الجمهورية الجزائرية الديموقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي و البحث العلمي
Minstère de l'Enseignement Supérieur et de la Recherche Scientifique

CENTRE UNIVERSITAIRE DE MILA
INSTITUT DES SCIENCES ET DE LA TECHNOLOGIE

Réf. ***/12***

Mémoire de fin d'étude
Présenté pour l'obtention du diplôme de

Licence Académique

Domaine : **Mathématiques et Informatique**
Filière : **Informatique**

La gestion commerciale d'un magasin de vente en gros

Thème

Présenté par :
1- Daas Abderrahmene

Dirigé par :
-Mme F.Benabderrahmene

Année universitaire 2011-2012

Remerciement

Nous Remercions En tout premier lieu ALLAH le tout puissant qui m'a donné la force, la volonté et le courage pour accomplir ce modeste travail.

Nous tenons à exprimer nos remerciements avec un grand plaisir et un grand respect à notre encadreur Mme F.Benabderrahmane, ses conseils, sa disponibilité et ses encouragements qui nous ont permis de réaliser ce travail dans les meilleures conditions.

Nous désirons remercier tous nos enseignants pour toutes les connaissances qu'ils nous ont inculquées.

DEDICACES

A mes très chers parents qui m'ont beaucoup soutenu, encouragé et qui ont fait de moi ce que je suis aujourd'hui ;

A mes chères sœurs

Imane , Soumeya ;

A mon beau frère

Yahia ;

A tous mes amis en particulier Fayçal, Zaki, Radoine, Nassim, Amine, Fouad, Brahim, Halim, Nori ;

A tous mes amis en CUM et en particulier 3emeanneé informatique ;

A tous ceux que j'aime tant et que je n'ai pas cités ;

Je dédie ce mémoire …

DAAS ABDERRAHMENE

Sommaire

1. Introduction :

Avant l'invention de l'ordinateur, on enregistrait toutes les informations manuellement sur des supports en papier ce qui engendrait beaucoup de problèmes tels que la perte de temps considérable dans la recherche de ces informations ou la dégradation de ces dernières. ..Etc.

Il ne fait désormais plus aucun doute que l'informatique est la révolution la plus importante et la plus innovante qui a marqué la vie de l'humanité moderne. En effet, les logiciels informatiques proposent maintenant des solutions à tous les problèmes de la vie, aussi bien dans les domaines professionnels que pour les applications personnelles. Les méthodes de conception et de développement ont vu l'avènement d'autant de technologies qui facilitent leur mise en place et leurs donnent des possibilités et des fonctionnalités de plus en plus étendues.

2. Présentation du problème et objectifs :

La finalité de notre projet est de développer un système d'information pour la gestion commerciale d'un magasin de vente en gros.

L'idée proposée donc consiste à faire l'automatisation de la gestion commerciale du magasin pour résoudre les problèmes rencontrés dans le magasin de vente en gros.

Et qui se traduise par :

- La quantité volumineuse d'informations concernant le magasin de vente en gros tels que la réception des marchandises, la vente, les clients et les fournisseurs, les produits...etc.
- Les fonctions diverses exécutés par le grossiste.

3. Choix de la méthode :

Nous nous sommes basés sur l'application de l'une des méthodes orientées objet qui est la méthode OMT (Object modelling technique) en utilisant le standard des langages objet qui est la notation UML (Unified Modeling Language).

4. Plan de travail :

Notre travail est organisé en trois chapitres principaux, qu'on peut résumer comme suit :

Dans le chapitre I on va présenter la méthode OMT qu'on a choisi pour développer notre projet. Donc on va présenter les étapes principales suivies par la méthode pour le développement d'un système, ainsi que les modèles utilisés pour le décrire.

Dans le chapitre II on va essayer de faire une intégration de la notation UML dans la méthode OMT, donc on va suivre les étapes de la méthode OMT et à chaque fois en intègre les notations d'UML en utilisant les formalismes et les schémas associés.

Dans le chapitre III on va développer notre application, en suivant les étapes d'OMT, et en utilisant les notations d'UML.

Chapitre I
introduction à la méthode OMT

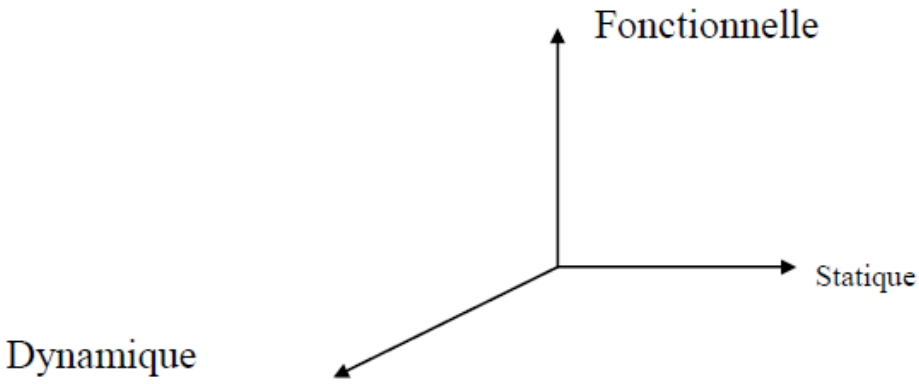
1. Introduction :

1.1. Définition :

La méthode OMT est une méthode d'analyse orientée objet pour le développement des systèmes d'information.

Développé à la fin des années 80 par RUMBAUGH, et comme les autres méthodes de modélisation elle nécessite une description complète d'un système la prise en compte de trois dimensions.

- La dimension statique : c'est la description de la structure des objets, notamment via le Diagramme d'objets.
- La dimension dynamique : c'est la description des règles d'évolution (changements) des objets au cours du temps via le Diagramme d'états.
- La dimension fonctionnel : description des traitements / transformation des données du système via le Diagramme des flux de données (DFD).



1.2. Les trois modèles d'OMT :

- Le Modèle OBJET : représente les différents types d'objets, et les relations structurelles statiques entre les objets.
- Le Modèle dynamique : représente les interactions temporelles entre les objets les événements.
- Le Modèle fonctionnel (Transformations) : représente les dépendances fonctionnelles.

1.3. Cycle de développement d'OMT :

<table>
<tr><th>PHASE</th><th>MODELES</th></tr>
<tr><td rowspan="3">ANALYSE</td><td>Modèle objet (structure statique) { objet, classe, attribut, liens-association, opérations-méthodes, généralisation, héritage</td></tr>
<tr><td>Modèle dynamique (séquencement des interaction) { Evénement, opérations, état/transition, Diagramme d'état</td></tr>
<tr><td>Modèle fonctionnel (transformation de données) { DFD (Data Flow diagram), entités externe (acteur), processus, répertoire de données, flot de données</td></tr>
<tr><td>CONCEPTION SYSTEME</td><td>➢ Architecture</td></tr>
<tr><td>CONCEPTION OBJET</td><td>➢ Détails d'implémentation</td></tr>
<tr><td>IMPLEMENTATION</td><td>➢ Utilisation d'un langage de programmation/SGBD</td></tr>
</table>

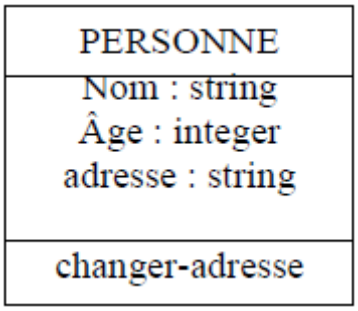
2. le modèle objet :

2.1. Objet et classe :

- Objet : c'est un concept, une abstraction, ou une chose ayant une signification pour le problème traité.
- Classe : c'est une description abstraite d'un ensemble d'objets possédant les mêmes caractéristiques.
- Attribut : c'est une propriété nommée d'une classe, Ce sont des variables stockant des informations sur l'état de l'objet.

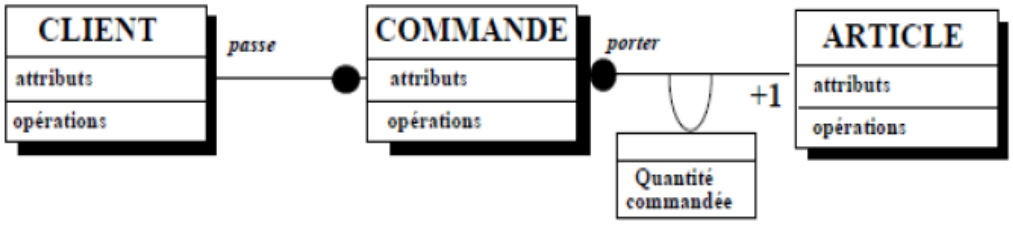
2.2. Opération et méthode :

- Opération : fonction ou transformation appliquée aux objets ou classes.
- Méthode : implémentation d'une opération sur une classe.
  Exemple :



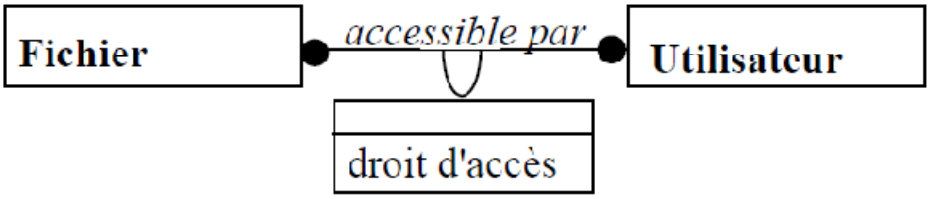
2.3. Lien et association :

- Lien : connexion physique ou conceptuelle entre objets.
- Association : décrit un ensemble de liens ayant même structure et même sémantique entre deux ou plusieurs classes.
- Multiplicité : nombre d'instances d'une classe participant à une association.
  Exemple :



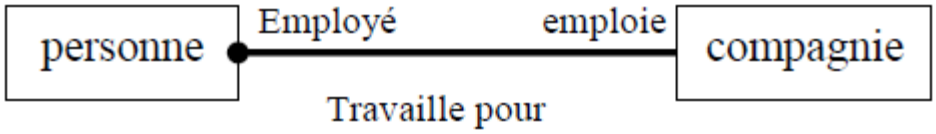
2.4. Construction additionnelles:

- Attribut de liens: Propriété d'une association décrivant des valeurs de données pour chaque lien de l'association.
  Exemple :



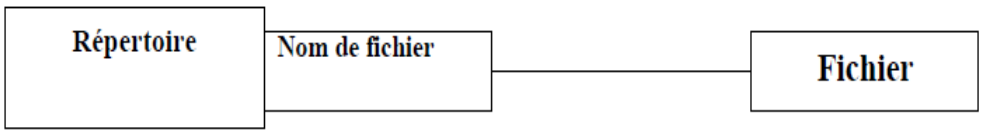
- Rôle : objectif de fin d'une association associé à un sens.

Exemple :

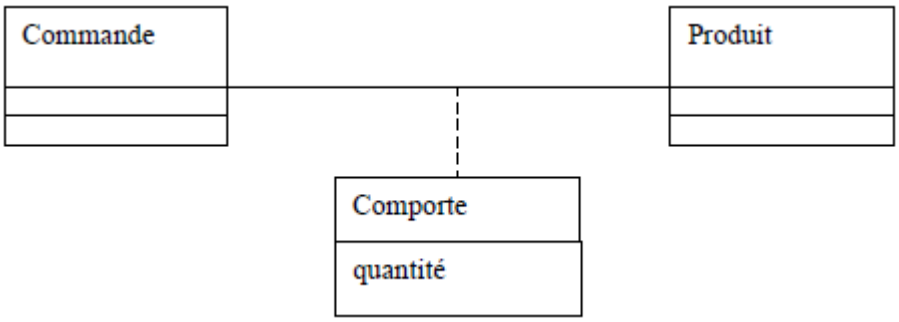


- Qualification : attribut spécial qui réduit la multiplicité effective des associations un-à-plusieurs et plusieurs-à-plusieurs.

Exemple :

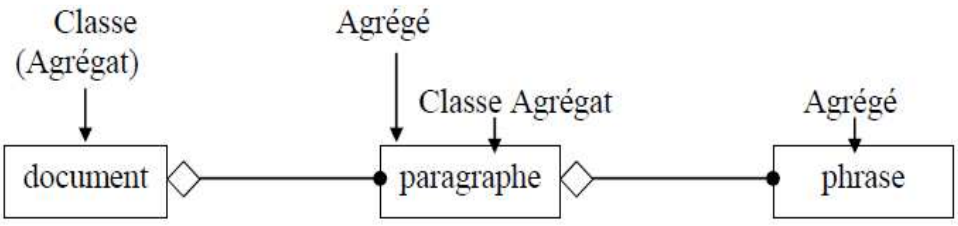


- Classe associative : Parfois, un attribut dépend de deux identifiants, appartenant à deux classes différentes.



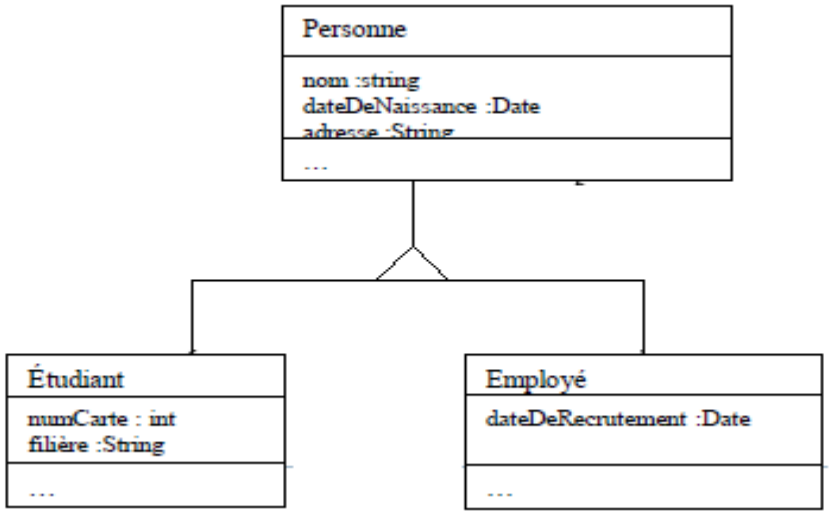
L'attribut «quantité» n'appartient ni à la commande qui peut contenir plusieurs produits, ni au produit qui peut figurer dans plusieurs commandes. On place l'attribut «quantité» dans l'association «comporter».

- Agrégation : Relation « partie-de » entre des objets composants et des objets composites.
  Exemple :

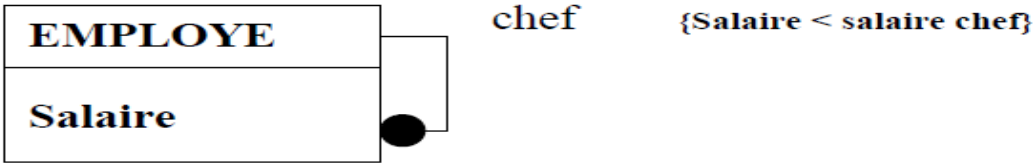


- Généralisation et héritage : relation entre une classe et une ou plusieurs de ses versions.
  La superclasse contient les attributs et opérations communes.
  Chaque sous-classe peut avoir ses propres attributs et ses propres opérations.

Exemple :



- Les contraintes : sont des relations fonctionnelles entre entités d'un modèle objet. Le terme entité englobe les objets, les classes, les attributs, les liens et les associations.
  Exemple :



2.5.Module :

Un module est une construction logique permettant de regrouper les classes, les associations et les généralisations. Un modèle objet est composé d'un ou plusieurs modules.

2.6. Les étapes de modélisation :

2.6.1. Identification des classes :

- ✓ A partir des besoins : noms et phrases nominales.
- ✓ Les classes incluent les entités physiques ainsi qu'abstraites.
- ✓ Eliminer les classes redondantes (ex : client et usager).

2.6.2. Le dictionnaire de données :

- ✓ Ecrire en langage naturel une description des différentes classes identifiées.

2.6.3. Identification des associations :

- ✓ Association correspond souvent aux verbes d'une phrase.
- ✓ Toute dépendance entre deux ou plusieurs classes représente une association.
- ✓ Une référence d'une classe à une autre représente une association.

2.6.4. Identification des attributs :

- ✓ Analyse des noms dans les phrases possessives.
- ✓ Qualificatifs / états des objets…

2.6.5. La hiérarchie de classes :

- ✓ Généraliser à partir des attributs communs, associations ou opérations communes (technique ascendante).
- ✓ Spécialiser à partir de l'application (technique descendante).

2.6.6. Vérification des chemins d'accès :

- ✓ Les questions courantes qui peuvent être posées doivent trouver réponse en naviguant dans le modèle.
- ✓ Lorsqu'une valeur unique est attendue, est ce que le chemin dans le graphe du modèle la donne ? (cardinalités des associations).

2.6.7. Définitions des modules :

- ✓ Le groupement de classes en modules peut être fait suivant les types de traitements effectués. En fait, il s'agit de définir ici les sous-systèmes possibles.

