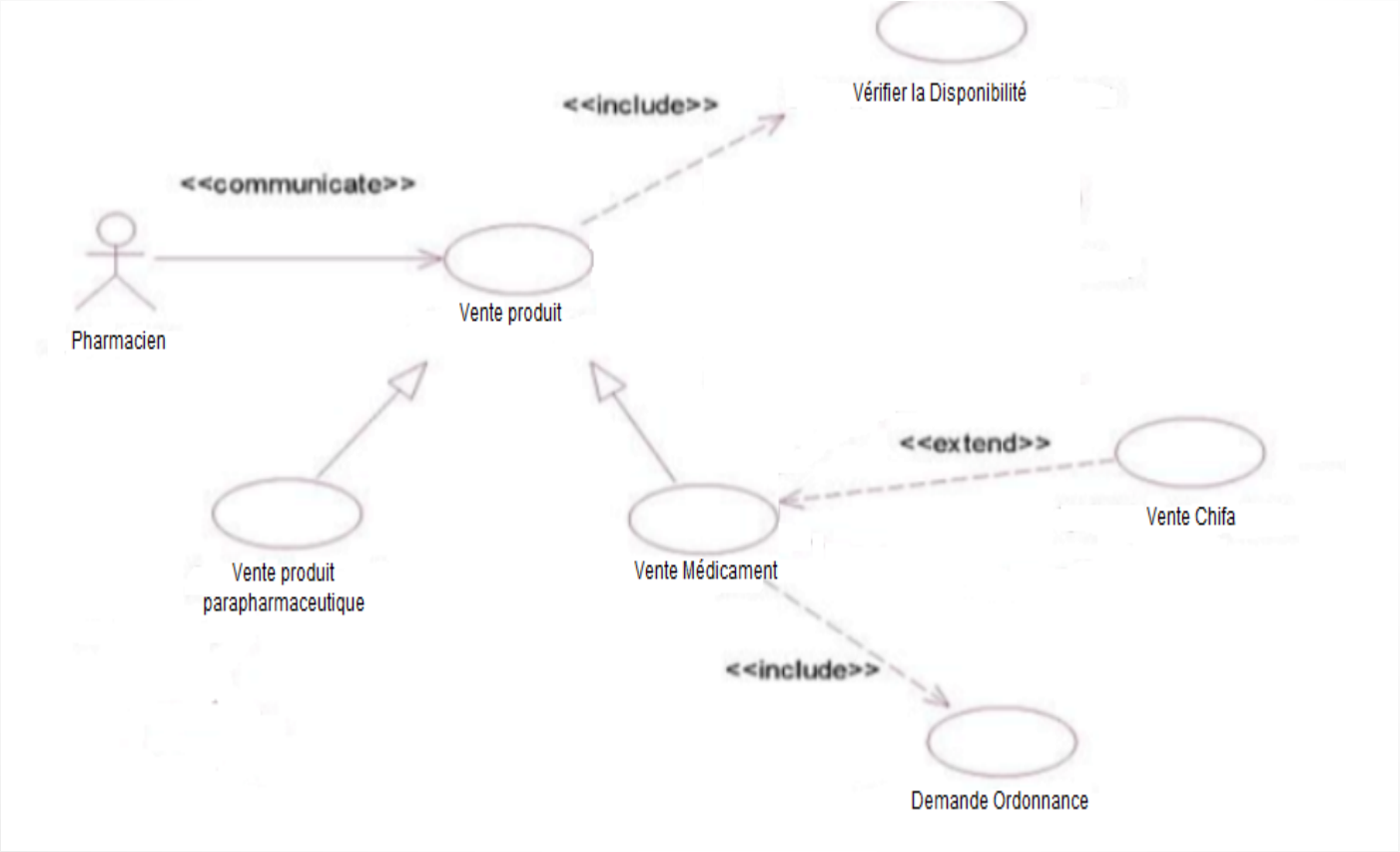


Figure 4.10 Factorisation d'inclusion et d'extension

Conclusion

Les cas d'utilisation servent à :

- exprimer les exigences fonctionnelles conférées au système par les utilisateurs lors de la rédaction du cahier des charges ;
- vérifier que le système répond à ces exigences lors de la livraison;
- déterminer les frontières du système ;
- écrire la documentation du système ;

Les cas d'utilisation offrent une technique de représentation qui convient au dialogue avec l'utilisateur car son formalisme reste proche du langage naturel.