Plan

Introduction

La technologie de l'information n'a cessé de s'évoluer depuis ces dernières décennies. Les méthodes, les techniques et l'utilisation de ces nouvelles technologies sont en constante mutation grâce notamment à l'importance et à la place qu'ils créent dans la vie quotidienne. L'on ne peut plus s'en passer d'un ordinateur. Ce dernier a chamboulé complètement la vie journalière de l'individu. Et l'Homme, de son côté, n'a cessé d'inventer des nouvelles formes voire d'améliorer ce domaine qu'est les nouvelles technologies de l'information et de la communication.

Face à cette concomitante évolution des méthodes et des technologies, un besoin d'impliquer les compétences des responsables aux activités de gestion dans le domaine de l'informatique s'émerge notamment au niveau de la gestion des informations dans l'entreprise. Etant donné que l'objectif de l'entreprise est d'augmenter le rendement, l'informatisation de l'entreprise répond parfaitement à ce besoin réel d'optimisation et de performatisation de la production.

L'accès des entreprises aux ordinateurs a permis d'économiser du temps notamment pour les employés passant la majeure partie de leur temps à faire les mêmes opérations de vérification. Cependant, pour un peu plus de rapidité au sein de l'entreprise, l'informatisation s'avère être la solution pour une meilleure qualité de gestion au sein de l'entreprise.

Rappelons que l'ordinateur a une fonctionnalité de traitement dans le but de arranger et de trier les données. En d'autres termes, l'ordinateur comble les lacunes de l'intelligence humaine. Certes, le P.C. ou (Personal Computer) a une fonction compréhensive, explicative et contribue largement à progresser l'esprit créatif de l'Homme. A juste titre, l'homme et le machine sont complémentaires d'un point de vue d'utilité et de capacité.

L'entreprise elle-même en variation constante n'est plus à même de gérer la production des biens et services à partir des outils et instruments obsolètes de la gestion. De la sorte qu'elle devrait recourir au système informatisé dans une perspective de gain en temps, en argent et en ressources humaines. C'est là l'intérêt d'avoir une gestion informatisée dans l'entreprise.

Ce projet s'inscrit dans ce cadre et justifie ce besoin de faire évoluer la structure interne de l'entreprise et de rendre le système existant plus performant.

L'objectif étant de minimiser le temps et de diminuer les tâches à faire, des problèmes se posent entre autres : comment peut-on optimiser le flux d'informations tout en garantissant une meilleure gestion de la production dans l'entreprise ? Mais ce qui est très important pour nous c'est d'avoir une situation en temps réel de la gestion du stock. Ainsi la question qui se pose est la suivante : dans quelle mesure pourrait-on avoir une situation et un rapport de mouvement du stock en temps réel ?

Pour ce faire, nous allons adopter le plan suivant :

En première partie nous tenterons de faire une brève présentation de l'entreprise et d'en établir la problématique. Puis en deuxième partie, nous allons nous focaliser sur la solution à mettre en œuvre. Dans la troisième partie, nous entrerons dans la conduite de projet c'est-à-dire de l'analyse de l'existant jusqu'à l'ordonnancement et la planification. Quant à la quatrième partie, elle permettra une conception générale et détaillé du projet. Finalement, ceci aboutira au déploiement c'est-à-dire sa mise en œuvre.

1. Présentation

Le développement et la conception d'une application de données nécessitent la présentation de l'entreprise, l'évaluation de la structure et des moyens informatiques pour faire en sortir la problématique et de voir les solutions existantes. C'est une étape importante dans le développement d'un logiciel afin d'éviter un logiciel ne répondant pas au besoin réel de l'entreprise. Cette partie présentera un aperçu global de l'entreprise.

1.1 Présentation de l'entreprise

1.1.1. Présentation générale de la société

La société XXX est une société importante et spécialisée depuis 1993 dans le déstockage de matériel neuf. Les principales activités de l'entreprise sont : la reprise, la revente de parcs informatiques, téléphoniques et multimédias, ainsi que dans le recyclage des équipements en fin de cycle de vie.

Elle fait partie de l'une des plus grandes sociétés de déstockage et reste le fournisseur de nombreux sites de déstockage très connus. De ce fait, ses activités ne se limitent pas seulement à la reprise et la revente. Elle reprend également les matériels auprès des grossistes, importateurs ou particuliers et la difficulté consiste à négocier auprès de ces derniers. Dans ce cas, le souci est de voir l'état réel des matériels pour ainsi évaluer les risques quant à la revente.

1.1.2. Mode de production de la société

Toutefois, la société XXX est une organisation dont la fonction économique principale est de produire des biens et des services (équipement informatique, téléphoniques, multimédias, le recyclage des équipements). Les produits finis de la société XXX sont des équipements destinés à une clientèle susceptible de les acheter et de les payer. A ce titre, la clientèle est bien établie, elle provient un peu partout de la France. Dans une perspective d'augmenter la clientèle, la société XXX projette d'intensifier sa vente et pourra ainsi doubler son chiffre d'affaires.

1.2 Structure et moyens informatiques

1.2.1. Une structure non informatisée

La structure de la société XXX lui vaut l'envergure d'une grande société de déstockage de par la présence d'un grand entrepôt de stock. Cet entrepôt, grâce à sa surface de 1500m², peut stocke plus de 1000 palettes de matériels ce qui montre déjà l'ampleur et la carrure de l'entreprise. Face à cette structure et cette place qu'elle tient dans le marché de déstockage, la société XXX ambitionnerait de lancer son propre affaire via le web. En effet, elle désirerait faire une vente ligne de matériels déstockés. Dans ce cas, elle serait totalement autonome et ne dépendra plus de ces nombreux sites de déstockage.

En ce qui concerne les moyens informatiques, une toute petite partie de la gestion de l'entreprise est actuellement informatisée. Effectivement, très peu de travail dans le processus de production sont seulement informatisées. La majorité des opérations sont faites manuellement entre autres les bon de commande, bon de livraison, saisie des commandes…et ceci exclusivement traitée en Excel.

De ce fait, une portion de la comptabilité est seulement traitée sur informatique. Dans ce contexte toutes les opérations de « back office » sont très difficiles à réaliser à partir de données fiables. En d'autres termes, la société XXX présente alors une lacune dans le flux d'information.

Cependant, il faudrait identifier les principaux acteurs et pouvoir ainsi déboucher sur un schéma de circulation de l'information.

En effet, l'analyse de flux d'information est primordiale dans la gestion informatique de l'entreprise. Il faut cependant distinguer les acteurs internes des acteurs externes.

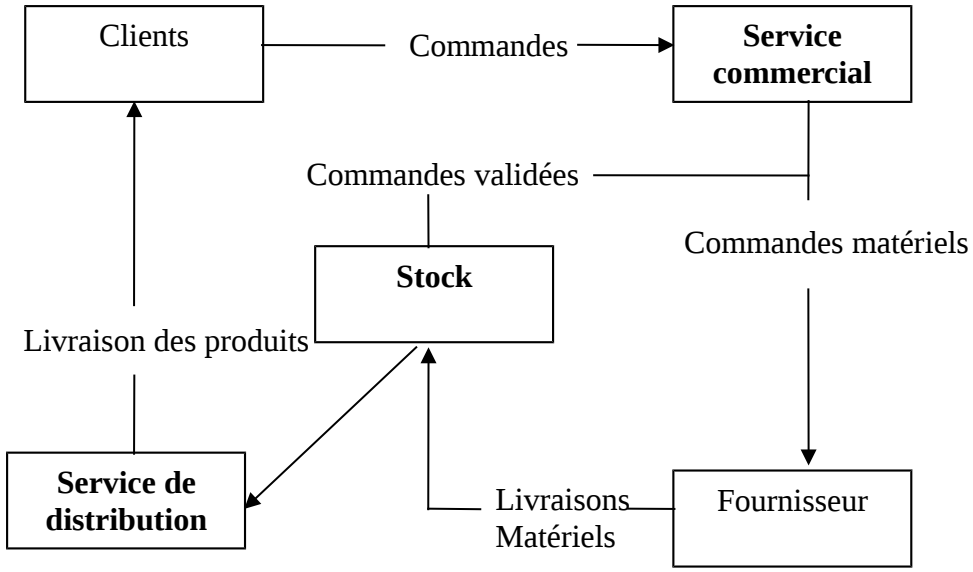
1.2.2. Identification des acteurs

Par exemple, les acteurs internes sont le service commercial, le service comptabilité et le service de distribution, le service d'approvisionnement, le service de dépôt, tandis que les acteurs externes sont les clients, les fournisseurs. Toutefois, il faudrait souligner dans le langage informatique que le sens donner au mot acteur ne désigne pas forcément une personne mais indique généralement un groupe de personnes.

Un schéma simpliste des mouvements des acteurs se passant au niveau du service commercial fait montre d'une circulation d'information bien établie.

1.2.3. Exemple de circulation des informations

Figure1 : Modèle de Circulation des informations au niveau du service commercial



Ce schéma met en évidence les acteurs internes (service commercial, stock, service de distribution) et les acteurs externes (clients et fournisseur).

Toutefois, la structure d'entreprise n'est pas seulement représentée par le service commercial, mais elle s'applique aussi aux différents domaines de la gestion de l'entreprise à savoir les autres services. Dans la société XXX, les services fonctionnent ensemble pour le bon déroulement de l'organisation humaine et financière de l'entreprise.

La connaissance des flux d'informations nous permettra en mieux de voir en générale la structure de l'entreprise et les moyens informatiques mis à disposition.

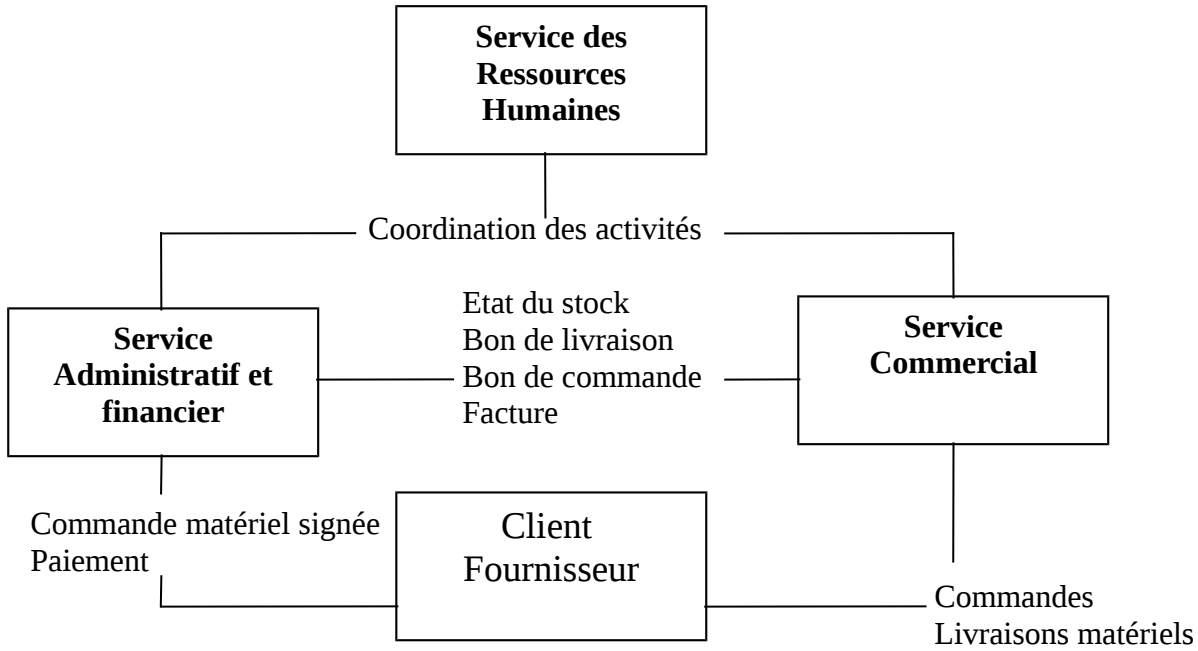
1.2.4. Les flux dans l'entreprise

Effectivement, les flux sont habituellement matérialisés par des documents, factures, commandes ou bons de livraison. Le développement d'un logiciel de technologie informatique et de communication traduira notamment cette dématérialisation des documents. Voilà pourquoi il est opportun de faire un schéma global des flux d'informations réels dans la société XXX.

Comme nous l'avons dit précédemment la société XXX est une société de déstockage de matériel neuf. De ce fait, il existerait un échange d'information important au niveau des services dans l'entreprise à savoir le service commercial, le service de distribution et le service de dépôt, le service d'approvisionnement.

Au niveau de chaque service, il présenterait des sous-services : dans le service commercial, il y a la vente, dans le service financier, il y a la comptabilité et dans le service de distribution il y a la livraison,etc. Voici un schéma montrant une circulation des flux d'informations au niveau de chaque service :

Figure 2 : Circulation des flux des informations entre les acteurs internes et les acteurs externes



Cette figure met en exergue les différentes activités de l'entreprise en l'occurrence les échanges internes et externes.

1.2.5. La circulation des flux de la société XXX

Les flux comportent une dimension informationnelle importante surtout dans la gestion informatisée de la société. Ils jouent un rôle important dans le système d'information de

gestion. Ces flux sont traduits en diagramme des flux ou schéma de circulation de l'information, important pour l'entreprise et le développement d'un système de gestion améliorée lesquels sont déjà mis en évidence antérieurement.

1.3 Problématique

Malgré les échanges de flux important entre les services de la société XXX notre attention se focalisera surtout sur la connaissance de l'état du stock en temps réel lequel impliquera régalement une contribution réelle des autres services. Le problème réside dans le fait que l'inventaire physique du stock nécessite une mobilisation réelle des ressources humaines. Une personne est aujourd'hui rémunérée quasiment à plein temps pour faire régulièrement un inventaire physique du stock.

Tout d'abord il faudrait améliorer la gestion des approvisionnements c'est-à-dire la gestion du stock par le biais d'un meilleur système d'inventaire efficace. Pour ce faire, la méthode appliquée sera une méthode consistant à restituer systématiquement un stock après la vente ou consommation.

Par ailleurs XXX souhaite lancer un site Web de vente en ligne pour augmenter son chiffre d'affaire. La fiabilité des informations concernant le stock est donc primordial car il faut pouvoir satisfaire en temps réel les demandes des clients. Ce site web facilitera la vente mais également les relations de la société XXX avec les clients.

En résumé, la société XXX, dans un objectif ultime d'améliorer son système d'information, cherche à gérer les matériels disponibles (stock), l'entrepôt, les clients et les fournisseurs, et éventuellement les échanges de flux d'information interne. Pour la société XXX, il est impossible pour le moment d'entrer dans ce système de vente à cause de la méconnaissance du stock disponible. Le risque de vendre des produits en rupture de stock est trop important.

Toutefois, la gestion de stock est concerne plusieurs acteurs. Par rapport à ce qu'on vient de mentionner précédemment, le schéma est le suivant : ce sont les fournisseurs qui approvisionnent le stock, l'entrepôt stock les matériels, les clients commandent et reçoivent la marchandise et enfin ce sont les services internes de l'entreprise qui gèrent tout ça.

1.4 Les solutions existantes

1.4.1. Approche des solutions existantes

En évaluant les besoins de l'entreprise et les problèmes résidant dans la gestion de l'entreprise, la société XXX souhaite ainsi restructurer totalement son système d'information afin d'améliorer la gestion de son stock et la traçabilité de ses produits.

Il est vrai que de nombreuses solutions existent pour répondre à ce besoin. Ce qui nous vient à l'esprit en premier ce sont les ERP (*Enterprise Resource Planning*) appelés également PGI (*Progiciels de Gestion Intégrés*), l'objectif étant la gestion de l'ensemble des processus de l'entreprise. En effet ce sont des applications dont le but est de coordonner l'ensemble des activités d'une entreprise. Du fait de cette fonction intégrant le système en entier (gestion des ressources humaines, gestion comptable, vente, distribution, approvisionnement), le PGI serait une base de données contenant toutes les informations nécessaires à l'entreprise. Dans ce cas, le PGI est capable d'établir la communication avec les fournisseurs, les clients. En d'autres termes, elle permettrait de relier les acteurs de production entre eux. En théorie, sa

principale fonction serait de construire une base de données informatique permettant de faire toutes les tâches suivante :

- La gestion des stocks ;
- La gestion des ventes ;
- La gestion comptable ;
- La gestion financière ;
- La gestion de la production ;
- La gestion des achats.

De ce fait, les ERP ou PGI présentent une multitude d'avantages entre autres :
- La cohérence et la clarté des informations ;
- L'indivisibilité du système d'information facilitant la communication interne et externe ;
- La réduction des coûts (exemple : réduction du nombre de salariés ayant pour tâche la saisie des données à comptabiliser).

Pourtant, dans notre cas les ERP ne sont pas adaptés à nos besoins. Etant donné que notre principale préoccupation c'est la gestion de la traçabilité des produits, alors le progiciel ne satisfait pas réellement à ce besoin.

Par exemple, les ERP n'offrent pas totalement des fonctions dédiées à la gestion de production surtout dans notre cas très spécial. Ils ne peuvent pas en aucun cas permettre une reconfiguration des articles à la demande. Ainsi, les progiciels n'ont pas également la fonctionnalité de traiter les produits et les sous-produits alors que le logiciel développé répond parfaitement à ce besoin.

Par soucis d'avoir une suivie et une connaissance de la disponibilité du stock, le progiciel ne permet pas de regrouper, d'inventorier et de définir le statut du stock qui peut s'évoluer au fur et à mesure que le stock entre dans l'entrepôt.

Bien que les ERP présentent plusieurs fonctionnalités, cela ne répondent pas complètement aux besoins de la société XXX notamment par rapport à la gestion du stock : le changement de statut des produits, l'éclatement ou le fusion des produits.

1.4.2. Evaluation des besoins par rapport aux solutions existantes

Cette brève analyse des solutions existantes nous montre qu'elles présentent de nombreux inconvénients par rapport aux besoins de la société XXX :

- Le périmètre fonctionnel des solutions existantes est trop large. En effet de nombreuses fonctionnalités sont inutiles (gestion de paie, gestion des ressources humaine, aide à la décision, groupware etc…) en particulier pour une PME comme XXX.
- Complexité pour la mise en œuvre et l'utilisation.
- Nécessité d'adapter certains processus de l'entreprise au logiciel.
- Difficultés d'appropriation par le personnel de l'entreprise.
- La gestion du stock de XXX est très spécifique. Les produits ont un statut qui peut changer plusieurs fois au cours de leur cycle de vie. Cette particularité ne permet pas d'adapter les solutions existantes qui sont assez rigides.

2. Solution à mettre en œuvre

2.1 Description

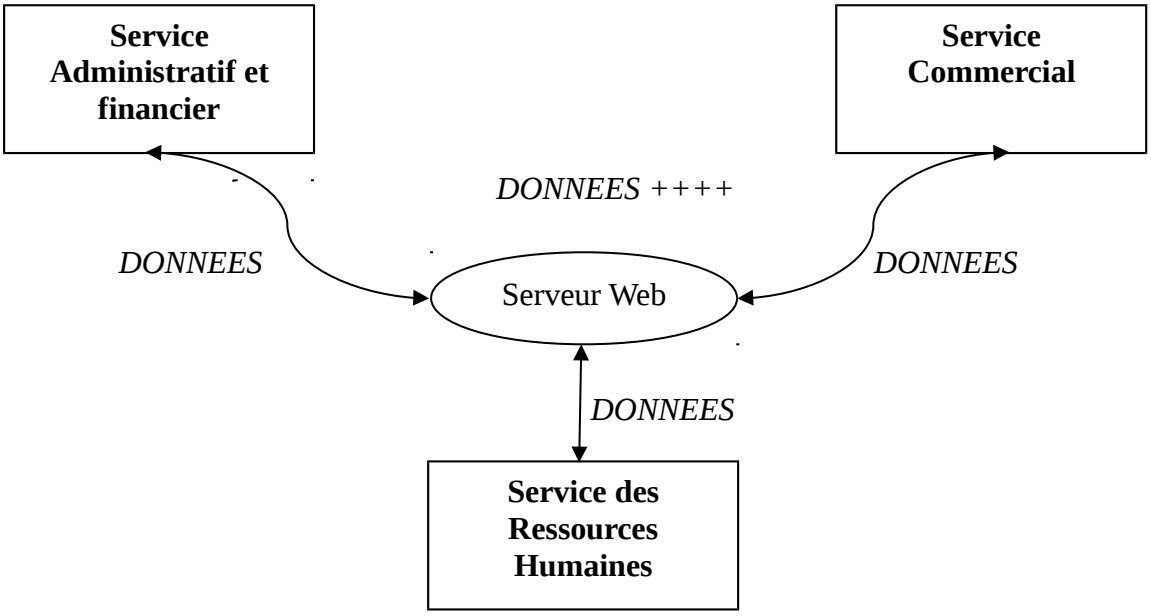
Etant donné que les services mis à disposition par les solutions existantes ne répondent pas parfaitement aux besoins de la société XXX, il faudrait chercher une autre alternative en évitant les inconvénients des solutions existantes. Cependant, il importe de développer un logiciel personnalisé permettant de satisfaire au maximum les acteurs (internes et externes) de la société XXX.

