Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid– Tlemcen
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études

pour l'obtention du diplôme de Licence en Informatique

Thème

Gestion D'un Magasin De Pièces De Rechange

GMPR

(Conception et réalisation d'une application)

Réalisé par :

- **Melle BORDJI Fatima Zohra**
- **Melle BOUAKKAZ Khadidja**

Encadré par :

- **M.BENMOUNA Youcef**

Présenté le 27 Juin 2013 devant la commission d'examination composée de MM.

- *M.BENAMAR Abdelkarim* *(Examinateur)*
- *M.MOUFFOK Benattou* *(Examinateur)*

Année universitaire : 2012-2013

Remerciements

Louange à Dieu Miséricorde et Miséricordieux qui nous a donné la force, la volonté et la patience durant toutes nos années d'études.

Nous remercions nos très chers parents qui nous ont accordé le courage, la volonté et le soutien pour bien réaliser ce travail.

Nous tenons à exprimer nos vifs remerciements à Mr BENMOUNA. Y, notre encadreur ; c'est sous sa direction scientifique que ce travail de mémoire à été réalisé. Nous tenons à le remercier pour ces conseils, ses encouragements et son orientation. Merci Monsieur l'encadreur pour votre gentillesse et votre patience. Qu'il trouve aussi ici l'expression de notre profonde gratitude d'avoir initié et dirigé ce travail.

Nous adressons toute notre gratitude à Mr MOUFFOK. B, pour avoir très gentiment accepté d'examiner ce travail. On la suit reconnaissante pour tous les efforts qu'il a fourni pour ce diplôme. Merci Mr pour votre soutien humain et scientifique et votre gentillesse.

Nous remercions également Mr BENAMMAR. A, chef de département d'informatique à l'université de Tlemcen. Il nous a fait un plaisir d'être examinateur de ce travail. Veuillez trouver ici de notre haute considération.

Nous exprimons aussi toute notre sympathie et notre reconnaissance à l'ensemble des membres du magasin que nous avons visité pour notre projet.

Enfin nos remerciements vont à toutes les personnes qui de près ou de loin nous ont aidés par leurs encouragements.

Dédicaces

A nos chers parents

A nos familles

A tous nos amis et connaissances

Nous dédions ce mémoire.

Table des matières

Introduction générale

Introduction :

Aujourd'hui, les entreprises évoluent dans un environnement de plus en plus concurrentiel, variable et complexe. La mondialisation, la fragmentation et la diversification croissante des marchés, la complexité des politiques commerciales représentent tant de menaces que d'opportunités pour les entreprises.

Par ailleurs, le développement d'une entreprise en matière de technologies de l'information exige de nouveaux moyens et supports pour échanger et diffuser l'information dans le but de réduire les contraintes de temps, d'espace et du coût, et facilite la gestion de l'entreprise.

De plus, les clients ne sont plus satisfaits de solutions standards, mais demandent plutôt, des solutions globales adéquates à leurs besoins spécifiques. Ils deviennent plus exigeants et demandent des qualités de produits plus élevées et des services plus personnalisés. Ce changement de l'environnement fait qu'il y ait l'augmentation de l'incertitude, et par conséquent, retarde la réponse rapide et appropriée de l'entreprise.

En effet, pour bien assurer la bonne gestion d'une entreprise et avoir de compétitivité il faudrait utiliser un système informatisé. C'est dans ce sens que notre travail contribue à la gestion d'achat et de vente des pièces de rechange dans un magasin.

Problématique :

La gestion d'achat et de vente des pièces de rechange dans un magasin est constituée par les activités suivantes :

· Accueillir les différents clients et fournisseurs ;

· Recevoir les dossiers concernant l'achat et vente des pièces de rechange et les classer ;

· Etablir les commandes au besoin du client ;

· Enregistrer des pièces livrées et les arranger dans les différents rayons ;

· Etablir les rapports concernant l'achat et la vente des pièces de rechange ;

· Faire l'inventaire périodique.

Le système de gestion d'achat et de vente de pièces de rechange rencontre certaines difficultés :

- La perte des documents ;

- L'oubli de faire les livraisons à temps ;

- La redondance dans le saisi des données ;

- Les rapports incomplets ;

- Le manque d'informations pouvant contribuer à l'élaboration des rapports au moment opportun ;

- Temps de réponse très long ou lent ;

- Difficultés dans l'inventaire périodique.

En effet, le département chargé d'achat et de vente des pièces de rechange dans un magasin a besoin d'un système plus performant que celui utilisé actuellement.

La raison qui nous a poussé à choisir le sujet de la gestion d'achat et de vente des pièces de rechange dans un magasin qui nous a demandé de concevoir un logiciel susceptible de gérer l'achat et la vente des pièces de rechange.

L'objectif principal de ce travail de recherche est d'informatiser le département chargé d'achat et de vente des pièces de rechange dans un magasin.

Les intérêts de notre travail de recherche intitulé « Le Logiciel de gestion d'achat et de vente des pièces de rechange » sont :

Ce travail de recherche va nous permettre de nous familiariser à des recherches approfondies dans les magasins et les sociétés et avoir l'expérience dans la conception des logiciels.

Pour le magasin, ce travail va permettre à trouver rapidement les informations fiables et adéquates au moment opportun, et aussi la réalisation de la bonne gestion au sein du magasin.

Pour les employeurs du magasin ce logiciel permettra de faire l'amélioration de la rapidité dans la recherche des informations concernant la gestion d'achat et de vente des pièces de rechange.

L'objectif de ce travail de recherche est de concevoir un logiciel appelé **« Gestion du magasin de pièces de rechange »** qui va contenir la base des données et les interfaces permettant la bonne gestion du magasin dans le domaine d'achat et de vente des pièces de rechange.

