Groupe 2

Diagramme UML des cas d'utilisation

Chargé du cours : Dr. APEKE K. Séna

Groupe 2

GAGNON Efoé Schamma

KOUSSAWO Ayélé Lucie

PAMAZI Manawèzouwé Essozimna

Table des matières

I. Introduction

L'Homme en tant qu'être vivant intelligent développe le souci permanent de se doter d'une vie simple et aisée. Dans ce sens, il élabore et améliore sans cesse des mécanismes lui permettant d'atteindre cet objectif. L'activité de génie logiciel, qui est une émanation de ce talent humain, est au cœur du développement des logiciels, qui sont des solutions applicatives à des problèmes concrets de la vie. La réalisation de cette activité s'inscrit dans une démarche globale et de grande envergure, partant de la compréhension du problème jusqu'à la mise en œuvre effective de la solution. Pour ce faire, l'ingénieur génie logiciel, maître d'œuvre par excellence pour la réalisation de ce type de projet, dispose d'un certain nombre d'outils lui permettant de mener à bien toutes les tâches. L'UML (*Unified Modeling Langage)* est particulièrement utilisé dans le cadre de la conception et de la modélisation. UML utilise trois axes de modélisation notamment l'axe fonctionnel, l'axe statique et l'axe dynamique. Dans le cadre de cet exposé, nous allons développer l'aspect fonctionnel d'UML. Ceci se fera à travers la présentation du diagramme des cas d'utilisation. Nous introduirons notre travail par une présentation rapide des généralités d'UML. Ensuite nous nous focaliserons sur le diagramme des cas d'utilisation proprement dit, sa définition, ses éléments constitutifs et sa mise en œuvre. Nous terminerons avec deux exercices permettant d'élucider les concepts abordés tout au long de ce travail.

II. Généralités sur UML

UML est un langage de modélisation très complet, qui couvre de nombreux aspects du développement des logiciels, comme les exigences, l'architecture, les structures et les comportements. Depuis sa normalisation, en 1997, UML a fortement évolué, passant d'un langage peu formel, principalement destiné à la documentation, à un langage suffisamment précis pour que des applications puissent être générées à partir des modèles.

A. Définition

UML est un acronyme qui signifie *Unified Modeling Language* soit *Langage de Modélisation Unifié*. UML est un langage essentiellement graphique, facile à lire et à comprendre et qui s'applique à l'analyse et à la conception des logiciels. C'est une norme qui définit les diagrammes et les conventions à utiliser lors de la construction de modèles décrivant la structure et le comportement d'un logiciel. Les modèles sont des diagrammes constitués d'éléments graphiques et de texte. En clair, UML n'est pas une méthode, mais un langage.

B. Les diagrammes UML

Les diagrammes UML 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 depuis UML 2.3. Ils sont organisés en deux groupes :

- les **diagrammes de structure** ou **diagrammes statiques** : diagramme de paquetage, diagramme de composants, diagramme structurel, diagramme de déploiement, diagramme de structures composites, diagramme d'objets, diagramme de profils ;

- les **diagrammes de comportement** pour modéliser l'aspect dynamique d'un système : diagramme de cas d'utilisation, diagramme d'activités, diagramme de séquence, diagramme de temps, diagramme de StateMachine, diagramme d'aperçu d'interactions, diagramme de communication.

III. Le diagramme des cas d'utilisation

A. Définition

Le diagramme de cas d'utilisation (DCU) est un diagramme UML utilisé pour une représentation du comportement fonctionnel d'un système logiciel. Le diagramme de cas d'utilisation décrit ce qu'un système fait du point de vue d'un observateur externe. L'accent est mis sur ce qu'un système fait, plutôt que sur la façon dont il le fait.
En langage UML, un diagramme de cas d'utilisation peut servir à résumer les informations des utilisateurs du système (également appelés acteurs) et leurs interactions avec ce dernier. La création de ce type de diagramme requiert un ensemble de symboles et de connecteurs spécifiques. Lorsqu'ils sont bien conçus, les diagrammes de cas d'utilisation peuvent aider à représenter :

- les scénarios dans lesquels le système interagit avec des personnes, des organisations ou des systèmes externes ;
- les objectifs que le système permet aux entités (appelées acteurs) d'atteindre ;
- la portée du système.

Avant de se lancer dans la réalisation d'un logiciel, il faut comprendre, clarifier et structurer les attentes et les besoins du client. Généralement, le diagramme des cas d'utilisation constitue la première étape dans la démarche 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.

Ce diagramme décrit :

- le système ;
- les acteurs ;
- les cas d'utilisation.

B. Les composantes du diagramme de cas d'utilisation

1. Le système

a) Définition

Un système représente une application dans le modèle UML. Il est identifié par un nom et regroupe un ensemble de cas d'utilisation qui correspondent aux fonctionnalités offertes par l'application à son environnement. L'environnement est spécifié sous forme d'acteurs liés aux cas d'utilisation.

b) Représentation graphique

Un système se représente par un rectangle contenant le nom du système et les cas d'utilisation de l'application.