2.6.8. Itération et raffinement :

- ✓ Un modèle est rarement correct à la 1ère passe.
- ✓ Des objets peuvent être revus après le passage au modèle dynamique et fonctionnel.

3. le modèle dynamique :

S'exprime à l'aide des concepts suivants :

- ✓ Evénement
- ✓ Etat
- ✓ Scénario
- ✓ Transition
- ✓ Opération

3.1.Evénement et état :

- ✓ Evénement : un fait est survenu à un instant donné.
- ✓ Etat : L'intervalle entre deux événements reçus par un objet.

3.2. Scénario : Une séquence d'événements qui surviennent durant une exécution particulière du système.

3.3. Transition : changement d'état causé par un événement.

3.4. Opération : comment l'objet réagit aux événements.

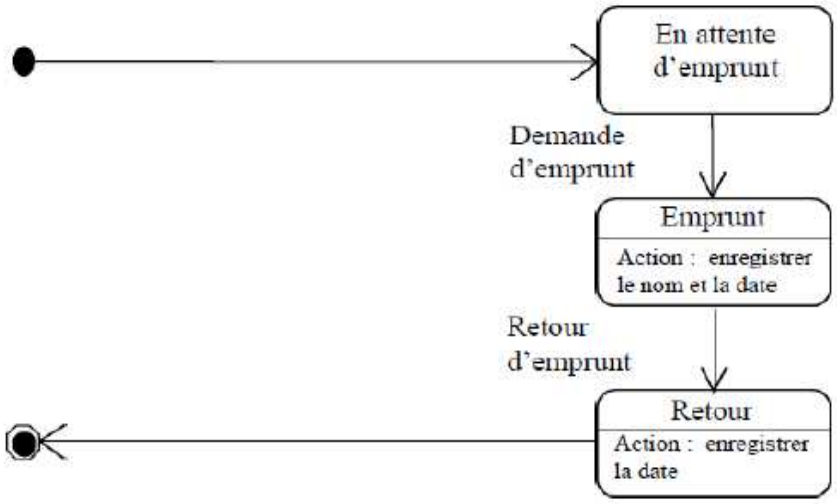
Deux types d'opération :

- ✓ Une activité : opération dont l'exécution dure un certain laps de temps. Elle est associée à un état.
- ✓ Une action : est une opération instantanée, elle est associée à une transition d'état (événement). Une action peut envoyer un événement à un objet.

3.5. Diagramme d'état :

Rallie événement et états et décrit le comportement d'une classe d'objets.

Exemple1 :diagramme état-transition simplifié d'un livre



3.6. Construction du modèle dynamique :

3.6.1. Identification des cas d'utilisation :

- ✓ Un cas d'utilisation décrit une catégorie de scénarios pour un aspect de fonctionnement d'un système (relié à un acteur).
- ✓ Identifier les différents acteurs intervenants dans le système
- ✓ Pour chaque acteur, décrire les différents chemins d'utilisation du système (les interactions) et faire apparaître toutes les interactions possibles.

3.6.2. Préparation des scénarios :

- ✓ Préparation d'un ou plusieurs cas typiques de dialogues entre l'utilisateur et le système afin d'avoir une idée du comportement du système.
- ✓ Le dialogue fait apparaître des scénarios des cas normaux d'utilisation (sans condition d'erreurs) et des scénarios d'exception (avec erreurs).

3.6.3. Identification des événements :

- ✓ Identification des événements externes par examen des scénarios : signal, entrée, décision, interruption, transition et action depuis et vers l'utilisateur.
- ✓ Le scénario décrit sous forme de texte est ensuite formalisé sous forme de diagramme appelé : Trace d'événements.

3.6.4. Construction du diagramme d'états :

- ✓ Préparer un diagramme d'états pour chaque objet dont le comportement dynamique n'est pas trivial, en indiquant les événements qu'il émet et qu'il reçoit.

- ✓ Chaque scénario ou suivi d'événements est un chemin dans le diagramme d'états
- ✓ Commencer par les suivis d'événements directement liés à la classe à modéliser.
- ✓ Incluez les autres scénarios dans le diagramme, en repérant les points où le scénario diffère des précédents. Ces points doivent correspondre à un état existant du diagramme.
- ✓ Dans ce cas attacher la nouvelle séquence comme nouvelle alternative dans les chemins préalablement définis.

3.6.5. Vérification de la correspondance des événements entre les objets :

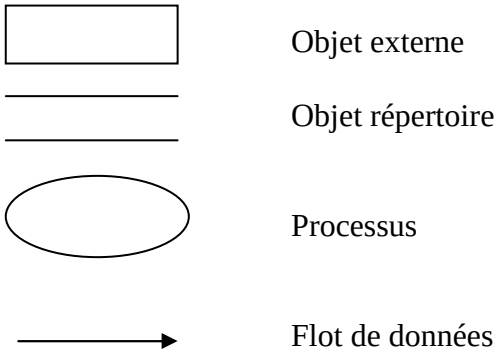
- ✓ Chaque événement doit lier un émetteur à un récepteur
- ✓ Les états sans suivants ni précédents ne doivent correspondre qu'à des états initiaux ou finaux d'une séquence d'interaction.

4. le modèle fonctionnel :

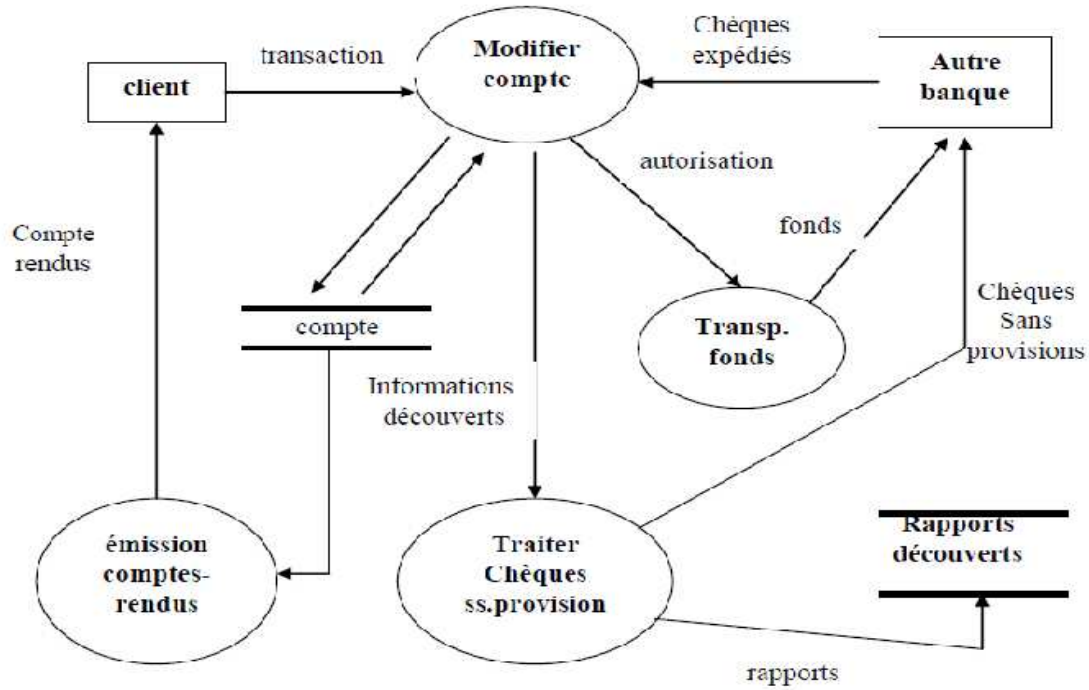
4.1. Définition d'un DFD (diagramme des flux de données) :

C'est un graphe symbolisant des flots des données à partir d'objets sources vers des objets destinataires et à travers un processus de transformation.

Concept :



Exemple de DFD :



4.2. Construction du modèle fonctionnel :

4.2.1. Identification des valeurs d'entrée et de sortie : paramètres des événements entre le système et l'environnement.

4.2.2. Construction du diagramme à flots de données.

4.2.3. Description des fonctions : chaque fonction doit être décrite soit en utilisant le langage naturel, soit du pseudo – code soit en utilisant des tables de décisions …

4.2.4. Identification des contraintes : les contraintes sont des dépendances fonctionnelles entre objets. En général, ce sont des **conditions d'entrée/sortie.**

4.2.5. Spécifications des critères d'optimisation : spécifier les valeurs qui doivent être maximisées, ou minimisées (par exemple, minimiser le nombre de messages entre systèmes, minimiser le temps de verrouillage d'un compte).

5. description du système :

Se fait en plusieurs étapes :

5.1.Décomposition du système :

- ✓ Subdiviser le système en un certain nombre de composants

- ✓ Un sous-système englobe les parties du système qui partagent les mêmes propriétés.
- ✓ Un sous-système est un paquetage de classes, d'associations, d'opérations, d'événements, et de contraintes reliés. Il faut ici privilégier les hiérarchies d'agrégats et de généralisation.
- ✓ Chaque sous-système comporte une interface bien définie avec les autres sous-systèmes.
- ✓ Chaque sous-système doit être défini de manière à minimiser les interactions avec les autres sous-systèmes.

5.2.Identification de la concurrence :

- ✓ Dans le modèle d'analyse, comme dans le monde réel tous les objets sont concurrents. Dans l'implémentation, les logiciels ne sont pas tous concurrents, parce qu'un processeur peut supporter plusieurs objets. Un des rôles de la conception du système est d'identifier les objets logiciels concurrents
- ✓ Identification des objets concurrents : « Deux objets sont concurrents s'ils peuvent recevoir simultanément un événement sans interagir ».
- ✓ Définition des tâches concurrentes

5.3.Allocation des sous-systèmes aux processeurs et aux tâches :

- ✓ Chaque sous-système concurrent doit être alloué à un matériel particulier qui peut être soit un processeur, soit une unité fonctionnelle spécialisée.
- ✓ Estimer les besoins en ressources, en s'appuyant sur des besoins de performances.
- ✓ Répartition matériel – logiciel
- ✓ Allouer les sous-systèmes logiciels à un processeur pour satisfaire les besoins de performances et limiter les communications inter-processeurs.

6. Conception des objets :

Les étapes :

- ✓ Combiner les trois modèles pour obtenir les opérations sur les classes : conversion des actions et activités du modèle dynamique, et des processus du modèle fonctionnel en opérations liées aux classes du modèle.
- ✓ Concevoir les algorithmes pour implémenter les opérations.
- ✓ Optimiser de la conception : ajouter des associations redondantes pour minimiser les coûts d'accès, réorganiser les calculs pour une plus grande efficacité, sauvegarder les attributs calculés pour éviter la redondance de calculs complexes.
- ✓ Implémenter du contrôle : Il existe différentes stratégies pour implémenter le contrôle des états dans le modèle dynamique.
- ✓ L'état comme adresse dans un programme.
- ✓ Implémentation directe d'une machine d'états finis.
- ✓ Utilisation des tâches concurrentes : un objet peut être implémenté comme une tâche d'un système d'exploitation ou d'un langage de

programmation. Les événements sont implémentés comme des appels inter-tâches.

- ✓ Ajuster la structure de classes pour accroître l'héritage : extraire des comportements communs, réorganiser les classes et opérations.
- ✓ Concevoir les associations : différentes possibilités d'implémentation d'une association : objet distinct ou ajout d'attributs à une des classes associées ou les deux.
- ✓ Déterminer la représentation des objets : déterminer les types primitifs, ou les nouveaux types par groupement d'objets.
- ✓ Empaqueter les classes et les associations en modules.

- Règles de représentation entre modèle objet et tables :

1. Représentation des classes d'objets en tables :

- ✓ Chaque classe est représentée par une ou plusieurs tables.

2. Représentation des associations en tables :

- ✓ Chaque association M-N est représentée par une table distincte.
- ✓ Une association 1-M est représentée par une table distincte ou peut être enfouie comme clé étrangère dans la table de l'une ou autre classe.
- ✓ Pour les associations 1-M ou 1-1, s'il n'y a pas de cycle, on dispose de l'option supplémentaire qui consiste à ranger l'association et les deux objets liés dans une seule table.
- ✓ Les noms de rôles sont incorporés en tant que partie du nom de l'attribut de la clé étrangère.
- ✓ Les associations n-aires se représentent par des tables distinctes.
- ✓ Une association qualifiée se représente en une table distincte avec au moins trois attributs : la clé primaire de chaque classe liée et le qualificatif
- ✓ Les agrégations suivent les mêmes règles que les associations.

3. Représentation de la généralisation d'héritage simple en tables :

- ✓ La super classe et chaque sous classe se représentent par une table.
- ✓ Pas de table de super classe et les attributs sont dupliqués dans chaque table de sous classe.
- ✓ Pas de tables de sous classent et on exporte tous les attributs des sous classes dans la super classe.

4. Représentation des héritages multiples :

- ✓ Système héritage disjoint, la super classe et les sous classes se représentent par une table.
- ✓ Système héritage par recouvrement, la super classe et chaque sous classe se représentent par une table, la relation de généralisation se représente par une table.

7. Conclusion de chapitre :

Dans ce chapitre nous avons présenté les étapes de la méthode OMT ,dans le 2eme chapitre, nous expliquerons les apports de la notation .

Chapitre II
Intégration d'UML dans OMT

1. Introduction :

UML est une notation unifiée permettant de modéliser un problème d'une façon standard, mais ne définit pas le processus de développement, permet de tirer profit des étapes de la méthode tout en respectant le standard de la notation UML.

Une intégration de cette notation dans la méthode OMT (c'est-à-dire suivi des étapes de la méthode OMT en utilisant les notations d'UML), comme UML 2.0 a été pensée pour pouvoir être utilisée avec n'importe quelle méthode objet, la combinaison des deux termes nous permet de construire des modèles lisibles, conviviaux et complets.

2. Apport d'UML a OMT :

2.1. La vue des besoins des utilisateurs :

- ✓ La démarche est pilotée par les besoins des utilisateurs : avec UML ce sont les utilisateurs qui guident la définition des modèles :
  - Les utilisateurs définissent ce que doit être le système.
  - Le but du système à modéliser est de répondre aux besoins de ses utilisateurs.
- ✓ Les besoins des utilisateurs servent de fil rouge : tout au long de cycle de développement (itératif et incrémentales) :
  - A chaque itération de la phase d'analyse, on clarifie, affine et valide les besoins des utilisateurs.
  - A chaque itération de la phase de conception et de réalisation on veille a la prise en compte des besoins des utilisateurs.
  - A chaque itération de la phase de test, on va vérifie que les besoins des utilisateurs sont satisfaits.

2.2. La vue des processus : cette vue est très important dans les environnements multitâches, elle montre :

- ✓ La décomposition en termes de processus.
- ✓ Les interactions entre les processus.
- ✓ La synchronisation et la communication des activités parallèles.

2.3. La vue de déploiement : cette vue est très important dans les environnements distribués, décrit les ressources matérielles et la réception dans logiciel dans ces ressources :

- ✓ La disposition et nature physique des matérielles, ainsi que leurs performances.
- ✓ L'implantation des modules principaux sur les nœuds du réseau.
- ✓ Les exigences en termes de performances.

3. Etude comparative entre la méthode OMT et le langage UML :

3.1. Analyse des concepts communs :

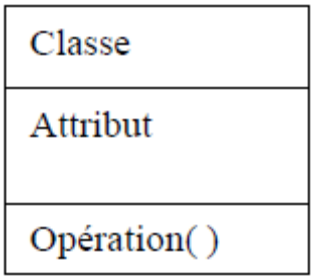
3.1.1. Démarche :

- ✓ Les deux méthodes sont basées sur trois grandes étapes.
  - Analyse
  - Conception
  - Implémentation

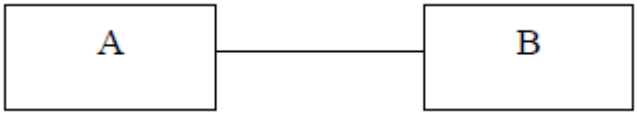
- ✓ Les deux méthodes englobent exactement le même domaine de développement et d'abstraction.

3.1.2. Le formalisme :

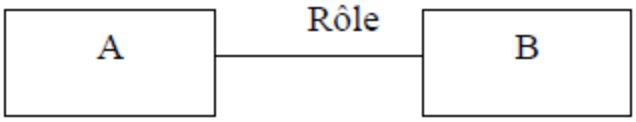
- ✓ Les classes :



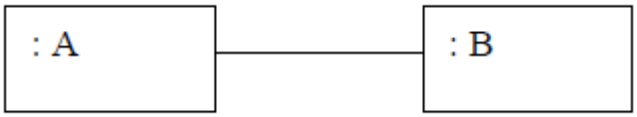
- ✓ Associations :



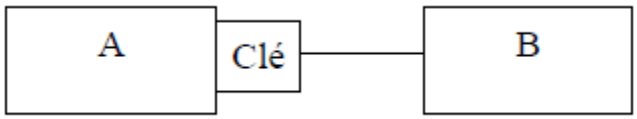
- ✓ Rôle :



- ✓ Les liens :



- ✓ Qualification :



3.2.Analyse des différences :

Le diagramme de déploiement chez UML est une description de l'implémentation physique du système qui n'a pas d'équivalent chez OMT.

