Chapitre 3: Spécifications des exigences du logiciel

Avant de développer un système, il faut savoir ce qu'on va développer et pourquoi le développer.

Une exigence ou besoin du logiciel (***software requirement***) est tout énoncé à propos du système à concevoir avec lequel toutes les parties prenantes impliquées sont en accord et qui doit devenir vrai afin de résoudre adéquatement le problème du client.

D'après Larman en 2002 c'est :"***A capability that the system must deliver or a condition that it must be satisfied in order to address a need of a stakeholder***".

Selon la norme IEEE 830-1993 une **exigence** est une condition ou capacité dont un utilisateur a besoin pour résoudre un problème ou atteindre un objectif.

Les exigences d'un système représentent l'ensemble d'éléments qui doivent être implémentées et qui doivent être exprimées selon une méthode bien définie car elles sont une des sources d'échec des projets de développement du logiciel. Les exigences du logiciel sont de deux types :

- ***Les exigences fonctionnelles (besoins fonctionnels)*** qui concernent ce que doit faire le système, le système du point de vue de son utilisateur et qui répondent à la question **Quoi ?**
- ***Les exigences non-fonctionnelles (besoins non-fonctionnels)*** qui regroupent les contraintes du système ou les choix techniques qui répondent à la question **Comment?**

L'**ingénierie des exigences** est le processus qui a pour objet d'établir et de maintenir un accord avec les parties prenantes sur les exigences du système à construire d'un point de vue **contractuel** et **technique** et dont le but ultime est de réaliser un système conforme au juste besoin.

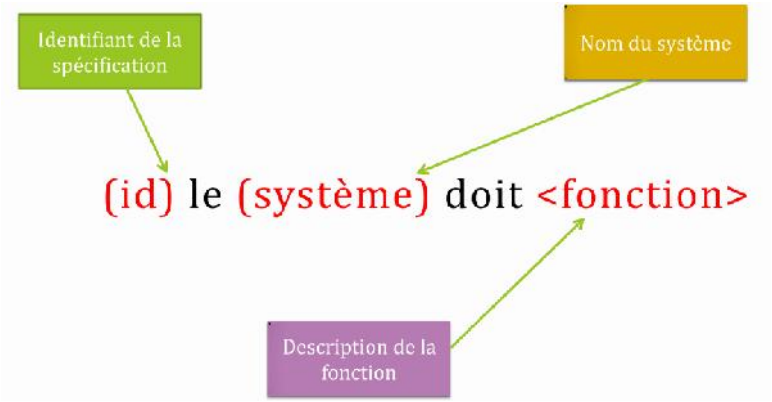
La spécification des besoins permet de comprendre comment est le système du point de vue utilisateur et d'un point de vue interne. La méthode UP (Unified Process) propose deux modèles pour l'expression des exigences ou besoins : le **modèle de spécifications** et le **modèle des cas d'utilisation.**

Le modèle de spécifications basé sur le **langage naturel structuré** convient aussi bien pour les **spécifications fonctionnelles** que les **spécifications non fonctionnelles.** Le modèle des cas d'utilisation qui utilise les diagrammes de cas d'utilisation d'UML s'adapte mieux aux **spécifications fonctionnelles**.

Les outils utilisés pour la spécification des exigences vont d'un simple éditeur de texte (un bloc-notes) aux outils dédiés (StarUML, DOORS, RequisitePro, RTM, etc.).

I. Modèle de spécifications

Le modèle de spécifications est un ensemble de phrases **bien formées** et **numérotées** où chacune est appelée **spécification.** Chaque spécification décrit une **seule fonction du système** et possède un **numéro unique.** Un exemple de structuration formelle d'une spécification bien formée peut être le suivant :



Exemple

Spécifications fonctionnelles :

1. Le système GAB (distributeur) de biller doit vérifier la validité de la carte CIB insérée
2. Le système GAB doit valider le code PIN entré par l'utilisateur
3. Le système doit allouer une somme maximum de 200 000 FCAMER à l'utilisateur
4. Le système doit mettre à jour le solde de l'utilisateur

Spécifications non fonctionnelles :

1. Le système du GAB doit être écrit en C#
2. Le système doit utiliser des listes chaînées
3. Le système du GAB doit utiliser un cryptage 256 bits pour les données sensibles
4. Le système doit vérifier le PIN en moins de 03 secondes

II. Recensement des spécifications

Le recensement des exigences est une activité délicate qui se heurte souvent à plusieurs difficultés relatives aux :

- **Problèmes de compréhension** : Les développeurs et le client ne parlent pas le même langage.
- **Problèmes de volatilité** : Une spécification « valide » peut ne plus l'être après une courte période de temps
- **Problèmes humains** : Conflits, Rétention d'information, etc.
- **Problèmes de portée** : Savoir l'étendue d'une spécification. À quel sous-système elle appartient ?