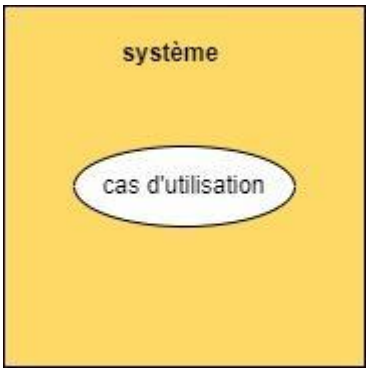


Fig 1 : Représentation du système

2. Les acteurs

a) Définition

Un acteur représente un rôle joué par une entité externe (utilisateur humain, dispositif matériel ou autre système) qui interagit directement avec le système étudié. Exemple : un agent, le SI d'une banque.

Un acteur peut consulter et/ou modifier directement l'état du système, en émettant et/ou en recevant des messages susceptibles d'être porteurs de données.

Un même acteur peut jouer plusieurs rôles à la fois.

b) Identification des acteurs

Pour identifier les acteurs, il faut se poser la question suivante : quelles sont les entités externes au système et qui réagissent directement avec le système ?

Les acteurs candidats sont systématiquement :

- Les utilisateurs humains directs : on fait donc en sorte d'identifier tous les profils possibles ;
- les autres systèmes connexes qui interagissent aussi directement avec le système étudié, souvent par le biais de protocoles bidirectionnels.

Contrairement à ce que l'on pourrait croire, tous les acteurs n'utilisent pas forcément le système. On appelle **acteur principal** celui pour qui le cas d'utilisation produit un résultat observable. Par opposition, on qualifie d'**acteurs secondaires** les autres participants du cas d'utilisation. Les acteurs secondaires sont souvent sollicités pour des informations complémentaires ; ils peuvent uniquement consulter ou informer le système lors de l'exécution du cas d'utilisation.

c) Représentation

La représentation graphique standard de l'acteur en UML est l'icône appelée **stick man**, avec le nom de l'acteur sous le dessin. On peut également figurer un acteur sous la **forme rectangulaire d'une classe, avec le mot-clé <<actor>>.** Une troisième représentation (**intermédiaire entre les deux premières**) est également possible avec certains outils, comme cela est indiqué ci-après.

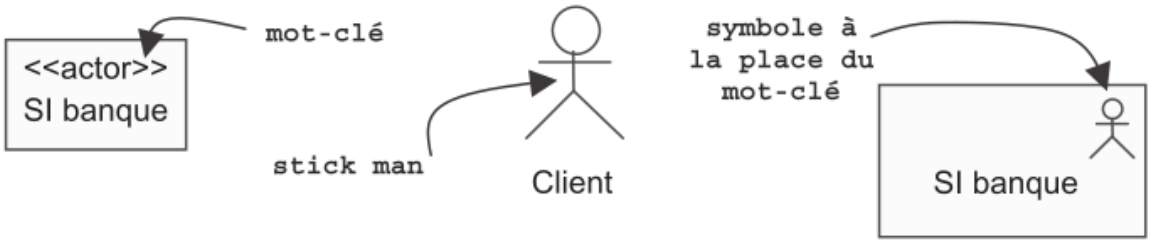


Fig 2 : Représentations graphiques possibles d'un acteur

NB *:* Une bonne recommandation consiste à faire prévaloir l'utilisation de la forme graphique du stick man pour les acteurs humains et une représentation rectangulaire pour les systèmes connectés.

d) Relation d'héritage entre les acteurs

UML permet de décrire une relation d'héritage entre acteurs encore appelée de **relation de généralisation** ou de **spécialisation**. L'acteur1 est une généralisation de l'acteur2 si tous les cas d'utilisation accessibles à l'acteur1 sont accessibles à l'acteur2 également mais pas l'inverse.

La relation d'héritage entre acteurs se représente par une flèche allant de l'acteur qui hérite vers l'acteur hérité.

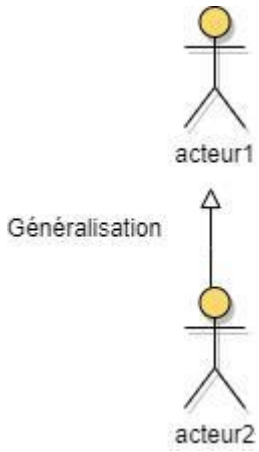


Fig 3 : Relation d'héritage entre acteurs

e) Diagramme de contexte statique

Plutôt que de répertorier simplement les acteurs textuellement, on peut réaliser un premier diagramme appelé **diagramme de contexte statique**. Il suffit pour cela d'utiliser un diagramme de classes dans lequel chaque acteur est relié par une association à une classe centrale unique représentant le système. Ce qui permet en outre de spécifier le nombre d'instances d'acteurs connectées au système à un moment donné.