Dans notre travail de recherche, la technique utilisée est définie comme un ensemble des moyens tel que :

- Technique documentaire

- Technique d'observation

- Technique d'interview

- Bibliographie et sites Internet

Notre travail est subdivisé en quatre chapitres essentiels:

- Introduction générale

- Chapitre I : Etude du système existant

- Chapitre II : Le langage de modélisation UML

- Chapitre III : La conception du nouveau système d'information

- Chapitre IV : L'implémentation du nouveau système

- Conclusion générale.

Chapitre I

L’étude du système existant

I.1- Introduction

Toutes les entreprises et les magasins commerciaux utilisent les fonctions de réception, d'acquisition et de distribution des marchandises. Le produit acheté est un élément de base pour la bonne gestion du magasin. Pour pouvoir l'informatiser, il faut d'abord se familiariser avec le système existant, en faisant l'étude et l'analyse pour résoudre ses différents problèmes.

Pour faire l'analyse du système existant, nous avons procédé par plusieurs étapes. Nous avons utilisé des méthodes et techniques différentes pour avoir les informations complètes auprès des utilisateurs du système.

En ce qui concerne la documentation, nous avons utilisé différents documents du magasin de pièces de rechange poids lourds « ABDERRAHIMI Nourreddine » qui décrivent le fonctionnement du magasin qui sont :

- Des classeurs: les commandes, les bons de livraison, les fiches de stock et les factures.

-Les documents qui sont en Excel.

- Les documents utilisés par l'agent de stock pour les entrées et sorties de stock (registre de stock).

Pour l'interview, nous avons fait un entretien avec le personnel du magasin :

Mlle H.Chahrazed, la secrétaire.

Monsieur ABDERRAHIMI Nourredine, le patron.

Concernant l'observation, nous avons passé 3 jours au magasin afin d'observer le fonctionnement du système.

I.2- Présentation du magasin

- Raison social: «ABDERRAHIMI Nourredine ». agent agrée par S.N.V.I.
- Date de création: 06 Juillet 1991

- Adresse: au 1025, les Dahlias – Kiffane Tlemcen
- Nombre des employés : 03
- Activités :

1. Garage : vente des pièces.
2. Administration : établir les bons de commande, les bons de livraison les fiches de stock et les factures.
3. Dépôt : stockage des pièces.

- **Organigramme du magasin de P.R**

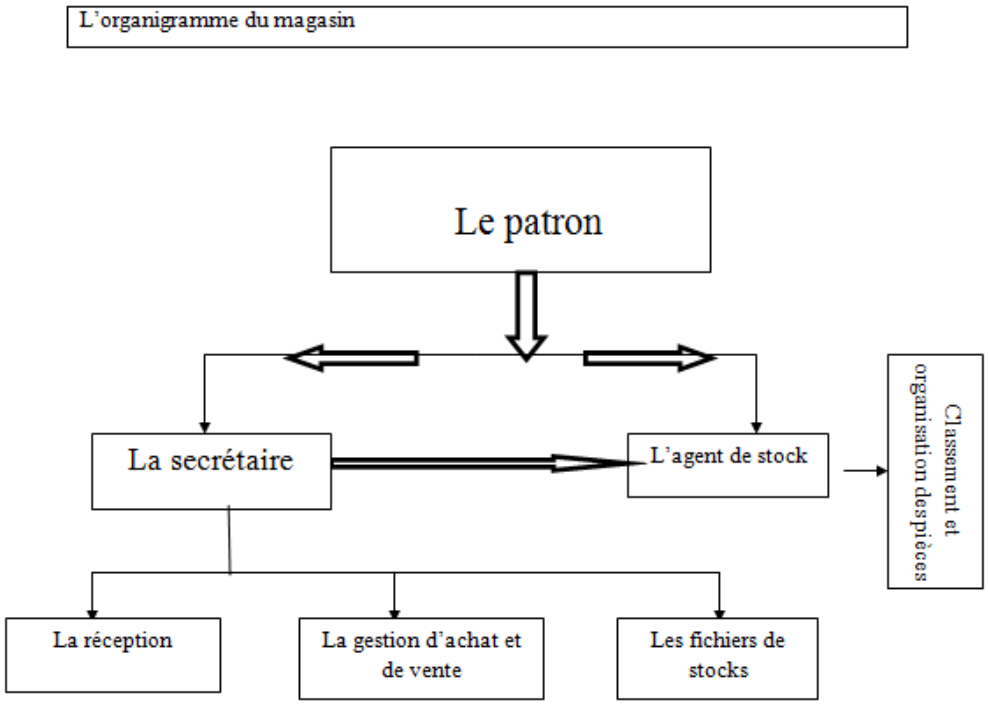

Figure I.1 **L'organigramme du magasin.**

I.3- Le système d'achat et de vente des pièces de rechange

Le magasin avec son système d'achat et de vente des pièces de rechange, utilise le fichier Excel pour le traitement des entrées et sorties.

Les étapes du fonctionnement du système sont les suivantes :

Pour la demande :

1. Le client se présente à la secrétaire pour demander la pièce, le patron lui envoie à l'agent de stock. Le nouveau client est enregistré dans un fichier Excel appelé « Client » où les clients sont enregistrés, après la vérification du bon de commande, ce dossier est classé dans un classeur des commandes de l'année concernée.

2. La secrétaire vérifie dans le fichier Excel appelé « Pièce » où les pièces sont enregistrées.

3. Si la pièce existe, la secrétaire établit le bon de livraison et la facture en Excel pour le payement, si la pièce n'existe pas, la secrétaire établi la liste des pièces à commander et l'envoi à un fournisseur (A.P.C, sociétés et entreprises privées).