Malgré ces obstacles, divers sources peuvent être exploitées dans le processus de recensement des spécifications

- Utilisateurs directs

- Personnes ayant une relation avec le système
- Autres systèmes avec lesquels va interagir le logiciel
- Le matériel sur lequel va être déployé le logiciel
- Contraintes juridiques et administratives
- Contraintes techniques
- Objectifs métier (but ou intention guidant la découverte d'exigence)

a) Quelques techniques de collecte des exigences

Plusieurs moyens sont utilisés pour recenser les spécifications.

i. Observation

Souvent utilisée par les ergonomes, cette technique est efficace pour bien comprendre le comportement des utilisateurs sur leur poste de travail ou autour du poste de travail. En notant toutes les astuces qu'un utilisateur trouve pour compenser les manques d'une application (Post-it, carnets, notes…), on prend la mesure de ses limites et de ses failles qu'il faudra combler. On peut leur demander en quoi consiste leur travail, ce qu'ils sont en train de faire ou filmer et enregistrer des sessions de travail.

ii. Le brainstorming ou « remue-méninges »

Idéal pour « défricher » les besoins encore flous et mal organisés des utilisateurs lors du démarrage d'un projet, son principe consiste à organiser une ou plusieurs réunions courtes (environ une heure) durant lesquelles chaque participant est autorisé à dire tout ce qui lui paraît important pour le projet. L'idée est de générer autant d'idées que possible et que personne ne se censure, on élargit, là encore, le domaine des possibles. On peut écrire les idées sur un tableau ou un transparent et à la fin, chacun choisit les trois meilleures idées. Un facilitateur amène le groupe à hiérarchiser les résultats. Le cas échéant, on réduira le périmètre par la suite, d'où la distinction entre besoin initial et besoin à traiter.

*On pourra utiliser des outils de **mind mapping**. Le **mind mapping** est une façon de représenter graphiquement des idées, concepts ou informations. La technique du schéma heuristique ou carte mentale (mind map) part d'une idée centrale ou d'une question, et permet d'organiser et de représenter l'information d'une manière visuelle et structurée de telle façon qu'elle invite à la découverte de nouvelles idées ou informations. La carte peut être réalisée avec des images et de la couleur.*

iii. Entrevues (*interviews*)

C'est la technique la plus directe pour approfondir les besoins, mais peut-être pas la plus simple, puisqu'elle nécessite une **certaine expérience** et un réel savoir-faire. On aura, au préalable, préparé un questionnaire pour guider et optimiser l'entretien. On peut demander à chaque personne interviewée, des détails spécifiques du système, sa vision du problème et de sa solution, des idées à proposer, des sources d'information intéressantes, des dessins et diagrammes.

iv. Questionnaire

C'est une autre technique encore utilisée mais dont on a peut-être un peu abusé à une certaine époque, notamment lorsqu'on sondait la satisfaction des utilisateurs pour faire évoluer une application. Il permet de recueillir rapidement les avis de plusieurs personnes sur un certain nombre de points. Combinant questions fermées, ciblées, et questions ouvertes, le questionnaire doit envisager toutes les alternatives et ne pas induire les réponses.

v. Prototypage

Sa forme la plus simple est un ***prototype sur papier*** qui représente un ensemble de schémas montrant le système en action ou des maquettes des interfaces utilisateurs conçues dans des logiciels de dessin ou dans environnement de développement. L'idée est de recevoir le feedback des utilisateurs assez rapidement.

vi. Atelier facilité ou workshop

Forum d'échanges, un groupe de travail se réunit pour travailler sur un thème donné ou sur un groupe de besoins. Le rôle de l'animateur, l'organisation matérielle ou l'équipement de la salle, le processus structuré (convocation, rédaction des comptes rendus et validation), sont des facteurs clés de la réussite de ces ateliers. L'objectif est de conduire à une prise de décision consensuelle, en séance si possible.

vii. Analyse de l'existant

Il s'agit, ici, d'examiner les applications existantes pour en évaluer les forces et les faiblesses, et de mettre au point la stratégie d'évolution : doit-on réutiliser certaines fonctions ? Dans quelle proportion doit-on remplacer des fonctions existantes ? Comment les améliorer ? On parle souvent de *gap analysis*.

III. Modèle des cas d'utilisation

A. Acteur

Les utilisateurs du système sont appelé « acteurs ». Un acteur représente un rôle joué par une entité externe (utilisateur humain, dispositif matériel ou autre système) qui interagit directement avec le système étudié. 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.

a) Identification des acteurs

Pour identifier les acteurs, les questions suivantes peuvent être posées :

- Qui ou qu'est-ce qui utilise le système ?
- Quel est leur rôle dans l'interaction ?
- Qui installe le système ?
- Qui démarre ou arrête le système ?
- Quels sont les systèmes qui interagissent avec ce système?
- Qui fournit les informations au système ?
- Quels sont les évènements qui ont lieu à un moment donné ?