2.1.1. L'application web dans l'entreprise

Dans ce projet, nous avons préconisé qu'une application web serait la mieux adaptée. En effet la partie du système d'information concernée par cette application serait installée sur un serveur Web et tous les modules seront développés autour de cette base de données centralisée. La centralisation des données permettrait la synchronisation et l'interdépendance des services et des acteurs entre eux.

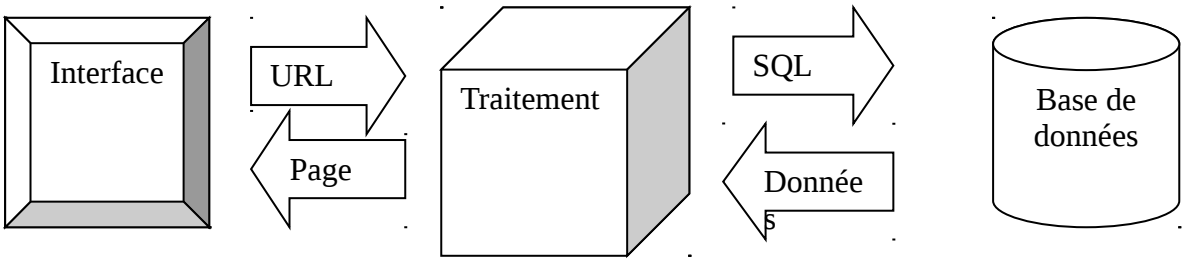
Ceci est illustré dans la figure suivante :

Figure 3 : Exemple d' échanges de flux d'information à travers un serveur web



Ce schéma montre de l'importance d'un serveur web dans une entreprise même si sa fonction reste juste au stockage de données.

Figure 4 : Architecture de l'application web dans l'entreprise



d'échange de do nte : l'échange des don es s'effectuent plus directement mais à travers le serveur web. Ce qui facilite l'organisation interne de l'entreprise et précisément améliore les échanges de flux entre les acteurs. Grâce au serveur web, les données des services sont accessibles en temps réel. Le serveur web permet ainsi d'emmagasiner les données afin d'obtenir des informations fiables émanant des autres services.

2.1.2. Description de l'application

Notre travail consiste alors à développer un logiciel permettant la gestion automatique des fournisseurs, du stock et des clients.

Si l'on reprend ce qui a été dit, nous pourrons répondre aux inconvénients des solutions existantes :

- Les fonctionnalités répondront aux besoins exacts de la société XXX sans modules inutiles et complexes.

- Le logiciel développé sur mesure sera simple à mettre en œuvre et à utiliser.

- Le logiciel sera développé en fonction des processus de XXX et non l'inverse. Il n'y aura donc pas à modifier en profondeur les habitudes de travail des employés.

- Le logiciel sera développé en concertation avec tous les employés de l'entreprise pour répondre au besoin de chaque poste de travail.

- Le logiciel permettra de répondre à la spécificité de la gestion du stock des produits de XXX.

- Le langage web (HTML) est multi-plateforme. Aucune installation ni configuration de logiciel sur les ordinateurs n'est nécessaire ce qui facilitera notamment l'accès à l'application par des PDA.

En bref, la conception du logiciel permettra ainsi de :

- la gestion des tiers (prospects, des clients, fournisseurs, salariés)
- la gestion des documents et l'archivage des documents (devis, bons de commandes, bons de livraisons, factures, avoirs financiers,…)
- la gestion des spécificités de l'activité de l'entreprise (négoce, fabrication, prestations de services…)
- la gestion des achats
- la gestion des livraisons

- la gestion du ou des stocks

2.1.3. L'architecture n-tiers

L'architecture de l'application sera de type *n-tiers* (structurée en plusieurs couches distinctes et indépendantes). Toutefois, il importe de mettre en évidence et de voir en détail ce que c'est l'architecture de type *n-tiers.*

Dans cette application, les trois couches (couche présentation, couche métier, couche d'accès aux données) sont intimement et fortement liées et s'opèrent sur la même machine. Ici donc, l'architecture client-serveur n'est plus en évidence, on parlerait plutôt d'informatique centralisée.

Etant donné que la société XXX emploi un bon nombre de salariés, nous ne trouvons plus dans un contexte simple-utilisateur mais dans un contexte multi-utilisateurs c'est-à-dire que l'application est manipulée par plusieurs acteurs. Le contexte est ici un contexte mutli-utilisateurs et non un contexte simple-utilisateur, donc cela suppose la mise en œuvre de deux types d'architectures un-tiers. Soulignons toutefois que l'architecture en tiers spécifie l'organisation des éléments d'exploitation mis en œuvre pour réaliser le système. Chaque *tiers* indiquera une responsabilité technique des différents acteurs d'exploitation d'un système.

Ainsi, dans un contexte multi-utilisateurs, nous distinguerons :

- des applications sur site central (*mainframe*) ;
- des applications partagées sur des machines indépendantes communiquant par partage de fichiers.

a) Application sur site central (ou mainframe)

Comme son nom l'indique, cette application propose la centralisation des données c'est-à-dire que les utilisateurs ont la possibilité se connecter sur un serveur central et peut ainsi accéder directement aux données stockées dans le serveur. Ce qui est normal et pratique pour un accès multi-utilisateurs. Par conséquent, cette application présente plusieurs avantages du fait de la puissance qu'elle procure par rapport à l'utilisation optimale des ressources et le traitement des données. Ainsi, elle offre une grande disponibilité de données fiables et une administration certaine de la base des données.

b) Application un-tiers déployée

Dans un environnement multi-utilisateurs, il est vrai que plusieurs acteurs entrent en interaction mais dans ce type d'application, on met l'accent sur le système réseau c'est-à-dire le déploiement de l'application se fait sur plusieurs machines.

Effectivement, ce type d'application ne répond seulement qu'aux besoins d'une personne isolée ou d'un petit groupe de personne. Dans notre cas, la société XXX s'intéresse à l'inventaire et l'état des stocks disponibles, ici la mise en œuvre de l'application tient sa source dans le service de stock. En d'autres termes, l'on pourrait envisager un moteur de base de données de stock afin que les utilisateurs puissent avoir accès en temps réel. Cependant, le souci de la société XXX ne reste pas là, les opérations de comptabilité sont aussi à prendre en compte. Ce qui fait que l'application ne devrait pas seulement répondre aux besoins d'une seule personne isolée.

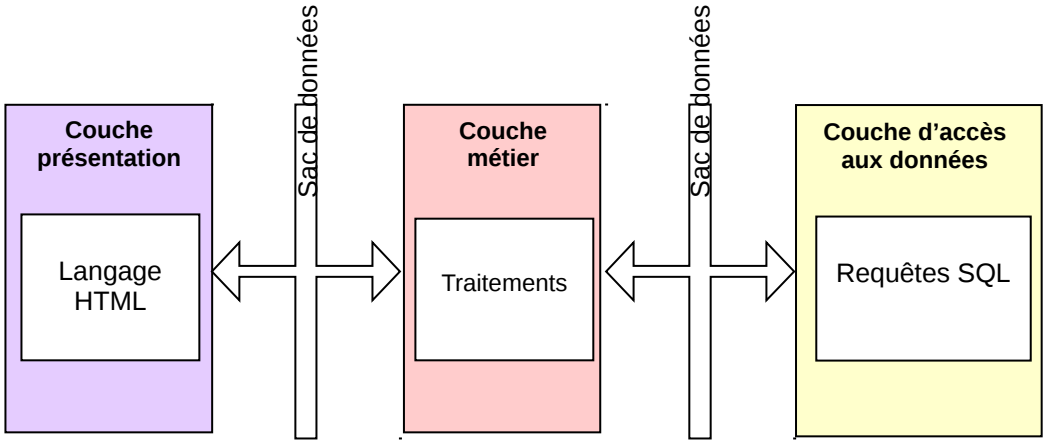
Dans ce cas, l'inconvénient demeure dans l'instabilité du réseau. L'envoi simultané des données concomitant avec l'exécution des requêtes pourrait saturer le réseau. Il serait non seulement difficile de manipuler et de consulter les données mais également de conserver la fiabilité et l'intégrité des données. Ce type d'application pourrait convenir à la société XXX dans la mesure où les données sont exploitées par de petits groupes de travail. Voilà pourquoi l'intérêt de créer une application personnalisée.

c) Présentation de l'architecture d'application n-tiers

Voici la présentation de notre architecture d'application *n-tiers* :

- Une couche présentation (interface graphique) dotée de plusieurs composants de gestion spécifiques.
- Une couche métier (cœur du système) où est concentrée "l'intelligence" de l'application. Chaque module de calcul devra être en mesure d'utiliser les possibilités offertes par le module voisin.
- Un sac de données génériques pour les échanges entre chaque couche. Pour faciliter la communication entre les couches de l'application, il sera un mis en place un « sac » de données. Il sera distribué à travers chaque objet de l'application pour déposer des données de types divers.

Figure 5 : Présentation de l'architecture d'application



L'architecture *n*-tiers ou l'architecture distribuée ou architecture multi-tiers désigne le partage d'applications entre de multiples services contrairement à la multiplication des niveaux de service. Pourtant, l'architecture *n-tiers* implique la prise en compte des niveaux d'abstraction d'une application à savoir :

La distribution est servie par l'utilisation de composants introduisant les concepts orientés objets. Nous allons encore revoir en détail ces concepts orientés objets un peu plus tard. Quoique l'on puisse envisager une communication certaine entre les machines et les composants même étant hétérogène et implantés distinctement. La communication devient alors facile.

d) Les trois niveaux d'abstraction de l'architecture n-tiers

L'on entend que l'application se découpera en trois niveaux d'abstraction :

1) La couche présentation ou la couche vue appelée encore IHM (Interface Homme/Machine) permet l'interaction de l'application avec l'utilisateur. Cette couche gère toutes les actions directes à l'application c'est-à-dire la face dévoilée sur écran et accessibles à tous les utilisateurs. C'est en quelque sorte la présentation de l'information, cela veut dire que c'est l'interface réelle de l'application à développer. Cette couche devrait être accessible et ergonomique autant que possible ;

2) Le traitement des données se fait en 2 sous-parties :

- les traitements locaux : s’intéressent aux traitements du dialogue avec l'IHM, particulièrement pour faciliter leur emploi ;
- les traitements globaux : appelées aussi *Business Logic* (logique applicative) contiennent les règles simples et internes qui régissent l’entreprise.

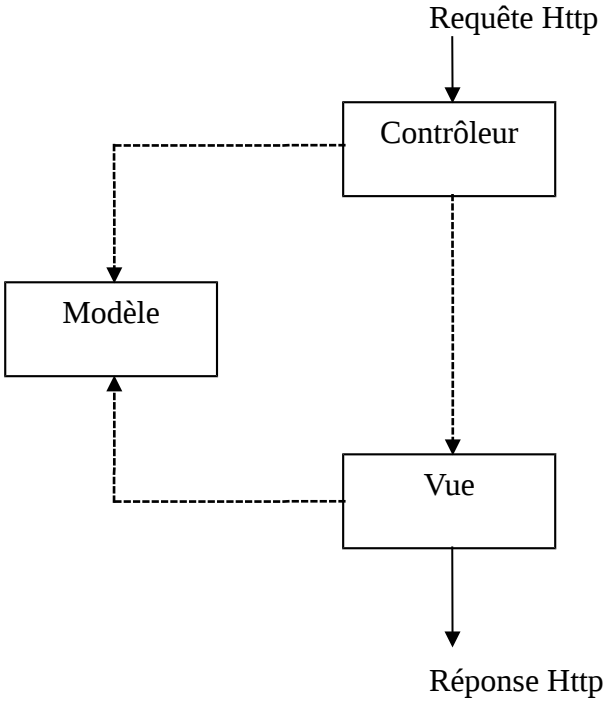
3) Les données : l'accès aux données rassemblent les mécanismes permettant la gestion des informations entreposées par ou pour l'application.

2.1.4. Développement de l’application à l’aide du framework php-mvc [PHP.MVC]

PHP-MVC est un framework libre de droit utilisé pour le développement d’application web en langage PHP. PHP-MVC implémente le patron de conception Modèle – Vue – Contrôleur qui permet de structurer l’application en couches distinctes. L’on recherche plutôt à distinguer nettement les couches présentation, traitement et accès aux données.

Voici un schéma montrant un modèle MVC global :

Figure 6 : Modèle global MVC



Comme on l’avait déjà mentionné précédemment, l’objectif de MVC est de séparer les couches d'une application (en 3 couches distinctes, au minimum, et le plus souvent). On va donc distinguer :

1. Le modèle encapsule la logique métier mais également la manipulation des sources de données. Dans ce cas, l’on aura recourt aux designs patterns lesquels pourrait être utile. Par exemple : TableDataGateway, ActiveRecord ....

2. En ce qui concerne la vue : elle présente les données au client les demandant (un utilisateur humain ou un autre programme). On utilisera PHP pour transcrire les

varaiables HTML. Pourtant les motifs qu'on pourrait introduire sont divers à savoir Factory , Composite, TemplateView;

3. Le contrôleur : quant à lui serait l'analyste des requête du client, il a accès aux données. C'est lui qui formate le tout et expédie à la partie de présentation. L'on peut dire que c'est le composant important du modèle MVC.

Ce framework sera utilisé pour le développement de la solution proposée car il permet de fournir des briques logicielles et impose une rigueur et une méthode de travail pour obtenir une application fiable et facile à maintenir. Ce framework permet la conception de l'implémentation d'applications Web de taille importante. Les tâches peuvent être gérées par différentes personnes ce qui est nécessaire dans notre cas puisque plusieurs développeurs vont intervenir sur le projet.

2.1.5. Base de données

Une partie importante du développement sera consacrée à la base de données. La gestion des données se fera grâce à l'utilisation du système de gestion de base de données relationnelles gratuit MySQL [MYSQL].

SQL *(Structured Query Language)* en fait est un langage informatique normalisé servant à faire des opérations sur des bases de données. C'est une partie importante dans le *langage de manipulation de données* permettant de faire des recherches, des ajouts ou supprimer des données dans les bases de données.

Cette contrôle fait partie des tâches de l'administrateur de l'application. Il a la possibilité de modifier, de créer ou d'organiser la base des données. Il a également les compétences d'autoriser ou d'interdire l'accès de la base des données à certaines personnes.

En effet il s'agira entre autres d'établir le tableau des données proprement dit dans ce cas une manipulation réelle des données s'impose.

Dans la table des données, l'on essayera de structurer les données. Pour ce faire, il faut faire attention aux clefs primaires et à l'unicité. Il faudrait estimer les contraintes (contraintes de validation, contraintes d'intégrité référentielle,…). SQL est reconnu par tous les SGBD (Système de Gestion de Base des Données).

La base de SQL est d'établir une syntaxe claire et des instructions ne présentant pas des ambigüités.

2.2 Méthode de travail

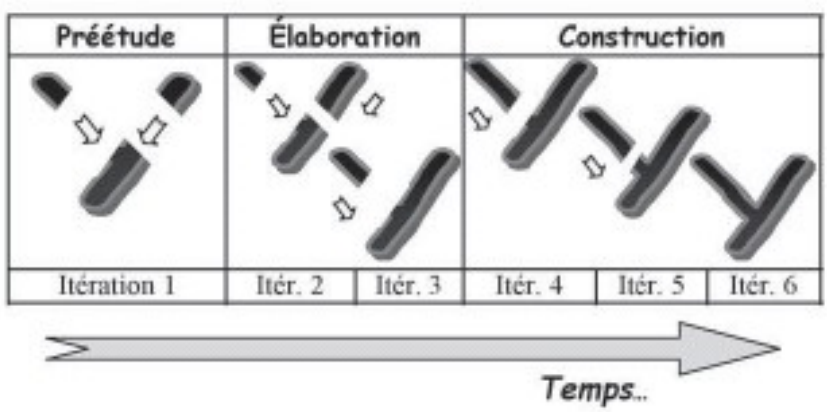
La méthode de travail à adopter se focalisera sur le processus de développement en Y [ROQUES & VALLEE]. Ici, il s'agira de mettre en œuvre et de définir réellement les besoins initiaux (fonctionnels et techniques) afin d'aboutir à la partie de conception générique jusqu'à la conception détaillée permettant de définir la structure logique de l'application.

Le langage de modélisation à utiliser ici est la modélisation unifié UML. UML se définit comme un langage de modélisation graphique et textuel destiné à comprendre et exposer des besoins, indiquer et éclaircir des systèmes, ébaucher des architectures logicielles, imaginer des solutions et communiquer des points de vue.

Par rapport au processus unifié, il s'agit plutôt d'un développement logiciel établit sur UML. Il est fréquent et incrémental, centré surtout sur l'architecture et dont la base est les cas d'utilisation. Pourtant, la gestion d'un tel processus est articulée d'après 4 phases à savoir : préétude *(inception)*, élaboration, construction et transition.

En adoptant le développement en Y, ROQUES & VALLEE a reproduit ces phases de gestion suivant la figure suivante :

Figure 7 : Plan d'itération suivant l'axe de gestion du projet



*Source : **[ROQUES & VALLEE]** Pascal Roques, Franck Vallée. UML 2 en action. Troisième édition. Paris : Eyrolles, 2004. ISBN : 2-212-11462-1, p. 17*

- L'itération 1 développe les fonctions d'acceptation du principe du système et intègre les instruments prévus pour le développement.
- L'itération 2 est concentrée sur l'architecture ; elle peut être examinée comme le prototype de réalisation technique.
- L'itération 3 avance dans l'exécution des fonctions les plus prioritaires de manière à montrer une première version de déploiement pour les utilisateurs. Elle permet également l'amélioration et le complément de l'architecture technique.

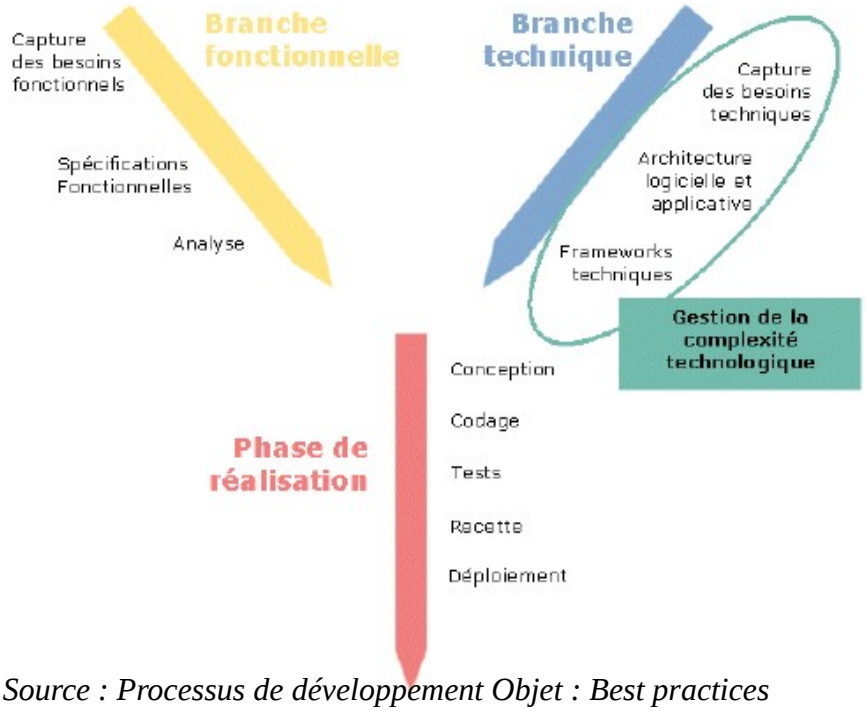
Le développement se fera en deux grandes phases. La première correspond au développement des couches métier, accès aux données et du sac de donnée. La seconde correspond au développement de la couche graphique (présentation) qui se fera en étroite collaboration avec les employés de la société XXX. Chaque étape fera l'objet d'une étude correspondant à la branche fonctionnelle du développement en Y jusqu'à la conception. Dans ce cas là, le développement en Y est notre principale méthode dans la réalisation de l'application.

La constitution d'une documentation technique à l'aide de commentaires (au format Doxygen [DOXYGEN]) dans le code se fera au fur et à mesure de l'avancement du projet.
Ici, il s'agit d'établir le manuel d'usage pour l'utilisation finale de l'application. Elle nous servira à former les utilisateurs du logiciel. C'est une partie importante du travail dans la mesure au fur et à mesure on aura à codifier les techniques et la documentation au format Doxygen.