3.2.1. La démarche de conception :

OMT	UML	Observation
	Diagramme de classes	Structure statique de classes et

Modèle Statique		dimensions
	Diagramme d'objets	Objets et relations
	Diagramme de composant	Composants physique
	Diagramme de déploiement	Implémentation physique des composants
Modèle dynamique	Diagramme d'activités	Comportements d'une opération sous formes d'actions
	Diagramme de séquence	Chronologie des objets et des interactions
	Diagramme Etat-transition	Comportements d'une classe à partir ses états possibles
	Diagramme de communication	Représentation spatiale des objets, des liens est des opérations
Modèle fonctionnel	Diagramme des cas d'utilisation	Définis les besoins des utilisateurs

3.2.2. Le formalisme :

✓ Les objets :

OMT	UML
Nom : Classe	Nom : Classe

✓ Classe abstraite :

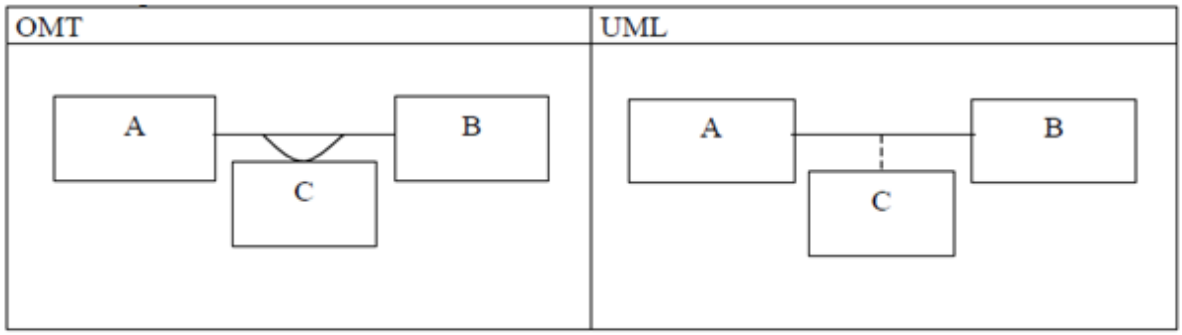
En OMT une classe abstraite est repérée par la valeur 0, en UML le nom de la classe est en italique

OMT	UML
Classe abstraite 0	*Classe abstraite*

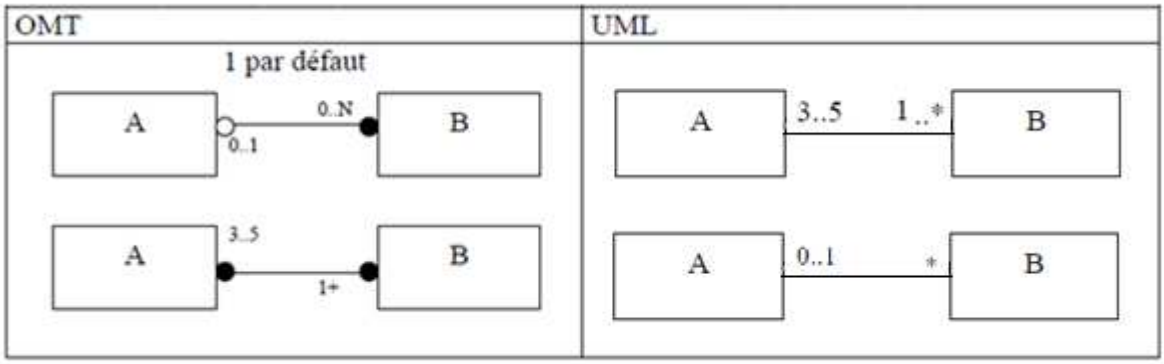
✓ Méta classe :

OMT	UML
Métaclasse {Métaclasse}	<<Métaclasse>> Métaclasse

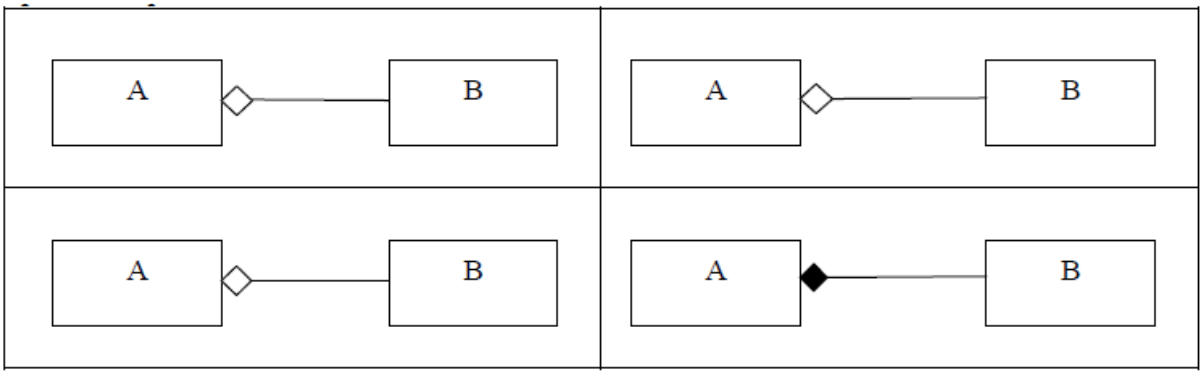
✓ Classe-association :



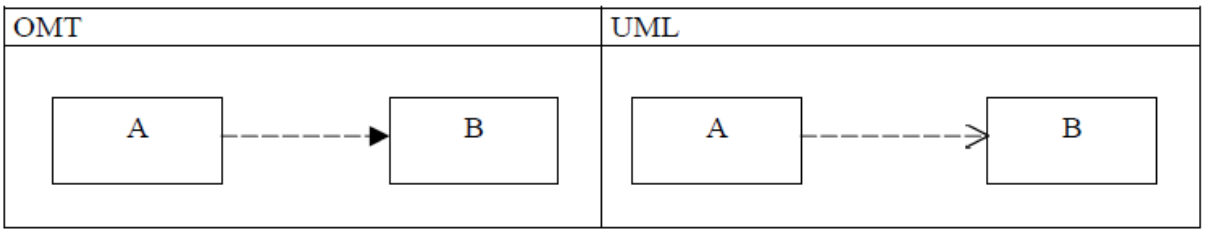
✓ Multiplicité :



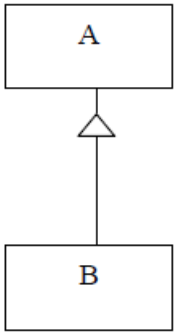
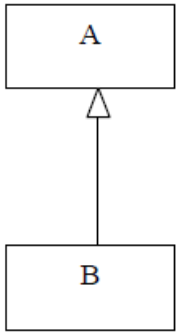
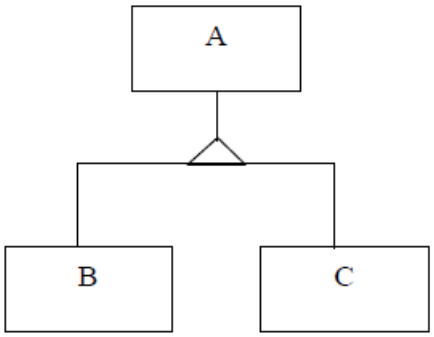
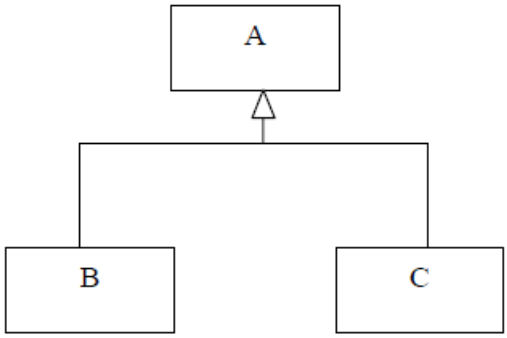
✓ Agrégation :



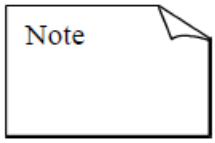
✓ Dépendance :



✓ Héritage :

OMT	UML
	
	

✓ Note :

OMT	UML
Texte entre accolades {Note}	

4. Les diagrammes dans OMT:

4.1.Le modèle objet : dans ce modèle, UML utilise deux diagrammes pour représenter l'aspect statique du système :

- Le diagramme d'instance (diagramme objet en UML).
- Le diagramme de classe : c'est le même que celui d'UML.

4.2.Le modèle dynamique : dans ce modèle, UML utilise aussi deux plus diagrammes pour représenter l'aspect dynamique du système :

- Le diagramme de trace d'éventement : le diagramme de séquence d'UML est une version plus complète de diagrammes de trace d'éventement d'OMT.
- Le diagramme d'état.

4.3.Le modèle fonctionnel : dans ce modèle, OMT utilise des diagrammes de flots de donnees.qui ont la même sémantique que les diagrammes d'activités en UML.

Chapitre III
Etude de cas : gestion d'une grossisterie

1. Introduction :

Mon étude de cas consiste à développer une application pour la gestion commerciale d'un magasin de vente en gros.

Le système que je compte développer a pour but d'atteindre les objectifs suivants :

- ✓ La réception des marchandises avec mise à jour du stock.
- ✓ La gestion des informations concernant les clients.
- ✓ La gestion des ventes avec mise à jour du stock.
- ✓ La gestion des informations concernant les produits.
- ✓ Etablissement de statistiques.

a. La réception des marchandises avec mise à jour du stock :

Cette fonctionnalité permet la réception des marchandises d'une livraison d'un fournisseur.

Le grossiste peut ajouter un nouveau fournisseur, nouveau produit,
Il peut gérer les informations des deux, il peut enregistrer un bon de livraison et il peut imprimer un bon de réception.

Après la réception le système mettre à jour le stock des produits livrés.

b. La gestion des ventes avec mise à jour du stock :

Cette fonctionnalité permet la satisfaction d'une commande disponible d'un client de magasin.

Le grossiste peut ajouter on nouveau client, il peut gérer les informations concernant le client, il peut enregistrer le bon de commande.
Il peut enregistrer la vente en plus il peut imprimer une facture.

c. La gestion des informations concernant les clients :

Cette fonctionnalité permet au grossiste de gérer les informations concernant le client.

Le grossiste peut supprimer, modifier ou ajouter un nouveau client.

d. La gestion des informations concernant les produits :

Cette fonctionnalité permet au grossiste de gérer les informations concernant le produit.

Le grossiste peut supprimer, modifier ou ajouter un nouveau produit.

e. Etablissement de statistiques :

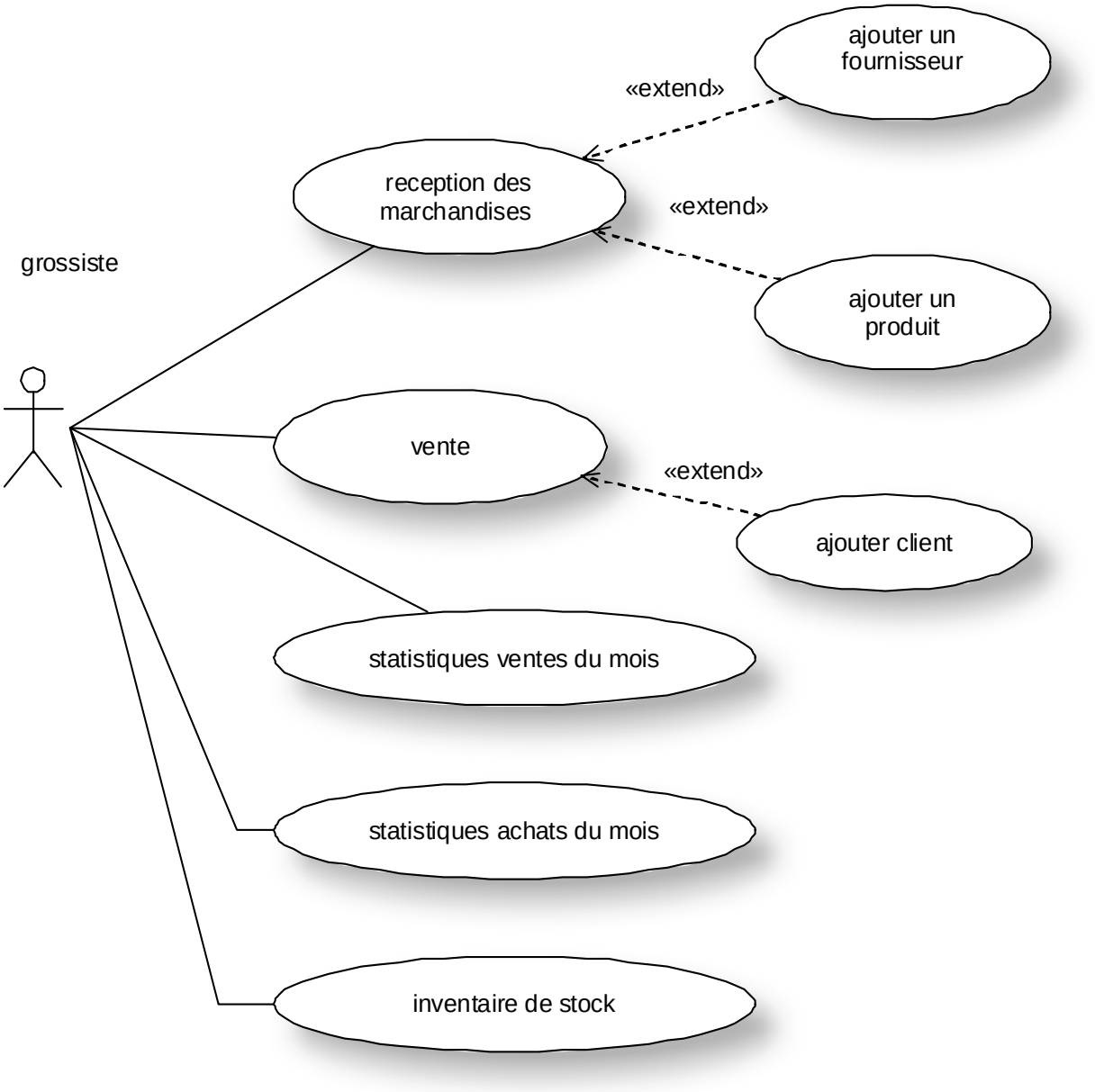
Cette fonctionnalité permet au grossiste de consulter les différentes statistiques concernant les achats et les ventes du mois.

Le grossiste peut consulter les statistiques achats du mois, les statistiques du mois et il peut consulter l'inventaire de stock.