Il en ressort de cette série de questions que, les acteurs candidats sont systématiquement:

- les utilisateurs humains directs : faites donc en sorte d'identifier tous les profils possibles, sans oublier l'administrateur, l'opérateur de maintenance, etc.;
- les systèmes connexes qui interagissent directement avec le système étudié, souvent par le biais de protocoles bidirectionnels.

Remarques

R1 : Acteurs logiques vs acteurs physiques

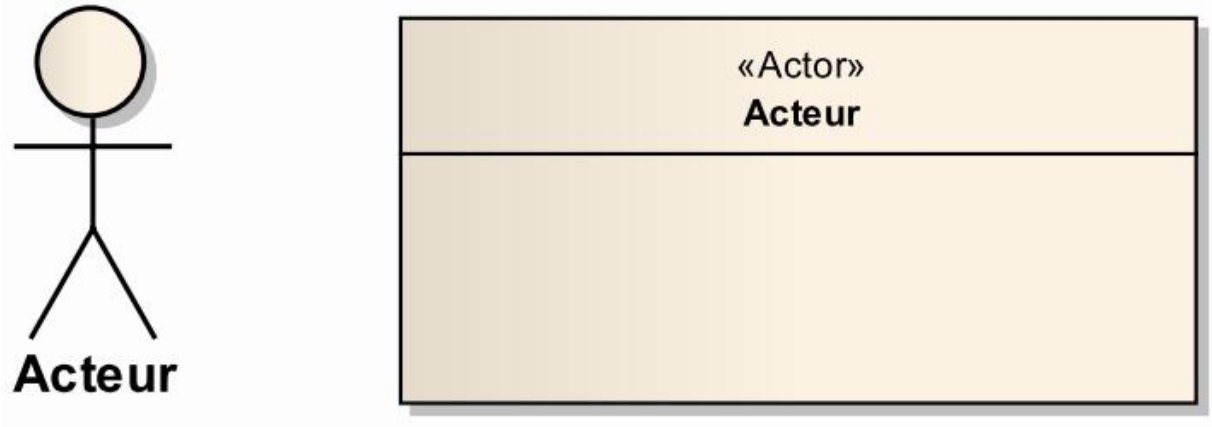
Éliminez autant que faire se peut les acteurs « physiques » au profit des acteurs « logiques » : l'acteur est celui qui bénéficie de l'utilisation du système. Cette règle permet en particulier de s'affranchir des technologies d'interface qui risquent d'évoluer au fil du projet.

R2 : Rôle vs entité concrète

Une même entité concrète peut jouer successivement différents rôles par rapport au système étudié, et être modélisée par plusieurs acteurs. Réciproquement, le même rôle peut être tenu simultanément par plusieurs entités concrètes, qui seront modélisées par le même acteur. Par exemple, même si une seule personne physique peut jouer successivement les rôles de Libraire et Webmaster vis à vis d'un site web, il s'agit bien de deux acteurs distincts, de deux profils différents.

b) 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 représenter un acteur sous la forme rectangulaire avec le mot-clé <<actor>>.



c) Nom d'un acteur

Les acteurs doivent être nommés en utilisant leur rôle au *singulier*, pas au pluriel.

Exemple : visiteur, client

d) Exemple d'application

Identifier les acteurs dans les textes suivants :

Texte 1 : « *Le client parcourt le catalogue de produit et ajoute les produits qui lui plaisent dans son panier. Quand le client souhaite finaliser son achat, il fournit ses informations sur sa carte et sur son adresse de livraison. Le système vérifie la carte du client ensuite valide la transaction.* »

Réponse : Internaute, client, agent service client, SIBanque

Texte 2 : «*Un championnat de damier se déroule comme suit : Un administrateur de l'application crée un championnat. Les participants peuvent s'inscrire comme joueurs dans le championnat. L'administrateur crée l'ensemble des parties. Les participants, une fois inscrits, peuvent consulter leur liste de parties. Les participants, une fois inscrits, peuvent jouer leurs parties. Nous ne nous intéressons qu'aux coups joués par chacun des deux joueurs. Les participants peuvent consulter leur classement.*»

Réponse : participant, joueur, administrateur

B. Cas d'utilisation

Les cas d'utilisation décrivent les interactions entre les utilisateurs (acteurs) du système et le système lui-même. Un cas d'utilisation est une description "narrative" de comment est utilisé le système. En d'autres termes, un cas d'utilisation représente un ensemble de séquences d'actions qui sont réalisées par le système et qui produisent un résultat observable pour un utilisateur particulier. A cet effet, Il représente le système du point de vue de son utilisateur. Il convient donc pour chaque acteur identifié, de rechercher les différentes intentions « métier » selon lesquelles il utilise le système. Enfin les cas d'utilisation sont une technique de capture des besoins fonctionnels du système et sont toujours déclenchés par les acteurs.