2.3 Processus de développement en Y

Le cycle de développement en Y est représenté globalement par la figure suivante :

Figure 7 : Cycle de développement en Y



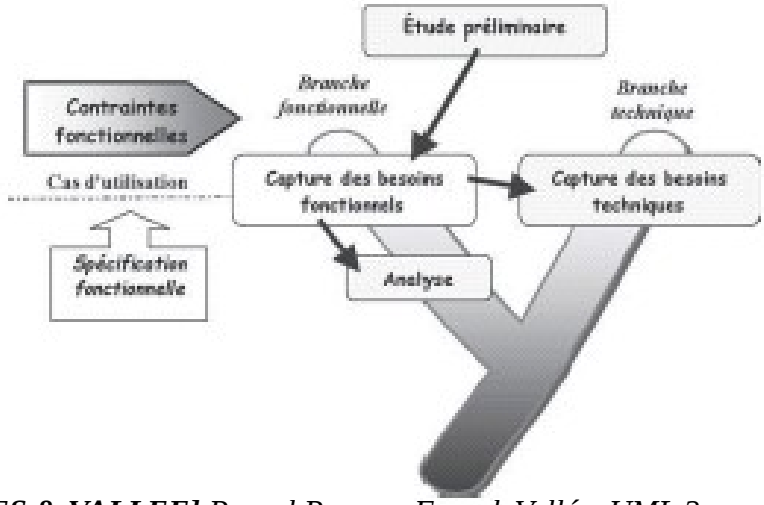
Source : Processus de développement Objet : Best practices

Nous essayerons alors d'étendre ce cycle de développement en essayant de se focaliser sur la branche fonctionnelle, la branche technique et la phase de réalisation.

2.3.1. Les notions dans la capture des besoins fonctionnels

La capture des besoins fonctionnels est la première étape de la branche gauche du cycle en Y. Cette branche forme l'étude préliminaire c'est-à-dire une ébauche déjà de l'application.
Transposer dans 2UTP, la capture des besoins fonctionnels se résume dans la figure suivante :

Figure 8 : Situation de la capture des besoins fonctionnels dans 2UTP



*Source : **[ROQUES & VALLEE]** Pascal Roques, Franck Vallée. UML 2 en action. Troisième édition. Paris : Eyrolles, 2004. ISBN : 2-212-11462-1, p.62*

a) Identification des cas d'utilisation

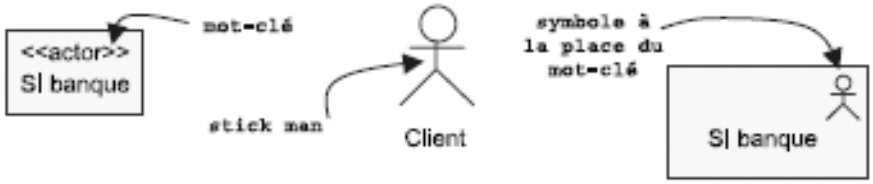
L'identification des cas d'utilisation met en exergue essentiellement l'identification des acteurs d'abord, puis les cas d'utilisation et le scénario.

On l'avait déjà définit précédemment, mais à titre de rappel un acteur indique un rôle joué par une entité externe (utilisateur humain, dispositif matériel ou autre système) qui interfère directement avec le système étudié.

Un acteur peut effectuer des changements ou consulter et/ou modifier directement l'état du système, en émettant et/ou en recevant des messages susceptibles d'être porteurs de données. On les identifie par rapport aux utilisateurs du système. On les représente souvent par une représentation graphique standard en UML.

Voici des représentations possibles d'un acteur.

Figure 9 : Exemple de représentations graphiques possible d'un acteur



*Source : **[ROQUES]** Pascal Roques. UML 2 par la pratique. Cinquième édition. Paris : Eyrolles, 2006. ISBN : 2-212-12014-1. P.16*

Dans notre étude, nous allons prendre le modèle *stick man* lequel va être présenté un peu plus tard dans le projet. Ce qui importe ici c'est de définir les acteurs potentiels dans l'entreprise. Mais afin d'avoir une meilleure appréhension des acteurs, la phase suivante est de décrire les cas d'utilisation.

b) Description des cas d'utilisation

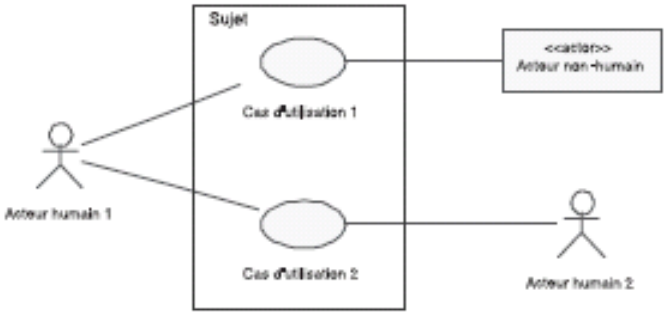
Un cas d'utilisation en anglais *« use case »* est l'ensemble des enchainements d'actions qui sont réalisées par le système et qui créent un résultat observable intéressant pour un acteur particulier. Les cas d'utilisation décrivent amplement les exigences fonctionnelles du système. Chaque acteur devrait avoir un cas d'utilisation néanmoins, un cas d'utilisation comporte un acteur principal et éventuellement un deuxième acteur secondaire ou plusieurs même.

Pour identifier les cas d'utilisation, le processus est le suivant : il convient de rechercher les différents desseins métier utilisés par le système mais également de voir dans le cahier des charges les attentes du système par rapport aux services fonctionnels.

Souvent, on représente les cas d'utilisation sur un schéma ovale relié par des associations appelées également lignes reliées à leurs acteurs (*stick man*).

En voici un exemple :

Figure 10 : Exemple de diagramme de cas d'utilisation



c) Organisation des cas d'utilisation

L'organisation des cas d'utilisation concerne les relations et l'utilisation des cas d'utilisation dans l'application. Dans ce cas, il faudrait détailler un peu plus les rôles de chaque acteur. Il faut voir alors si l'acteur est principal ou secondaire et établir également sa contribution dans le système d'information.

Organiser les cas d'utilisation ajouter les relations d'inclusion (formalisée par le mot clé « include »), d'extension « extend », et la généralisation entre les cas d'utilisation. On aboutit alors à un regroupement de packages pour pouvoir définir les blocs fonctionnels. De ce fait, on arrive à une structuration des cas d'utilisation.

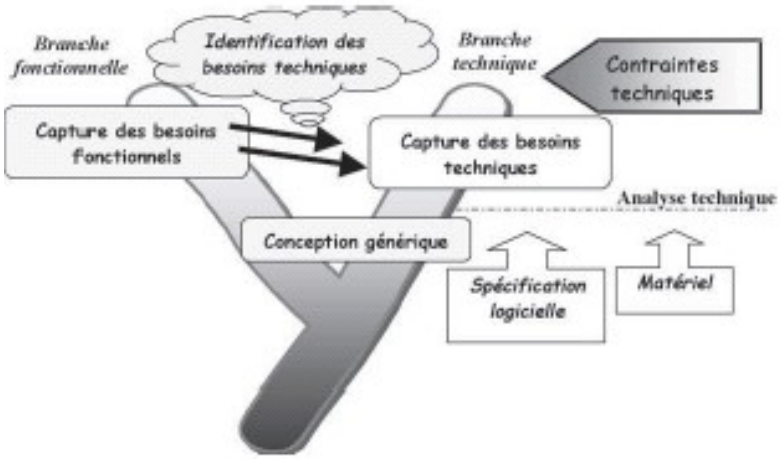
d) Identification des flux entre applications

La description des cas d'utilisation doit être une manière d'identifier les flux entre applications servant ainsi à la coordination entre les différentes applications participant au système. Nous l'avons déjà évoqué précédemment, il s'agit d'intégrer des applications comme une contrainte non fonctionnelle. Ceci justement dans le but de décrire les échanges entre applications. L'important c'est de faire une planification des applications. Il faudrait alors impliquer un peu plus la répartition des activités entre les activités. La phase modélisation métier est donc importante.

2.3.2. Les notions dans la capture des besoins techniques

Complémentaire aux besoins fonctionnels, les besoins techniques représentent les contraintes techniques du modèle de spécification du logiciel. Elle est également primordiale pour la conception architecture. C'est une activité de la branche droite du développement en Y. Il s'agit également de connaître les matériels à déployer (machines, réseaux, les progiciels à intégrer,etc.). Ceci nous permettra alors de déterminer les problèmes à résoudre par rapport aux cas d'utilisation technique. Comme dans l'autre branche, le développement de l'architecture technique est représenté par la figure suivante :

Figure 11 : Situation de la capture des besoins techniques dans 2UTP



*Source : **[ROQUES & VALLEE]** Pascal Roques, Franck Vallée. UML 2 en action. Troisième édition. Paris : Eyrolles, 2004. ISBN : 2-212-11462-1, p.94*

a) Spécification technique

La spécification technique a déjà été montrée un peu plus haut notamment en ce qui concerne le style d'architecture à utiliser. L'architecture la mieux adaptée pour notre projet est l'architecture n-tiers. Toutefois cela nécessite la configuration effective des machines conditionnée également par l'environnement du système.

Il importe alors de bien établir la spécification du logicielle par rapport aux besoins fonctionnels c'est-à-dire identifier les contraintes techniques à travers les besoins techniques pour aboutir à une capture des besoins techniques. La mission principale ici est de bien décrire les cas d'utilisation en l'occurrence les cas d'utilisation technique mais également l'organisation en couches logicielles.

Le plus important à retenir en ce qui concerne la spécification technique est de déterminer les prérequis techniques primordiaux dans la mise en œuvre du système d'information de la société.

b) Le modèle de déploiement

Parallèlement à la spécification technique, la spécification d'architecture joue un rôle d'influence par rapport au modèle de déploiement. En effet, l'établissement du diagramme de déploiement nous permettra de représenter l'architecture physique supportant l'exploitation du système. Il peut s'agir des nœuds qui peuvent être interconnectés formant ainsi un réseau

d'éléments physiques correspondant aux supports physiques (serveur, routeur,…). Ceci s'applique également sur le partage des artefacts logiciels.

Ici, ce qui nous importe c'est le système client/serveur faisant référence à au moins deux types de composants à savoir le client, le serveur et le base des données.

c) Le modèle de logicielle et développement des couches

Un modèle de spécification logicielle est souvent fondé en deux itérations.

Le modèle originel consiste à chiffrer les besoins des différents exploitants du système et à en détacher les cas d'utilisation techniques. Lors de la deuxième itération, le modèle de spécification est réaménagé en couches de responsabilités techniques de manière à affiner les exigences.

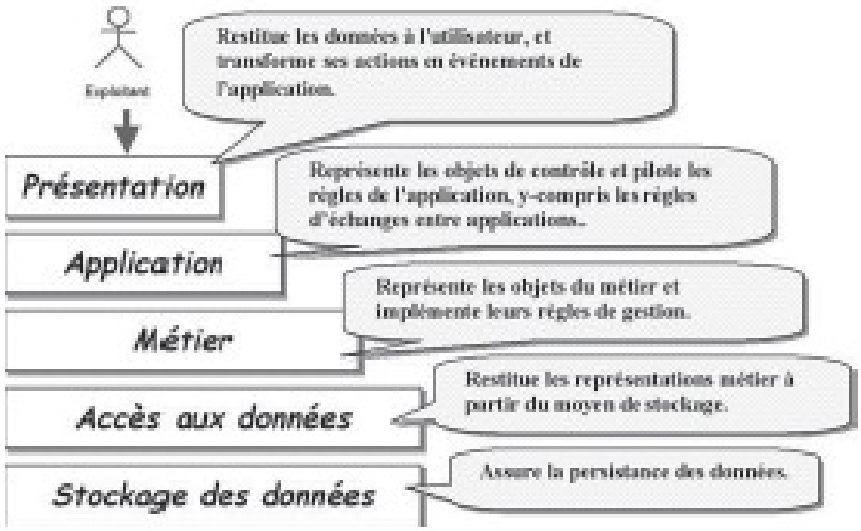
Il ne faudrait pas non plus écarter le concept de couche logicielle. Une couche logicielle montre un ensemble de spécifications ou de exécutions qui respectivement relatent ou mettent en œuvre un ensemble de responsabilités techniques et homogènes pour un système logiciel.

Les couches s'entassent en niveaux pour couvrir des modifications logicielles successives, de sorte que la couche d'un niveau ne puisse utiliser que les services des couches des niveaux inférieurs.

Pour le modèle UML, les couches logicielles sont représentées par les packages. En effet, les couches logicielles sont essentielles surtout pour affiner la spécification technique.

Voici un exemple de style d'architecture en 5 couches

Figure 12 : Style d'architecture en 5 couches



Source in **[Rumbaugh 91]** *Object-Oriented Modeling and Design, J. Rumbaugh, 1991, Prentice Hall* cité dans ***[ROQUES & VALLEE]*** *Pascal Roques, Franck Vallée. UML 2 en action. Troisième édition. Paris : Eyrolles, 2004. ISBN : 2-212-11462-1, p.104*

d) Définition des concepts techniques

Chaque étape de la conception de l'application est importante. La définition des concepts techniques est essentielle surtout dans un contexte où les termes techniques sont variables. Cependant, il faudrait établir un dictionnaire de termes techniques justement pour échapper à l'incohérence entre les terminologies et les termes techniques. Constituer une base de données directement à partir de données brutes serait la source de graves problèmes. Construire le dictionnaire des données, c'est déterminer d'une manière très rigoureuse les données élémentaires non calculées, exemptes de défauts, qui seront réellement utiles.

La méthode est la suivante :

- Établir la liste des données élémentaires : ce sont les données qui ne peuvent être décomposées. Ainsi, une même donnée peut être considérée comme élémentaire dans une application mais devra être décomposée en plusieurs données élémentaires dans une autre.

- Epurer les données : lorsque deux noms de données recouvrent la même réalité et lorsqu'un nom de données recouvre plusieurs réalités. Il s'agit là la source de graves confusions et doivent être absolument supprimé avant de constituer une base de données.

- Distinguer les données calculées des données non calculées : les données non calculées ne doivent pas être intégrées à la base de données à construire.

- Appliquer le principe de pertinence : les données brutes qui figurent sur les supports de données présentes dans le schéma de circulation de l'information ne sont pas nécessairement utiles eu égard aux finalités de l'application à développer.

- Dresser le dictionnaire des données : pour chacune des données élémentaires, calculées ou non calculées, exemptes de synonymies et de polysémies, et retenues pour leur pertinence, on précise le type et le domaine de définition. C : pour les données calculées et NC : pour les données non calculées. Le domaine de définition peut être défini soit en précisant la nature et la longueur des données soit par extension.

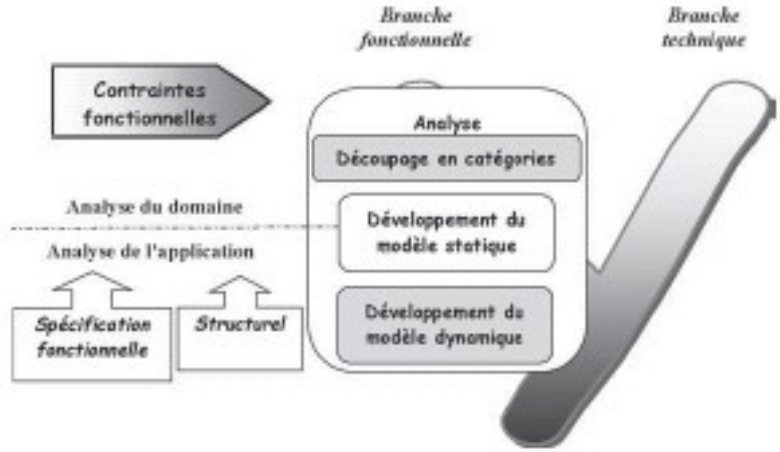
2.3.3. Développement du modèle statique

Après le découpage en catégories de l'analyse, le développement statique intervient. Elle se situe sur la branche gauche du cycle en Y.

Tout d'abord l'identification des classes lors de l'étude des cas d'utilisation nous permettra de faire l'affinement de ces classes c'est-à-dire que le diagramme de classes sera le point essentiel dans le développement orienté objet.

Voici la figure illustrant ce développement.

Figure 13 : Situation du développement du modèle statique dans le 2TUP



*Source : **[ROQUES & VALLEE]** Pascal Roques, Franck Vallée. UML 2 en action. Troisième édition. Paris : Eyrolles, 2004. ISBN : 2-212-11462-1, p.134*

Le processus de développement statique est la suivante :

- Identification des concepts du domaine et modéliser en tant que classes,
- Identification les associations pertinentes entre les concepts,
- Ajout des attributs aux classes du domaine,
- Compréhension de la différence entre modèles d'analyse et de conception,
- Utilisation des diagrammes d'objets pour illustrer les diagrammes de classes,
- Utilisation des classes d'association, contraintes et qualificatifs,
- Structuration du modèle en *packages*.

2.3.4. Développement du modèle dynamique

En troisième activité de l'étape de l'analyse laquelle se situe sur la branche gauche du cycle en Y.

En commençant par identifier les acteurs et les cas d'utilisation, l'on essayera de dessiner le diagramme de séquence « système » et après le dessin du diagramme de contexte dynamique serait approprié afin de répertorier les messages que les acteurs envoient et reçoivent au système. Dans ce modèle dynamique, nous mettrons plus l'accent sur le diagramme d'états.

Il serait opportun, dans ce cas de voir ce que c'est un état :

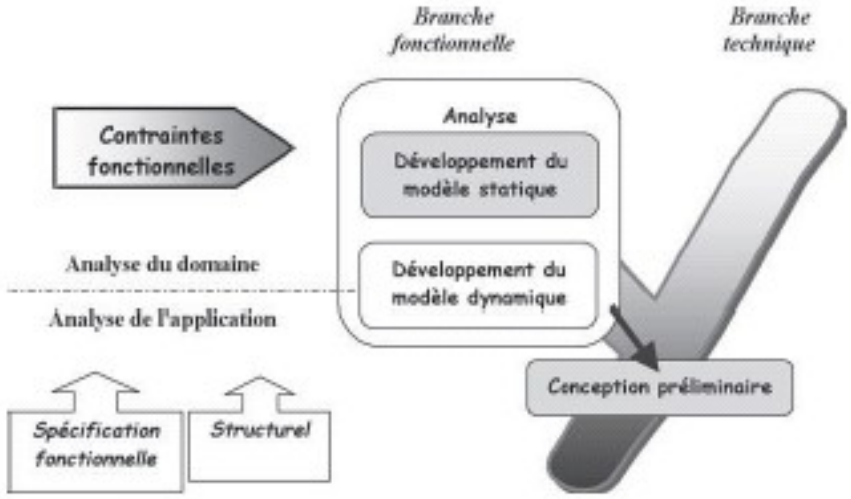
Un *état* représente une situation durant la vie d'un objet pendant laquelle :

- il satisfait une certaine condition ;
- il exécute une certaine activité ;
- ou bien il attend un certain événement.

L'on parle également d'état initial et d'état final représentant la dynamique de base du système d'informations.

Pour illustrer tout ca, voici la figure de la situation du développement du modèle dynamique

Figure 14: Situation du développement du modèle dynamique dans le 2TUP



*Source : **[ROQUES & VALLEE]** Pascal Roques, Franck Vallée. UML 2 en action. Troisième édition. Paris : Eyrolles, 2004. ISBN : 2-212-11462-1, p.166*

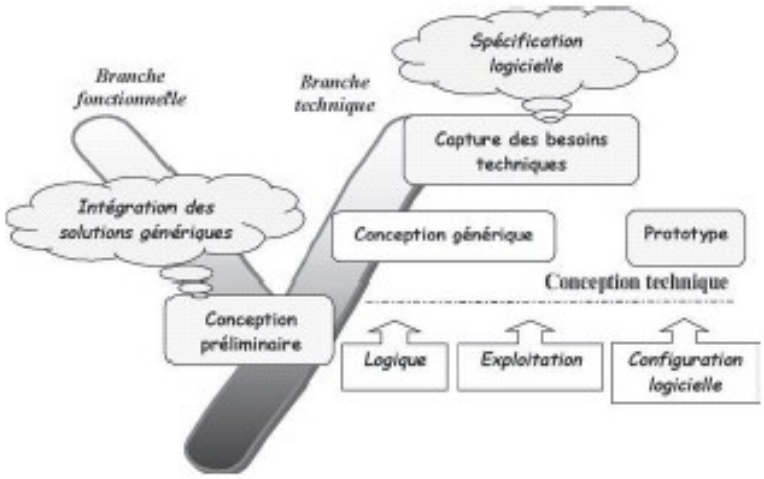
2.3.5. Conception générique

Dans cette partie, on va essayer de développer les solutions répondant aux spécifications techniques présentées précédemment.