Pour la commande envers le fournisseur :

1. Le fournisseur envoie un bon de commande où il enregistre les pièces et la quantité à commander.

2. Arrivée d'une pièce :

Quand le fournisseur envoie les pièces commandées et le bon de livraison, l'agent de stock appelle le mécanicien pour contrôler si les pièces sont conformes à la commande ou les pièces ne sont pas endommagées, si les pièces sont conformes à la commande, l'agent de stock enregistre les pièces entrant dans le fichier Excel (Ficher de stock) et il les arrange dans le magasin. Au cas contraire, l'agent de stock établit la fiche de pièces endommagées et l'envoie au directeur commercial pour le retour des pièces non conformes ou endommagées.

Le système n'est pas capable de faire la gestion de tout le calcul, il y a même la redondance dans l'élaboration des rapports où l'agent de stock fait le copier coller.

I.4- Les définitions des mots ainsi que les documents utilisés

- Le bon de commande : Document comptable qui enregistre les données en rapport avec les commandes envoyées aux fournisseurs et aux magasins.

- Le bon de livraison : Est un document comptable utilisé pour l'enregistrement des données concernant les livraisons faites.

- La facture : Est un document comptable relatif à un achat ou un service rendu.

I.5- Lacunes du système existant

Le système actuel utilise des fichiers Excel pour la gestion d'achat et de vente des pièces de rechange.

Les différents inconvénients du système existant sont les suivants :

- Difficulté de contrôler des achats et ventes

Compte tenu des fichiers Excel utilisés dans le magasin, ce n'est pas facile de :

- Gérer les commandes

- Gérer les livraisons

- Avoir les informations directes concernant les clients et les fournisseurs

- Générer des rapports.

- La gestion inefficace des pièces

Il est difficile de contrôler :

- Les pièces de rechange achetées

- Les pièces de rechange vendues

I.6- Proposition des solutions

Pour faire face à ces différents problèmes que rencontre le magasin dans ses activités journalières, un système informatisé est nécessaire pour remédier à ces problèmes. Ce système devrait permettre au magasin de faire la saisie de toutes les entrées et sorties en rapport avec l'achat et la vente.

Notre travail de recherche aboutira aux objectifs suivants :

1) Suivi des commandes pour les fournisseurs et les clients

2) Eviter la perte des documents

3) Faciliter le rangement des pièces

4) Faciliter l'inventaire périodique.

Chapitre II

Le langage de modélisation UML

II.1- Introduction [1]

Le génie logiciel et la méthodologie s'efforcent de couvrir tous les aspects de la vie du logiciel. Issus de l'expérience des développeurs, concepteurs et chefs de projets, ils sont en constante évolution, parallèlement à l'évolution des techniques informatiques et du savoir-faire des équipes.

Comme toutes les tentatives de mise à plat d'une expérience et d'un savoir-faire, les méthodologies ont parfois souffert d'une formalisation excessive, imposant aux développeurs des contraintes parfois contre-productives sur leur façon de travailler.

Avec la mise en commun de l'expérience et la maturation des savoir-faire, on voit se développer à présent des méthodes de travail à la fois plus proches de la pratique réelle des experts et moins contraignantes.

UML, qui se veut un instrument de capitalisation des savoir-faire puisqu'il propose un langage qui soit commun à tous les experts du logiciel, va dans le sens de cet assouplissement des contraintes méthodologiques.

UML signifie Unified Modeling Language. La justification de chacun de ces mots nous servira de fil conducteur pour cette présentation.

II.2- Historique [2]

- **Les premières méthodes d'analyse (années 70)**

Découpe cartésienne (fonctionnelle et hiérarchique) d'un système.

- **L'approche systémique (années 80)**

Modélisation des données + modélisation des traitements (Merise ...).

- **L'émergence des méthodes objet (1990-1995)**

Prise de conscience de l'importance d'une méthode spécifiquement objet :

- comment structurer un système sans centrer l'analyse uniquement sur les données ou uniquement sur les traitements (mais sur les deux) ?

- Plus de 50 méthodes objet sont apparues durant cette période (Booch, Classe-Relation, Fusion, HOOD, OMT, OOA, OOD, OOM, OOSE...).

- Aucun méthode ne s'est réellement imposée.

Les premiers consensus (1995)

- OMT (James Rumbaugh) : vues statiques, dynamiques et fonctionnelles d'un système
  - ✓ issue du centre de R&D de General Electric.
  - ✓ Notation graphique riche et lisible.
- OOD (Grady Booch) : vues logiques et physiques du système
  - ✓ Définie pour le DOD, afin de rationaliser de développement d'applications ADA, puis C++.
  - ✓ Ne couvre pas la phase d'analyse dans ses 1ères versions (préconise SADT).
  - ✓ Introduit le concept de package (élément d'organisation des modèles).
- OOSE (Ivar Jacobson) : couvre tout le cycle de développement
  - ✓ Issue d'un centre de développement d'Ericsson, en Suède.
  - ✓ La méthodologie repose sur l'analyse des besoins des utilisateurs.

- **L'unification et la normalisation des méthodes (1995-1997)**

En octobre 1994, G. Booch (père fondateur de la méthode Booch) et J. Rumbaugh (principal auteur de la méthode OMT) ont décidé de travailler ensemble pour unifier leurs méthodes au sein de la société Rational Software. Un an après, I . Jacobson (auteur de la méthode OOSE et des cas d'utilisation) a rejoint Rational Software pour travailler sur l'unification. Unified Modified Language (UML) est né.