a) Processus de création ou d'identification des cas d'utilisation

Ce processus se subdivise en quartes phases :

- Identifier les acteurs
- Identifier comment chaque acteur va exploiter le système
- Obtenir une liste de cas d'utilisation potentiels
- Retenir les cas les plus pertinents

b) Nom d'un cas d'utilisation

Il convient de nommer un cas d'utilisation par un verbe à l'infinitif suivi d'un complément, du point de vue de l'acteur et non du système. Il est conseillé de respecter les critères suivants lors de l'attribution d'un nom à un cas d'utilisation :

- Pas d'espace, utilise la convention UpperCamelCase
- Décrit un comportement, utiliser des verbes
- Le nom doit être parlant et aussi court que possible
- Le nom est unique

Exemple : ChercherUnProduit, ConsulterSonClassement

c) Acteur principal et secondaire

L'acteur principal est celui qui déclenche le cas d'utilisation (CU) tandis que l'acteur secondaire réagit au CU. En d'autres termes, il sera considéré comme acteur principal

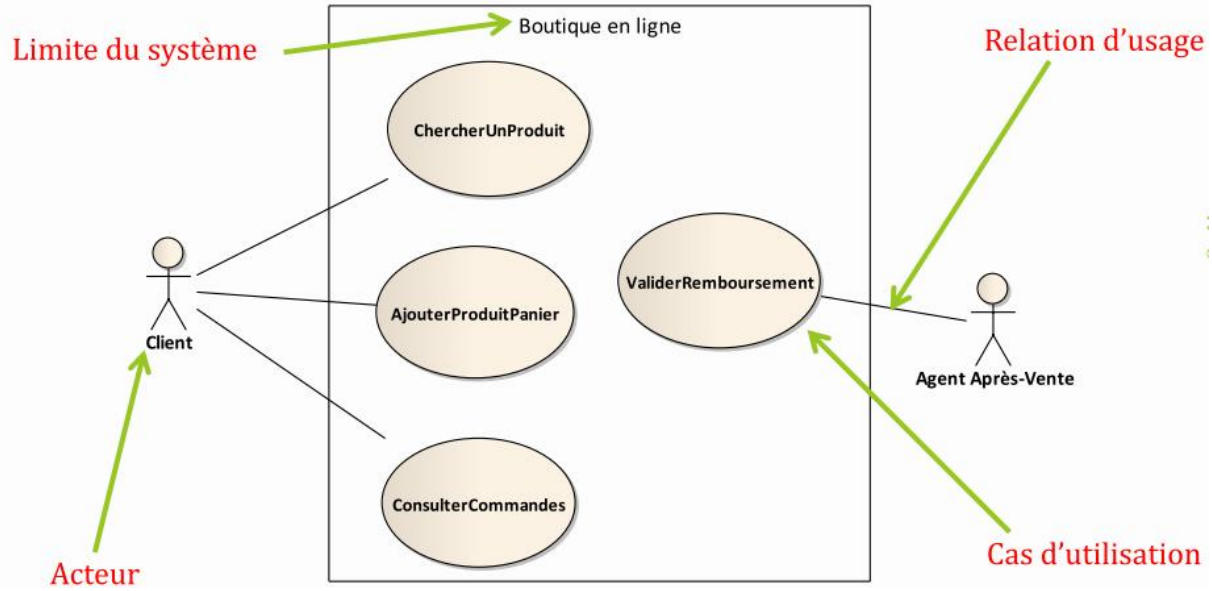
l'utilisateur pour qui le cas d'utilisation produit un résultat observable et comme acteur secondaire l'utilisateur qui participe à la réalisation du cas d'utilisation. Les acteurs secondaires sont souvent sollicités pour des compléments d'informations nécessaires à la réalisation d'un cas d'utilisation. Généralement, l'acteur principal est dessiné à gauche et le secondaire à droite du diagramme de cas d'utilisation.

d) Le diagramme de 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. Chaque association signifie **« participe à »**. L'utilisation de la flèche sur l'association entre le cas d'utilisation et l'acteur, signale un sens unique de transmission d'information. En revanche sans la flèche l'échange d'informations entre l'acteur et le système est bidirectionnel. Un cas d'utilisation doit être relié à au moins un acteur.