Se situant dans la branche droite de Y, la conception est qualifiée de générique du fait de l'indépendance des aspects fonctionnels spécifiés dans la branche gauche.

La conception générique met en exergue le diagramme de classes, les *frameworks* utilisés, la réutilisation des composants techniques sans oublier le diagramme de composants voir les composants de configuration logicielle.

Figure 15: Situation de la conception générique dans le 2TUP



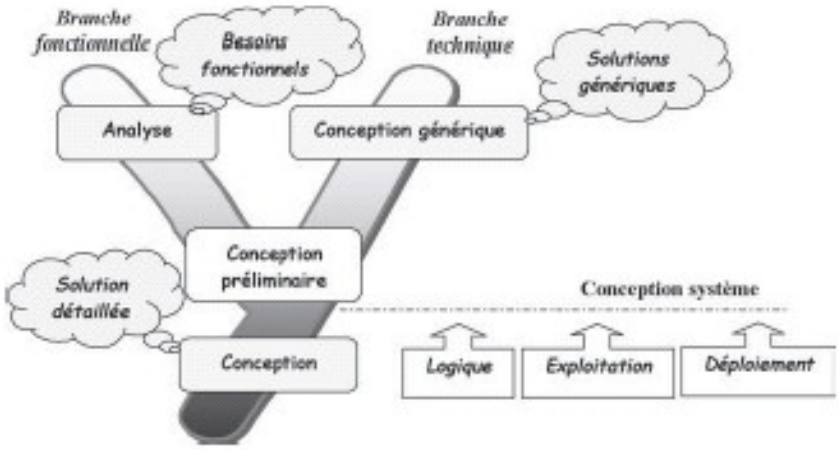
*Source : **[ROQUES & VALLEE]** Pascal Roques, Franck Vallée. UML 2 en action. Troisième édition. Paris : Eyrolles, 2004. ISBN : 2-212-11462-1, p.201*

2.3.6. Conception préliminaire

Dans l'étape du processus 2UTP, cette conception préliminaire est sans doute l'étape la plus sensible du fait de l'importance et le résultat que les études fonctionnelles et techniques produit.

Ceci est mis en exergue dans la figure suivante :

Figure 16 : Situation de la conception préliminaire dans le 2TUP



*Source : **[ROQUES & VALLEE]** Pascal Roques, Franck Vallée. UML 2 en action. Troisième édition. Paris : Eyrolles, 2004. ISBN : 2-212-11462-1, p.234*

En effet, c'est dans cette partie que la fusion entre les études fonctionnelles et les études techniques s'effectue. Il convient alors de signaler que cette conception devrait passer par plusieurs activités :

- passer de l'analyse objet à la conception,
- intégrer les fonctions métier et applicatives du système dans l'architecture technique,
- adapter la conception générique aux spécifications fournies par l'analyse.

2.3.7. Conception détaillée

Arriver à la dernière phase et non le moindre de la modélisation avec UML, nous tenterons à partir de la modélisation des besoins de construire les documents, les interfaces, les tables lesquels forment le codage et la solution.

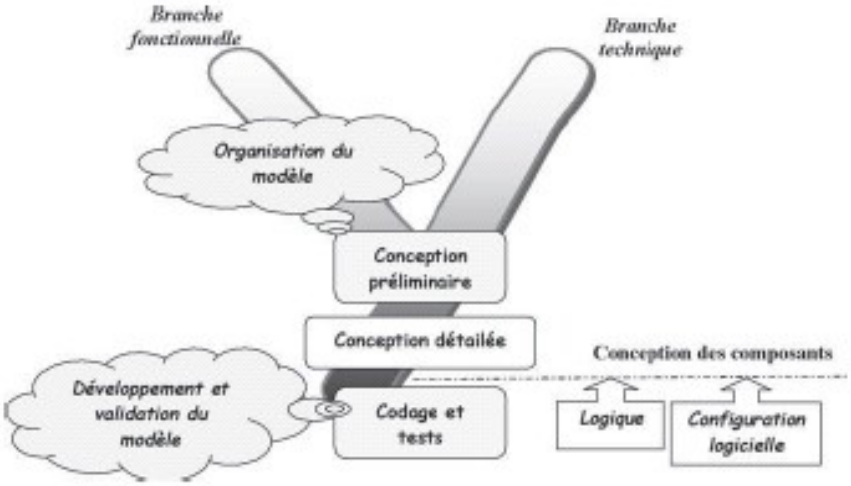
Le modèle logique de la conception détaillée s'inscrit dans l'organisation définie par la conception préliminaire. C'est dans cette étape qu'on génère un important volume d'informations. Les *frameworks* techniques et les regroupements au métier y jouent alors un rôle essentiel.

C'est dans cette partie de l'analyse qu'on construit les classes, les vues d'IHM, les interfaces, les tables et les méthodes codés puis complémenteront la configuration du logiciel.

Tout ceci sera effectivement réalisé avant la phase de codage.

Nous le résumerons dans la figure suivante :

Figure 17 : Situation de la conception détaillée dans le 2TUP

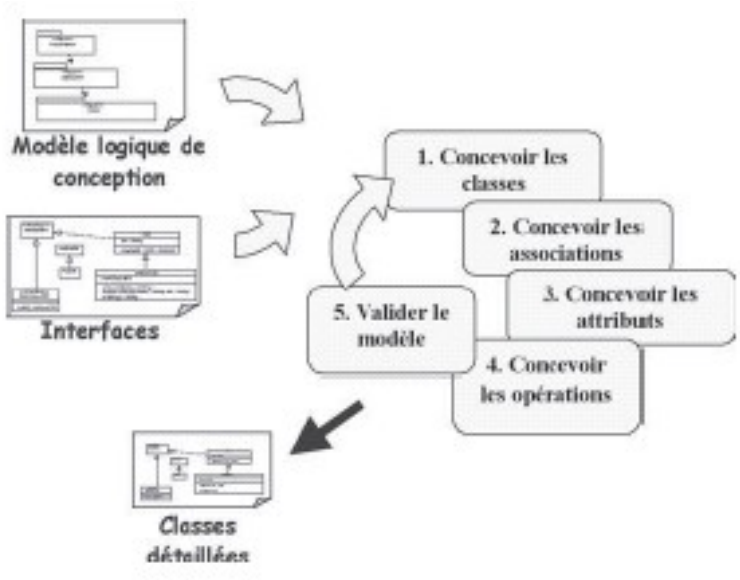


*Source : **[ROQUES & VALLEE]** Pascal Roques, Franck Vallée. UML 2 en action. Troisième édition. Paris : Eyrolles, 2004. ISBN : 2-212-11462-1, p.270*

A cette étape, il faut le dire, toutes les questions relatives à la conception sont déjà résolues. Mais l'essentiel est de faire la conception logique du micro-processus c'est-à-dire le modèle logique. Il s'agit de combiner les diagrammes déjà réalisés précédemment (diagramme de classes, diagramme d'interactions, diagramme d'activités).

La figure suivante nous montre ce micro-processus :

Figure 18 : Micro-processus de conception détaillée



*Source : **[ROQUES & VALLEE]** Pascal Roques, Franck Vallée. UML 2 en action. Troisième édition. Paris : Eyrolles, 2004. ISBN : 2-212-11462-1, p.271*

3. Conduite de projet

3.1 Analyse de l'existant

Dans cette partie, nous essayerons de développer en long et en large l'existant dans l'entreprise XXX. Pour cela, il va falloir établir le flux d'informations réel dans la société. Comme nous l'avons déjà mentionné précédemment, la société XXX souhaite avoir une meilleure maîtrise de ses activités de production.

Le résultat obtenu doit donc répondre au besoin de gestion de stock des produits dont le statut peut évoluer à tout moment. Ainsi donc, les principales fonctionnalités du logiciel sont :
- La gestion commerciale ;
- La gestion de la comptabilité ;
- La gestion de la distribution ;
- La gestion de l'approvisionnement ;
- La gestion du stock.

La circulation des flux d'information dans la société se fait à partir du schéma suivant :
La Figure ci-dessous montre les flux de circulation globaux dans la société XXX notamment ses relations avec les acteurs externes (client et fournisseur) et les relations entre les différents services et leur attribution respective. Etant donné que chaque service dans la société XXX a des rôles bien déterminés, nous allons, cependant, identifier les rôles de chacun c'est-à-dire les tâches qu'ils ont à entreprendre pour la bonne marche de l'entreprise.

Acteur 1 : Le Service Commercial

Tout d'abord le service commercial est en relation directe avec les clients. Déjà il établit les prévisions de ventes pour les produits et élabore les stratégies pour l'atteinte des objectifs de l'entreprise. Il est en charge également des suivis de stock et de l'évolution des stocks disponibles. Le service marketing fait partie également de ce service lequel est chargé de conduire des études de marché pour l'élaboration des stratégies à adopter dans la vente. Ce qui fait donc, que le service commercial se charge de la mise en application de ces stratégies de vente. En d'autres termes, le commercial conçoit le fichier clients de la société XXX. L'on pourrait ainsi partager les tâches du commercial comme suit : constituer un fichier client (développement, gestion et qualification du client), établir les devis jusqu'à la négociation et convaincre le client pour aboutir à la commande. Il incombe au commercial alors de faire les bons de commande, les devis.

Acteur 2 : Le Service de la comptabilité

Le comptable est en charge des travaux de gestion des comptes de la société XXX. Le rôle premier de la comptabilité est d'enregistrer l'ensemble des flux important impliquant l'entreprise. Par exemple : le paiement des salaires des salariés à la fin du mois, l'achat des matières premières aux fournisseurs. Mais ici, les travaux journaliers du comptable consistent plutôt à l'enregistrement courant des opérations (facture, relance, …). Toutefois, c'est le service comptabilité qui prend en charge de vérifier l'état du stock c'est-à-dire la disponibilité des produits pour pouvoir ainsi en faire part au service commercial quant aux produits à vendre.

Acteur 3 : Le Service de la distribution

Il s’en charge de la livraison des produits auprès des clients. Comme le service commercial, il est en contact direct avec les clients. Il peut être un prestataire de service ou fait partie de la société elle-même.

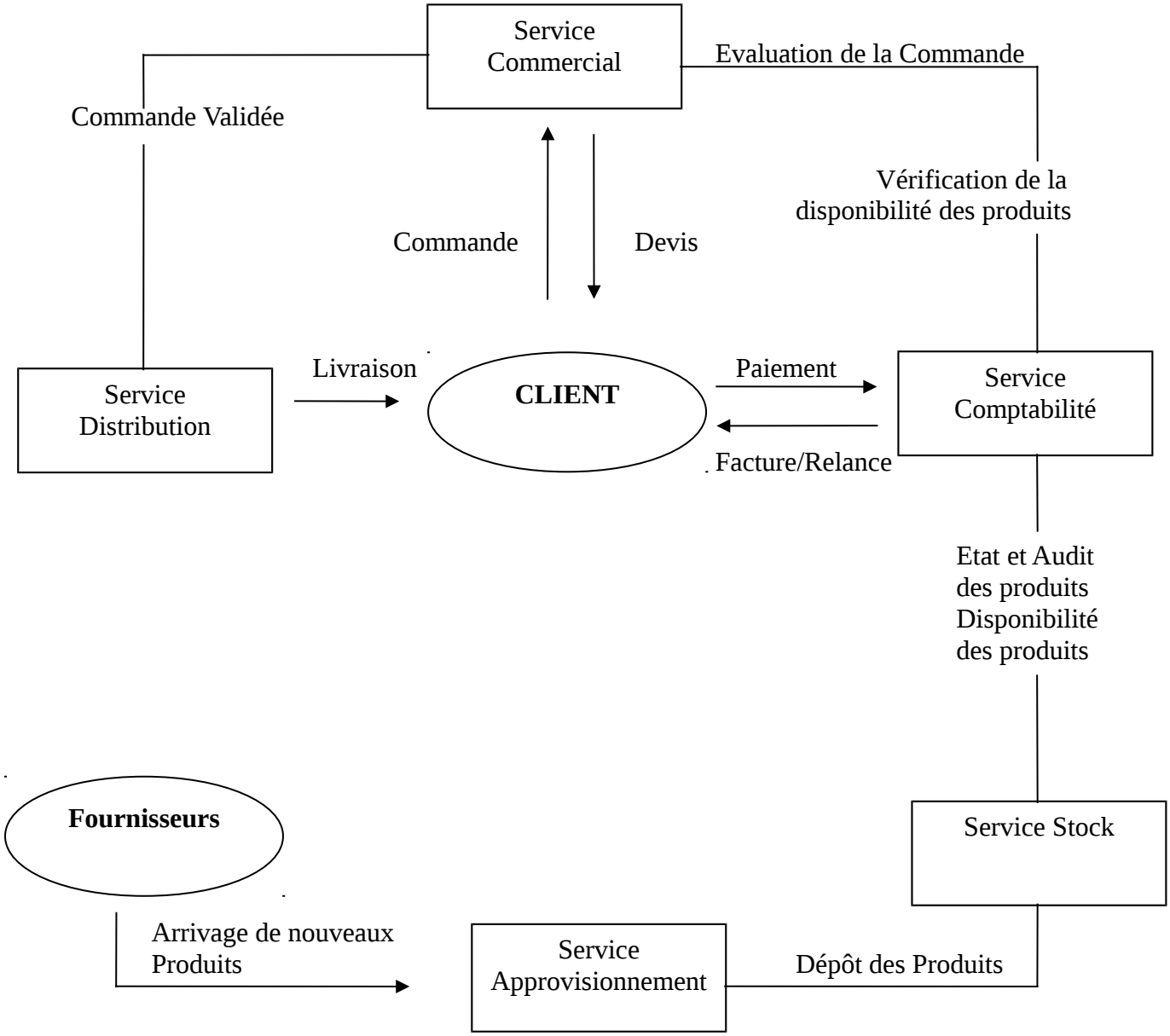
Acteur 4 : Le Service de l’Approvisionnement

L’approvisionnement consiste à fournir les matières premières c'est-à-dire la gestion des relations avec les fournisseurs qui assurent les produits et les services nécessaires pour l’entreprise. Dans ce cas, le service de l’approvisionnement doit tenir compte de quelques critères : la quantité, la qualité, le délai, le coût et la source des produits. Ce service est en relation directe avec les fournisseurs. Néanmoins, le service d’approvisionnement est en étroite collaboration avec le service stock (l’entrepôt). A ce titre, c’est le service approvisionnement qui fournit les factures auprès des fournisseurs.

Acteur 5 : Le Service de Stock
Ce service assure la disponibilité des stocks dans l’entrepôt. Il évalue les besoins en stock c'est-à-dire s’occupe de l’inventaire et de la valorisation des stocks. Il veille à l’état des produits, assure l’audit des produits et fait part de la traçabilité des produits au niveau des autres services. C’est également ce service qui garantit le suivi des commandes auprès du service de distribution en outre la réception et la vérification des marchandises (établissement des bons d’entrée des produits).

Toutefois une grande partie de ces actions ne sont pas encore informatisées. Ainsi, notre objectif est de développer un logiciel capable de répondre aux besoins de la société XXX.

Figure 19 : Circulation des flux d'information dans la société XXX



3.2 Spécification des besoins

3.2.1. Modélisation fonctionnelle

La modélisation fonctionnelle consiste à la réalisation des cas d'utilisation, c'est-à-dire essayé de faire une première approche de ce que peut être les fonctionnalités par rapport à son utilisation et les mouvements des acteurs. Il s'agit ici d'avoir une vue fonctionnelle du modèle à concevoir.

3.2.2. Modélisation dynamique

a) Identification systématique des acteurs et des cas d'utilisation

L'identification des acteurs et des cas d'utilisation a permis d'avoir une première vue des acteurs principaux et des acteurs secondaires dans la société XXX. Voici un tableau illustrant tout cela.

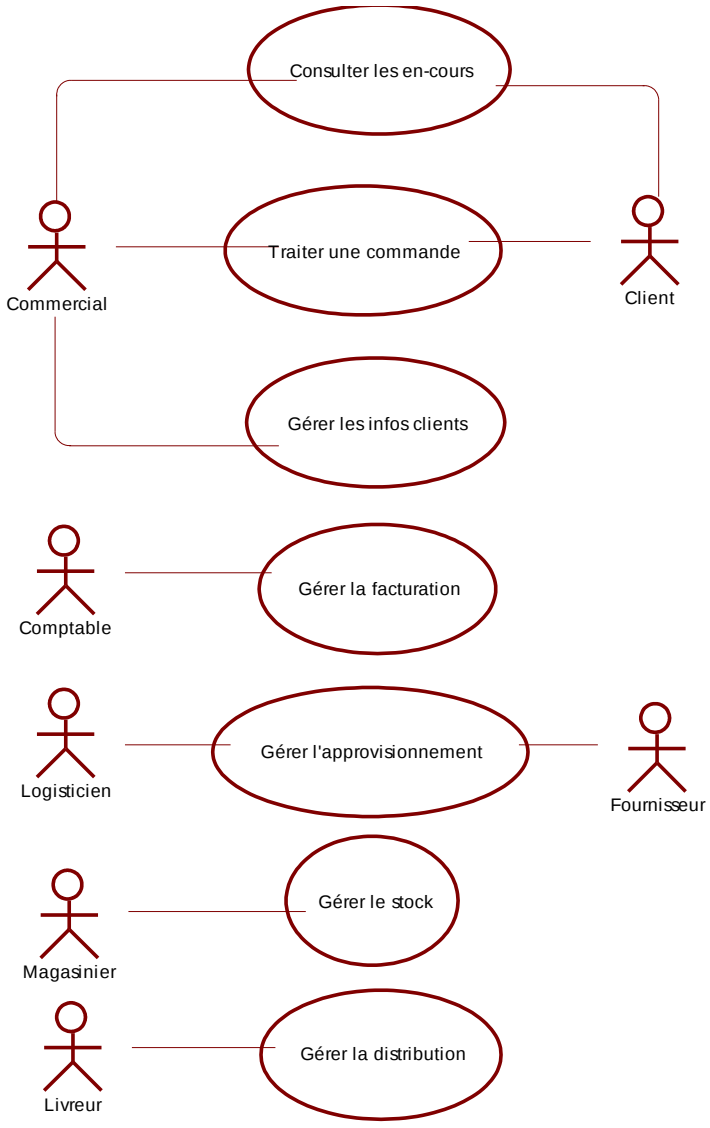
Tableau 1 : Liste des Acteurs et messages par cas d'utilisation

Cas d'utilisation	**Acteurs principales, Acteurs Secondaires**	**Messages émis/reçus par les acteurs**
Traiter une commande	Commercial	émet: création, modification, annulation commande
		reçoit: conditions commande
	Client	reçoit: confirmation commande
Gérer les infos clients	Commercial	émet: création client
Consulter les en-cours	Commercial	reçoit: en-cours
	Client	reçoit: en-cours
Gérer la facturation	Comptable	reçoit: facture
		reçoit: paiement
		émet: relance
		émet: règlement achat
Gérer l'approvisionnement	Logisticien	émet: commande des produits
		émet: dépôt des produits
		reçoit: état de la disponibilité des produits
	Fournisseur	émet: réponse à la demande des produits
		reçoit: paiement des achats
Gérer le stock	Magasinier	émet: état des produits
		émet: audit des produits
		reçoit: dépôt des produits
Gérer la distribution	Livreur	reçoit: commande validée
		émis: livraison commande

b) Diagramme des cas d'utilisation

Après avoir identifié les cas d'utilisation et ses acteurs, nous pouvons alors produire les diagrammes des cas d'utilisation de chacun de ces acteurs.