Les travaux sur ce langage ont continué avec son adoption par de grands acteurs industriels comme HP, Microsoft, Oracle ou Unisys. Ce travail a abouti en 1997 à UML 1.0. Le langage a été soumis par Rational Software et ses partenaires à l'OMG comme réponse à un appel d'offres sur la standardisation des langages de modélisation.

L'appel d'offres de l'OMG a recueilli un avis favorable, puisque 6 réponses concurrentes sont parvenues à l'OMG. IBM et Object Time (méthode ROOM pour les systèmes temps réel réactifs) ont décidé de rejoindre l'équipe UML ; leur proposition était en fait une extension d'UML 1.0.

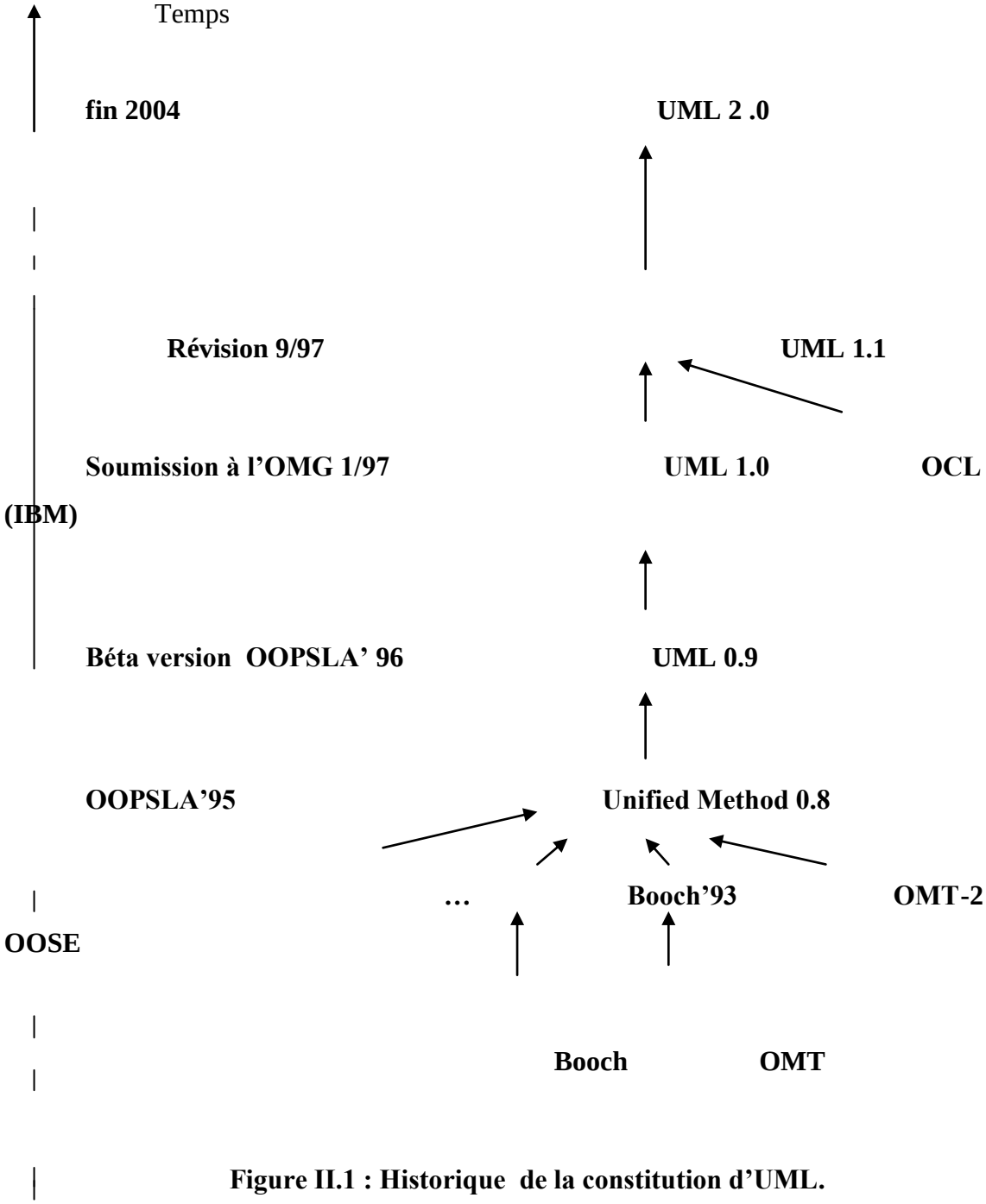

Figure II.1 : Historique de la constitution d'UML.

II.3- Définition UML

UML (en anglais Unified Modeling Language ou « Langage de modélisation unifié ») est un langage de modélisation graphique et textuel destiné a comprendre et a

décrire des besoins, spécifier et documenter des systèmes, esquisser des architectures logicielles, concevoir des solutions et communiquer des points de vue.

Il est apparu dans le monde du génie logiciel, dans le cadre de la « conception orientée objet ». Couramment utilisé dans les projets logiciels, il peut être appliqué a toutes sortes de systèmes ne se limitant pas au domaine informatique.

UML l'analyse objet, en offrant :

Différentes vues (perspectives) complémentaires d'un système, qui des guide l'utilisation des concepts objets.

Plusieurs niveaux d'abstraction, qui permettent de mieux contrôler la complexité dans l'expression des solutions objets.

UML est un support de communication

Sa notation graphique permet d'exprimer visuellement une solution objet.

L'aspect formel de sa notation limite les ambiguïtés et les incompréhensions.