Exemple : Considérons le texte : « *Le client parcourt le catalogue de produit et ajoute les produits qui lui plaisent à son panier. Quand le client souhaite finaliser son achat, il fournit ses informations sur sa carte et sur son adresse de livraison. Le système vérifie la carte du client ensuite valide la transaction.* »

Le diagramme de cas d'utilisation correspondant est :

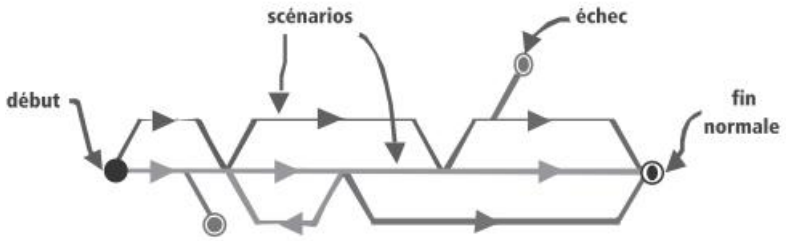


e) Documentation d'un cas d'utilisation

i. Scénarios ou Enchaînements

Un scénario ou enchainement est une suite d'actions numérotées. Il représente une suite spécifique d'interactions entre les acteurs et le système à l'étude. On peut aussi dire que c'est une « instance » d'un cas d'utilisation ou un chemin particulier dans sa combinatoire. Il existe deux types de scénarios : le scénario « nominal » ou « principal » qui satisfait les objectifs des acteurs par le chemin le plus direct de succès et les scénarios alternatifs qui comprennent les déviations ou interruptions qu'a subi un scénario nominal. Ces déviations peuvent se terminer

par un succès (fin normale) ou un échec (erreur). La figure ci-dessous illustre un cas d'utilisation avec ses scénarii.



ii. Fiche de description textuelle d'un cas d'utilisation

Sommaire d'identification (**Obligatoire**)	**CU**: ChercherUnProduit (**Nom du cas d'utilisation**)
	ID:1 (**Numéro d'identification unique**)
	Date de création et modification : 16/09/2016
	Version : 1.0
	Résumé ou Description brève : chercher des produits afin d'alimenter éventuellement le panier ou faire des comparatifs entre les produits
	Acteurs primaires : Client (**liste des acteurs primaires**)
	Acteurs secondaires : (**liste des acteurs secondaires**)
Description des scénarios (**Obligatoire**)	**Préconditions** : le client doit être authentifié (**l'état du système avant que le CU ne démarre, ce qui doit être vrai avant l'exécution du CU**)
	Enchaînement principal ou nominal 1. Le CU démarre quand le client clique sur le lien « chercher » (flux du CU) 2. Le client entre la référence ou le nom du produit 3. Le système affiche la liste des produits triés par nom
	Postconditions : (**l'état du système après le CU, ce qui doit être vrai après l'exécution**)
	Enchaînements alternatifs : (**Flux se déclenchant sous certaines conditions**)
Exigences non-fonctionnelles (**Optionnel**)	Il peut s'agir ici de **performance**, de **sécurité**, **d'ergonomie, etc. On complètera par exemple la description des scénarios par des copies d'écran de la maquette.**

Remarque : **Cas d'utilisation = ensemble de séquences d'actions**

Une erreur fréquente concernant les cas d'utilisation consiste à vouloir descendre trop bas en termes de granularité. Un cas d'utilisation représente un ensemble de séquences d'actions réalisées par le système, et le lien entre ces séquences d'actions est précisément l'objectif métier de l'acteur. Le cas d'utilisation ne doit donc pas se réduire systématiquement à une seule séquence, et encore moins à une simple action.

iii. Mise en œuvre des actions

Comme signaler précédemment, un scénario ou enchaînement est une suite d'action numérotées. La première action est formulée comme ceci : **Le CU démarre quand <l'acteur><fonction>.** Chaque action est formulée comme ceci : **<quelque chose / quelqu'un> <fonction>.**

Exemple :

1. Le CU démarre quand le client clique sur le bouton « se connecter »
2. Le client entre son nom d'utilisateur dans la zone « login »
3. Le client entre son mot de passe dans la zone « mot de passe »
4. Le système vérifie la validité du nom d'utilisateur et du mot de passe

Exemple d'une mauvaise formulation

« Les informations du client sont entrées et vérifiées »

Dans cette formulation, on ne connait pas : Qui entre les informations ? Dans quoi ? Quelles sont ces informations ? Pour s'affranchir de ces erreurs, il ne faut pas s'exprimer en voix passive. Il faut se poser les questions : qui, quoi, quand et où ? et chercher à donner une réponse précise.

iv. Contrôle de flux

Pour éviter de créer des enchaînements alternatifs, on peut faire des instructions de contrôle de flux à l'intérieur d'un enchaînement en utilisant les mots clés **si, pour et tantque**.

Exemple : GérerUnePhotoDansLAlbum

1. Le CU démarre quand l'utilisateur clique sur une photo dans l'album
2. **Si** l'utilisateur clique sur le bouton « supprimer »

   2.1 Le système supprime la photo de l'album
3. **Si** l'utilisateur clique sur le bouton «Noir et Blanc»

   3.1 Le système transforme la photo en noir et blanc

v. Mise en œuvre des enchainements alternatifs

Les noms des enchaînements alternatifs sont formulés de la manière suivante : **Enchaînement Alternatif** : **NomCU**: **NomEnchaînementAlternatif**. L'id de l'enchaînement alternatif obéit à une numérotation hiérarchique. Par exemple, si l'ID d'un CU est **5**, l'id de son premier enchaînement alternatif serait **5.1.**

Pour chaque CU il y a un seul enchaînement principal et plusieurs EA. Eviter de donner trop d'enchaînements alternatifs et regrouper les EA similaires.