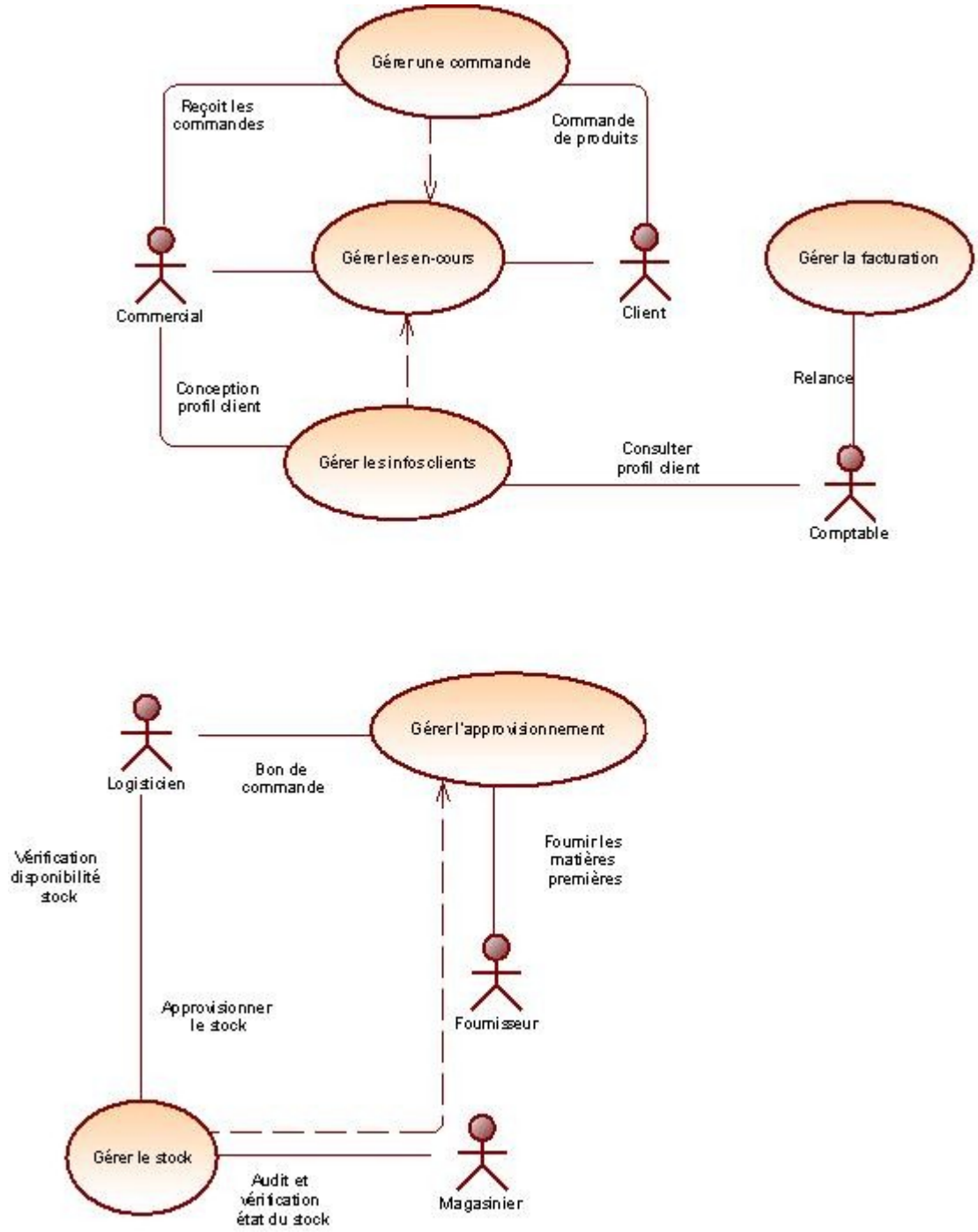
Figure 20 : Exemple de Diagramme de cas d'utilisation de la société XXX



Dans un premier temps, nous avons identifié en gros, les cas d'utilisation contenant la totalité du déroulement normal du système. Mais nous pouvons quand même avoir une version améliorée notamment en mettant en exergue les différents acteurs et les interactions existantes entre eux.

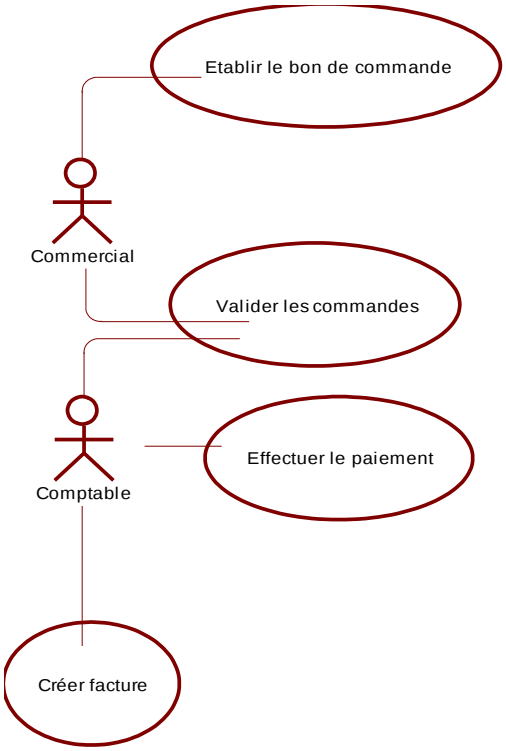
Toutefois, nous pouvons encore une version encore plus détaillée de ces cas d'utilisation, notamment en essayant d'approfondir chacun des attributions des acteurs. De ce fait, nous allons réaliser des diagrammes de cas d'utilisation pour chaque type de cas d'utilisation. Il faut le souligner, il s'agit pour la société XXX d'améliorer sa gestion de stock ce qui fait que notre attention se focalisera sur la gestion de stock néanmoins sans faire abstraction des tâches des autres acteurs.

Figure 21 : Diagramme de cas d'utilisation entre les différents acteurs



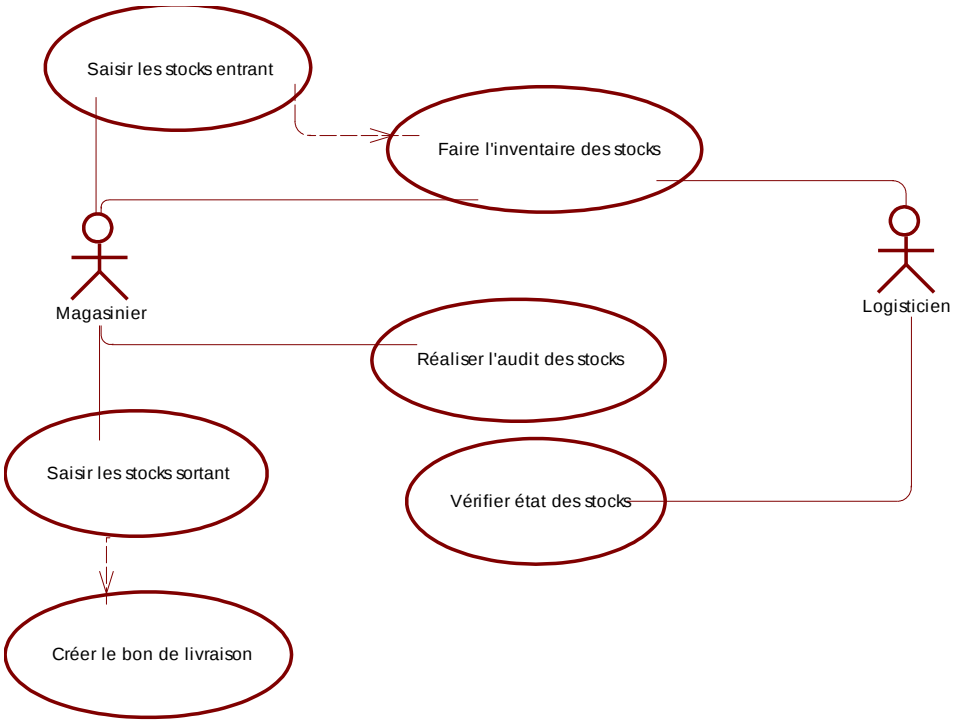
Etant donné que ce qui nous intéresse le plus c'est la gestion de stock, nous essayerons de centrer nos cas d'utilisation sur la gestion de stock. Nous insistons alors sur la gestion de stock et les flux qui y circulent. Rappelons que la principale difficulté consiste à gérer le stock des produits dont le statut peut évoluer à tout moment. L'objectif pour la société XXX est la traçabilité des produits. Certes les acteurs internes et externes participent également à la gestion de stock. Voici une illustration de tout cela :

Figure 22 : Diagramme de cas d'utilisation gestion de stock entre le comptable et le commercial



Cette Figure montre partiellement les mouvements du service commercial et du comptable notamment par rapport à la gestion du stock. Nous allons voir plus tard une description un peu plus détaillée de ces actions.

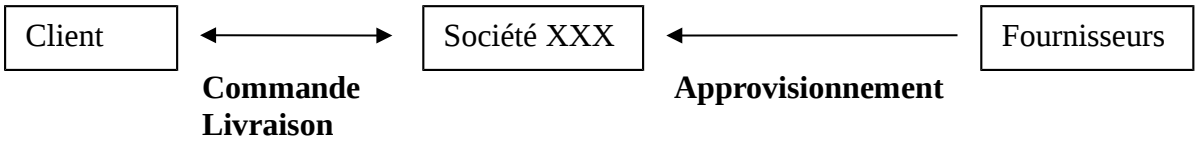
Figure 23 : Diagramme de cas d'utilisation au niveau de la gestion de stock entre les principaux acteurs



La Figure ci-dessus est une illustration simpliste des mouvements et des flux dans la gestion de stock particulièrement en ce qui concerne la traçabilité des stocks et l'évolution des statuts des stocks. La description textuelle de ces cas d'utilisation nous montrera davantage les flux réels dans la gestion des stocks.

c) Description textuelle des cas d'utilisation

Nous avons pu voir plusieurs cas d'utilisation susceptible de répondre à nos besoins, cependant nous n'en retiendrons que quelques uns. Voici un schéma sommaire de ce que pourrait le mouvement global dans la société XXX.



Ici nous allons surtout faire une description des cas d'utilisation prioritaires et utiles. Nous n'en retiendrons que : saisir devis, valider commande, créer bon de commande, modifier commande, créer facture, gérer le traçage et suivi des marchandises.

Sommaire d'identification

Titre 1 : Saisir devis
Résumé
Après que le client a pu consulter les articles et les marchandises disponibles, il effectue sa commande auprès du commercial. Ce dernier doit faire un devis pour le client.
Acteurs
Commercial : acteur principal
Précondition
- Client ayant déjà pris connaissance des procédés pour l'achat des articles
- Disponibilité de l'article

Scénario nominal
- Le client consulte les articles en vente via le web
- Grâce au contact inscrit dans le site, il établit le contact avec le commercial
- Le commercial vérifie la disponibilité de l'article
- Le commercial créée le devis

Titre 2 : Valider commande
Résumé
La validation de la commande ne se fait qu'après avoir reçu l'aval du client. Le commercial peut alors valider la commande et passe directement à l'étape suivante.
Acteurs
Commercial : acteur principal
Précondition
Devis accepté par le client
Scénario nominal
- Le client contact le commercial pour valider la commande
- Le commercial entame maintenant les procédures de vente

Titre 3 : Créer bon de commande

Résumé

La commande validée, le commercial crée maintenant le bon de commande à soumettre auprès du comptable et le service de stock.

Acteurs

Commercial : acteur principal

Comptable et magasinier : acteurs secondaires

Précondition

Commande validée

Scénario nominal

Prendre les informations nécessaires du client pour le bon de commande

Titre 4 : Modifier commande

Résumé

Il se peut que la commande du client change soudainement si le client trouve d'autres articles intéressants. Dans ce cas, la difficulté réside dans l'ajout et la suppression des articles. Il se peut également que le client annule carrément la commande.

Acteurs

Commercial : acteur principal

Précondition

- Le client désire annuler sa commande
- Le client veut ajouter d'autres articles à sa commande
- Le client veut annuler la commande de certains articles

Scénario nominal

- Le client contact le commercial pour que ce dernier modifie la commande
- Le client doit refaire un autre bon de commande

Titre 5 : Créer facture

Résumé

Pour le paiement de la marchandise, le comptable doit fournir au client une facture justement pour vérifier et permettre, éventuellement, la relance du client.

Acteurs

Comptable : acteur principal

Précondition

Le paiement se fait à partir de la connaissance de la facture

Scénario nominal

- Le bon de commande servira de pièce pour la création de la facture
- Le client effectue le paiement à partir de la facture qu'on lui fourni

Titre 6 : Gérer les infos clients

Résumé

Un client commande un produit, mais avant le commercial doit créer le profil du client. La finalité de cette action est l'entrée des infos du client dans le système. Toutefois, le comptable peut également consulter les informations du client, justement pour le suivi des paiements et éventuellement les relances auprès du client.

Acteurs

Commercial : acteur principal

Comptable : acteur secondaire

Précondition

Le client a déjà établi une commande auprès de la société XXX.

Scénario nominal
- Le commercial saisi les informations du client
- Les informations introduites doivent être complètes
- Le commercial actualise les informations du client, des mouvements du client par rapport à la commande et le paiement
- Le comptable vérifie également la situation du client quant au paiement des factures

Titre 7 : Gérer le stock

Résumé

La vente et la commande auprès des fournisseurs dépend de la disponibilité des stocks et de l'état des stocks dans le dépôt. Ainsi, l'objectif est d'avoir un statut actualisé de l'état du stock pour la vente des produits et l'achat des matières premières auprès des fournisseurs.

Acteurs

Magasinier : acteur principal

Logisticien : acteur secondaire

Précondition

L'évolution des stocks devrait être disponible pour tous les autres acteurs de production de la société XXX

Scénario nominal
- Le magasinier saisi les stocks entrant
- Il vérifie la quantité et la qualité des marchandises entrant dans le dépôt
- De ce fait, le magasinier prend en charge la vérification et l'audit des marchandises à vendre
- Le magasinier saisi les stocks sortant destinés à la vente

Titre 8 : Gérer le traçage et le suivi des marchandises

Résumé

Etant donné que le statut et l'état des articles dans l'entrepôt sont en constante évolution, la connaissance en temps réel du statut de ces marchandises serait indispensable dans la mesure où c'est l'objectif même du logiciel. Ainsi, la responsabilité revient à la gestion du stock et de l'entrepôt même justement pour permettre aux autres acteurs d'avoir une connaissance temporelle des variations du stock.

Acteurs

Magasinier : acteur principal

Logisticien/Comptabilité/Commercial : acteurs secondaires

Précondition

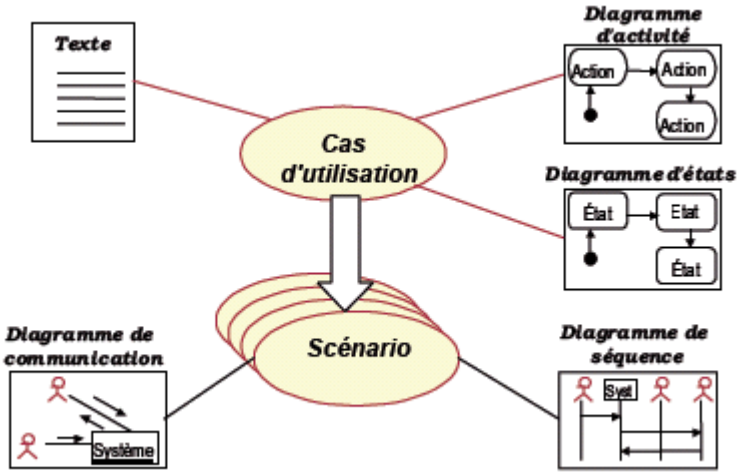
L'état réel du stock devrait être disponible sur le système

Scénario nominal
- Les stocks arrivés dans l'entrepôt doivent subir les contrôles et les audits du magasinier
- Le magasinier doit faire un rapport auprès du service approvisionnement de l'état du stock
- Il appartient au logisticien de faire un bon de commande auprès de la comptabilité pour l'achat d'articles
- Les marchandises destinées à la vente ne doivent pas sortir de l'entrepôt à condition que les bons de commande sont en bonne et du forme
- Le magasinier ne doit pas faire sortir des marchandises sans l'aval du comptable ou du commercial

d) Description graphique des cas d'utilisation

La description textuelle est importante dans une modélisation UML. Pourtant, elle doit être complétée par d'autres diagrammes dynamiques entre autre on peut utiliser un diagramme.

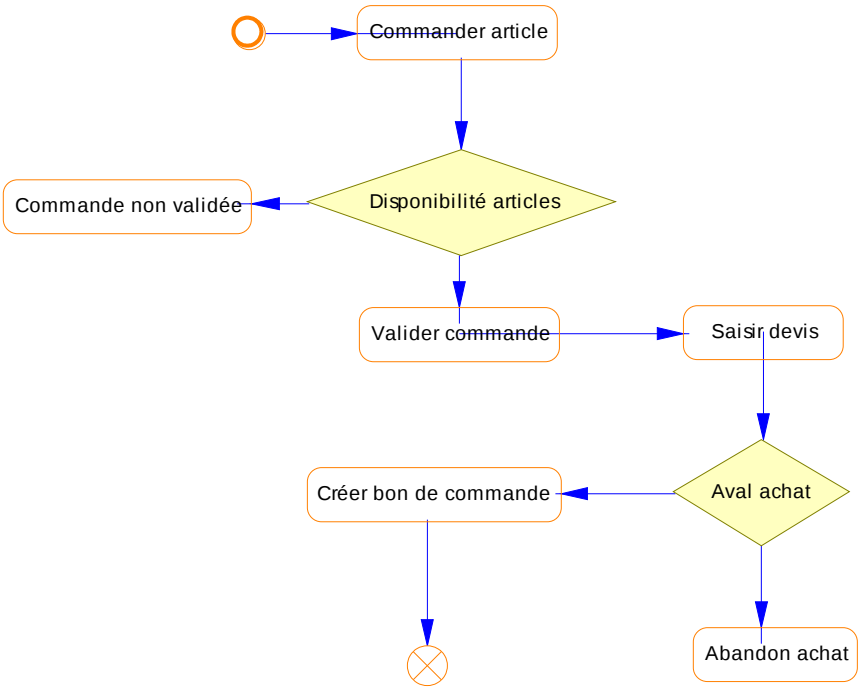
Figure 24 : Types de diagrammes dynamiques utilisables pour documenter les cas d'utilisation



*Source : **[ROQUES & VALLEE]** Pascal Roques, Franck Vallée. UML 2 en action. Troisième édition. Paris : Eyrolles, 2004. ISBN : 2-212-11462-1, p.75*

Dans notre cas, le diagramme d'activités est recommandé. Le diagramme d'activités nous permettra de consolider les enchaînements de la fiche textuelle précédemment établie.

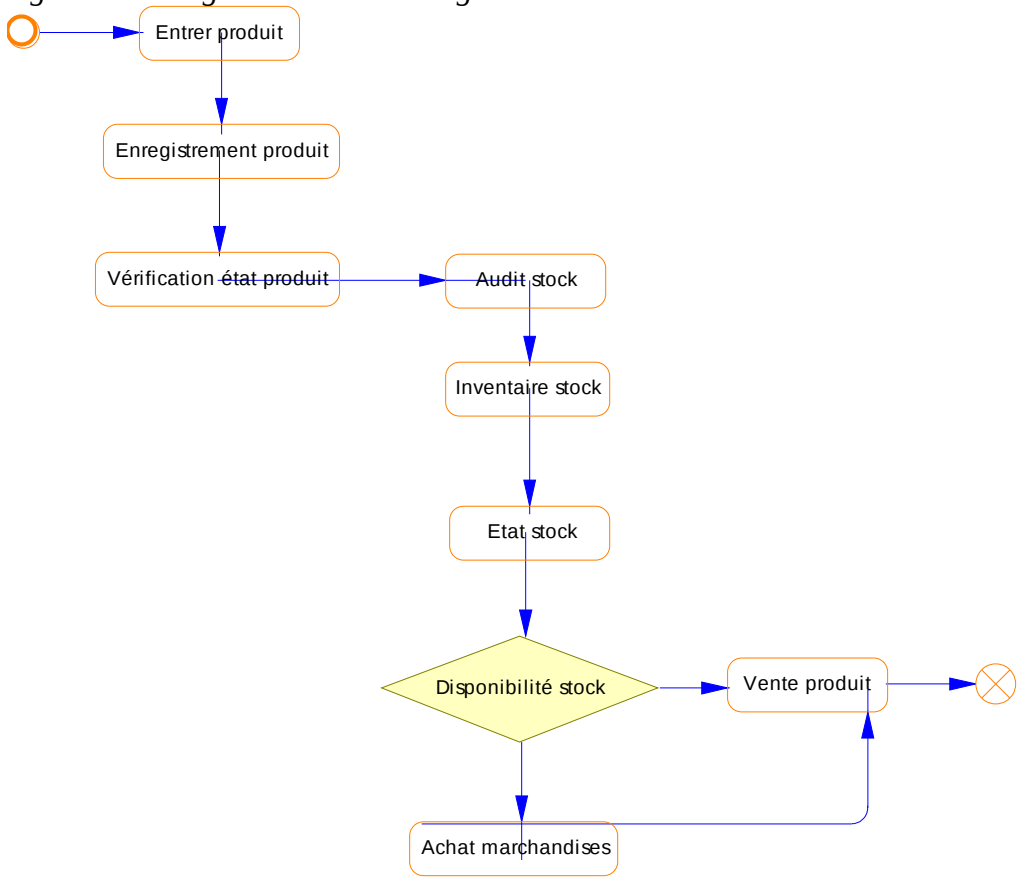
Figure 25 : Diagramme d'activités de créer bon de commande



Le diagramme des activités ci-dessus montre les actions sans ambigüité et pour identifier d'un seul coup d'œil les scénarios transcrits précédemment. Ce diagramme décrit le système et les erreurs éventuels.

Le diagramme suivant des activités suivant nous montrera davantage les scénarios quant à la gestion de stock et la traçabilité des produits.