Bien que ce diagramme ne fasse pas partie des diagrammes UML « officiels », il est souvent trouvé utile dans les cas de projets réels.

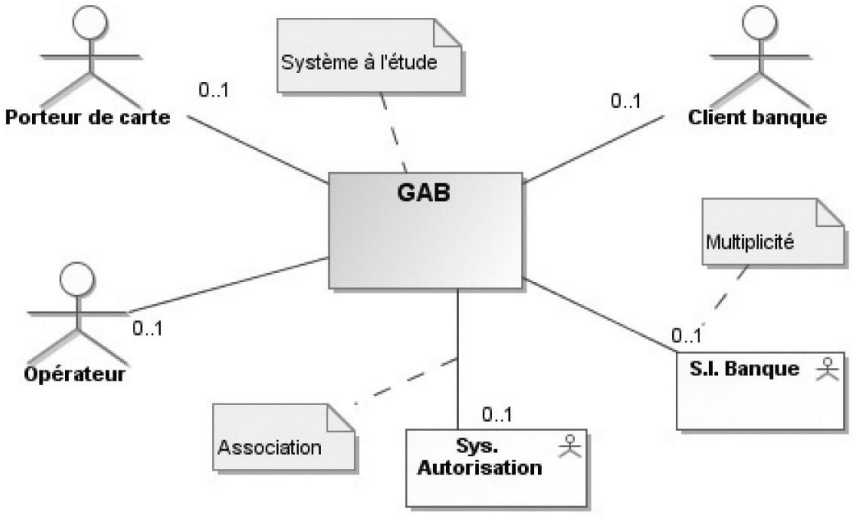


Figure 4 : Diagramme de contexte statique du GAB

3. Les cas d'utilisation

a) Définition

Un cas d'utilisation (« **use case** ») représente un ensemble de séquences d'actions qui sont réalisées par le système et qui produisent un résultat observable pour un acteur.

Chaque cas d'utilisation spécifie un comportement attendu du système considéré comme un tout, sans imposer le mode de réalisation de ce comportement. Il permet de décrire ce que le futur système devra faire, sans spécifier comment il le fera. Les cas d'utilisation permettent de définir les limites du système et les relations entre le système et son environnement. Un cas d'utilisation est une manière spécifique d'utiliser le système.

b) Identification des cas d'utilisation

Chaque cas d'utilisation correspond à une fonction métier du système, selon le point de vue d'un de ses acteurs. Pour chaque acteur, il convient de :

- rechercher les différentes intentions métier avec lesquelles il utilise le système ;
- déterminer dans le cahier des charges les services fonctionnels attendus du système.

c) Représentation

Un cas d'utilisation se représente par une ellipse contenant l'intitulé du cas d'utilisation. Il est recommandé de nommer les cas d'utilisation par un verbe à l'infinitif suivi d'un complément, du point de vue de l'acteur (et non pas du point de vue du système).

Figure 5 : Représentation d'un cas d'utilisation

d) Relations entre les cas d'utilisation

Il existe 3 types de relation entre cas d'utilisations :

- **La relation d'inclusion (include)** : le cas d'utilisation source inclus c'est-à-dire contient obligatoirement le comportement du cas d'utilisation destination.

Figure 6 : Représentation de la relation d'inclusion entre cas d'utilisation

- **La relation d'extension (extends)** : Le cas d'utilisation source étend c'est à dire ajoute son comportement (optionnellement) au comportement du cas d'utilisation destination.

Figure 7 : Représentation de la relation d'extension entre cas d'utilisation

- **La relation de généralisation** : on dit qu'un cas A est une généralisation de B si B est un cas particulier de A.

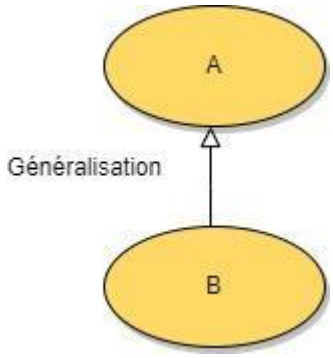


Figure 8 : Représentation de la relation de généralisation entre cas d'utilisation

e) Organisation des cas d'utilisation

Avec UML, il est possible d'organiser les cas d'utilisation et les regrouper en un ensemble cohérent afin de définir un bloc fonctionnel qui s'appelle **package**. Plusieurs stratégies sont possibles : procéder au regroupement par acteur, par domaine fonctionnel, etc.