Exemple :

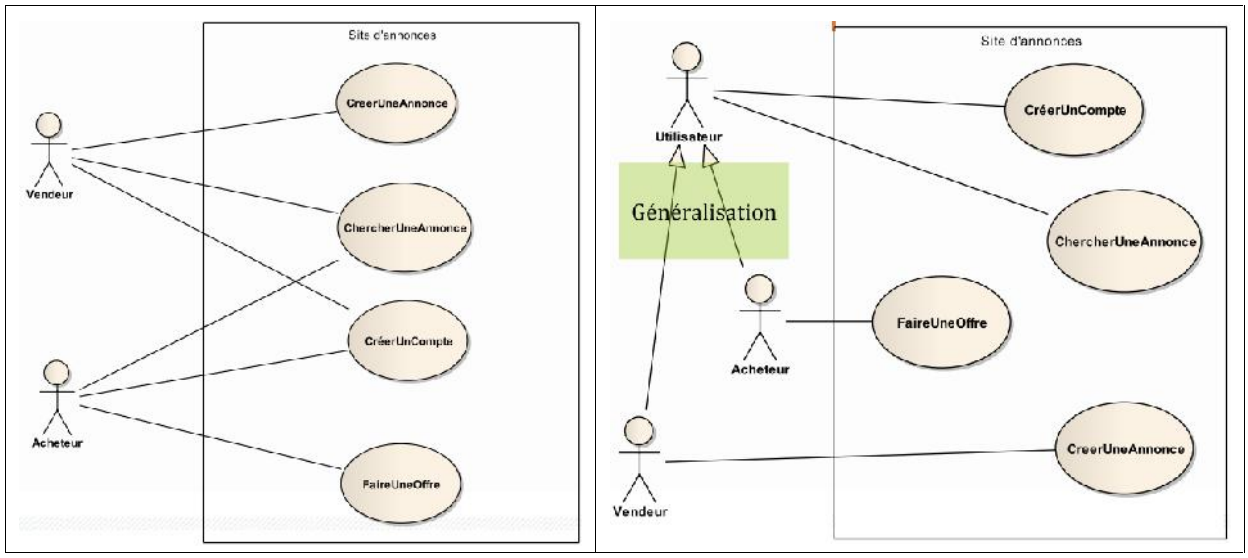
CU: CréerUnCompte
ID: 8
Date de création et modification : 16/09/2016
Version : 1.0
Résumé ou Description brève Création d'un compte utilisateur
Acteurs primaires : Utilisateurs
Acteurs secondaires : Aucun
Préconditions : Aucune
Enchaînement principal ou nominal 1. L'utilisateur clique sur le lien « s'inscrire » 2. L'utilisateur entre son nom d'utilisateur 3. L'utilisateur entre son mot de passe 4. Le système valide le nom d'utilisateur et le mot de passe 5. Le système crée un compte pour l'utilisateur
Postconditions : Le compte du client est créé
Enchaînements alternatifs CompteExistant MotDePasseInvalide Annulation

EA: CréerUnCompte**:**CompteExistant
ID: 8.1
Date de création et modification : 16/09/2016
Version : 1.0
Résumé ou Description brève : Le système informe l'utilisateur que le nom d'utilisateur a déjà été pris
Acteurs primaires : Utilisateurs
Acteurs secondaires : Aucun
Préconditions : Le compte entré par l'utilisateur est réservé par un autre utilisateur
Enchaînement principal ou nominal 1. L'EA démarre à l'étape 4 de l'enchaînement principal 2. Le système affiche que le compte a déjà été donné à un autre utilisateur 3. L'EA revient à l'étape 2 de l'enchaînement principal
Postconditions : Aucune
Enchaînements alternatifs