Figure 26 : Diagramme d'activités gestion de stock



e) Réalisation du diagramme de séquence système

Dans la modélisation UML, il est fortement recommandé de représenté les cas particuliers à partir du *diagramme de séquence* système.

Le diagramme de séquence décrit la dynamique d'un système, en détaillant de façon temporelle ses fonctions et ses sous-fonctions. Il décrit les interactions entre objets de façon séquentielle (d'où son nom) ainsi que les messages et les flux de données échangés.

Les messages

Les objets communiquent entre eux par le biais de messages. Ce message représente une méthode (fonction) que l'objet destinataire doit exécuter à sa réception. Les envois de messages sont représentés par des flèches horizontales entre deux objets (ou plutôt d'une ligne de vie à une autre).

La ligne de vie

Afin de détailler les fonctions et sous fonctions dans le temps chaque instance (objet) comporte une ligne de vie qui informe quand l'objet est actif ou pas, ainsi que ses interactions avec d'autres objets du système.

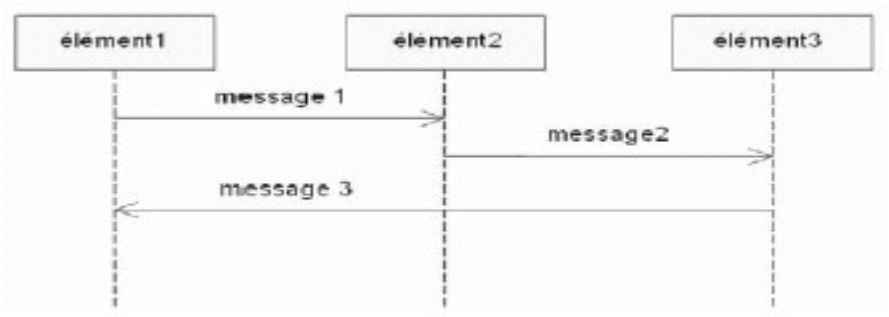
Autrement dit, Un *diagramme de séquence* est un diagramme UML qui fournit une représentation graphique de la technologie d'échange de messages entre des objets et des acteurs pour un cas d'utilisation, l'exécution d'une opération, ou une interaction des classes, en mettant l'accent sur leur chronologie.

Un diagramme de séquence véhicule le même genre d'informations qu'un diagramme de communication, à ceci près qu'il se concentre sur la chronologie des messages échangés entre les objets plutôt que sur la structure des objets.

L'un des principaux avantages d'un diagramme de séquence sur un diagramme de communication est qu'il permet de référencer des interactions courantes et de spécifier facilement des scénarios alternatifs ou parallèles en utilisant des fragments d'interaction. Ainsi, vous pouvez décrire dans un seul diagramme de séquence un nombre d'interactions liées qui nécessiterait plusieurs diagrammes de communication.

Les bases du diagramme de séquence système sont représentées dans la Figure suivante :

Figure 27 : Bases du diagramme de séquence UML2

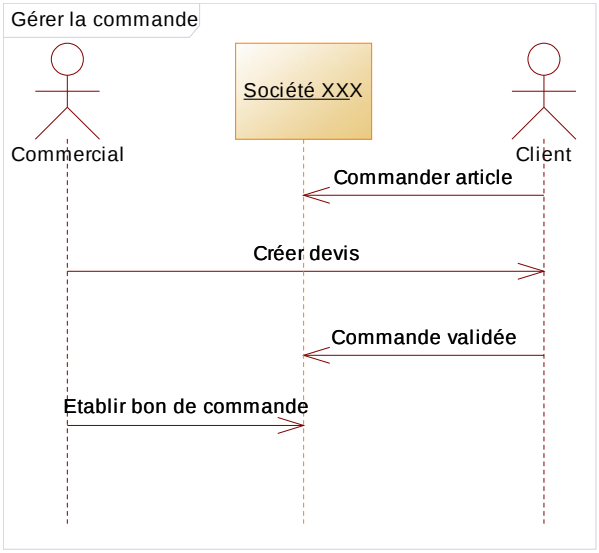


Nous allons réaliser le diagramme de séquence système par rapport au scénario nominal de gestion de stock. Nous présentons l'acteur principal à gauche, puis un objet unique représentant le système et éventuellement les acteurs secondaires à droite.

Ici, nous allons transcrire sous forme de diagramme de séquence les interactions citées dans les scénarios textuels:

- L'acteur principal *agent (magasinier, commercial, comptable ou logisticien)* à gauche ;
- Un participant représentant le système ;
- L'acteur secondaire à droite.

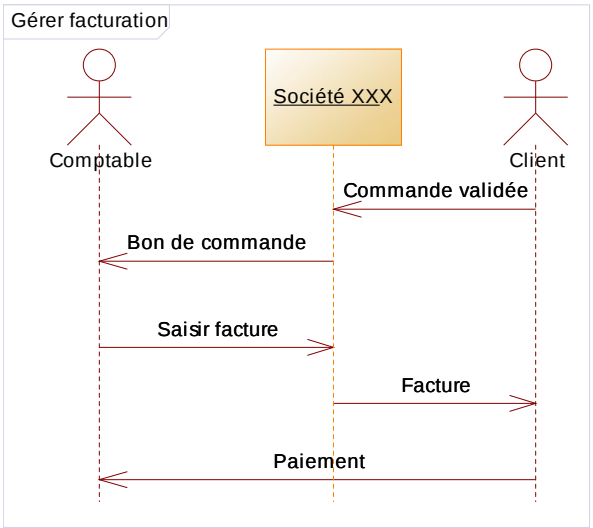
Figure 28 : Diagramme de séquence gérer la commande



Le diagramme réalisé ci-dessus est plutôt simpliste. Il s'agit juste à étaler le rôle du commercial vis-à-vis de la société XXX et du client.

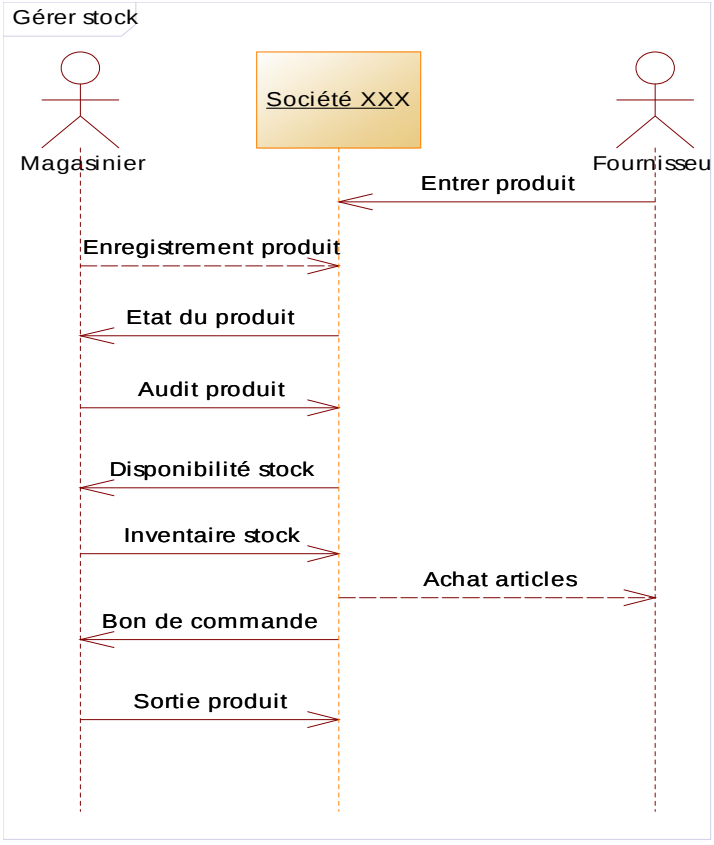
On a mis en évidence l'interaction entre le commercial et le client. La séquence est la suivante : le client commande un article par l'intermédiaire de la société XXX (représentée par le commercial), le commercial crée un devis à fournir auprès du client. C'est après que le client pourrait prendre sa décision d'acheter ou pas le produit. Et comme nous l'avons déjà démontré, la validation de la commande est le début des interactions et des mouvements de flux dans la société. Elle enclenche le processus de production au niveau de chaque service.

Figure 29 : Diagramme de séquence gérer la facturation



Dans le cas de la gestion de la facturation, l'interaction se passe entre le comptable et le client. Après la validation de la commande, avant la livraison le client doit effectuer le paiement pour recevoir la commande. Alors, le comptable saisie la facture pour le client.

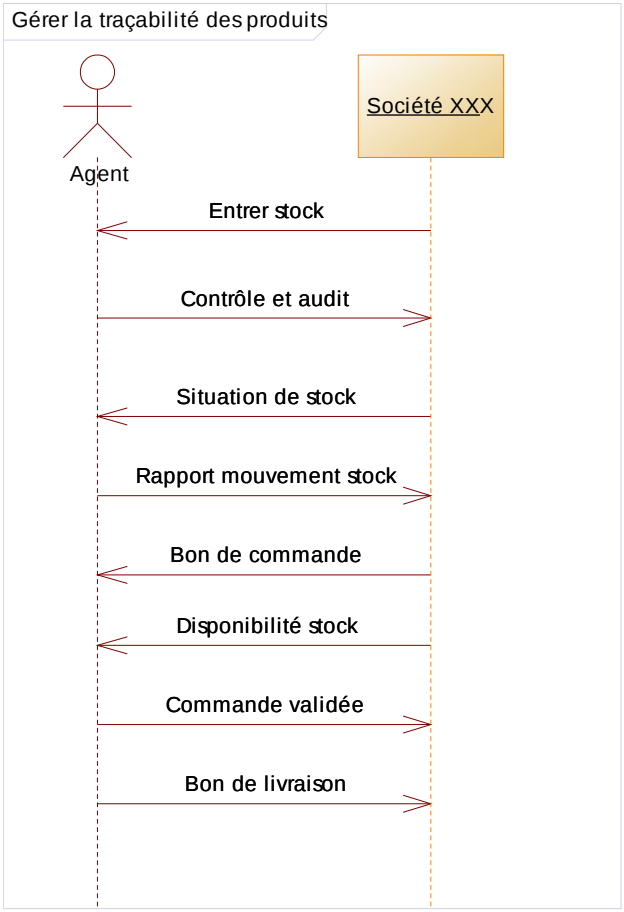
Figure 30 : Diagramme de séquence gérer le stock



Le diagramme formalisant la gestion de stock est résumée dans la figure ci-dessus. Nous pouvons y voir les messages émis entre les acteurs commençant par l'entrée du produit jusqu'à sa sortie. Ce diagramme nous permet d'agencer au mieux les actions entreprises au niveau de la gestion de stock.

Dans ce diagramme, l'on observe la participation du fournisseur (acteur externe) en effet, la gestion de stock dépend principalement de l'entrée de stock dans l'entrepôt. Si aucun produit n'entre dans l'entrepôt, aucun mouvement ne se fait alors entre les acteurs.

Figure 31 : Diagramme de séquence gérer la traçabilité des produits



f) Organisation des cas d'utilisation

Dans cette étape, nous allons surtout essayer d'améliorer nos diagrammes et nos descriptions.

Avec UML, il est en effet possible d'étayer et d'agencer les cas d'utilisation de deux façons différentes et complémentaires :

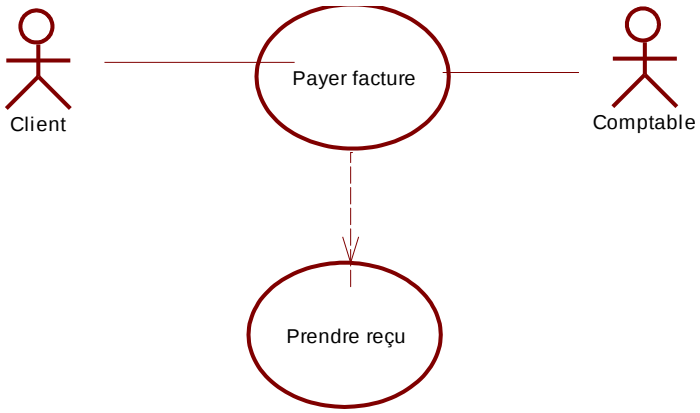
- en insérant des relations d'inclusion, d'extension et de généralisation entre les cas d'utilisation ;
- en les rassemblant en packages, afin de définir des blocs fonctionnels de plus haut niveau.

En effet, UML définit trois types de relations à savoir les relations d'inclusion, d'extension et de généralisation entre les cas d'utilisation.

L'inclusion permet de structurer et détailler un cas d'utilisation par d'autres cas d'utilisations (des sous-cas d'utilisations parfois appelés sous-fonctions du cas d'utilisation principal). Ces sous-cas sont obligatoires c'est-à-dire qu'ils doivent être exécutés pour que le cas principal soit considéré comme terminé et pour que l'objectif soit atteint.

Voici un exemple illustrant cette inclusion :

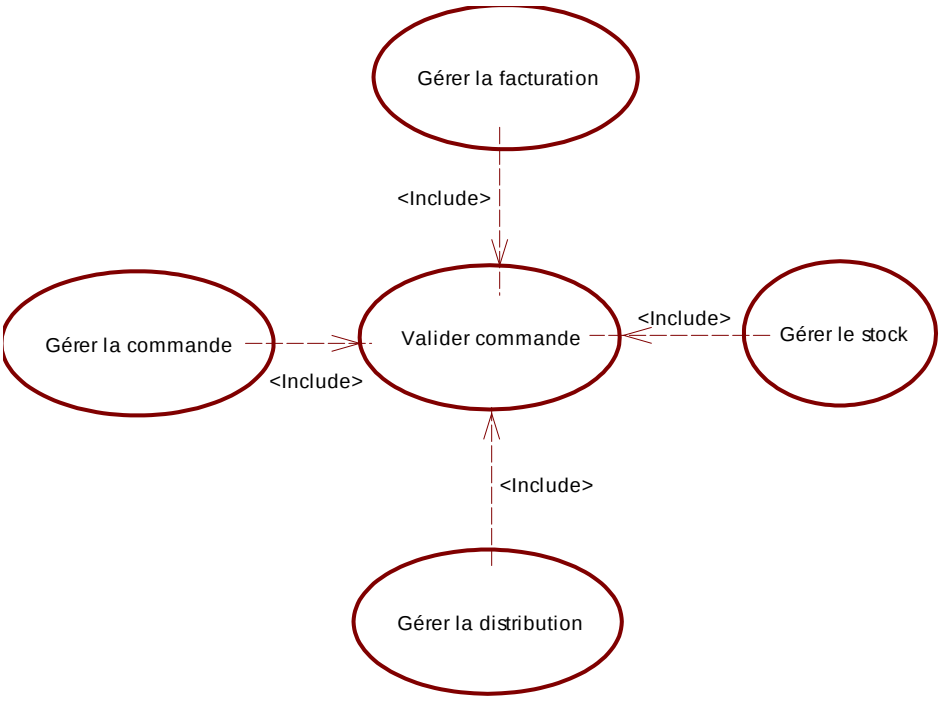
Figure 32 : Exemple de relation d'inclusion



L'exemple qui nous est proposé ici s'explique par le fait que le client paye sa facture et doit récupérer après son reçu (prouvant ainsi qu'il a payé la facture). La relation d'inclusion implique alors une action obligatoire sans laquelle le cas d'utilisation ne serait pas valide.

Dans notre cas, nous allons procéder à l'affinement de nos cas d'utilisation. Voici par exemple une généralisation de nos cas d'utilisation :

Figure 33 : Relation « include » entre cas d'utilisation



Si l'on examine en détail les descriptions textuelles des cas d'utilisation de la société XXX, on s'aperçoit rapidement que dans toutes les préconditions, on a spécifié que l'acteur principal du cas d'utilisation serait le client. En d'autres termes, la commande validée venant du client.

En fait, le processus de validation de la commande implique un flot d'événements entre les différents services du système.

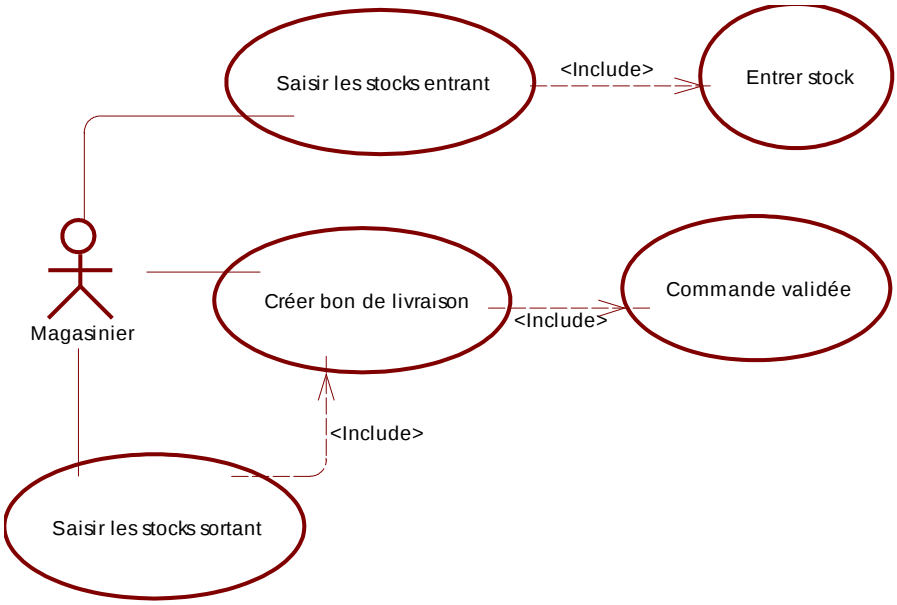
Les devis, la commande validée constituent les pièces maîtresses pour la mise en marche du système et du processus de vente dans chaque service.

L'enchaînement précité fait partie également de la capture des besoins.

La notion de cas d'utilisation inclus, ne s'exécute jamais seul, mais seulement lorsqu'il est appelé par un cas d'utilisation plus large.

Tous les cas d'utilisation sont commandés par la validation commande. Si aucune commande n'est pas validée par le client, alors les autres cas d'utilisation ne s'enchaîneront pas.

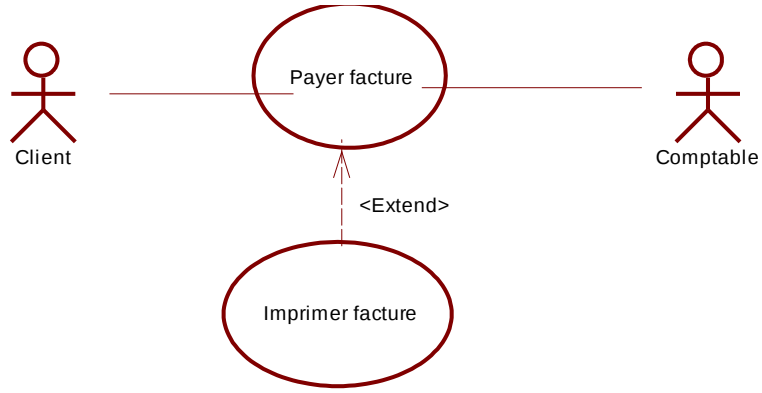
Figure 34 : Relation « include » cas d'utilisation gérer le stock



En examinant soigneusement la description textuelle des cas d'utilisation, nous avons encore identifié d'autres cas d'utilisation impliquant une inclusion au niveau des autres cas d'utilisation. Dans notre cas, la saisie des stocks entrant dépendent de l'entrer des produits dans le stock. Et la création de bon de livraison ne se fait pas qu'après une commande validée.

Par ailleurs en ce qui concerne la relation d'extension, elle est identique à la relation « include » à la différence prés que les sous cas d'utilisations sont optionnels. Ils permettent d'enrichir le cas d'utilisation principal. En effet, Relation d'extension : le cas de base en insère implicitement un autre, à un endroit spécifié indirectement dans celui qui étend. Le cas de base peut marcher tout seul, mais il peut également être complété par un autre, sous certaines conditions.

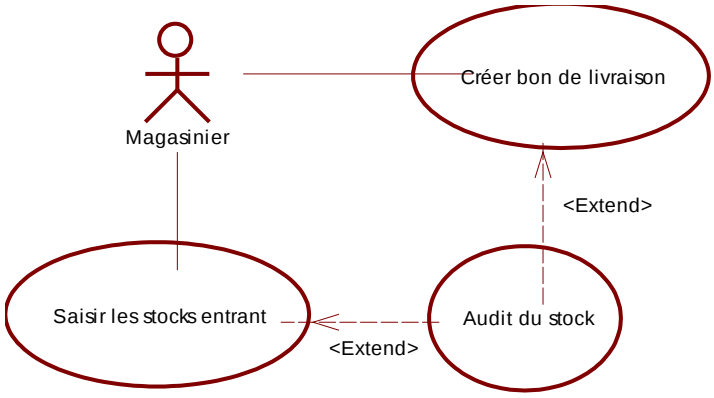
Figure 35 : Exemple de relation « exclusion »



Lorsque le client paye sa facture il a la possibilité de l'imprimer ou pas, ce qui n'empêche pas le paiement. Ce qui veut dire alors que ce n'est pas une obligation.