Son aspect visuel facilite la comparaison et l'évaluation des solutions.

Son indépendance (par rapport au langage d'implémentation, domaine d'application, processus) en font un langage universel.

Les points forts d'UML [3]

UML est un langage formel et normalisé, il permet un gain de précision et un gage de stabilité, ce qui encourage l'utilisation des outils.

UML est un support de communication performant, il cadre l'analyse et facilite la compréhension de représentation abstraite complexe.

Son caractère polyvalent et sa souplesse en font un langage universel.

Les points faibles d'UML

La mise en pratique d'UML nécessite un apprentissage et passe par une période d'adaptation.

UML n'est pas à l' origine des concepts objets, mais en constitue une étape majeur, car il unifie les différentes les différentes approches et en donne une définition plus formelle.

Le processus (non couvert par UML) est une autre clé de la réussite d'un projet. Or, l'intégration d'UML dans un processus n'est pas triviale et améliorer un processus est une tâche complexe et longue.

II.4- Les diagrammes UML

a- Définition d'un diagramme

Un diagramme UML est une représentation graphique, qui s'intéresse à un aspect précis du modèle. C'est une perspective du modèle.

Chaque type de diagramme UML possède une structure (les types des éléments de modélisation qui le composent sont prédéfinis).

Un type de diagramme UML véhicule une sémantique précise (un type de diagramme offre toujours la même vue d'un système).

Combinés, les différents types de diagrammes UML offrent une vue complète des aspects statiques et dynamiques d'un système.

Par extension et abus de langage, un diagramme UML est aussi un modèle (un diagramme modélise un aspect du modèle global).

b- Caractéristiques des diagrammes UML

Les diagrammes UML supportent l'abstraction. Leur niveau de détail caractérise le niveau d'abstraction du modèle.

La structure des diagrammes UML et la notation graphique des éléments de modélisation est normalisée.

II.5- Les différents types de diagrammes

a. Diagrammes structurels ou diagrammes statiques (UML Structure)

- **Diagrammes de cas d'utilisation :** use case diagram

Les diagrammes de cas d'utilisation représentent un ensemble de cas d'utilisation, d'acteurs et de leurs relations.ils représentent la vue statique des cas d'utilisation d'un système et sont particulièrement importants dans l'organisation et la modélisation des comportements d'un système.

✓ **Les cas d'utilisation :** Les cas d'utilisation décrivent sous la forme d'actions et de réactions, le comportement, ou tout simplement ce qui fait le point de vue d'utilisateur, encore appelé acteur. On recense de la sorte, l'ensemble des fonctionnalités d'un système en examinant les besoins fonctionnels de chaque acteur.

✓ **Les acteurs :** Un acteur représente un ensemble cohérent de rôles joués par les utilisateurs des cas d'utilisation en interaction avec ces cas d'utilisation. En règle générale, un acteur représente un rôle qu'un homme, une machine ou même un autre système joue avec le système. Il existe 4 grandes catégories d'acteurs :

- les acteurs principaux : les personnes qui utilisent les fonctions principales du système.

- les acteurs secondaires : les personnes qui effectuent des tâches administratives ou de maintenance.

- le matériel externe : les dispositifs matériels incontournables qui font partie du domaine de l'application et qui doivent être utilisés.

- les autres systèmes : les systèmes avec lesquels le système doit interagir.

✓ **Les relations entre les cas d'utilisation :** UML définit 3 types de relations standardisées entre cas d'utilisation, détaillées ci après :

- **La relation d'inclusion :** formalisée par dépendance « include »

Lors de description des cas d'utilisation, il apparaît qu'il existe des sous-ensembles communs à plusieurs cas d'utilisation, il convient donc de factoriser ces fonctionnalités en créant de nouveaux cas d'utilisation qui seront utilisés par les cas d'utilisation qui les avaient en commun.

- **La relation d'extension :** formalisée par la dépendance « extend »

La relation stéréotypée « extend » permet d'étendre les interactions et donc les fonctions décrites par les interactions. Le cas de base peut fonctionner tout seul, mais il peut également être complété par un autre, sous certaines conditions, et uniquement à certains points particuliers de son flot d'évènements (point d'insertion). On utilise principalement cette relation pour séparer le comportement optionnel (les variantes) du comportement obligatoire.

- **La relation de généralisation :**

La relation d'héritage ou de généralisation entre cas et plus subtile. La version 1.1 d'UML ne distinguait d'ailleurs pas « extend » et généralisation. Cette relation et à prendre au sens classique de spécialisation, inhérent à l'héritage. Ici, la généralisation peut être vue aussi comme un polymorphisme de cas.

✓ **Les scénarios :**

Un cas d'utilisation est une abstraction de plusieurs chemins d'exécution. Une instance de cas d'utilisation est appelée : « scénario ».

Après la description des cas d'utilisation, il est nécessaire de sélectionner un ensemble de scénarios qui vont servir à piloter l'itération en cours de développement.

Le choix et le nombre de scénarios à retenir est une étape difficile à réaliser : l'exhaustivité est difficile, voire impossible à atteindre. Le nombre d'instances pour un cas d'utilisation peut être très important, voire infini.

Les scénarios peuvent être classés en :

- **Scénarios principaux :** il correspond à l'instance principale du cas d'utilisation.

C'est souvent le chemin « normal » d'exécution du cas d'utilisation qui n'implique pas d'erreurs.

- **Scénarios secondaires :** il peut être un cas alternatif (un choix), un cas exceptionnel ou une erreur.

- **Diagramme de classe :**

Le diagramme de classes exprime la structure statique du système en termes de classes et de relations entre ces classes.