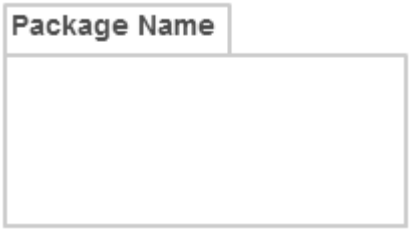


Figure 9 : Représentation d'un package de cas d'utilisation

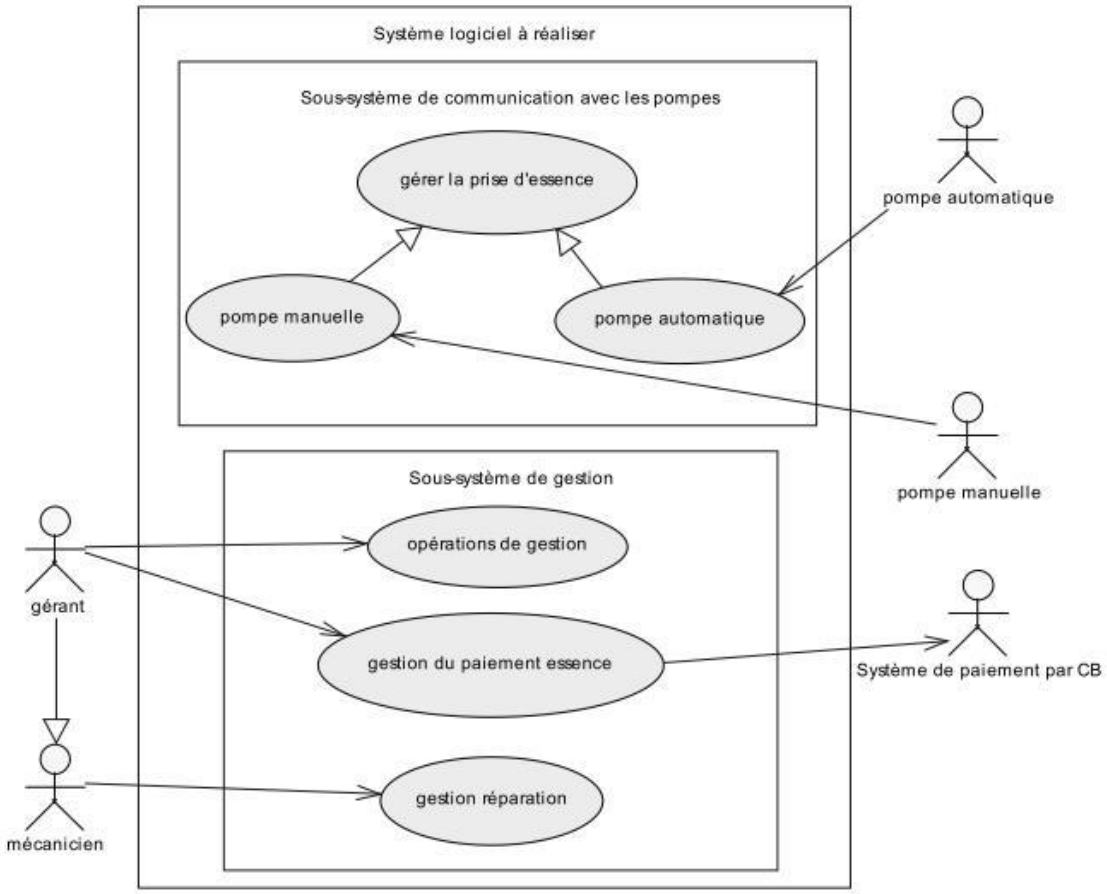


Figure 10 : Exemple de packages de cas d'utilisation

f) Description textuelle des cas d'utilisation

Une fois les cas d'utilisation identifiés, il faut encore les décrire. Pour détailler la dynamique du cas d'utilisation, la procédure la plus évidente consiste à recenser de façon textuelle toutes les interactions entre les acteurs et le système. Le cas

d'utilisation doit avoir un début et une fin clairement identifiés. Il faut également préciser les variantes possibles, telles que le **cas nominal**, les différents **cas alternatifs** et **d'erreurs**, tout en essayant d'ordonner séquentiellement les descriptions, afin d'améliorer leur lisibilité. La description textuelle d'un cas d'utilisation peut se faire à l'aide d'une **fiche de description textuelle**. Cette fiche n'est pas normalisée par UML. Une description textuelle classique se compose de trois parties :

- Partie 1 : Identification.
  - Titre : nom du cas d'utilisation
  - Résumé : description du cas d'utilisation.
  - Acteurs : descriptions des acteurs principaux et secondaires.
  - Dates : date de création et date de mise à jour.
  - Responsable : nom(s) du ou des responsables.
  - Version : numéro de la version.
- Partie 2 : Description des scénarios.
  - Les pré-conditions : état du système avant que le cas d'utilisation puisse être déclenché.
  - Les Scénarios : (un scénario représente une succession particulière d'enchaînements, s'exécutant du début à la fin du cas d'utilisation). Un cas d'utilisation contient en général un **scénario nominal** et plusieurs **scénarios alternatifs** (qui se terminent de façon normale) ou **d'erreur** (qui se terminent en échec).
  - Les post-conditions : elles décrivent l'état du système après l'issue de chaque scénario.
- Partie 3 : 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...).

Un enchaînement est l'unité de description de séquences d'actions.