Si on essaye d'affiner notre cas d'utilisation, nous obtenons la figure suivante :

Figure 36 : Relation « exclude » cas d'utilisation gérer stock



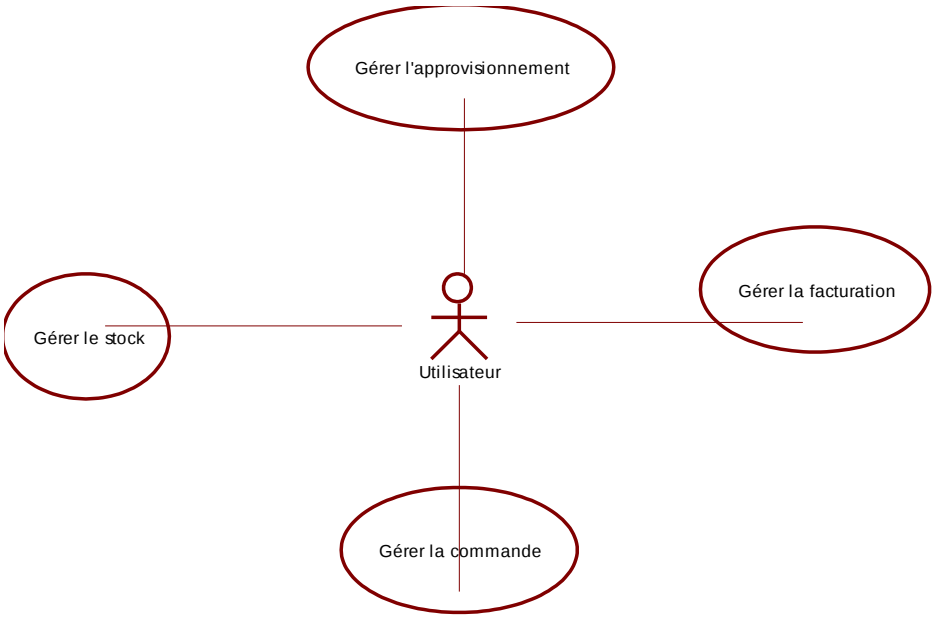
Finalement, nous avons également une autre manière d'organiser les cas d'utilisation. Il y a la relation de généralisation entre cas d'utilisation.

Les cas d'utilisation peuvent être hiérarchisés par généralisation/ spécialisation.

Les cas d'utilisation descendants héritent de la sémantique de leur parent. Ils peuvent comprendre des interactions spécifiques supplémentaires, ou modifier les interactions héritées.

Les acteurs communiquent de la même façon entre eux. Cependant, on peut généraliser tout cela de la manière suivante :

Figure 37 : Généralisation de l'acteur



Ici, nous avons assemblé les cas d'utilisation sur un seul utilisateur. Dans ce cas, l'utilisateur change au fur et à mesure où le cas d'utilisateur change également. Autrement dit, l'utilisateur peut être le comptable, le commercial, le magasinier ou le logisticien.

g) Représentation du contexte dynamique

Pour achever la préparation du diagramme d'états, nous allons maintenant établir un répertoire de l'ensemble des messages émis et surtout les messages dans la gestion de stock.

En effet, les messages reçus deviendront des événements qui déclencheront des transitions entre états et les messages émis vont donner lieu à des actions sur les transitions.

Le diagramme de séquence système réalisé précédemment liste déjà un certain nombre de messages. Nous intéressons maintenant à l'exhaustivité et à la « généricité ». Dans cette optique, nous préconisons de représenter graphiquement la totalité des messages échangés par le système avec ses acteurs sous la forme d'un diagramme de communication appelé *diagramme de contexte dynamique*.

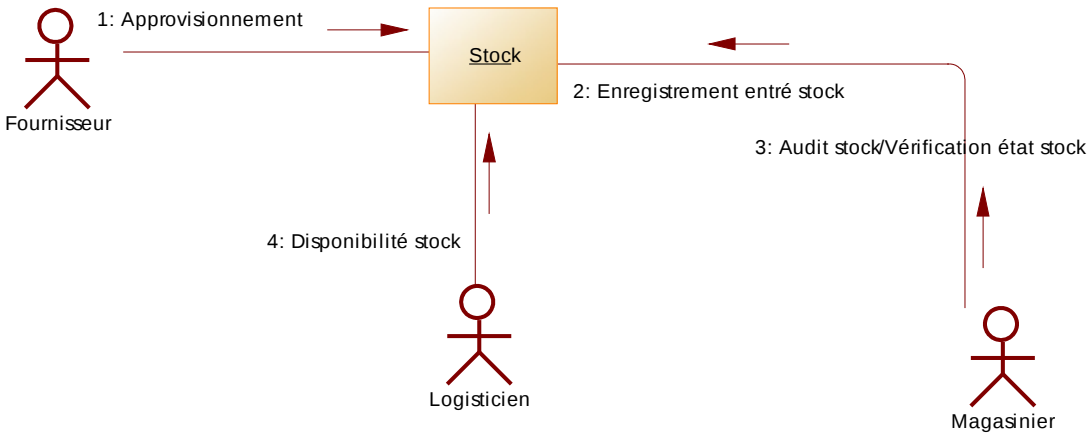
Un *diagramme de communication* est un diagramme UML fournissant une représentation graphique des interactions entre les objets d'un scénario de cas d'utilisation, l'exécution d'une opération, ou une interaction entre des classes, en s'intéressant particulièrement sur la structure du système.

Un diagramme de communication montre des acteurs, des objets (instances de classes) et leurs liens de communication (appelés liens entre objets), leur interrelation, ainsi que les messages qu'ils échangent. Les messages sont définis sur des liens entre objets qui correspondent à un lien de communication entre deux objets qui interagissent. L'ordre dans lequel les messages sont échangés est représenté par les numéros d'ordre.

En partant alors de nos diagrammes de séquence et tout particulièrement le diagramme de séquence gestion de stock, nous allons procéder à la réalisation du diagramme de contexte

dynamique. Toutefois, tous les échanges de message possible sont déjà formalisés dans les diagrammes de séquence précédemment réalisés. Mais nous allons quand même compléter ces diagrammes de séquence en réalisant le diagramme de communication correspondant.

Figure 38 : Diagramme non exhaustif du contexte dynamique de la gestion de stock



Le diagramme ci-dessus est une version sommaire de la dynamique au niveau de la gestion de stock. Etant donné qu'on essaye de simplifié autant que possible le système, nous essayons donc par la suite de ne pas trop compliquer les diagrammes. Par soucis de clarté et de compréhension au niveau du système, les diagrammes de séquence nous procurent déjà un aperçu global de notre système.

h) Réalisation du diagramme d'états

Maintenant qu'on a pu clairement compris les différents scénarios dans le système, essayons alors d'entamer la dernière étape de notre modélisation dynamique. La formalisation des scénarios nous a permis une connaissance d'ensemble des interactions entre objets permettant ainsi la représentation des règles de gestion dynamique du système.

Cependant, il faudrait se focaliser sur les classes aux comportements les plus riches de manière à développer précisément certaines des règles dynamiques. Dans ce cas, on a recourt au concept de *machine à états finis.* Ça consiste à s'intéresser au cycle de vie d'un objet générique d'une classe particulière au fil des interactions avec le reste du monde, dans les cas possibles. Nous représentons ces événements sous forme d'un *diagramme d'états.*

Selon ***[ROQUES & VALLEE]*** un *état* représente une situation durant la vie d'un objet pendant laquelle :

- il satisfait une certaine condition ;
- il exécute une certaine activité ;
- ou bien il attend un certain événement.

En effet, A différents moments de sa vie un objet peut se trouver dans différents états (caractérisés par des valeurs précises de ses attributs). Suivant son état il peut avoir accès à différentes méthodes.

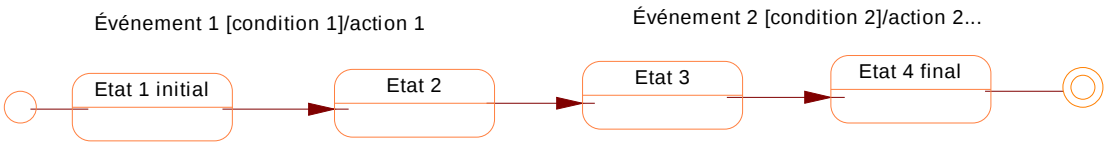
A titre de remarque, il existe deux états particuliers :

L'état initial : il représente l'objet au départ. Il n'y a qu'un seul état initial.

L'état final : Il représente l'objet à un moment où l'on peut quitter le système, parfois cela représente l'objet à la fin de sa vie. On peut avoir plusieurs états finaux.

Pour mieux illustrer tout cela, nous allons prendre un exemple de diagramme de représentation d'état-transition.

Figure 39 : Exemple de représentation des états initial et final



Dans le diagramme d'états il y a l'objet, l'événement et la transition.

Durant son existence, un objet passe par une succession d'états. L'état varie selon la vie de l'objet, en fonction des événements qui lui arrivent.

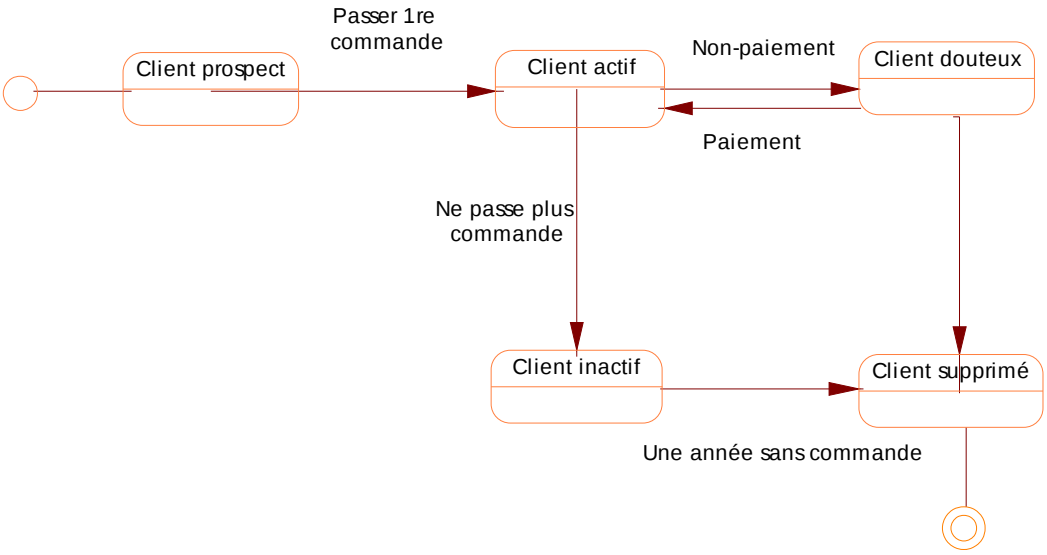
Pour qu'un objet change d'état il faut un événement. Tout comme il peut y avoir plusieurs états il peut y avoir également plusieurs événements. Les événements correspondent généralement à des méthodes qui influent sur un ou plusieurs objets.

Une transition est un lien orienté entre deux états qui illustre le fait qu'un objet peut passer d'un état à un autre. Une transition est généralement liée à un événement, mais pas toujours. Néanmoins, on peut avoir des transitions réflexives qui permettent de passer d'un état à un autre état identique.

En bref, un *diagramme d'états-transitions* est un diagramme UML qui fournit une représentation graphique d'une State Machine, le comportement public d'un classificateur (composant ou classe), sous la forme des changements de l'état du classificateur et des événements qui permettent la transition d'un état à l'autre.

Pour qu'on puisse avoir un aperçu réel et les situations réelles, nous proposons la figure suivante pour le diagramme d'état-transition de la gestion de commande d'abord.

Figure 40 : Diagramme d'état transition de l'objet client d'une gestion de commande



Dans ce premier exemple, nous pouvons voir les mouvements du client dans le système. Tout d'abord un état initial (client prospect) lequel passe une première commande voire plusieurs commandes. L'état client prospect devient alors client actif. L'action passer première commande a changé d'état.

Mais l'influence des événements change également l'état client actif en client douteux. En effet, si le client n'effectue pas le paiement, l'état change client actif change qui peut aller jusqu'à l'état final, c'est-à-dire la suppression du client du système. Pourtant, bien que le client effectue le paiement, il arrive également que pendant une période bien déterminée le client ne passe plus de commande. Dans ce cas, l'état client actif devient client inactif et débouche toujours sur l'état final, la suppression du client.

Nous voyons ici la gestion de la commande effectuée par le commercial ou un agent de la société XXX.

Mais pour avoir encore une vision réelle du système, nous allons également réaliser le diagramme d'état-transition de l'objet stock.

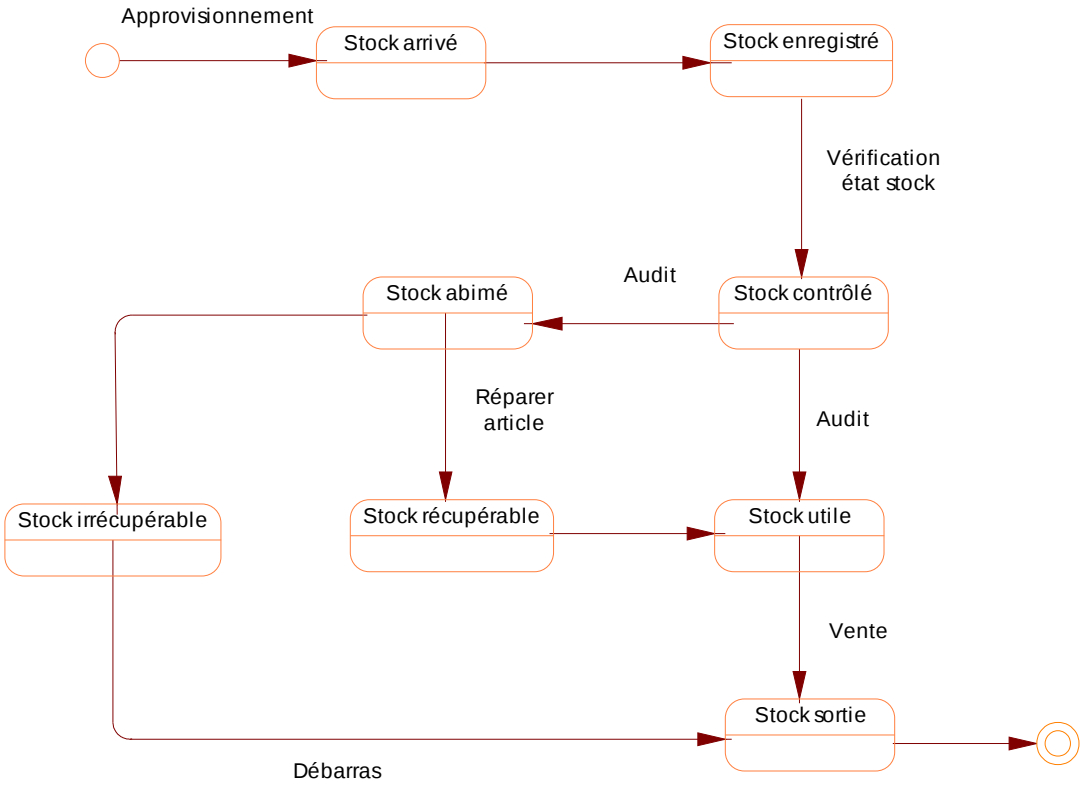
Ici, nous proposons le diagramme d'état-transition de l'objet stock qui se caractérise par les états suivants :

- Stock arrivé : arrivage du stock à un moment donné
- Stock enregistré : enregistrement du stock entrant
- Stock contrôlé : le contrôle du stock est systématique
- Stock irrécupérable : le contrôle du stock permet d'avoir un état réel du stock s'il est vendable ou non
- Stock utile : c'est le stock destiné à la vente
- Stock sortie : la commande du client implique une sortie de stock

Le changement d'état de l'objet stock se caractérise par les états-transition cités précédemment. Mais au niveau de la sortie du stock, une explication s'impose.

Lorsque l'état de l'objet stock passe au stock contrôlé, deux états peuvent apparaître : stock abimé et stock utile. Quand le l'objet stock est en état stock abimé alors l'état de l'objet devient (après événement réparation) stock récupérable ou stock irrécupérable. Toutefois, la finalité est toujours stock sortie (vente ou débarras).

Figure 41 : Exemple de diagramme d'état-transition du stock



3.2.3. Modélisation statique

a) La modélisation statique ou structurelle

Le diagramme de classe constitue peut être l'un des pivots indispensables de la modélisation avec UML. En effet, ce diagramme permet de donner et présenter la représentation statique du système à développer. Cette représentation est centrée sur quelques à savoir les concepts de classe et d'association. Chaque classe se décrit par les données et les traitements dont elle est responsable pour elle-même et vis-à-vis des autres classes. Les traitements sont matérialisés par des opérations.

Ainsi, la description du diagramme de classe se fonde sur :

- le concept d'objet,
- le concept de classe comprenant les attributs et les opérations,

- les différents types d’association entre classes.

b) Le concept objet

Avant de traiter le concept de classe, il nous paraît opportun d’avoir une première définition du concept objet. Nous remarquerons plus tard que la description d’un objet complétera simultanément la présentation du concept de classe

Dans le langage UML, un objet est un concept, une abstraction ou une chose possédant un sens dans le contexte du système à modéliser. Chaque objet a une identité et peut être distingué des autres sans considérer a priori les valeurs de ses propriétés.

Il est vrai qu’on aurait déjà du mentionner le concept objet précédemment mais compte tenu de l’importance de l’objet dans le diagramme de classes ou diagramme d’objet, la redéfinition du concept objet est très importante.

Au niveau conceptuel, un *objet* est un élément défini comme faisant partie intégrante du système décrit. Il représente un objet qui n'a pas encore été instancié car à ce stade les classes ne sont pas encore clairement définies.

Si vous devez poursuivre plus avant la mise en œuvre de votre modèle, l'objet qui a émergé lors de l'analyse se transformera probablement en instance d'une classe définie. Dans ce cas, un objet sera considéré comme une instance d'une classe.

Nous pourrons donc observer les trois situations suivantes :

- Lorsqu'un objet n'est pas une instance d'une classe - il n'a qu'un nom.
- Lorsqu'un objet est une instance d'une classe - il a un nom et une classe.
- Lorsqu'un objet est une instance d'une classe mais qu'il représente tout ou partie des instances de la classe - il a une classe mais pas de nom.

c) Structuration des classes, associations et attributs

Par rapport à la structuration des classes, nous essayerons également de voir les concepts classes, associations et attributs, avant qu’on passe à la réalisation des diagrammes de classe.

Une *classe* est une description d'un ensemble d'objets qui ont une structure et un comportement similaires et qui partagent des attributs, opérations, relations et sémantiques.

Une classe peut être créée dans les types de diagramme suivants :

- Diagramme de classes
- Diagramme de structure composite

La structure d'une classe est décrite par ses *attributs* et *associations*, et son comportement est décrit par ses *opérations*.

Les classes, et les relations que vous créez entre elles, forment la structure de base d'un MOO. Une classe définit un concept au sein de l'application modélisée, par exemple :

- un élément physique (par exemple un ordinateur),
- un élément commercial (une commande),
- un élément logique (un planning de diffusion),
- un élément d'application (un bouton OK),
- un élément de comportement (une tâche)

La représentation schématique d'une classe est la suivante : elle se représente à l'aide d'un rectangle comportant plusieurs compartiments. Il y a la désignation de la classe, la description des attributs et la description des opérations.

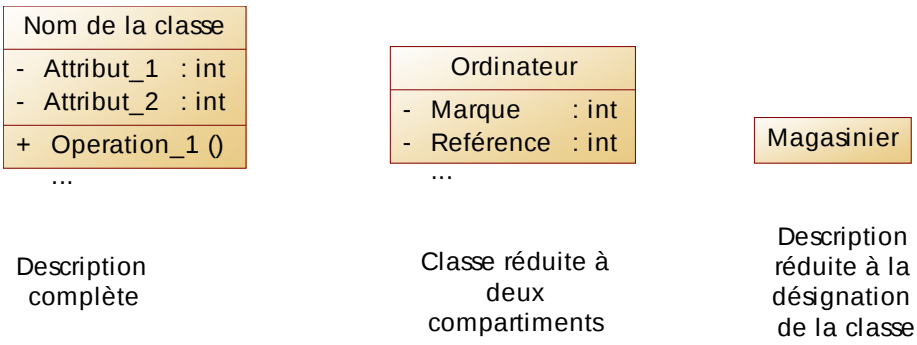
En bref, une classe représente un groupe d'objets ayant les mêmes propriétés (attributs), un même comportement (opérations), et une sémantique commune (domaine de définition). Ainsi, par rapport à l'objet, ce dernier reste une instance de la classe.