2. Intégration UML/OMT :

2.1.Le modèle fonctionnel :

2.1.1. Identification de diagramme des cas d'utilisation :



2.1.2. Les fiches descriptives :

2.1.2.1. Réception des marchandises :

Réception des marchandises
Permet la réception des marchandises et les ajouts dans le stock
Acteur principal : grossiste
En début : Saisir Numéro et Date du Bon de livraison. En cours : Si NVproduit alors Ajouter un fournisseur Finsi Pour chaque produit faire Saisir code produit Si NVproduit alors Ajouter un produit Finsi Mettre à jour le stock Finpour Enregistrer un Bon de livraison imprimer un Bon de réception. En fin : produits ajoutés. Bon de réception imprimé

2.1.2.2. Ajouter un fournisseur :

Ajouter fournisseur
Permet au grossiste d'enregistrer un nouveau fournisseur
Acteur principal : grossiste
En début : lire le code de fournisseur En cours : le système vérifie l'existence du code Si existe alors Afficher message « code existant » Sinon Lire les informations du fournisseur ajouter fournisseur Finsi En fin : afficher " fournisseur ajouté "

2.1.2.3. Ajouter un produit :

Ajouter produit
Permet au grossiste d'enregistrer un nouveau produit
Acteur principal : grossiste
En début : lire le code de produit En cours : le système vérifie l'existence du code Si existe alors Afficher message « code existant » Sinon Lire les informations du produit ajouter produit Finsi En fin : afficher " produit ajouté "

2.1.2.4. Vente :

vente
Ce cas traite l'opération de vente
Acteur principal : grossiste
En début : lire BC En fin : Le système vérifie l'existence du client Si NV client alors Ajouter clients Finsi Pour chaque article de la commande faire Lire (désignation, quantité commandé) vérifie l'existence de l'article Si article existe alors Afficher les autres informations de l'article Vérifier la disponibilité de l'article Si (qté.commandé <=qté.stock) alors Ajouter au panier Sinon Afficher (message d'erreur "quantité insuffisante ") Finsi Sinon Afficher (message d'erreur "article inexistant") Finsi Finpour

Enregistrer BC Enregistrer vente (inclure : établir facture et mettre a jour le stock) En fin Imprimer facture

2.1.2.5. Ajouter client :

Ajouter client
Permet au grossiste d'enregistrer un nouveau client
Acteur principal : grossiste
En début : lire le code de client En cours : le système vérifie l'existence du code Si existe alors Afficher message « code existant » Sinon Lire les informations du client ajouter client Finsi En fin : afficher " client ajouté "

2.1.2.6. Statistique achats du mois :

statistiques achats de mois
Permet de calculer et éditer les statistiques d'achats de mois
Acteur principal : grossiste
En début : demander les statiques achats de mois En cours : sélectionner la période Choisir si un produit ou tous Si tous_prod alors Calculer achats mois pour tous les produits Afficher les informations de tous produits achetés durant cette période sinon Calculer achats mois pour le produit sélectionné Afficher les informations de produit acheté durant ce période Finsi En fin : statistique achat éditée ou (sur imprimante ou sur écran)

2.1.2.7. Statistique ventes du mois :

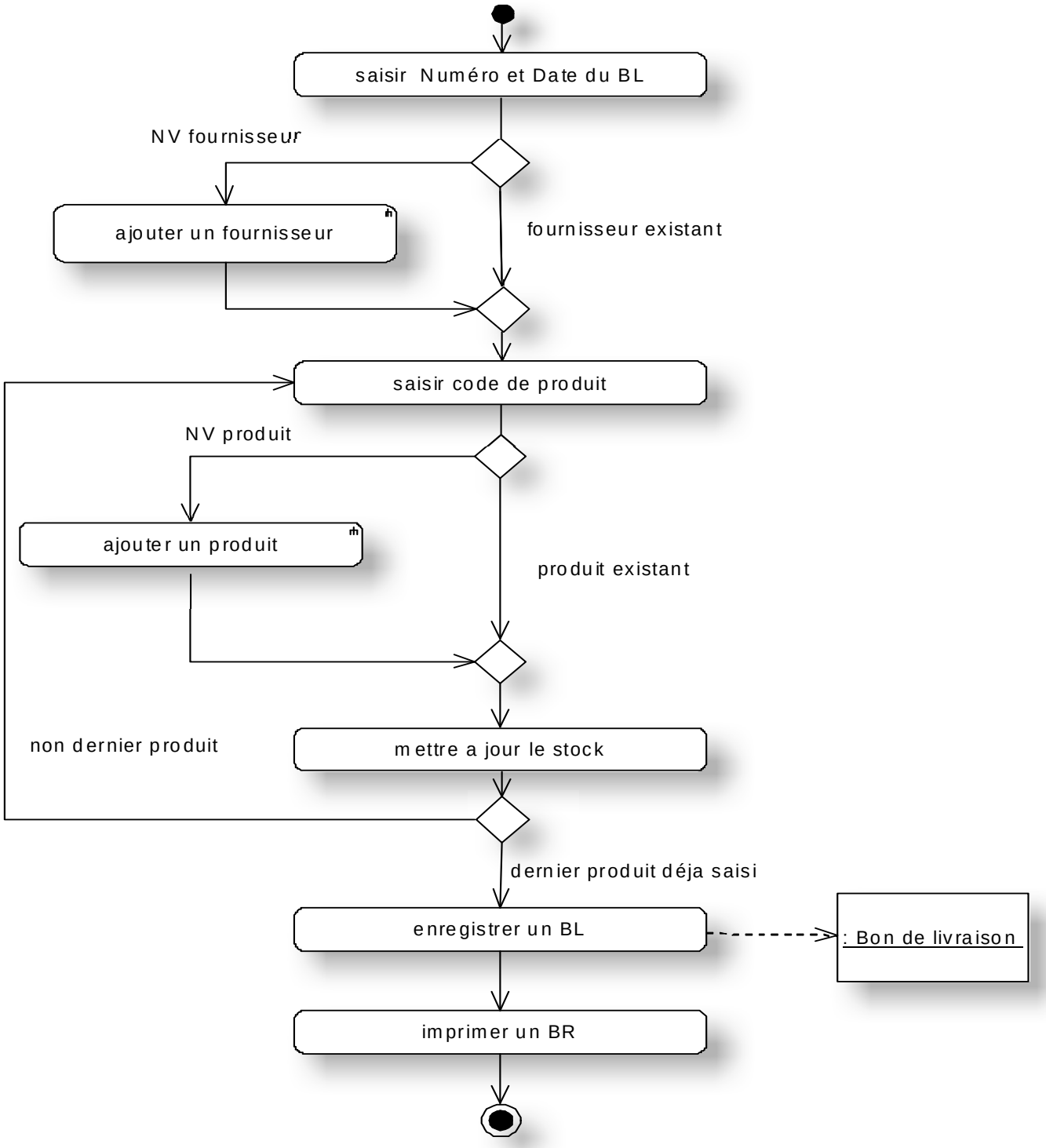
statistiques ventes de mois
Permet de calculer et éditer les statistiques de ventes de mois
Acteur principal : grossiste
En début : demander les statiques ventes de mois En cours : sélectionner la période Choisir si un produit ou tous Si tous_prod alors Calculer ventes mois pour tous les produits Afficher les informations de tous produits ventés durant ce période sinon Calculer ventes mois pour le produit sélectionné Afficher les informations de produit venté durant cette période Finsi En fin : liste affiché

2.1.2.8. Inventaire de stock :

Inventaire stock
Permet d'afficher l'inventaire de stock
Acteur principal : grossiste
En début : demander l'inventaire de stock En cours : afficher les informations de tous les articles En fin : inventaire édité

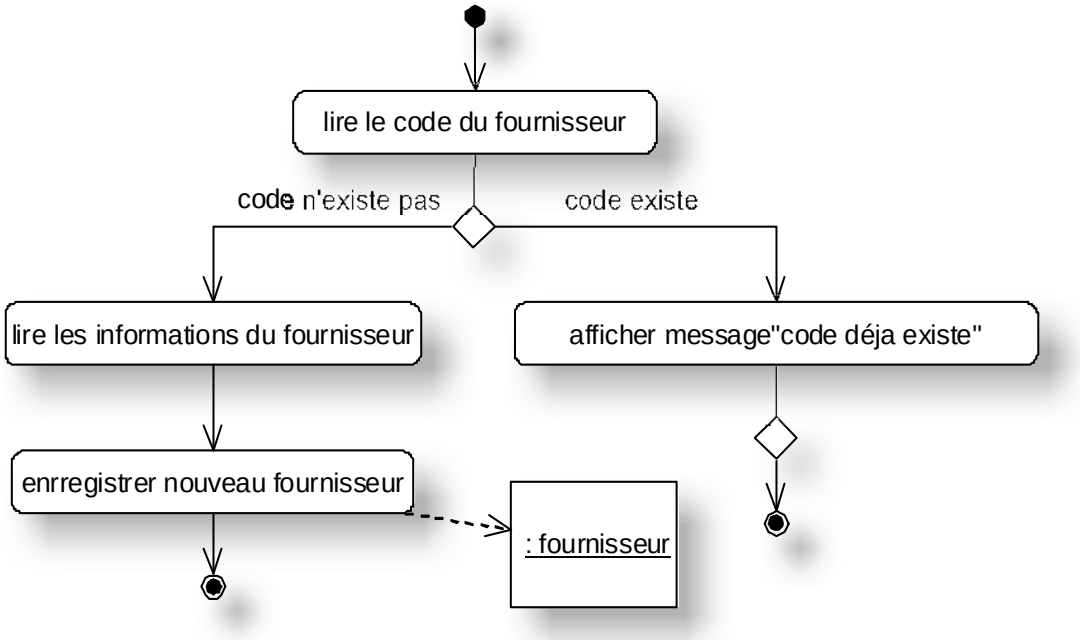
2.1.3. Diagrammes d'activités :

2.1.3.1. Réception des marchandises :



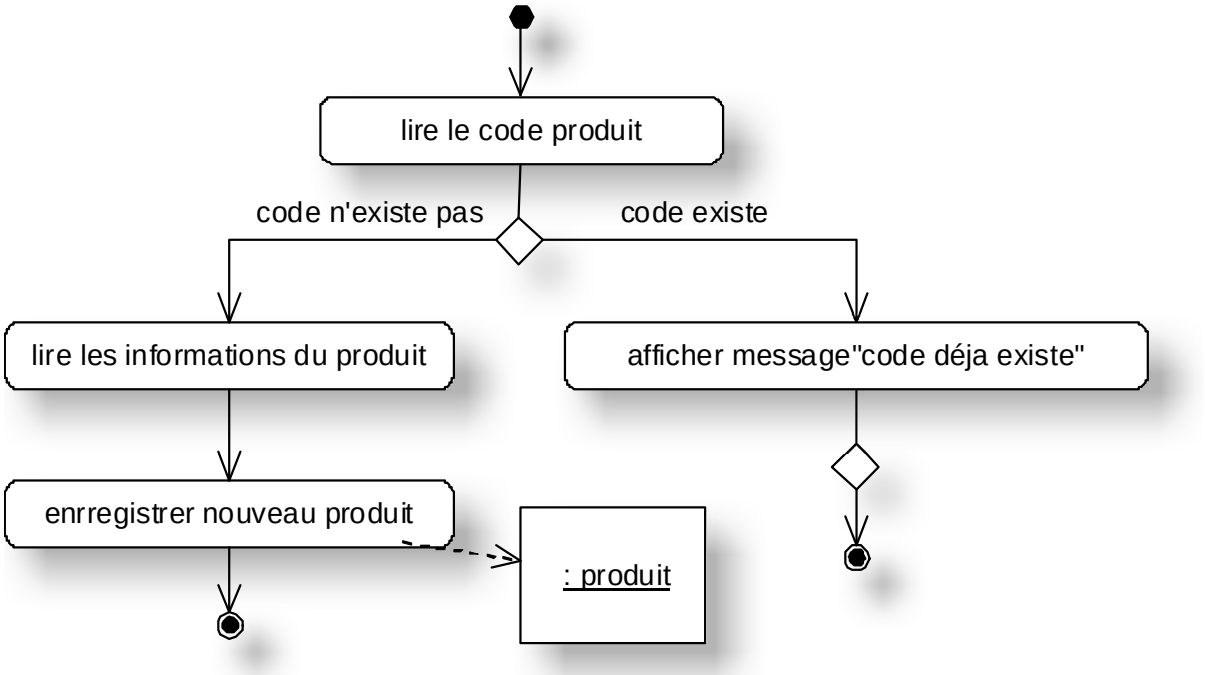
- On note l'identification d'une classe métier appelée « Bon de livraison ».

2.1.3.2. Ajouter un fournisseur :



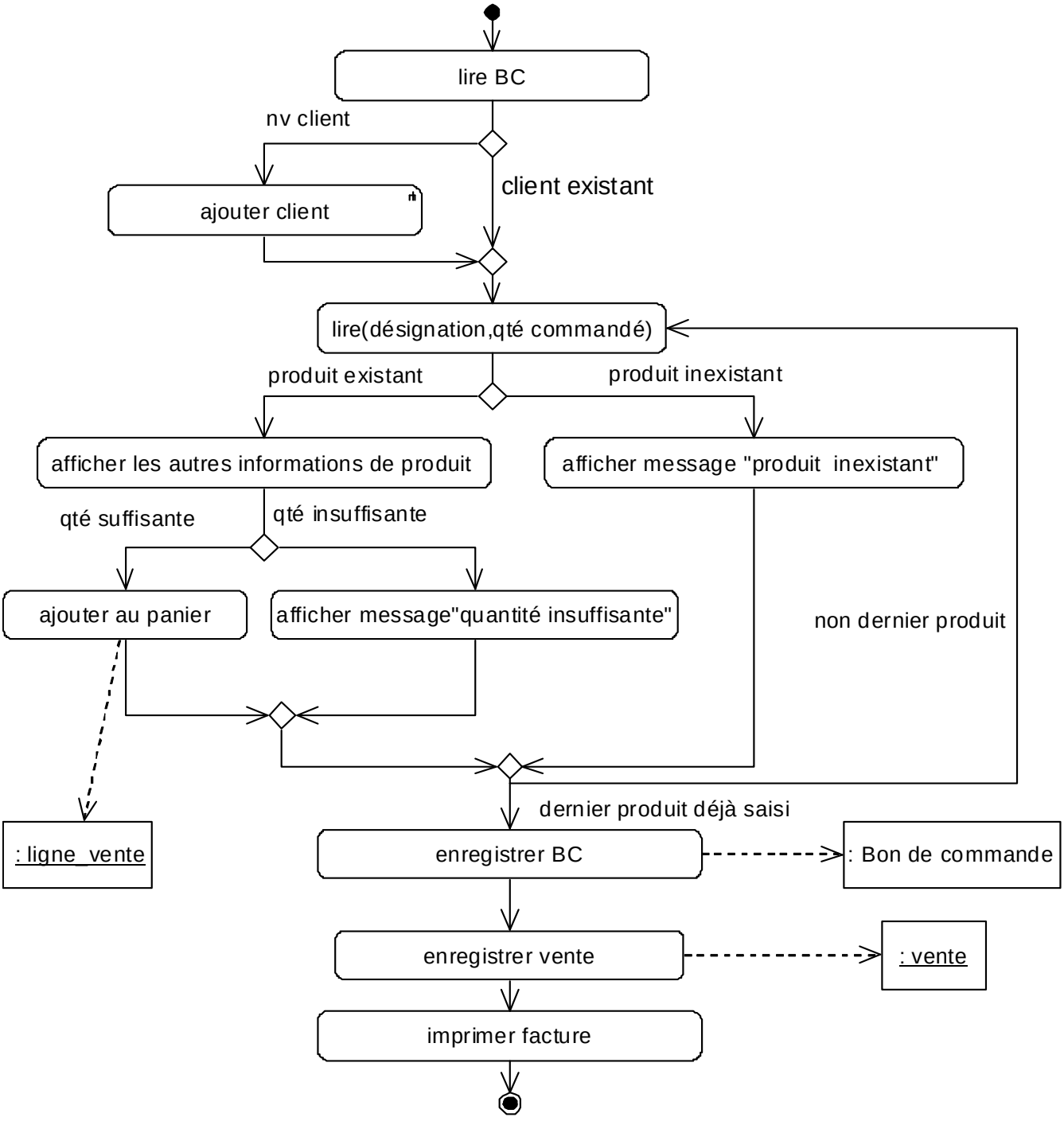
- On note l'identification d'une classe métier appelée « fournisseur ».

2.1.3.3. Ajouter un produit :



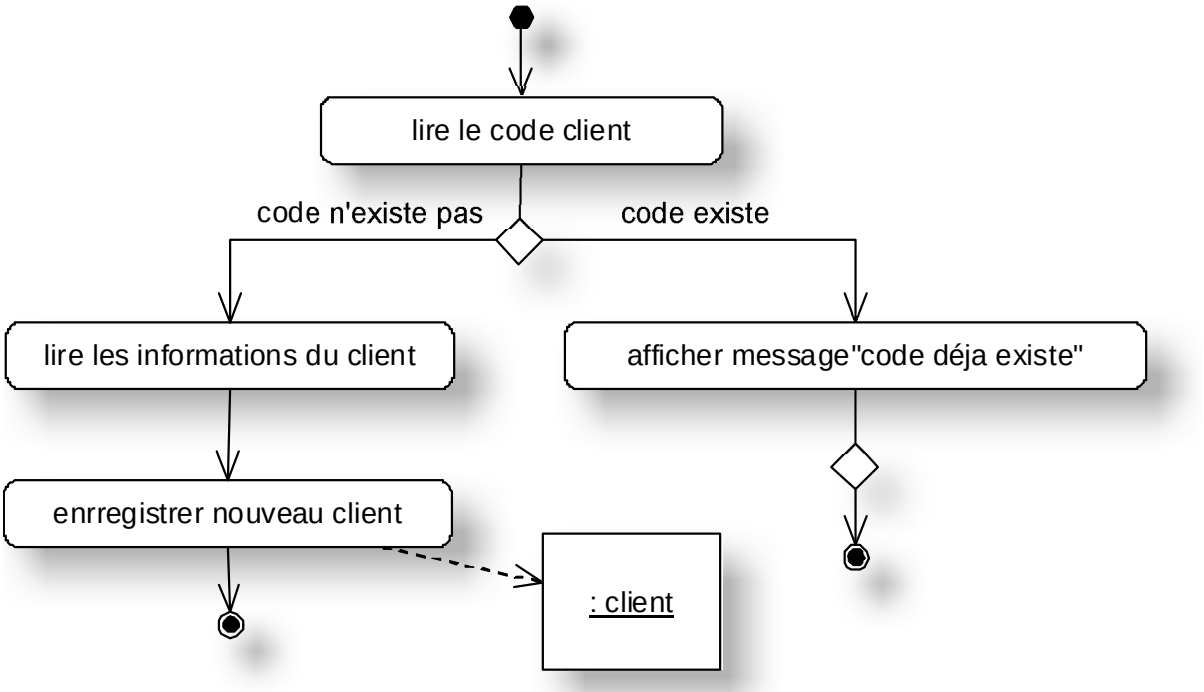
- On note l'identification d'une classe métier appelée « produit ».