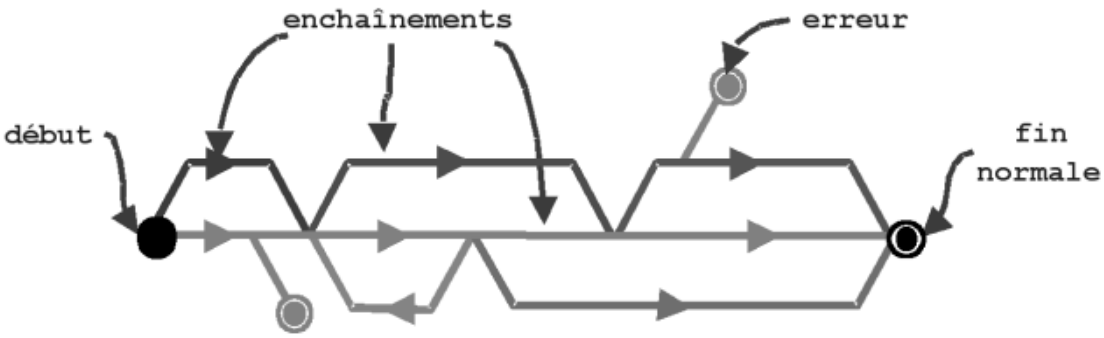


Fig 11 : Représentation des scénarios d'un cas d'utilisation

Sommaire d'identification (obligatoire)	Inclut titre, résumé, dates de création et de modification, version, responsable, acteurs...
Description des scénarios (obligatoire)	Décrit le scénario nominal, les scénarios (ou enchaînements) alternatifs, les scénarios (ou enchaînements) d'erreur, mais aussi les préconditions et les postconditions.
Exigences non-fonctionnelles (optionnel)	Ajoute, si c'est pertinent, les informations suivantes : fréquence, volumétrie, disponibilité, fiabilité, intégrité, confidentialité, performances, concurrence, etc. Précise également les contraintes d'interface homme-machine comme des règles d'ergonomie, une charte graphique, etc.

Tableau 1 : Exemple de structuration d'une fiche de description textuelle de cas d'utilisation

Descriptions textuelles essentielle et réelle d'un cas d'utilisation

C. Larman a introduit dans [Larman 97] la distinction entre cas d'utilisation essentiel et cas d'utilisation réel :

– **Essentiel** : décrit un processus, d'un point de vue analytique. Explicite un processus le plus indépendamment possible de l'environnement matériel/logiciel.

– **Réel** : décrit un processus, du point de vue de la conception. Explicite une solution en termes d'événements, d'interface utilisateur, d'entrées de données, etc.

C. Représentation du diagramme des cas d'utilisation

Le diagramme de cas d'utilisation est un schéma qui montre les cas d'utilisation (ovales) reliés par des associations (lignes) à leurs acteurs (icône du « stick man», ou représentation graphique équivalente). Chaque association signifie simplement « participe à ». Un cas d'utilisation doit être relié à au moins un acteur.

NB : Notez que depuis UML 2.0, un diagramme de cas d'utilisations peut être inclus dans un cadre accueillant tout le contenu graphique. Le cadre a pour intitulé le nom du diagramme et établit sa portée. C'est un rectangle avec un petit pentagone (appelé **tag de nom**) placé dans l'angle supérieur gauche, qui contient le type du diagramme et son nom. Le cadre n'est cependant pas obligatoire lorsque le contexte est clair. La spécification UML définit les tags de chaque type de diagramme, mais cela n'a pas de caractère obligatoire et chaque outil a fait ses propres choix.