L'intérêt du diagramme de classe est de modéliser les entités du système d'information.

Le diagramme de classe permet de représenter l'ensemble des informations finalisées qui sont gérées par le domaine.

Le diagramme de classes comporte généralement les éléments suivants :

✓ **Classe :** une classe est une description abstraite (condensée) d'un ensemble d'objets du domaine de l'application : elle définit leur structure, leur comportement et leurs relations.

Représentation : les classes sont représentées par des rectangles compartimentés :

- le 1er compartiment représente le nom de la classe
- le 2ème compartiment représente les attributs de la classe
- le 3ème compartiment représente les opérations de la classe

✓ **Attribut :** un attribut est une propriété nommée d'une classe qui décrit un ensemble de valeurs que les instances de cette propriété peuvent prendre. Une classe peut ne pas avoir, comme elle peut avoir un ou plusieurs attributs.

✓ **Opération :** une opération est une abstraction de ce que peut réaliser un objet et qui est réalisable par tous les objets de la classe. Une classe peut ne pas avoir comme elle peut avoir plusieurs opérations.

✓ **Visibilité :** la visibilité est l'un des détails les plus importants que l'on puisse spécifier pour les attributs et les opérations d'un classificateur. La visibilité d'une caractéristique détermine si d'autres classificateurs peuvent l'utiliser. En UML, on détermine 3 niveaux de visibilité : -public -protected -private.

✓ **Les interfaces :** une interface définit une limite entre la spécification des actions d'une abstraction et l'implémentation de la manière dont cette abstraction les exécute. Une interface est un ensemble d'opérations utilisées pour décrire un service d'une classe ou d'un composant.

✓ **Les collaborations :** la collaboration désigne un ensemble de classes, d'interfaces et d'autres éléments qui travaillent ensemble et fournissent un comportement de coopération plus important que la somme de toutes ces parties.

✓ **Les relations de dépendance, de généralisation et d'association :**

Une relation est une connexion sémantique entre des éléments.

Une relation est représentée par une ligne, chaque sorte de relation est représentée par un type de ligne différent.

Une dépendance est relation d'utilisation qui établi qu'un changement de spécification d'un élément. Une dépendance est symbolisée par une flèche en pointillés, dirigée vers l'élément à l'égard duquel existe la dépendance. On se sert des dépendances pour montrer qu'un élément en utilise un autre.

Une généralisation est une relation entre un élément général (appelé mère) et un élément dérivé de celui-ci, mais plus spécifique (désigné par une file). La généralisation implique que des objets de l'enfant puissent être utilisés partout ou le parent apparait,

elle est représentée par une flèche dont le trait est plein et dont la pointe creuse est dirigée vers le parent.

Une association est une relation structurelle qui précise que les objets d'un élément sont reliés aux objets d'un autre élément, en reliant deux classes et vice versa.

- **Diagramme d'objets :** object diagram

Les diagrammes d'objets permettent de mettre en évidence des liens entre les objets. Les objets, instances de classes, sont reliés par des liens, instances d'associations.

Les diagrammes d'objets utilisent les mêmes concepts que le diagramme de classes. Ils sont essentiellement utilisés pour comprendre ou illustrer des parties complexes d'un diagramme de classes.

- **Diagramme de composants :** composant diagram

Les diagrammes de composants décrivent les composants et leurs dépendances dans l'environnement de réalisation. En général, ils ne sont utilisés que pour des systèmes complexes. Un composant est une vue physique qui représente une partie d'implémentation d'un système. Un composant peut être du code, un script, un fichier de commandes, un fichier de données, une table … Il peut réaliser un ensemble d'interfaces qui définissent alors le comportement offert à d'autres composants.

- **Diagrammes de déploiement :** deployment diagram

Les diagrammes de déploiement montrent la disposition physique des différents matériels (les nœuds) qui entrent dans la composition d'un système et la répartition des instances de composants, processus et objets qui « vivent » sur ces matériels. Les diagrammes de déploiement sont donc très utiles pour modéliser l'architecture physique d'un système.

b. Diagrammes comportementaux ou diagrammes dynamiques (UML Behavior)

- **Diagramme de collaboration :** collaboration diagram

Le diagramme de collaboration permet de mettre en évidence les interactions entre les différents objets du système. Dans le cadre de l'analyse, il sera utilisé

- pour préciser le contexte dans lequel chaque objet évolue

- pour mettre en évidence les dépendances entre les différents objets impliqués dans l'exécution d'un processus ou d'un cas d'utilisation.

Un diagramme de collaboration fait apparaître les interactions entre des objets et les messages qu'ils échangent.

- **Diagramme de séquence :** sequence diagram

Un diagramme de séquence met en évidence le classement des messages par ordre chronologique. On forme un diagramme de séquence en plaçant d'abord les objets qui participent à l'interaction en haut du diagramme. Le long de l'axe des abscisses. En générale, on place l'objet qui débute l'interaction à gauche, puis on continue en progressant vers la droite, les objets les plus subordonnés étant tout à fait à droite. On place ensuite les messages envoyés et reçus par ces objets le long de l'axe des ordonnées, par ordre chronologique, du haut vers bas. Cela donne au lecteur une indication visuelle claire du flot de contrôle dans le temps.

En générale, les diagrammes de séquence contiennent :

✓ **L'objet :** est une manifestation concrète d'une abstraction à laquelle on peut appliquer un ensemble d'opérations et qui possède un état capable de mémoriser les effets de ces opérations. On représente un objet en soulignant son nom.