2.1.3.4. Vente :



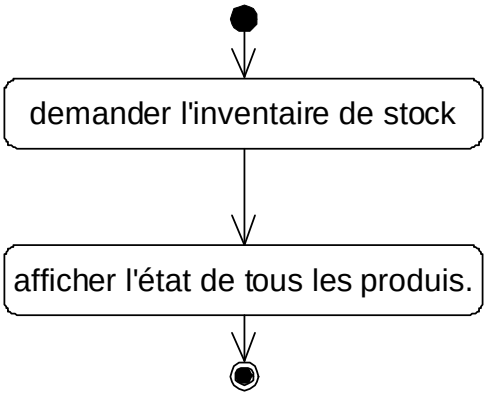
- On note l'identification d'une classe métier appelée « Bon de commande ».
- On note l'identification d'une classe métier appelée « vente ».
- On note l'identification d'une classe association appelée « ligne_vente ».

2.1.3.5. Ajouter client :

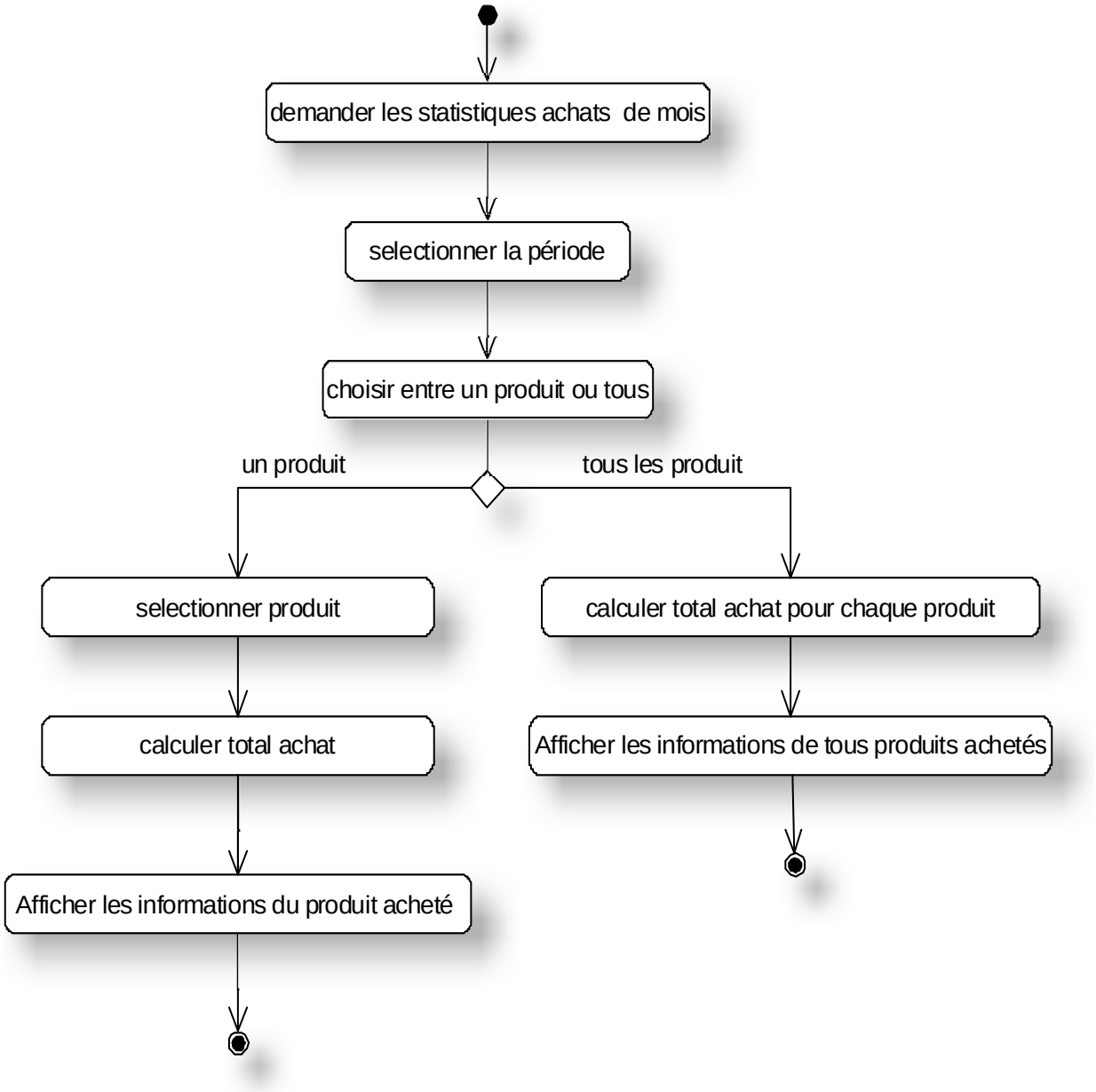


- On note l'identification d'une classe métier appelée « client ».

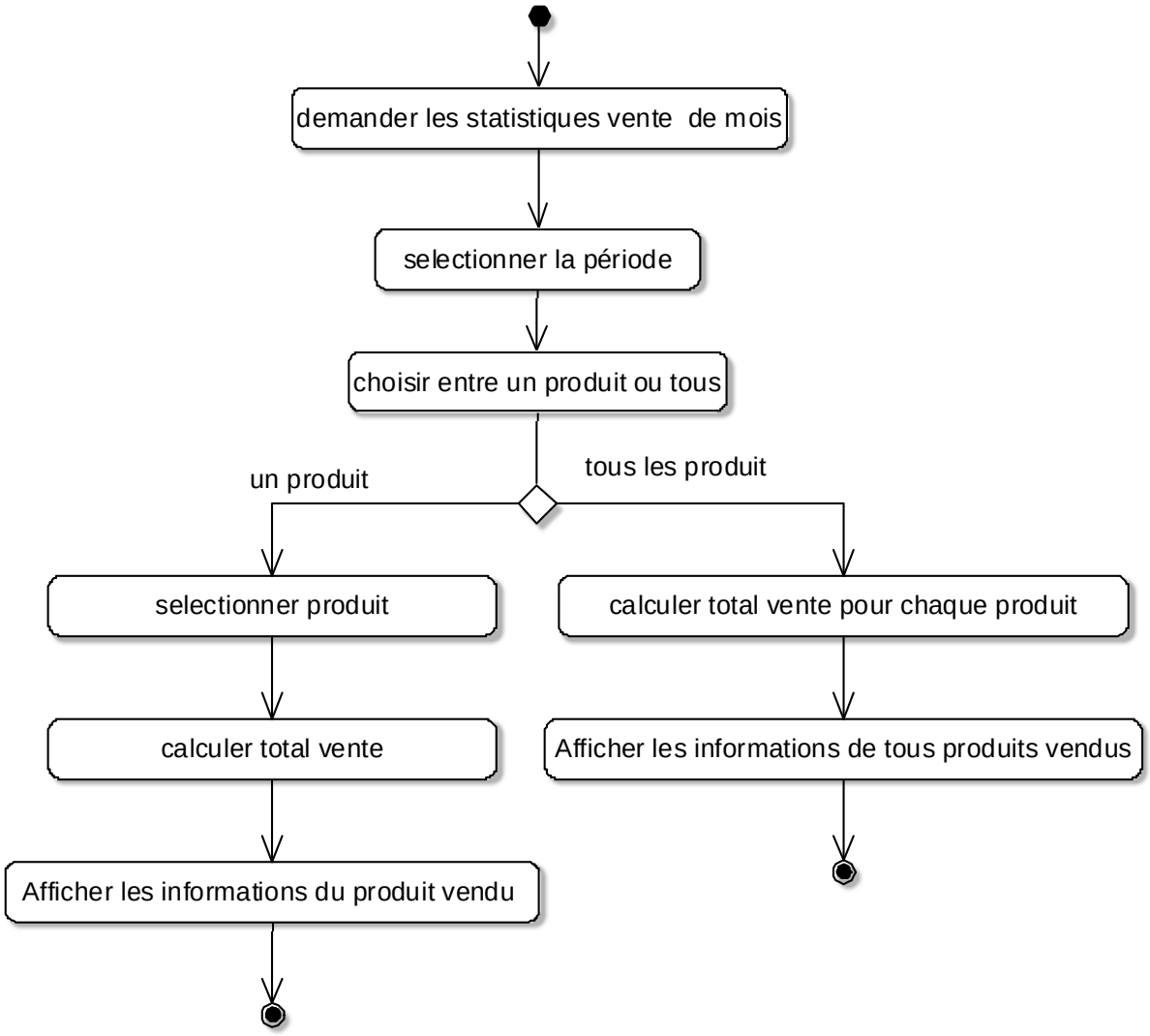
2.1.3.6. Inventaire de stock :



2.1.3.7. Statistique achats du mois :


demander les statistiques achats de mois
selectionner la période
choisir entre un produit ou tous
un produit
tous les produit
selectionner produit
calculer total achat
Afficher les informations du produit acheté
calculer total achat pour chaque produit
Afficher les informations de tous produits achetés

2.1.3.8. Statistique ventes du mois :

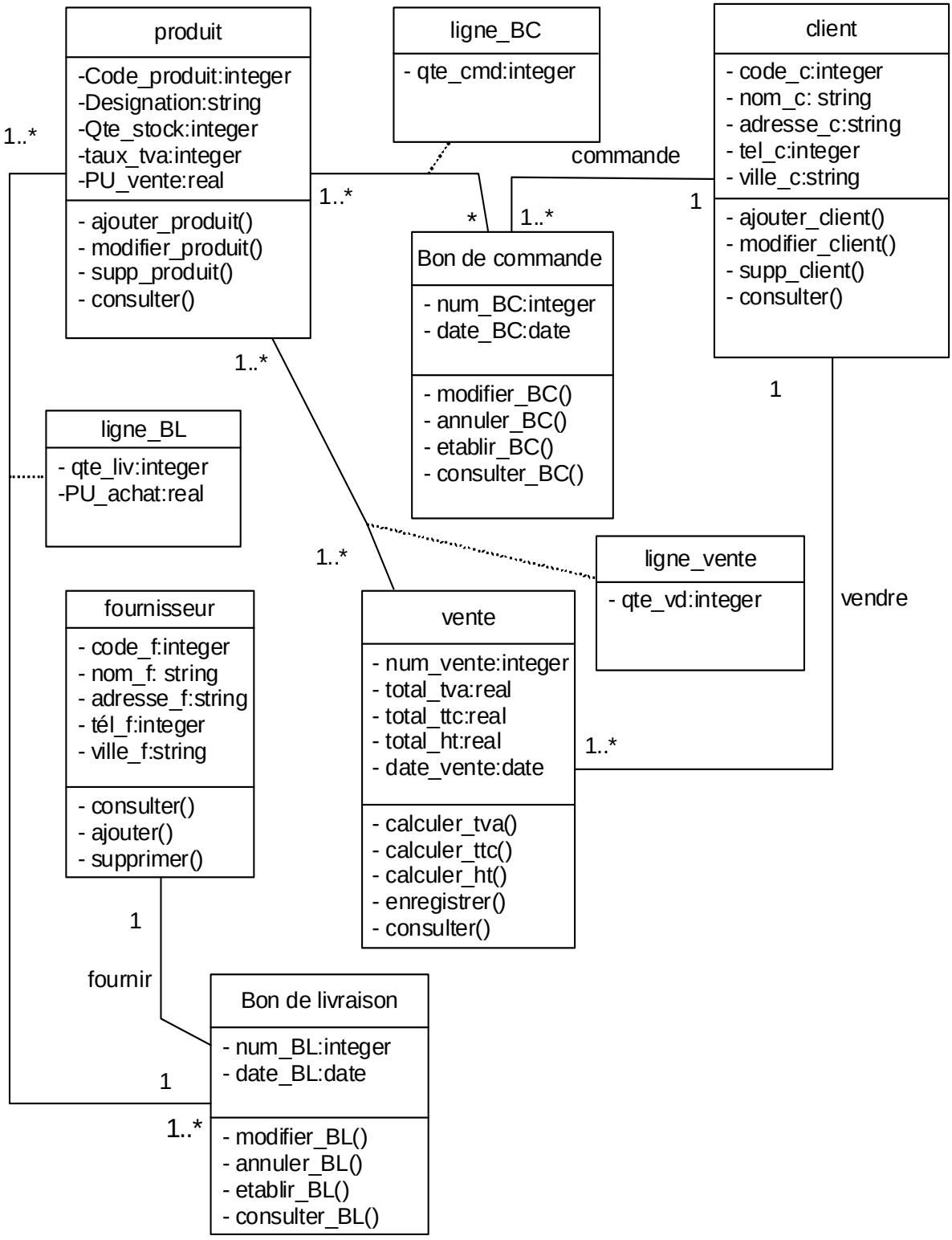


2.2.Le modèle objet :

2.2.1. Les règles de gestion :

- Un produit est livré plusieurs fois.
- Un BL contient plusieurs produits.
- Un produit est vendu plusieurs fois.
- Une vente concerne plusieurs produits.
- Un client effectue plusieurs ventes.
- Une vente est effectuée par un seul client.
- Un produit est commandé plusieurs fois.
- Un BC contient plusieurs produits.
- Un client établit plusieurs BC.
- Un BC est établi par un seul client.
- Un fournisseur établit plusieurs BL.
- Un BL est établi par un seul fournisseur.

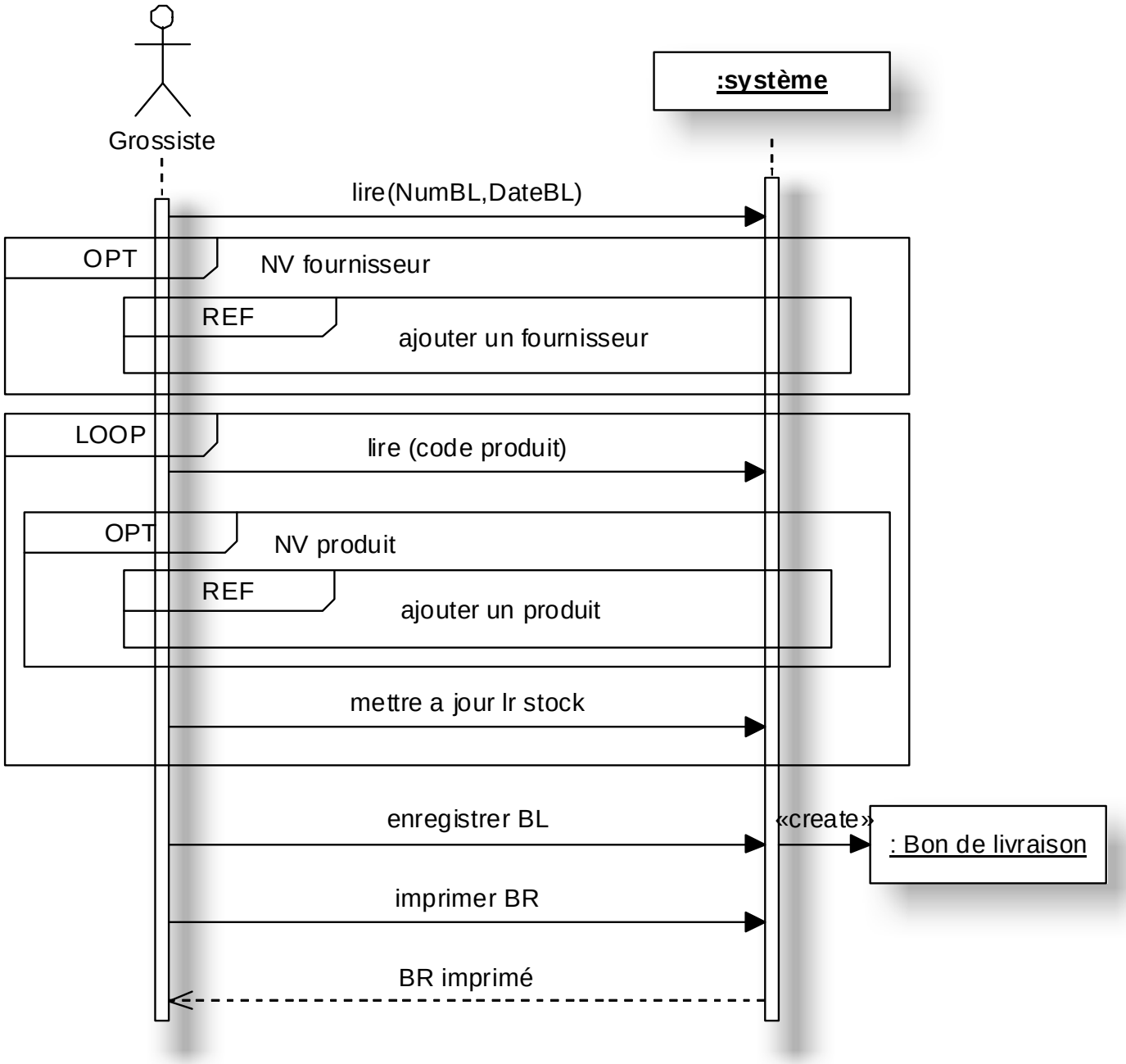
2.2.2. Le diagramme de classe métier :