Le formalisme général d'une classe nous montre qu'il est possible de le manipuler facilement. Cependant, il est possible d'avoir les situations suivantes pour la manipulation d'une description restreinte de classe :

- description uniquement du nom et des caractéristiques générales de la classe,
- description du nom de la classe et de la liste d'attributs.

Voici un exemple de classe décrit dans la figure suivante :

Figure 42 : Formalisme général d'une classe et exemples



Par ailleurs, l'attribut est une propriété nommée d'une classe (ou d'une interface) et qui décrit particulièrement ses caractéristiques.

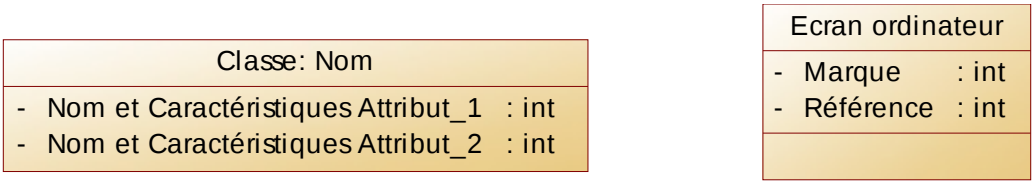
Un attribut peut être créé pour une classe ou une interface dans les types de diagramme suivants :

- Diagramme de classes
- Diagramme de structure composite
- Diagramme de composants

Une classe ou interface peut avoir un ou plusieurs attributs. Tous les objets d'une classe peuvent avoir les mêmes attributs, mais pas avec les mêmes valeurs.

Autrement dit, un attribut est une propriété de base d'une classe. Pour chaque objet d'une classe, l'attribut prend une valeur (sauf cas d'attributs multivalués).

Figure 43 : Formalisme d'attributs de classe et exemple



Toutefois, on peut qualifier le nom de la classe par ce qu'on appelle « stéréotype ».

En effet, la description complète des attributs d'une classe doit comporter un certain nombre de caractéristiques lesquelles nous allons mettre évidence en respectant le formalisme suivant :

Visibilité/Nom attribut : type [= valeur initiale {propriétés}]

– *Visibilité* : se reporter aux explications données plus loin sur ce point.

– *Nom d'attribut* : nom unique dans sa classe.

– *Type* : type primitif (entier, chaîne de caractères…) dépendant des types disponibles dans le langage d'implémentation ou type classe matérialisant un lien avec une autre classe.

– *Valeur initiale* : valeur facultative donnée à l'initialisation d'un objet de la classe.

– {*propriétés*} : valeurs marquées facultatives (ex. : « interdit » pour mise à jour interdite).

En d'autres termes, les noms d'attribut doivent être uniques au sein d'une classe. Néanmoins, on peut affecter le même nom à plusieurs attributs, à condition qu'ils soient placés dans des classes différentes.

Après qu'on ait vu ce que c'est l'attribut, passons maintenant à la notion d'opérations.

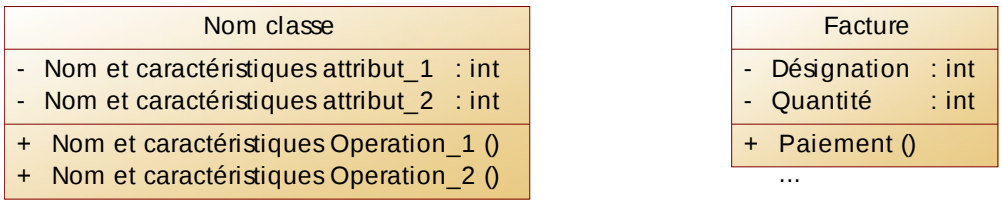
Une opération est une spécification nommée d'un service qui peut être requis par n'importe quel objet d'une classe afin de modifier son comportement. Il s'agit de la spécification d'une requête qu'un objet peut être appelé à exécuter. Autrement dit, une opération est une fonction qu'on peut appliquer aux objets d'une classe. Elle permet de décrire le comportement d'un objet. Par rapport à la méthode, cette dernière est l'implémentation d'une opération.

Une classe peut comporter un nombre illimité d'opérations, ou bien n'en comporter aucune.

Les opérations doivent être dotées d'un nom et d'une liste de paramètres. Plusieurs opérations peuvent porter le même nom au sein d'une classe, à condition toutefois que leurs paramètres soient différents.

Comme dans la classe et les attributs, le formalisme des opérations de classe peut être manipulé facilement. Une opération peut désigner soit uniquement par son nom soit par son nom, sa liste de paramètres et son type de résultat.

Figure 44 : Formalisme et exemple d'opérations de classe



Comme pour l'attribut également, la description complète des opérations d'une classe doit comporter certain nombre de caractéristiques lesquelles doivent respecter le formalisme suivant :

Visibilité Nom d'opération (paramètres) [:[type résultat] {propriétés}]

– *Visibilité* : se reporter aux explications données plus loin sur ce point.

– *Nom d'opération* : utiliser un verbe représentant l'action à réaliser.

– *Paramètres* : liste de paramètres (chaque paramètre peut être décrit, en plus de son nom, par son type et sa valeur par défaut). L'absence de paramètre est indiquée par ( ).

– *Type résultat* : type de (s) valeur(s) retourné(s) dépendant des types disponibles dans le langage d'implémentation. Par défaut, une opération ne retourne pas de valeur, ceci est indiqué par exemple par le mot réservé « void » dans le langage C++ ou Java.

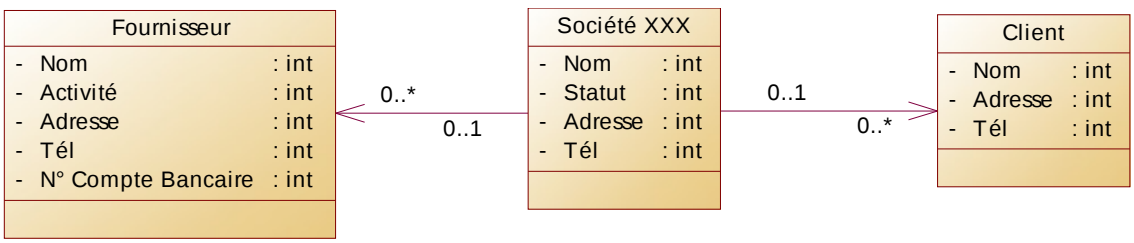
– {*propriétés*} : valeurs facultatives applicables (ex. : {query} pour un comportement sans influence sur l'état du système).

Toutefois dans la réalisation et la réalisation d'un diagramme de classes, il ne faut pas omettre de savoir ce que c'est une « association ».

Une association représente une relation sémantique durable entre deux classes. En d'autres termes, Une *association* représente une relation structurelle entre des classes ou entre une classe et une interface.

On peut la représenter sous une forme d'un trait plein entre deux symboles.

Figure 45 : Exemple de représentation d'une association entre trois classes

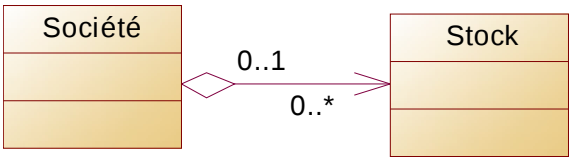


Une association peut avoir plusieurs types également, nous citons entre autre l'*agrégation.* Une *agrégation* est un type d'association dans lequel une classe représente un ensemble (un tout) composé d'éléments (les parties). Cette association spéciale est un lien de type "comporte un". Ceci signifie qu'un objet du tout comporte des objets de la partie.

En fait une agrégation est un cas particulier d'association non symétrique exprimant une relation de contenance. Les agrégations n'ont pas besoin d'être nommées : implicitement elles signifient « contient », « est composé de ».

Dans notre exemple nous pouvons avoir une agrégation suivante :

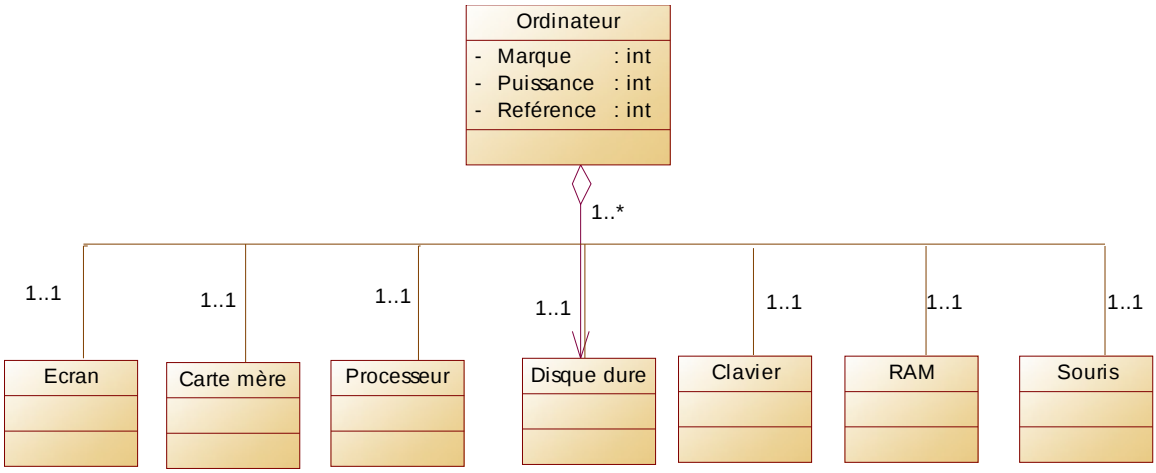
Figure 46 : Exemple d'agrégation entre deux classes



Dans l'exemple ci-dessus, l'agrégation signifie que la classe société forme le tout et elle peut comporter des stocks.

Pour avoir un peu plus de clarté dans notre cas, voici un exemple concret :

Figure 47 : Exemple d'agrégation classe ordinateur



Cette figure illustre bien le cas de notre classe ordinateur. L'ordinateur forme le tout. Pourtant un ordinateur comporte entre autre un écran, carte mère, processeur, disque dure, clavier, RAM, souris…

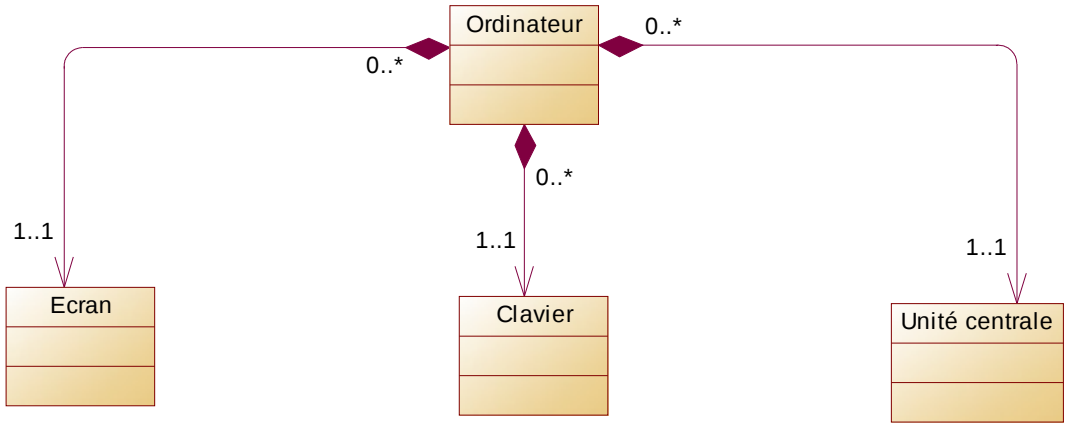
Par ailleurs, toujours dans les différents types d'association, nous avons la composition. Une *composition* est un type particulier d'agrégation dans lequel les parties sont fortement liées au tout. Dans une composition, un objet ne peut être partie que d'un seul composite à la fois, et le composite gère la création et la destruction de ses parties.

La composition est une relation d'agrégation dans laquelle il existe une contrainte de durée de vie entre la classe « composant » et la ou les classes « composé ». Autrement dit la

suppression de la classe « composé » implique la suppression de la ou des classes « composant ».

Nous aurons donc la figure suivante en ce qui concerne la notion de composition, si l'on tient toujours compte de notre exemple précédent.

Figure 48 : Exemple de composition classe ordinateur



La figure 48 nous montre que l'Ecran, le clavier et l'unité centrale ne forme qu'une partie de la classe objet ordinateur. Cependant, si l'on supprime la classe ordinateur, automatiquement les classes écran, clavier et unité centrale disparaissent également.

La structuration du modèle UML repose sur la structuration des classes. De ce fait c'est une étape importante dans la modélisation orientée objet.

Il s'agit ici d'affiner les classes déjà c'est-à-dire affiner en même temps les associations en ajoutant des attributs et éventuellement ajouter des opérations.

d) Diagramme des classes

Dans cette partie, nous essayerons de réaliser les relations structurelles entre classes. La présentation du projet nous rend compte d'une certaine spécification du système.

Par rapport à la gestion de la commande :

Une commande contient un devis, un bon de commande une facturation.

Un devis contient les attributions suivantes : nom de la société, adresse (code postale, ville), téléphone, télécopie, date, n° devis, code client, informations du client destinataire du devis (nom, adresse, tél), quantité, description, prix unitaire, taxe, montant.

Un bon de commande contient : le nom de la société, adresse, téléphone, télécopie, le numéro du bon de commande, date du bon de commande, les informations sur le client, nom et adresse du client, nom et signature de l'expéditeur, quantité, description, prix unitaire, sous total et total.

Une facture contient également : les informations générales sur la société (expéditeur), les références à se rappeler (numéro de la facture, date, n° du client, date commande, référence commande), informations générales sur le client (destinataire), adresse de livraison, référence

article, libellé article, taux TVA, quantité, prix unitaire hors taxe, taux remise, montant hors taxe, total HT et total TTC, si paiement par chèque ou virement n° RIB de la société.

Comme nous pouvons le voir la gestion de la commande englobe à la fois la gestion de la commande et la gestion de la facturation.

En ce qui concerne la gestion du stock, à la différence de la gestion de la commande, les principales attributions sont plutôt le suivi et le traçage des produits entrant et sortie, mais également l'évolution de leur statut.

On aura donc un rapport de mouvement du stock, état du stock, bon de commande (achat).

Le rapport de mouvement du stock contient la description, la référence, le nom et la quantité des stocks entrant et sortant.

L'état stock rend compte de l'inventaire du stock c'est-à-dire le nombre de produits disponible, leur état.

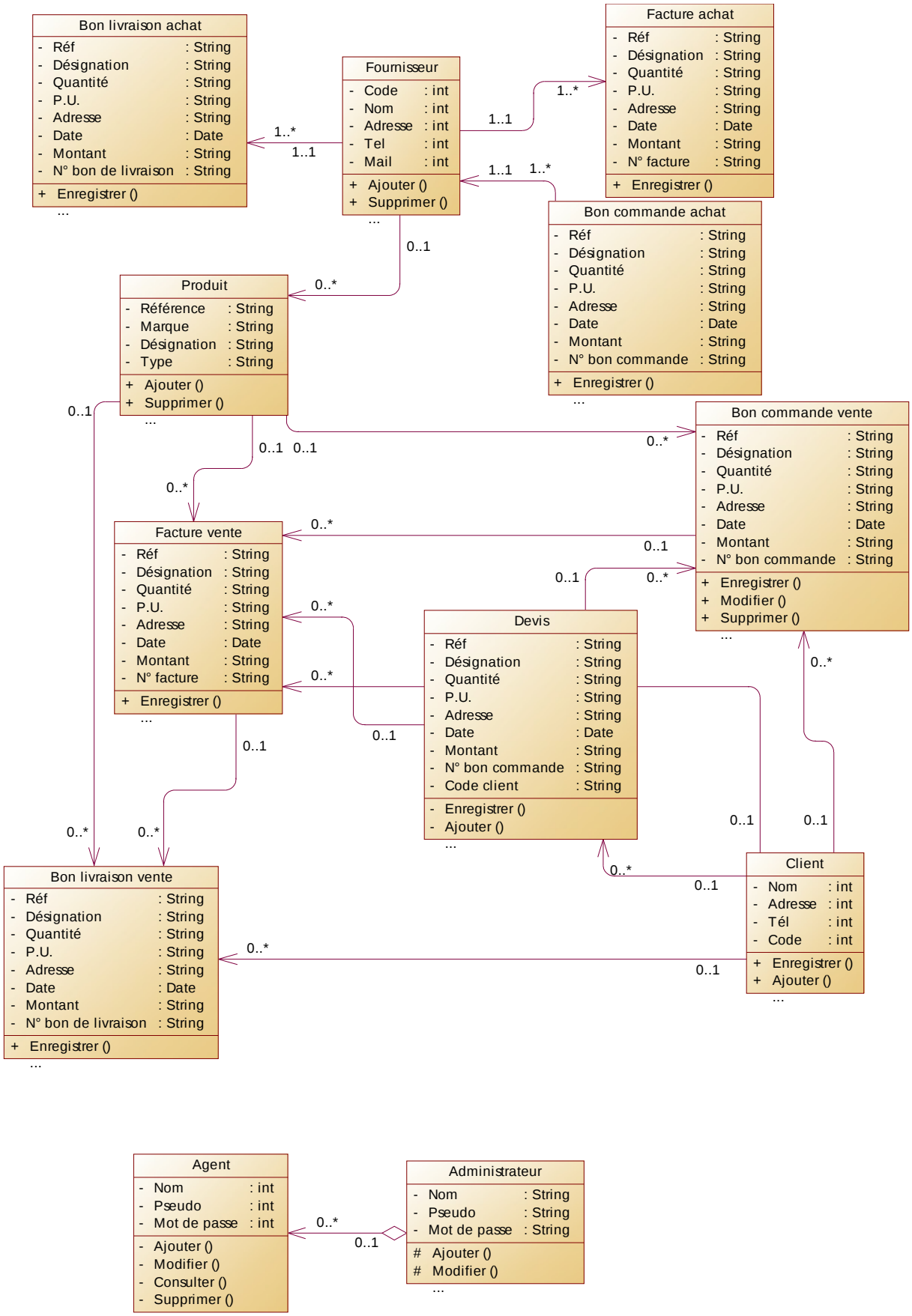
Quant au bon de commande (achat), c'est le même que celui dans le bon de commande (vente) sauf que le bon de commande est adressé au fournisseur. C'est pareil pour la facture (achat).

Le modèle présenté ci-dessous montre une vue statique de l'application. Ce modèle nous montre les différentes relations entre les différentes classes composant notre application.

Le schéma ci-dessous nous donne une vue générale de notre application. Compte tenu de ce qu'on a évoqué précédemment, nous avons les classes principales lesquelles vont nous être utiles à réaliser l'application.

En effet, le diagramme de classe représenté par le schéma suivant nous permet d'avoir de plus amples informations notamment sur l'application car il représente le modèle physique de l'application. On le représente alors en essayant d'établir des relations entre les classes. C'est en quelque sorte une ébauche de notre base de données.

Figure 49 : Diagramme de classes



Par rapport à ce diagramme faisons un peu le point entre les classes :

Le client

Un client peut avoir plusieurs factures ; il en est de même pour les commandes et les devis. Et le client quand il reçoit le produit, normalement il recevra également un bon de livraison.

Le Fournisseur :

Le fournisseur reçoit un ou plusieurs commandes de la société cela dépend des commandes effectuées par la société. Ainsi les commandes vont toujours avec une facture (achat) et un bon de livraison.

Facture, bon de commande, Devis, bon de livraison:

Il s'agit ici du côté de la vente. Ainsi, chaque facture, bon de commande, devis ou bon de livraison possède une référence, une date, un code client.

Facture, bon de commande, bon de livraison:

Du côté de l'achat, c'est pareil que comme dans la vente mais sans le devis. Toutefois chaque pièce doit posséder un code fournisseur, une référence, une date.

Produit :

Etant donné que ce qui nous intéresse le plus c'est la vente et la gestion du stock. Afin d'avoir une meilleure suivie des produit, un produit doit avoir une référence, une désignation, une marque, TVA et son prix d'achat, une catégorie.

Administrateur/agent :

L'administrateur ici est la personne responsable de la mise en œuvre de l'application. Il est en charge d'administrer l'application et de gérer la distribution au niveau de chaque acteur. Il a accès physiquement à l'application. L'agent quant à lui, gère toutes les informations et toutes les actions que ce soit au niveau des clients, des fournisseurs, les bons (commande ou livraison) et la facture. L'agent, ici, représente les personnes responsables des différentes activités de la société. Dans le schéma, nous mettons en évidence un lien d'héritage entre l'administrateur et l'agent. Car seul l'administrateur peut gérer les agents.