✓ **Le lien :** est une liaison sémantique entre objets. En générale, il s'agit d'une instance d'une association. Chaque fois qu'une classe est reliée à une autre par une association, il peut y avoir un lien entre les instances des deux classes, et chaque fois qu'un lien existe entre deux objets, le premier objet peut envoyer un message au deuxième.

✓ **Le message :** est la spécification d'une communication entre objets, qui transporte des informations et qui s'affiche dans le but de déclencher une activité. La réception d'une instance de message peut être considérée comme une instance d'un évènement.

- **Diagrammes d'états-transitions :** state machine diagram

Ils ont pour rôle de représenter les traitements (opérations) qui vont gérer le domaine étudié. Ils définissent l'enchaînement des états de classe et font donc apparaître l'ordonnancement des travaux. Le diagramme d'états-transition est associé à une classe pour laquelle on gère différents états : il permet de représenter tous les états possibles ainsi que les événements qui provoquent les changements d'état.

- **Diagramme d'activité** : activity diagram

Le diagramme d'activité est attaché à une catégorie de classe et décrit le déroulement des activités de cette catégorie. Le déroulement s'appelle "**flot de contrôle**". Il indique la part prise par chaque objet dans l'exécution d'un travail. Il sera enrichi par les **conditions** de séquencement. Il pourra comporter des synchronisations pour représenter les déroulements parallèles. La notion de couloir d'activité va décrire les responsabilités en répartissant les activités entre les différents acteurs opérationnels.

II.6- Processus UP

a- Définition du processus unifié

Le processus unifié est un processus de développement logiciel : il regroupe les activités à mener pour transformer les besoins d'un utilisateur en système logiciel.

C'est un patron de processus pouvant être adaptée à une large classe de systèmes logiciels, à différents domaines d'application, à différents types d'entreprises, à différents niveaux de compétences et à différents tailles de l'entreprise.

b- Les caractéristiques du processus unifié

- **Le processus unifié est piloté par les cas d'utilisation**

L'objectif principal d'un système logiciel est de rendre service à ses utilisateurs. Le processus de développement sera donc accès sur l'utilisateur. Les cas d'utilisation permettent d'illustrer ces services.

Ils détectent puis décrivent les besoins fonctionnels (du point de vue de l'utilisateur), et leur ensemble constitue le modèle de cas d'utilisation qui dicte les fonctionnalités complètes du système.

- **Le processus unifié est centré sur l'architecture**

Une architecture adaptée est la clé de voûte du succès d'un développement. Elle décrit des choix stratégiques qui déterminent en grande partie les qualités du logiciel (adaptabilité, performance, fiabilité…).

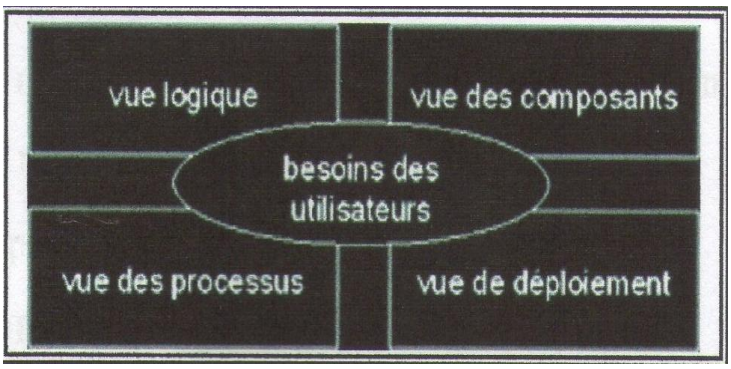

Figure II-2 : Les vues du processus unifié.

✓ **Liens entre cas d'utilisation et architecture**

Tout produit est à la fois forme et fonction. Les cas d'utilisation doivent une fois réalisés, trouver leur place dans l'architecture. L'architecture doit prévoir la réalisation de tous les cas d'utilisation. L'architecture et les cas d'utilisation doivent évoluer de façon concomitante.

✓ **Marche à suivre**

L'architecte crée une ébauche grossière de l'architecture, en partant de l'aspect qui n'est pas propre aux cas d'utilisation (plate forme..). Bien que cette partie de l'architecture soit indépendante des cas d'utilisation. L'architecte doit avoir une compréhension globale de ceux ci avant d'en esquisser l'architecture.

Il travaille ensuite, sur un sous ensemble des cas d'utilisation identifiés, ceux qui représentent les fonctions essentielles du système en cours de développement.

L'architecture se dévoile peu à peu, au rythme de la spécification et de la maturation des cas d'utilisation, qui favorisent, à leur tour, le développement d'un nombre croissant de cas d'utilisation.

Ce processus se poursuit jusqu'à ce que l'architecture soit jugée stable.

- **Le processus unifié est itératif et incrémental**

L'itération est une répétition d'une séquence d'instructions ou d'une partie du programme un nombre de fois fixé à l'avance ou tant qu'une condition définie n'est pas remplie, dans le but de reprendre un traitement sur des données différentes.

Elle qualifie un traitement ou une procédure qui exécute un groupe d'opérations de façon répétitive jusqu'à ce qu'une condition bien définie soit remplie.