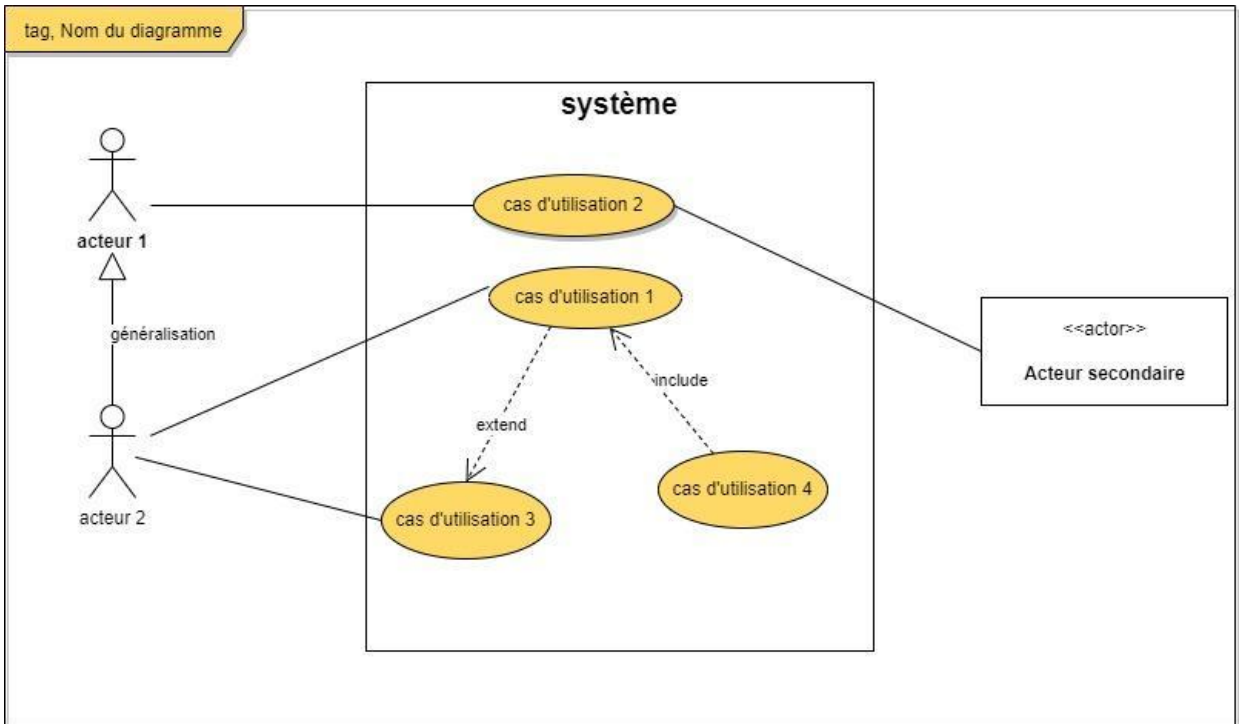


Fig12 : Représentation du diagramme de cas d'utilisation

Quelques règles conventionnelles :

- par défaut, le rôle d'un acteur est « principal » ; si ce n'est pas le cas, indiquez explicitement que le rôle est « secondaire » sur l'association, du côté de l'acteur ;
- dans la mesure du possible, disposez les acteurs principaux à gauche des cas d'utilisation et les acteurs secondaires à droite.

IV. Conclusion

Le diagramme UML des cas d'utilisation est très précieux pour bénéficier d'une vue globale sur une application informatique. Il permet de définir la portée du système et de visualiser immédiatement les liens entre acteurs et cas d'utilisation. En clair, ce diagramme permet de représenter les différentes fonctionnalités offertes par l'application à son environnement. De ce fait, il mérite qu'il lui soit accordée une place importante dans la démarche de conception et de modélisation d'un système.

EXERCICE D'ETUDE DE CAS : étude d'un terminal point de vente [1]

Cet exercice concerne un système simplifié de caisse enregistreuse de supermarché.

Le déroulement normal d'utilisation de la caisse est le suivant :

• Un client arrive à la caisse avec des articles à payer.

• Le caissier enregistre le numéro d'identification (CPU) de chaque article, ainsi que la quantité si elle est supérieure à un.

• La caisse affiche le prix de chaque article et son libellé.

• Lorsque tous les achats sont enregistrés, le caissier signale la fin de la vente.

• La caisse affiche le total des achats.

• Le client choisit son mode de paiement :

– numéraire : le caissier encaisse l'argent reçu, la caisse indique la monnaie à rendre au client ;

– chèque : le caissier vérifie la solvabilité du client en transmettant une requête à un centre d'autorisation via la caisse ;

– carte de crédit : un terminal bancaire fait partie de la caisse. Il transmet une demande d'autorisation à un centre d'autorisation en fonction du type de la carte.

• La caisse enregistre la vente et imprime un ticket.

• Le caissier donne le ticket de caisse au client.

Après la saisie des articles, le client peut présenter au caissier des coupons de réduction pour certains articles. Lorsque le paiement est terminé, la caisse transmet les informations sur le nombre d'articles vendus au système de gestion de stocks. Tous les matins, le responsable du magasin initialise les caisses pour la journée.