C. Modélisation Avancée de cas d'utilisation

a) Généralisation des acteurs

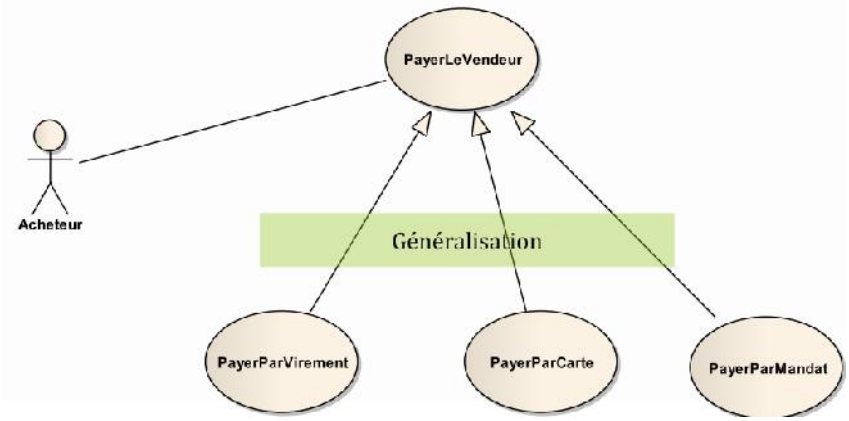
Des acteurs peuvent avoir beaucoup de CU en commun. Un acteur peut être différent d'un autre acteur par quelques CU supplémentaires. La généralisation répond au souci d'encombrement des diagrammes de CU. La généralisation des CU simplifie non seulement la présentation mais aussi la sémantique des CU.



b) Relations entre cas d'utilisation

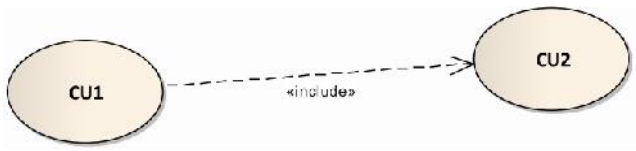
Généralisation des CU

La généralisation des CU est possible quand un CU est une spécialisation d'un autre CU.



Inclusion des CU

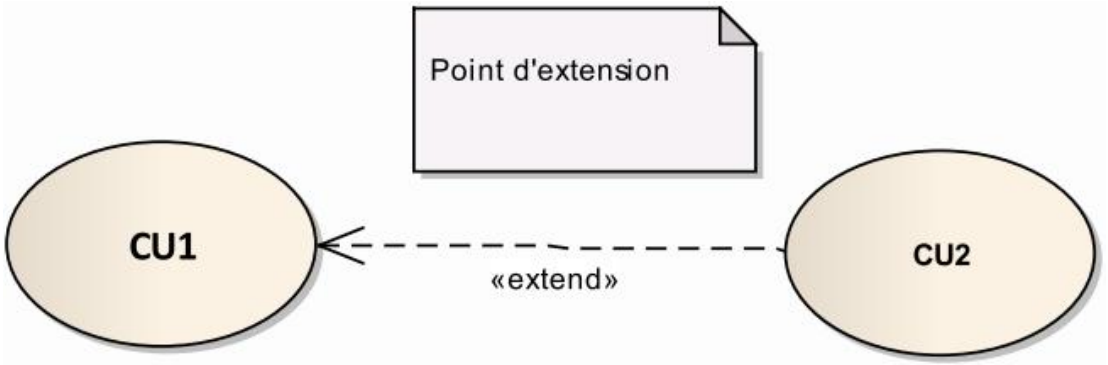
Formalisée par le mot-clé **« include> »,** l'inclusion est une relation entre deux CU (CU1 et CU2). CU 1 est appelé CU de base et CU2 est appelé CU d'inclusion. Quand l'enchaînement arrive au point d'inclusion, CU1 ne s'exécuter que lorsque CU2 s'exécute. L'inclusion évite la répétition.



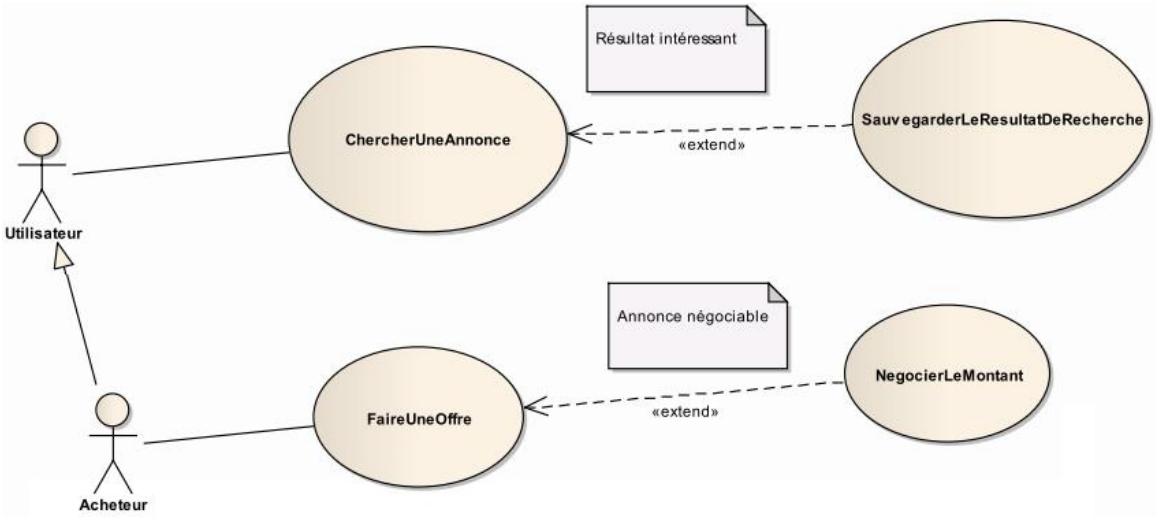
Dans la description textuelle, on peut utiliser le mot clé **Inclure** (**Nom Cas d'utilistion**) inclure un cas d'utilisation. Par exemple dans l'enchaine principal du cas d'utilisation **CeerUneAnnonce**, la première action peut être : 1. Inclure (S'authentifier)