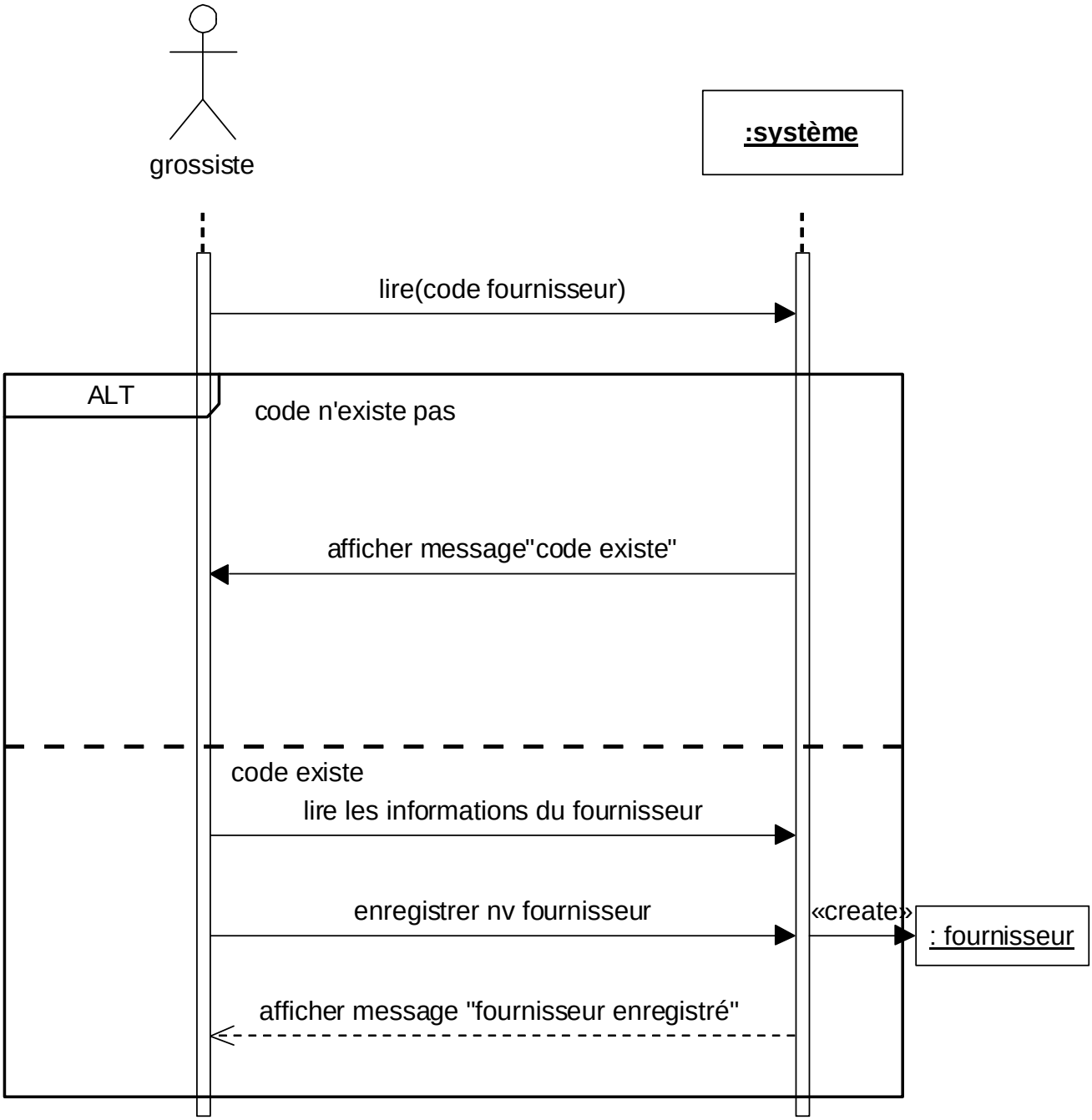
2.3. Le modèle dynamique :

2.3.1. Les diagrammes de séquence :

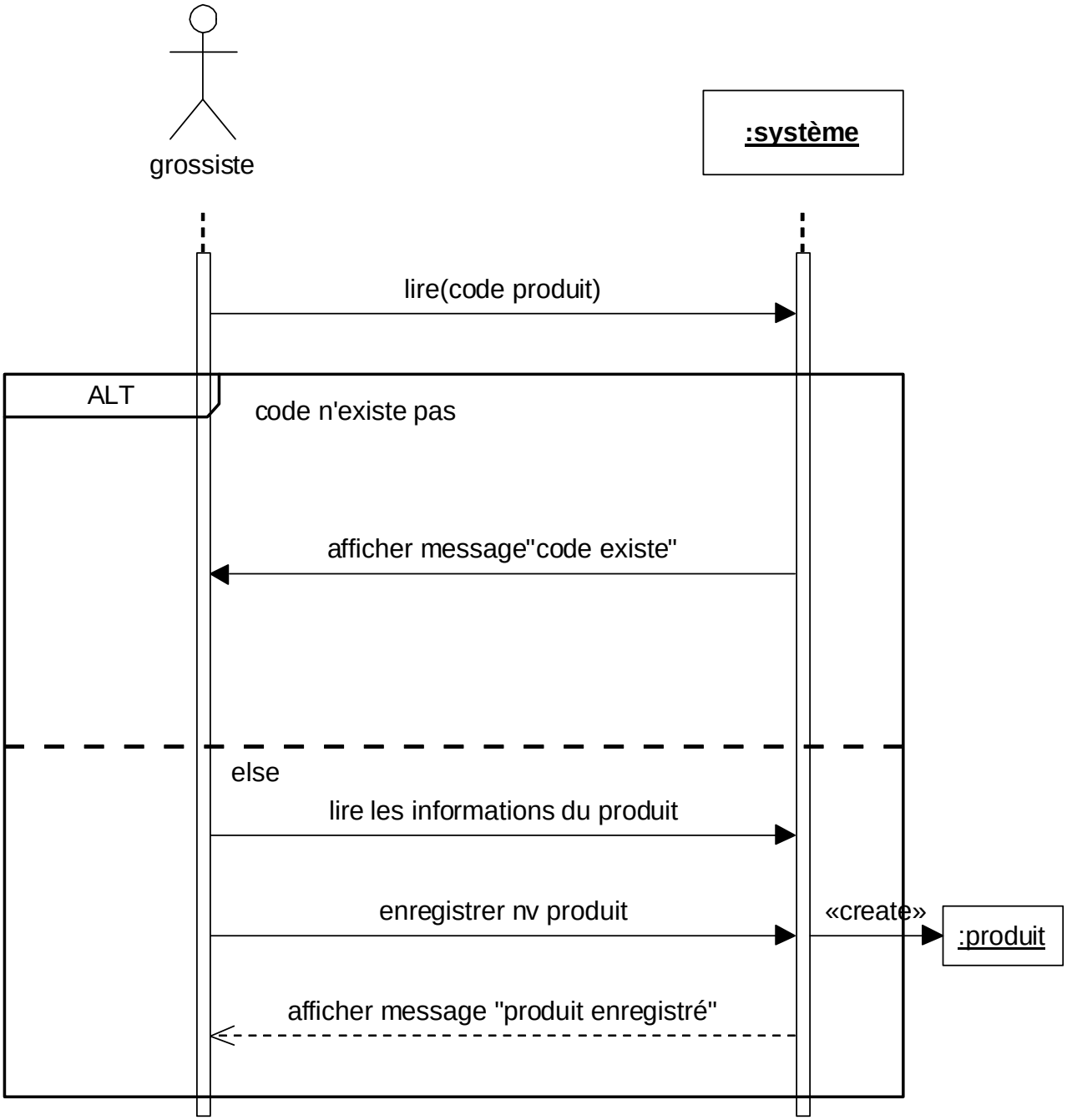
2.3.1.1. Réception des marchandises :



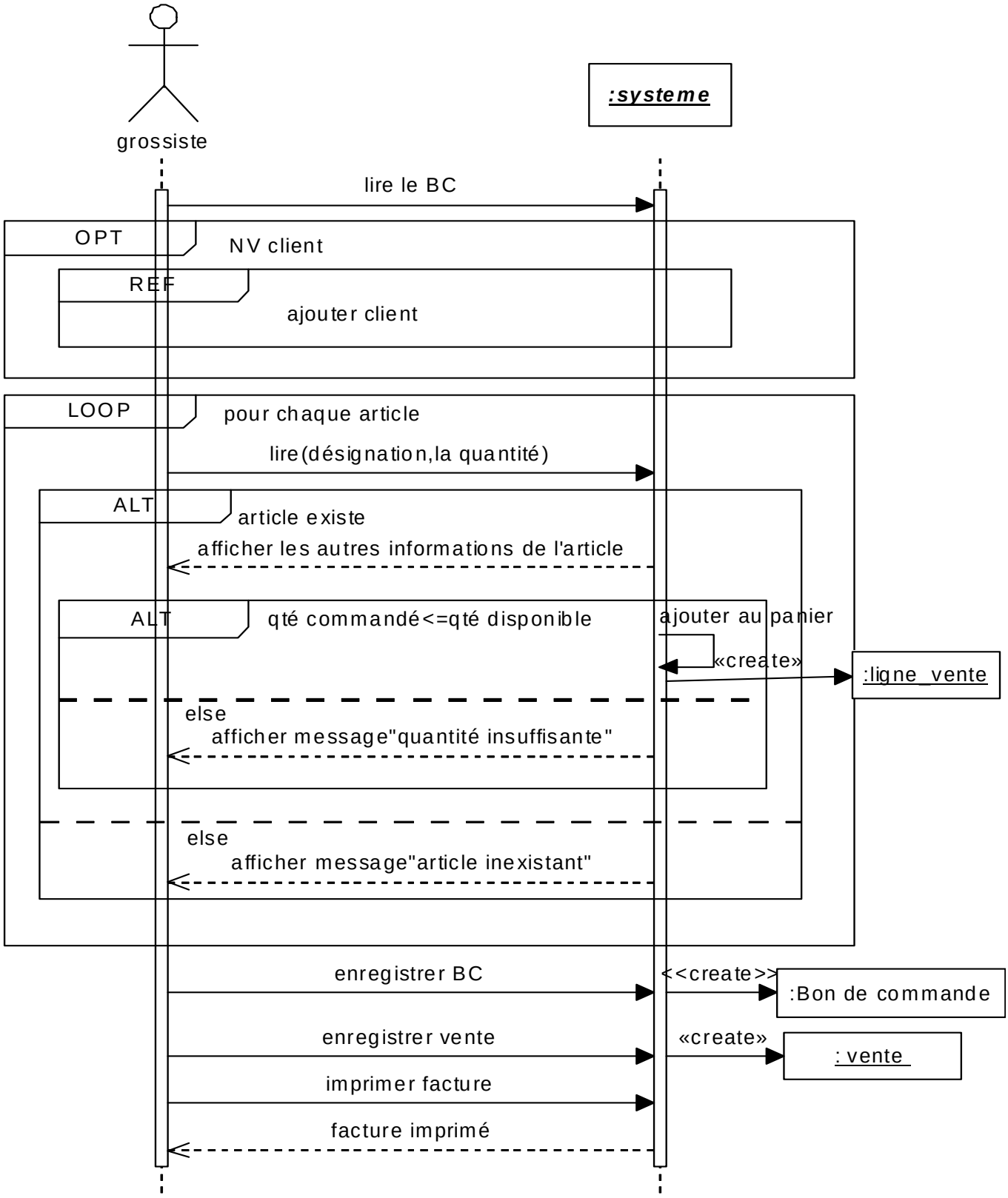
2.3.1.2. Ajouter un fournisseur :



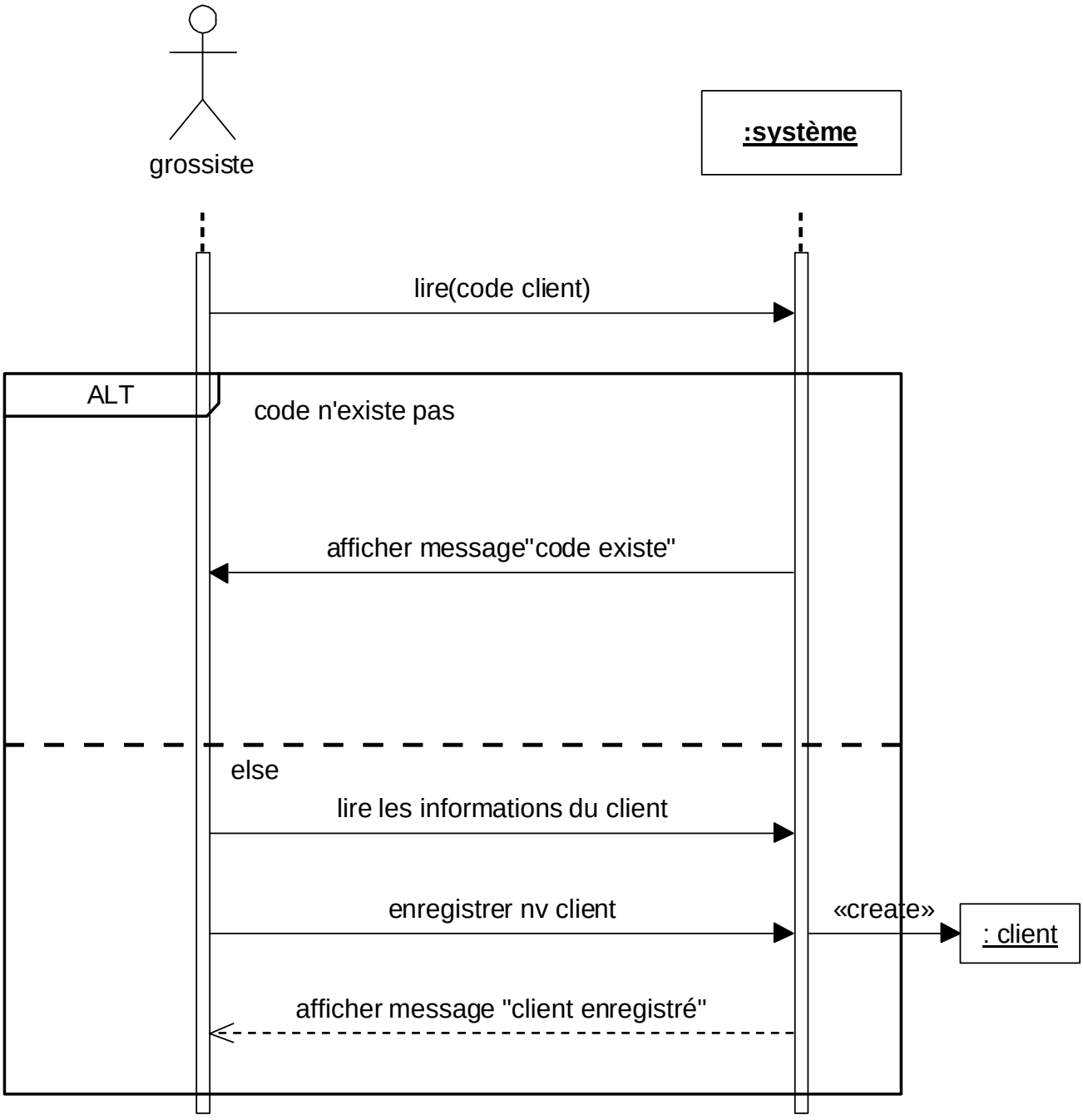
2.3.1.3. Ajouter un produit :