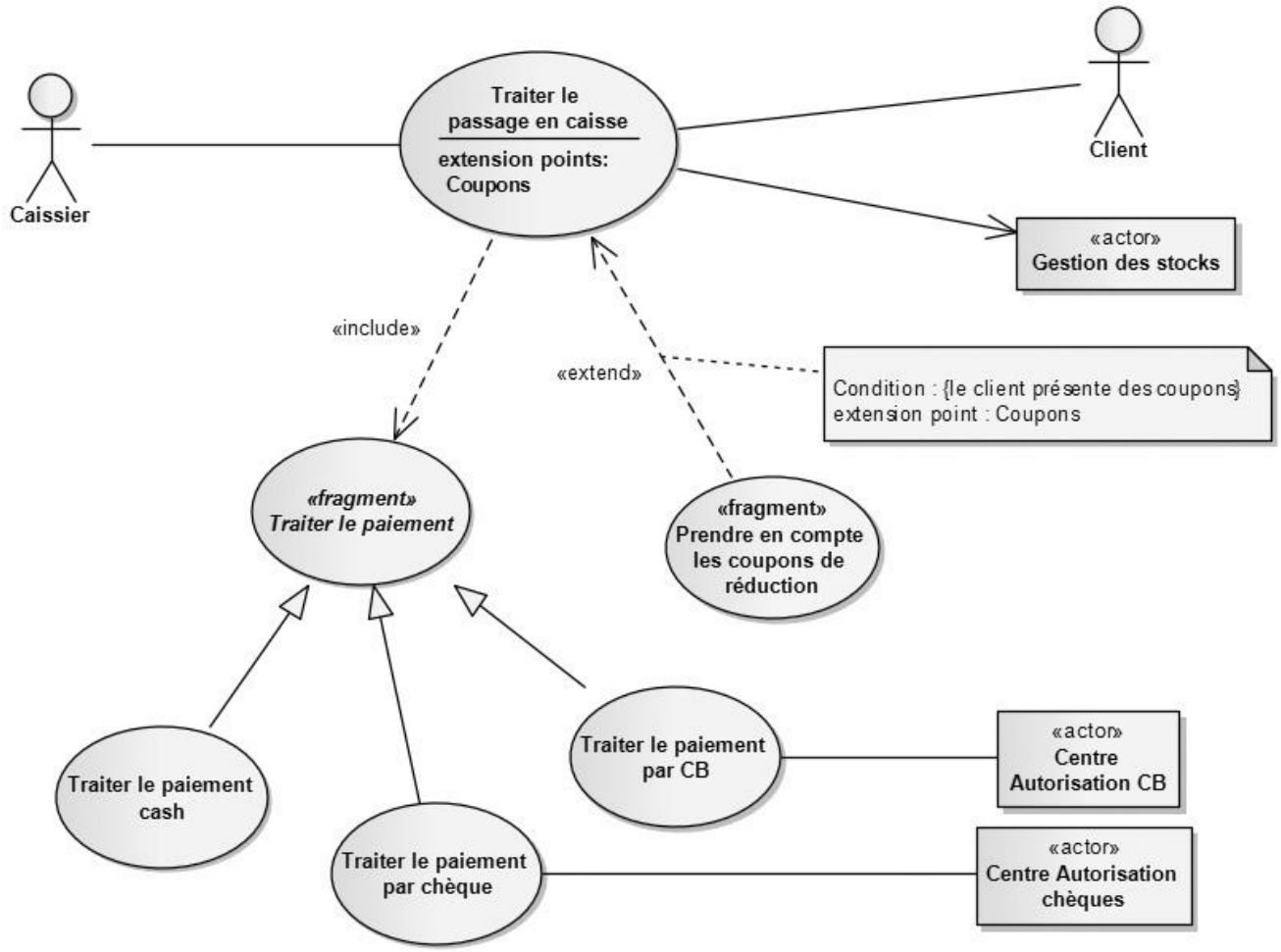
<u>Travail à faire</u> :

1) Réaliser le diagramme des cas d'utilisation du terminal de point de vente.
2) Écrivez une description textuelle essentielle et une description textuelle réelle du cas d'utilisation « Traiter le passage en caisse ».

[1] : Etude de cas complet avec corrigé et explications à retrouver dans le livre « UML 2.5 par la pratique » de la page 25 à la page 39.

CORRECTION

1) Diagramme des cas d'utilisation



2) Descriptions textuelles du cas d'utilisation « Traiter le passage en caisse »

Description textuelle essentielle

Sommaire d'identification

Titre : Traiter le passage en caisse

Type : essentiel détaillé

Résumé : un client arrive à une caisse avec des articles qu'il souhaite acheter. Le caissier enregistre les achats et récupère le paiement. À la fin de l'opération, le client part avec les articles.

Acteurs : Caissier (principal), Client (secondaire).

Date de création : 17/10/16

Date de mise à jour : 11/05/17

Version : 1.8

Responsable : Pascal Roques

Description des scénarios

Préconditions

Le TPV est en service ; un caissier y est connecté ;

Le catalogue produit est disponible.

Scénario nominal

Acteurs	Système
1. Ce cas d'utilisation commence quand un Client arrive à la caisse avec des articles qu'il souhaite acheter.	
2. Le Caissier enregistre chaque article. S'il y a plus d'un exemplaire par article, le Caissier indique également la quantité.	3. Le TPV valide le CPU et détermine le prix de l'article. Le TPV affiche la description et le prix de l'article en question.
4. Après avoir enregistré tous les articles, le Caissier indique que la vente est terminée.	5. Le TPV calcule et affiche le montant total de la vente.
6. Le Caissier annonce le montant total au Client.	
7. Le Client choisit le type de paiement : a. En cas de paiement cash, exécuter le cas d'utilisation « Traiter le paiement en liquide ».	

b. En cas de paiement par carte de crédit, exécuter le cas d'utilisation « Traiter le paiement par carte de crédit ». c. En cas de paiement par chèque, exécuter le cas d'utilisation « Traiter le paiement par chèque ».	
	8. Le TPV enregistre la vente effectuée et imprime un ticket.
9. Le Caissier donne le ticket de caisse au Client.	
10. Le Client s'en va avec les articles qu'il a achetés.	