Extension des CU

Formalisée par le mot-clé **« extend », l**'extension est une relation entre deux CU : CU1 et CU2. CU2 étend CU1 par un comportement optionnel qui a lieu sous une certaine condition. Cette condition est appelée « point d'extension »



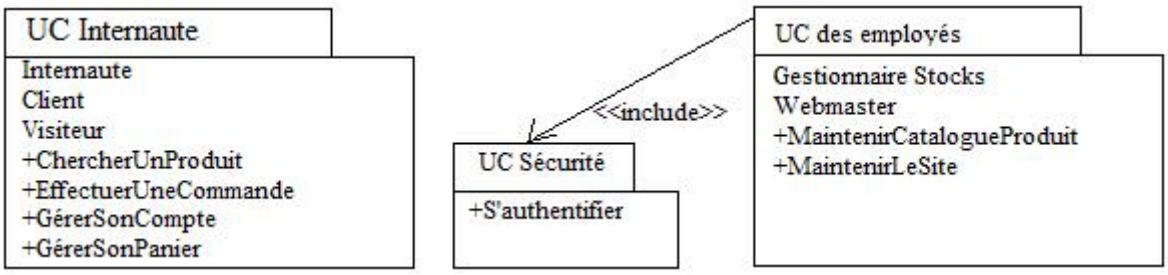
Exemple



Dans le scénario nominal l'extension pourrait être mise en œuvre comme suit :
1. L'utilisateur consulte les détails de l'offre
Point d'extension : NégocierLeMontant (si l'annonce est négociable)
2. Envoyer le montant au vendeur

Structuration en Packages

Le package est un mécanisme de regroupement d'éléments en UML tels que les cas d'utilisation, les acteurs, les classes, etc. Pour améliorer la lisibilité d'un modèle de cas d'utilisation, nous pouvons les regrouper en ensembles fonctionnels cohérents.



c) Classement des cas d'utilisation

Après tout ce travail d'identification des cas d'utilisation, nous pouvons maintenant les classifier en tenant compte des deux facteurs suivants :

- la priorité fonctionnelle : celle-ci est déterminée par le service Marketing de l'entreprise demandeur du logiciel;
- le risque technique : Celui-ci estimé par le chef de projet.

Cas d'utilisation	**Priorité**	**Risque**
ChercherDesProduits	Haute	Moyen
CréerUnCompte	Haute	Bas
EffectuerUneCommande	Moyenne	Haut
GérerSonCompte	Moyenne	Bas
GérérSonPanier	Haute	Bas

d) Traçabilité des cas d'utilisation

Un CU peut décrire une ou plusieurs spécifications. Une spécification peut être représentée par un ou plusieurs CU. La matrice de traçabilité définit les relations entre les spécifications et les CU.

	REQ01 : L'internaute pourra trouver le plus rapidement un produit	**REQ02** : l'internaute pourra saisir un critère de recherche (nom, marque, etc.)	**REQ03** : Le client doit pouvoir modifier les informations sur son compte	**REQ04** : Lorsque l'internaute est intéressé par un produit il peut l'enregistrer	**REQ05** : l'internaute doit à tout moment en ajouter, en supprimer un produit du panier	**REQ06** : l'internaute doit être capable d'imprimer un devis pour commander
ChercherDesProduits	X	X				
CréerUnCompte			X			
EffectuerUneCommande						X
GérerSonCompte			X			
GérérSonPanier				X	X	

D. Les bonnes pratiques

- Ne pas montrer le comportement, plutôt montrer la fonctionnalité
- Les diagrammes ne doivent pas être trop encombrés. Par exemple, au maximum 15 CU par diagramme
- Tous les CU doivent avoir le même niveau d'abstraction ;
- Les spécifications des CU doit être de la même taille (1/2 page à 1 page)
- Si les CU sont trop grands, utiliser les inclusions / extensions / généralisations / package ;

IV. Conclusion

Dans ce chapitre, il a été présenté les modèles utilisés pour la spécification des exigences du logiciel. Les techniques adoptées explorées sont celles proposées par la démarche UP. Dans le prochain chapitre il sera question d'apprendre à faire l'analyse statique du système.