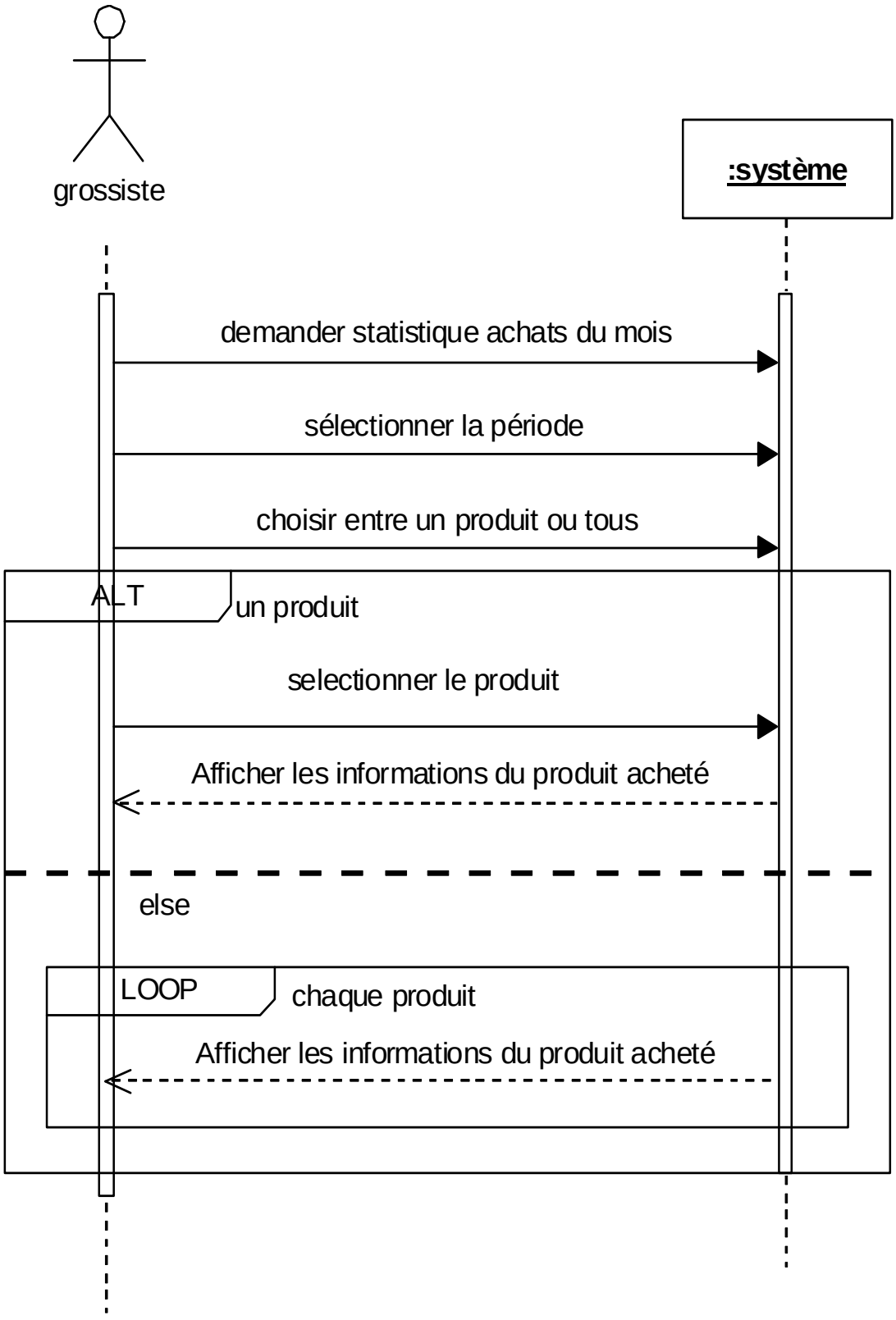
2.3.1.4. Vente :


grossiste
:systeme
lire le BC
OPT
NV client
REF
ajouter client
LOOP
pour chaque article
lire(désignation,la quantité)
ALT
article existe
afficher les autres informations de l'article
ALT
qté commandé<=qté disponible
ajouter au panier
«create»
:ligne_vente
else
afficher message"quantité insuffisante"
else
afficher message"article inexistant"
enregistrer BC
<<create>>
:Bon de commande
enregistrer vente
«create»
: vente
imprimer facture
facture imprimé

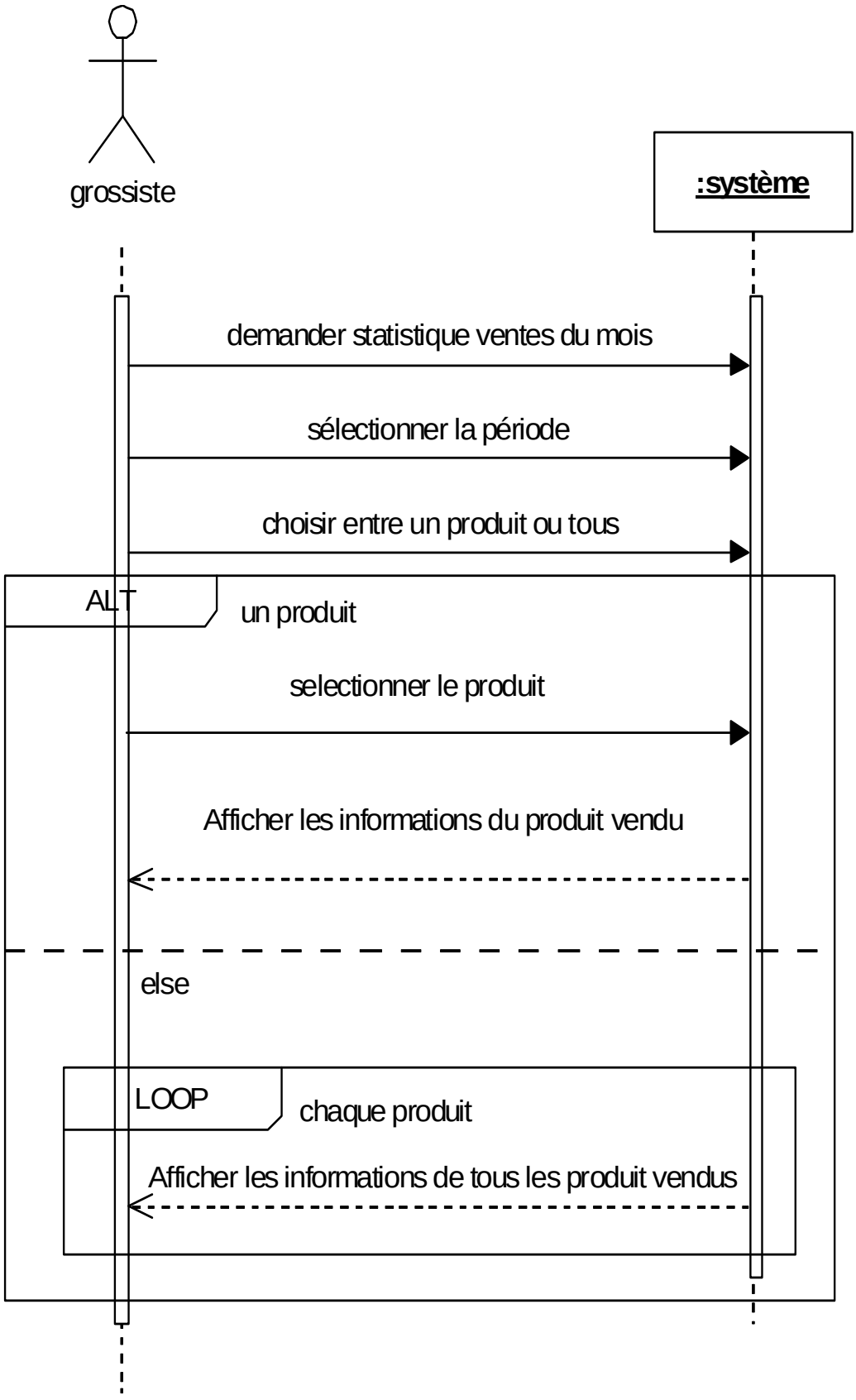
2.3.1.5. Ajouter client :



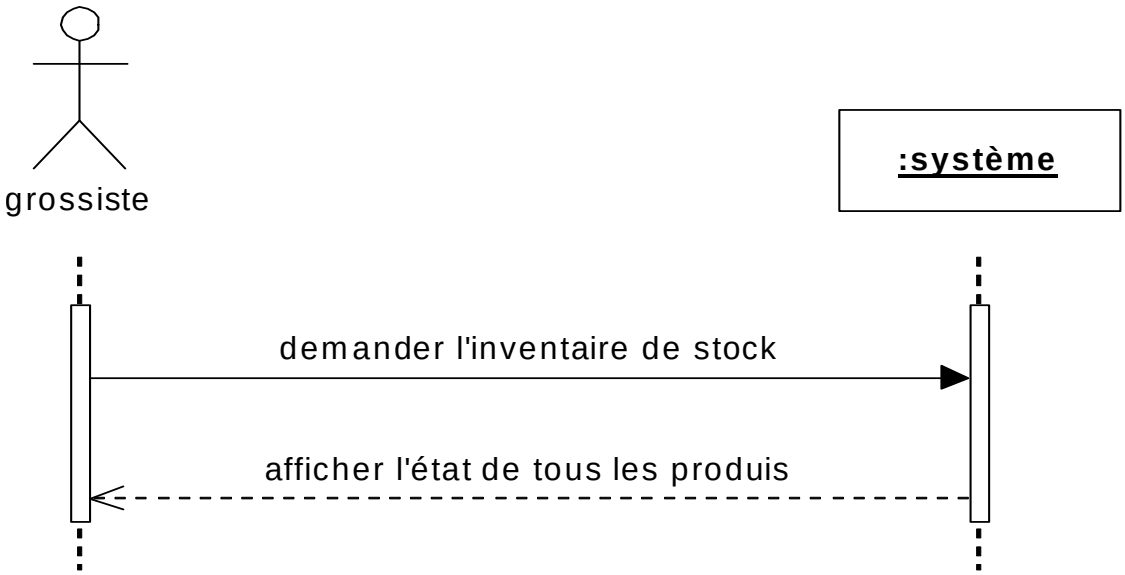
2.3.1.6. Statistique achats du mois :



2.3.1.7. Statistique ventes du mois :



2.3.1.8. Inventaire de stock :



Chapitre IV
Implémentation

1. Introduction :

Ce chapitre sera consacré à l'implémentation de notre système de gestion
Du magasin de vente en gros.

Dans un premier temps, la base de données relationnelle de notre système sera déduite depuis le diagramme de classe de conception abordé précédemment. par la suite, on présentera une brève description des différentes outils utilise pour la réalisation de notre application, à savoir de langage de programmation visuel basic 6, Access 2007.

Nous terminons ce chapitre par la présentation de quelques écrans de notre application.

2. Modèle relationnel :

2.1. Le passage du diagramme de classe au modèle relationnel :

Afin de pouvoir implémenter une base de données, il faut pouvoir traduire le modèle conceptuel en modèle logique. Cela signifie qu'il faut pouvoir convertir un modèle UML en modèle relationnel.

Le passage du modèle conceptuel au modèle relationnel est systématique en appliquant des règles de transformation.

2.2.Règles de passage :

- Transformation des entités /classes:

  Chaque classe devient une relation, l'identifiant de l'entité devient clé primaire pour la relation.

- Transformation des associations : Nous distinguons trois familles d'association :

  ✓ **Association 1..* :** la règle est la suivante :

  Il faut ajouter un attribut de type clé étrangère dans la relation fils de l'association.

  L'attribut porte le nom de la clé primaire de la relation père de l'association.

  ✓ **Association *..*:** la règle est la suivante :

  Association/classe – association devient une relation. La clé primaire de cette relation est la concaténation des identifiants des identités connecté à l'association. Chaque attribut devient clé étrangère si entité/classe connectée dont il devient une relation en vertu de la règle R1.

  Les attributs d'association/classe – association doivent être ajoutés a la nouvelle relation.ces attribut ne sont ni clé primaire, ni clé étrangère.

  ✓ **Association 1..1:** la règle est la suivante :

  Il faut ajouter un attribut de type clé étrangère dans la relation dérivée de l'entité ayant la cardinalité minimale égale à zéro.

  Dans le cas de diagramme UML il faut ajouter un attribut de type clé étrangère dans la relation dérivée de la classe ayant la multiplicité minimale égale à un.

2.3.Les tables de la base de données :

produit (code_produit , designation , PU_vente , qte_stock , taux_TVA)

client (code_c , nom_c , adresse_c , tel_c , ville_c)

fournisseur (code_f , nom_f, adresse_f , tel_f , ville_f)

vente (num_vente , taux_TVA , taux_HT , taux_TTC , date_vente , code_c)

commande(num_BC , date_BC , code_c)

livraison(num_BL , date_BL , code_f)

ligne_BL(code_produit , num_BL , qte_liv , PU_achat)

ligne_BC(code_produit , num_BC , qte_cmd)

ligne_vente(code_produit , num_vente , qte_vd)

3. le langage de programmation Visual Basic :

3.1. Environnement de développement :

Pour réaliser notre application, nous avons utilisé l'environnement Visuel Basic 6. Visuel Basic 6 est un outil de développement puissant et rapide pour la programmation d'applications pour Windows.

VB est un puissant outil d'écriture d'application sous windows et un environnement interactif très efficace pour mettre au point des fenêtres et des documents. Par cette combinaison, VB simplifie le travail de programmation, réduit le temps de mise au point d'une application et il permet aussi de gérer des données provenant de plusieurs sources de bases de données.

3.2. Choix du SGBD :

Pour créer la base de données de notre application, nous avons utilisé l'SGBD Microsoft Office Access 2007.

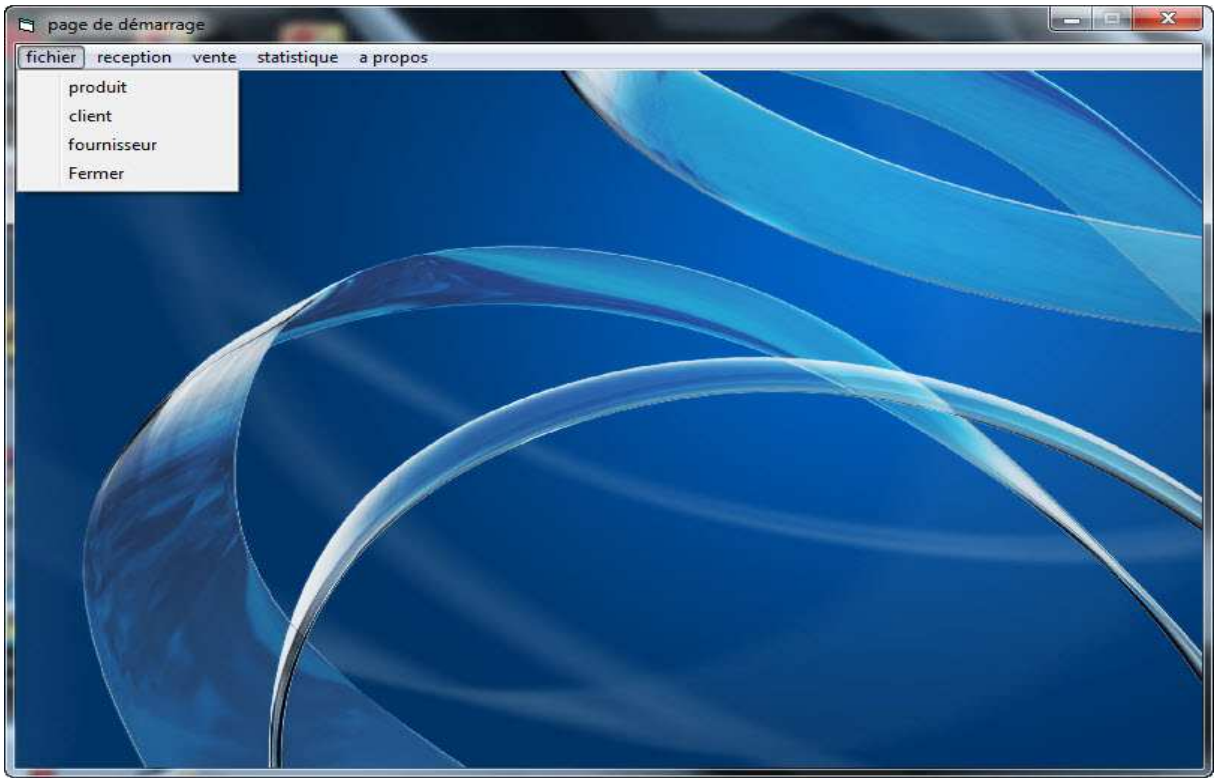
Avec Microsoft Office Access 2007, les travailleurs de l'information peuvent, en un clin d'œil et en toute simplicité, contrôler les informations dont ils disposent et créer des états à partir de celles-ci grâce à l'interface utilisateur Microsoft Office fluent et aux fonctionnalités de conception interactives qui n'exigent aucune connaissance approfondie en matière de base de données.

4. Les interfaces graphiques :

Dans ce qui suit, nous allons présenter quelques interfaces de notre application de gestion de stock.