Enchaînements alternatifs

A1 : numéro d'identification inconnu

L'enchaînement A1 démarre au point 3 du scénario nominal.

3. Le TPV indique au Caissier que le numéro d'identification de l'article est inconnu. L'article ne peut alors être pris en compte dans la vente en cours.

Le scénario nominal reprend au point 2 s'il y a d'autres articles ou au point 4 s'il n'y en a pas.

A2 : demande d'annulation d'article

L'enchaînement A2 démarre au point 2 du scénario nominal.

2. Le Caissier demande l'annulation du dernier article (prix erroné, etc.).

3. Le TPV enlève l'article de la vente en cours.

Le scénario nominal reprend au point 2 s'il y a d'autres articles ou au point 4 s'il n'y en a pas.

Enchaînements d'erreur

E1 : annulation de la vente

L'enchaînement E1 peut démarrer du point 2 au point 7 du scénario nominal.

2.7 Le Caissier annule l'ensemble de la vente et le cas d'utilisation se termine en échec.

Postconditions

• La vente est enregistrée dans le TPV.

NB : Il faudra également décrire chacun des cas d'utilisation spécialisés.

Description textuelle réelle

Le sommaire d'identification est similaire au précédent, mais le type devient : réel détaillé.

Scénario nominal

Acteurs	Système
1. Ce cas d'utilisation commence quand un Client arrive à la caisse avec des articles qu'il souhaite acheter.	
2. Le Caissier enregistre le code universel d'identification du produit dans le champ CPU de la fenêtre de dialogue de la caisse enregistreuse. S'il y a plus d'un exemplaire de l'article en question, le Caissier peut entrer la quantité dans le champ « Quantité », qui est positionné à « 1 » par défaut. Puis il appuie sur le bouton de validation : « Saisie article ».	3. Le TPV détermine le prix de l'article et ajoute les informations sur l'article à la vente en cours. Le TPV affiche la description (sur 6 lettres) et le prix de l'article en question dans le champ « Total ».
4. Après avoir enregistré tous les articles, le Caissier appuie sur le bouton « Fin de vente ».	5. Le TPV calcule et affiche le montant total de la vente dans le champ « Total ».
6. Le Caissier annonce le montant total au Client.	
7. Le Client choisit le type de paiement : • En cas de paiement cash, exécuter le cas d'utilisation « Traiter le paiement en liquide ». • En cas de paiement par carte de crédit, exécuter le cas d'utilisation « Traiter le paiement par carte de crédit ». • En cas de paiement par chèque, exécuter le cas d'utilisation « Traiter le paiement par chèque ».	
	8. Le TPV enregistre la vente qui vient d'être effectuée et imprime un ticket.
9. Le Caissier donne le ticket de caisse au Client.	
10. Le Client s'en va avec les articles qu'il a achetés.	

Exemple de définition des exigences non fonctionnelles pour la description du cas d'utilisation « Retirer de l'argent » dans le cas d'un GAB.

Exigences non fonctionnelles

Contraintes	Descriptif
Temps de réponse	L'interface du GAB doit réagir en l'espace de 2 secondes au maximum. Une transaction nominale de retrait doit durer moins de 2 minutes.
Concurrence	Non applicable (mono-utilisateur).
Disponibilité	Le GAB est accessible 7 jours sur 7, 24 h sur 24 (global[10]). L'absence de papier pour imprimer les tickets ne doit pas empêcher les retraits.
Intégrité	Les interfaces du GAB doivent être très robustes pour prévenir le vandalisme.
Confidentialité	La comparaison du code d'identification saisi sur le clavier du GAB avec celui de la carte doit être fiable à 10^{-6}.

Besoins d'IHM

Les dispositifs d'entrée/sortie à la disposition du Porteur de carte doivent être :

• Un lecteur de carte bancaire.

• Un clavier numérique (pour saisir son code), avec des touches « validation », « correction » et « annulation ».

• Un écran pour l'affichage des messages du GAB.

• Des touches autour de l'écran pour sélectionner un montant de retrait parmi ceux qui sont proposés.

• Un distributeur de billets.

• Un distributeur de tickets.

BIBLIOGRAPHIE

1) Roques P. (2018). UML 2.5 par la pratique - Etudes de cas et exercices corrigés (8è édition). Paris : Editions EYROLLES.
2) Xavier B., Isabelle M. UML2 pour les développeurs – Cours avec exercices corrigés. Paris : Editions EYROLLES. ISBN 2-212-12029-X
3) Bertrand LIAUDET, « UML ANALYSE FONCTIONNELLE - Diagrammes de cas d'utilisation, de séquence, d'activités » [En ligne]. Disponible : https://fr.scribd.com/document/404931365/UML-TP-UC-SeqSys-Acti-Correction-UC . [Accès le 29/01/2022]