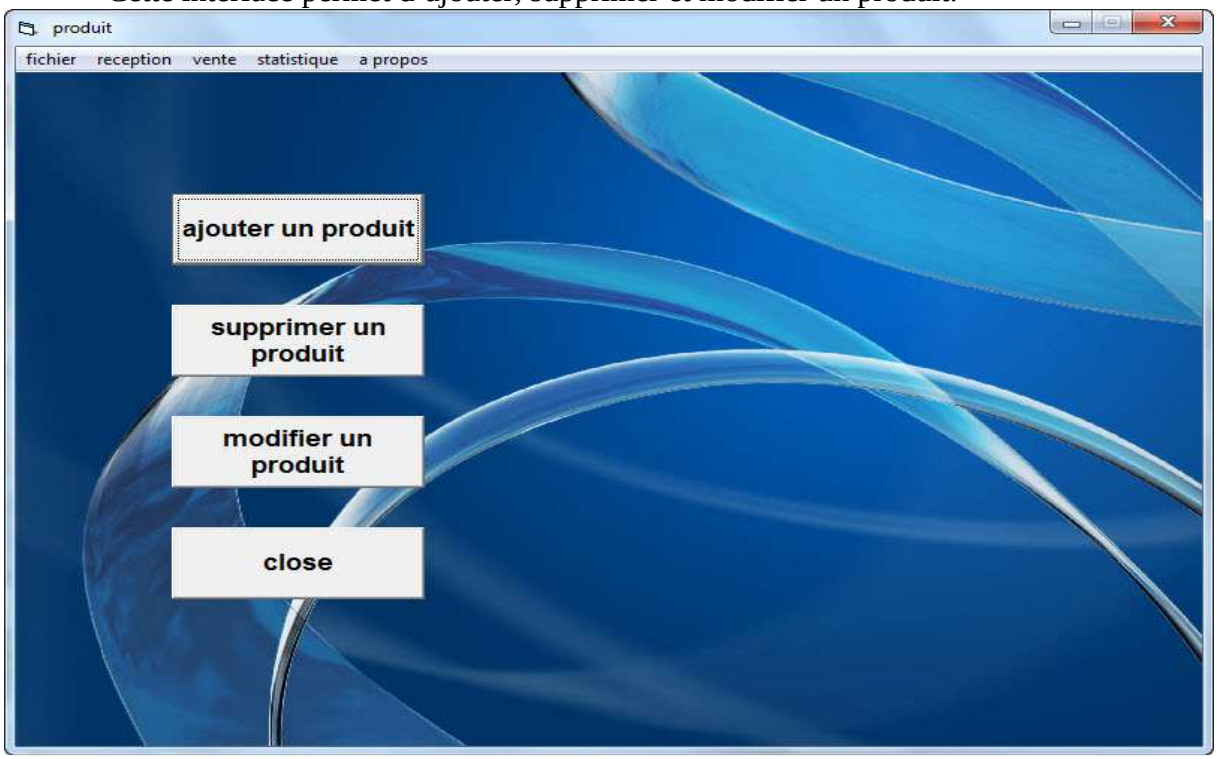
4.1. Page de démarrage :

Cette interface c'est le page de démarrage, permet d'accéder à différentes fonctionnalités de l'application.



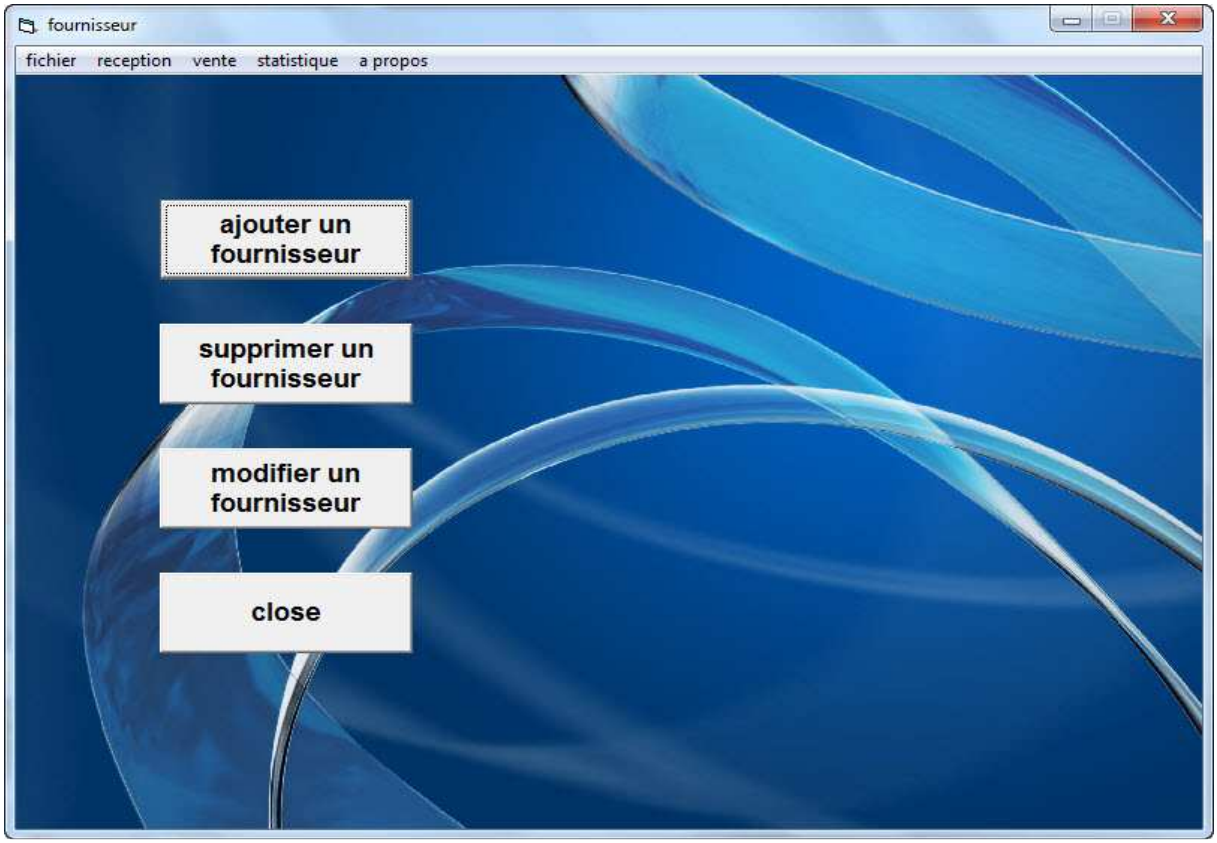
4.2.Mettre à jour produit :

Cette interface permet d'ajouter, supprimer et modifier un produit.



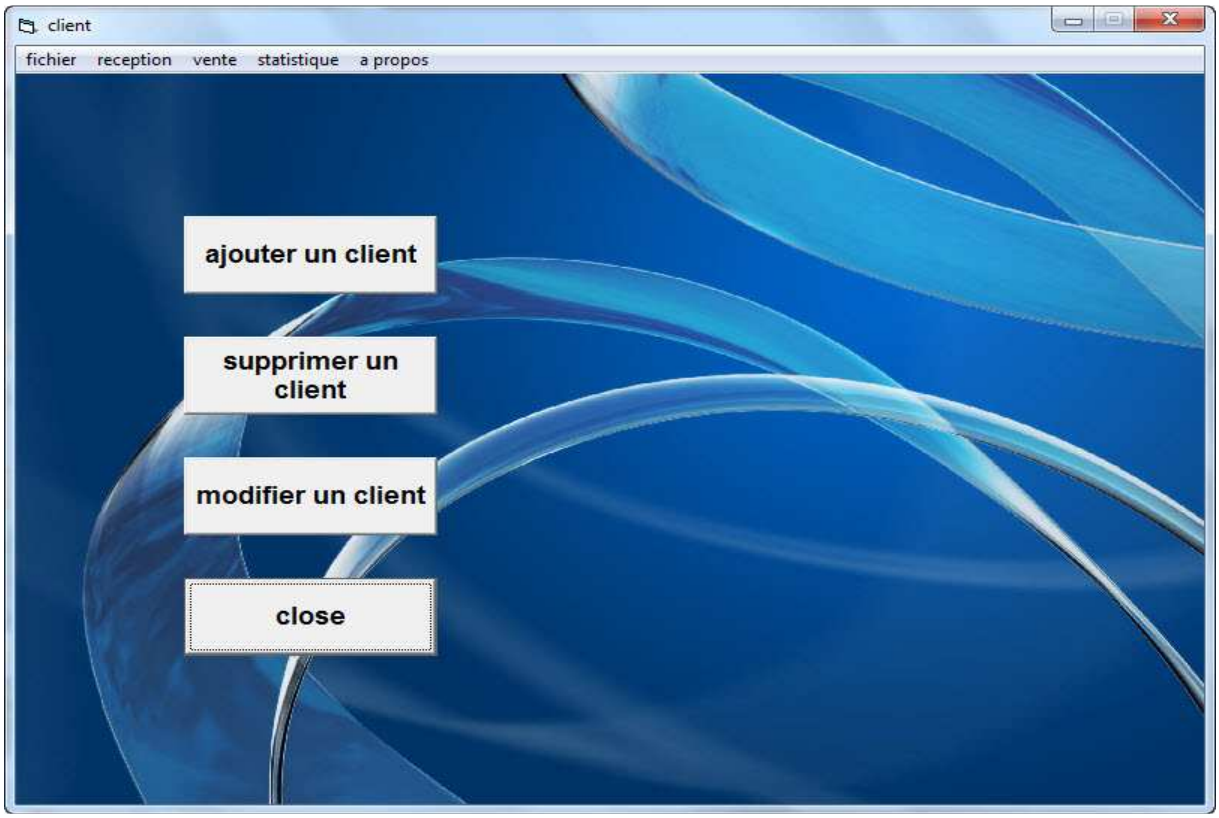
4.3.Mettre à jour fournisseur :

Cette interface permet d'ajouter, supprimer et modifier un fournisseur



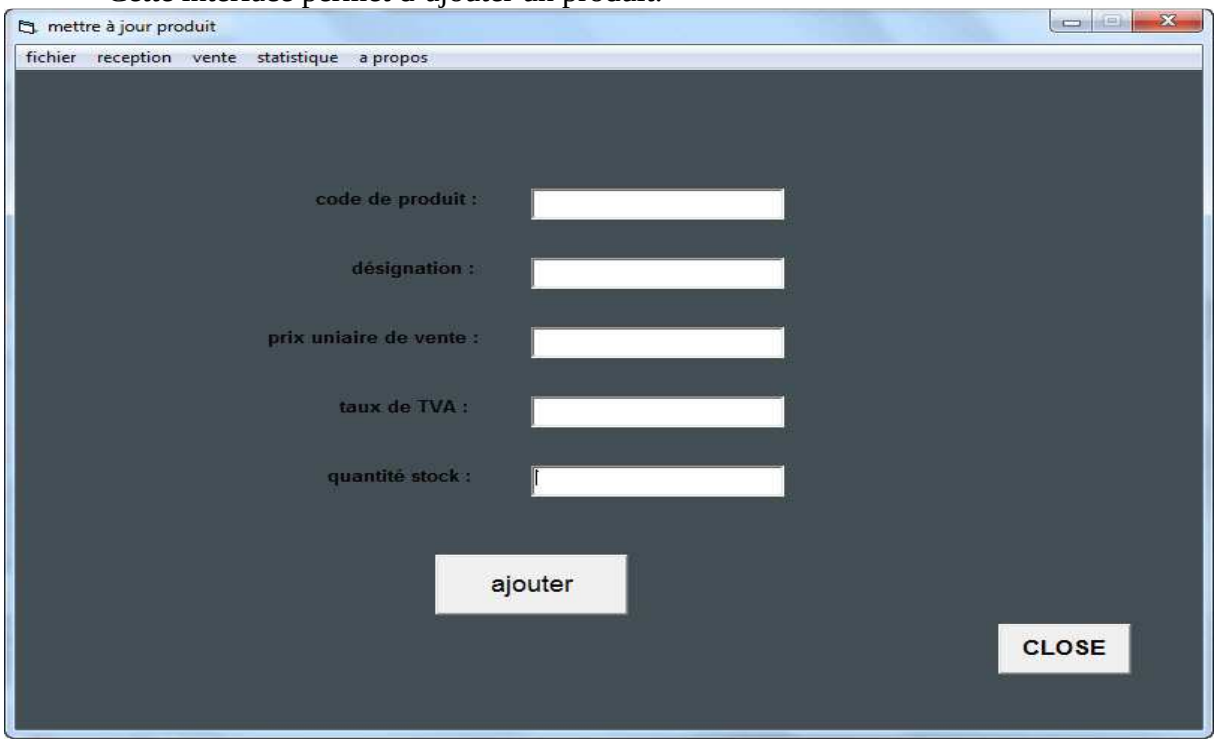
4.4.Mettre à jour client :

Cette interface permet d'ajouter, supprimer et modifier un client.



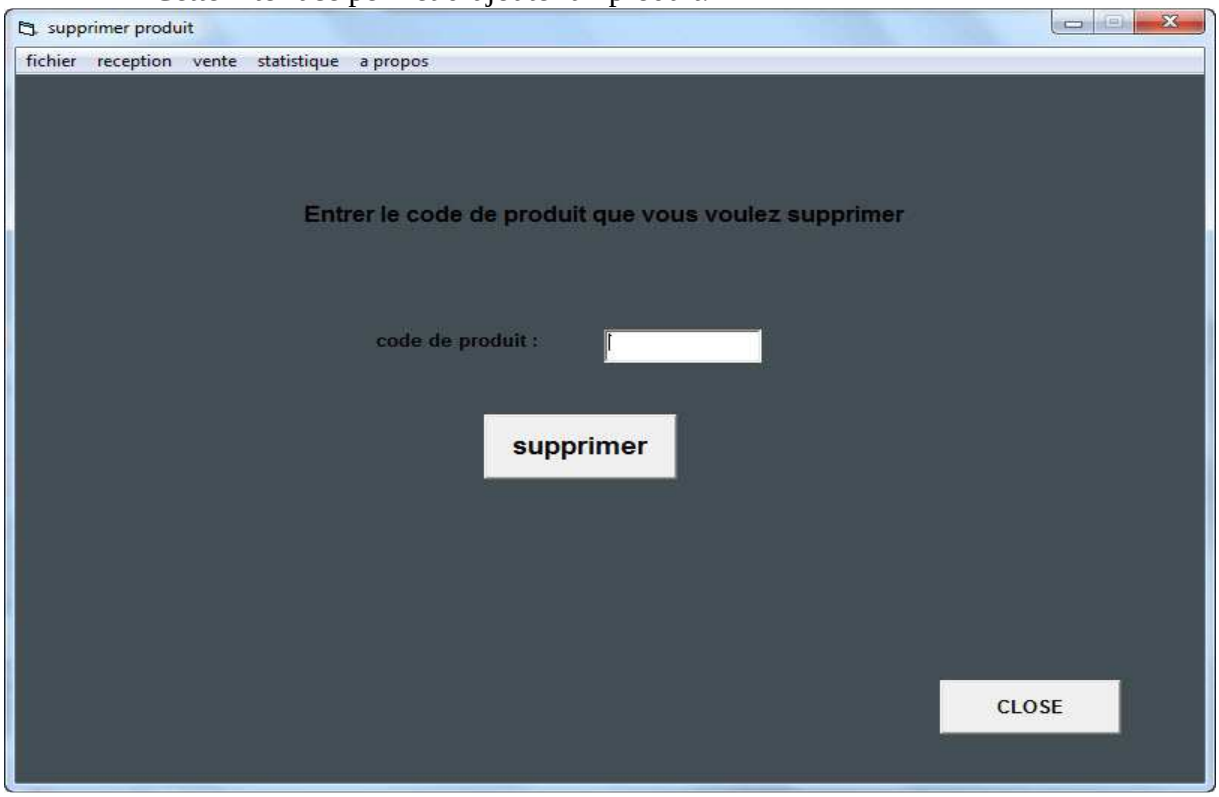
4.5.Ajouter un produit :

Cette interface permet d'ajouter un produit.



4.6. Supprimer un produit :

Cette interface permet d'ajouter un produit.



Conclusion générale

Nous voici arrivés au terme de notre travail.

Le travail que nous avons effectué consiste à faire la conception et la réalisation d'une application pour la gestion d'un magasin de vente en gros.

L'objectif principal de notre projet été de permettre au l'utilisateur de l'application de faire toutes les taches de la gestion de le magasin de vente en gros avec une interface claire et facile.

On peut résumer notre travail dans les deux parties suivantes :

- Partie théorique :

  Nous avons appliqué l'approche OMT/UML, qui nous a permet de résoudre des problèmes de représentation et de traduction des phénomènes réels et de leurs évolutions par des techniques de modélisation.

- Partie implémentation :

  Nous avons implémenté l'étude théorique sur un logiciel qui s'occupe d'automatiser la gestion de le magasin de vente en gros.

La réalisation de ce travail nous a donné l'occasion :

- ✓ Apprendre la méthode OMT et la notation UML.
- ✓ Appliquer l'approche OMT/UML.
- ✓ Enrichir nos capacités actuelles.
- ✓ Approfondir nos connaissances en matière de conception de système d'information.
- ✓ Découvrir le langage de programmation « visuel basic ».

Finalement nous disons que notre travail reste toujours ouvert pour des propositions d'améliorations éventuelles, ou pour des critiques.