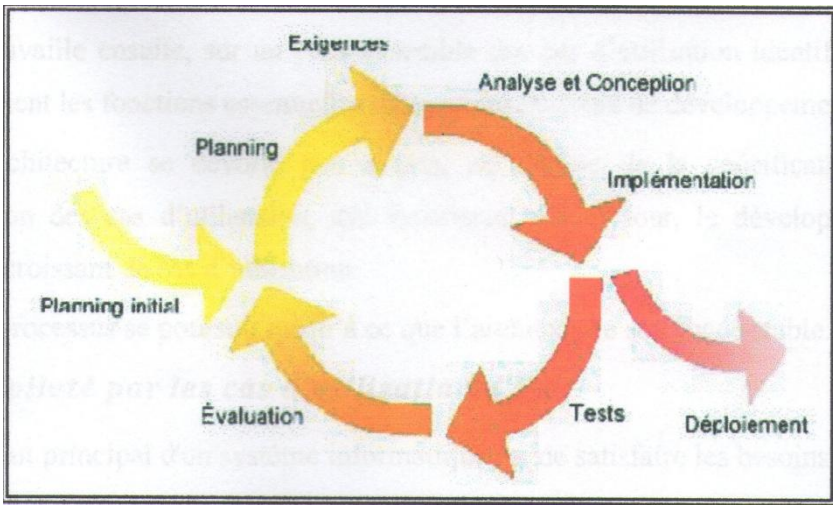

Figure II-3 : Déroulement du processus UP.

Une itération prend en compte un certain nombre de cas d'utilisation et traite en priorité les risques majeurs.

c- Le cycle de vie du processus unifié

Le processus unifié répète un certain nombre de fois une série de cycles.

Tout cycle se conclut par la livraison d'une version du produit aux clients et s'articule en 4 phases : création, élaboration, construction et transition, chacune d'entre elles se subdivisant à son tour en itérations.

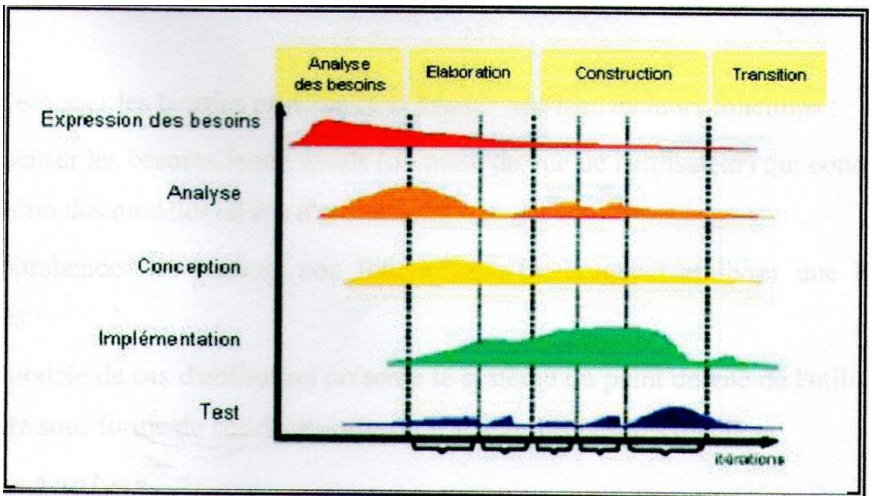

Figure II-4 : Présentation du cycle de vie du processus UP.

d- Les phases et la description et l'enchaînement d'activités

- **Phase de création**

Traduit une idée en vision de produit fini et présente une étude de rentabilité pour ce produit

- Que va faire le système pour les utilisateurs ?

- A quoi peut ressembler l'architecture d'un tel système ?

- Quels sont l'organisation et les coûts du développement de ce produit ?

On fait apparaître les principaux cas d'utilisation.

L'architecture est provisoire, identification des risques majeurs et planification de la phase d'élaboration.

- **Phase d'élaboration**

Permet de préciser la plupart des cas d'utilisation et de concevoir l'architecture du système. L'architecture doit être exprimée sous forme de vue de chacun des modèles.

Emergence d'une **architecture de référence**.

A l'issue de cette phase, le chef de projet doit être en mesure de prévoir les activités et d'estimer les ressources nécessaires à l'achèvement du projet.

- **Phase de construction**

Moment où l'on construit le produit. L'architecture de référence se métamorphose en produit complet, elle est maintenant stable. Le produit contient tous les cas d'utilisation que les chefs de projet, en accord avec les utilisateurs ont décidé de

mettre au point pour cette version. Celle ci doit encore avoir des anomalies qui peuvent être en partie résolue lors de la phase de transition.

- **Phase de transition**

Le produit est en version bêta. Un groupe d'utilisateurs essaye le produit et détecte les anomalies et défauts. Cette phase suppose des activités comme la fabrication, la COURS UML13.doc – janv. 2003 J.STEFFE – ENITA de Bordeaux 64 formation des utilisateurs clients, la mise en œuvre d'un service d'assistance et la correction des anomalies constatées.

II.7- Conclusion

Le langage UML nous apporte une aide à toutes les étapes d'un projet, comme il nous offre ainsi de nombreux avantages pour l'analyse et la conception d'un système d'information géographique. Par conséquent, nous avons combiné UML au processus unifié pour modéliser notre système.

Cette modélisation sera détaillée dans le chapitre suivant.

Chapitre III

La conception Du Système GMPR

III.1-Introduction

Ce chapitre porte sur la conception du système comme nous avons déjà cité dans le chapitre précédent ,notre choix s'est porté sur le processus UP, en conséquence , nous allons détailler trois étapes :

- Spécifier les besoins de notre système en définissant le diagramme de cas d'utilisation.
- Le diagramme de cas d'utilisation va être détaillé en plusieurs diagrammes de séquences.
- Décrire la structure statique de notre système par le diagramme de classe.

III.2- Modélisation du système GMPR

Pour modéliser le système GMPR, nous commençons par partager ce système en plusieurs cas d'utilisations. Chaque cas correspond à un ou plusieurs scénarios.

a- Diagramme de cas d'utilisation

Les cas d'utilisation et les acteurs GMPR sont schématisés dans le diagramme suivant (Fig. **III.1).**

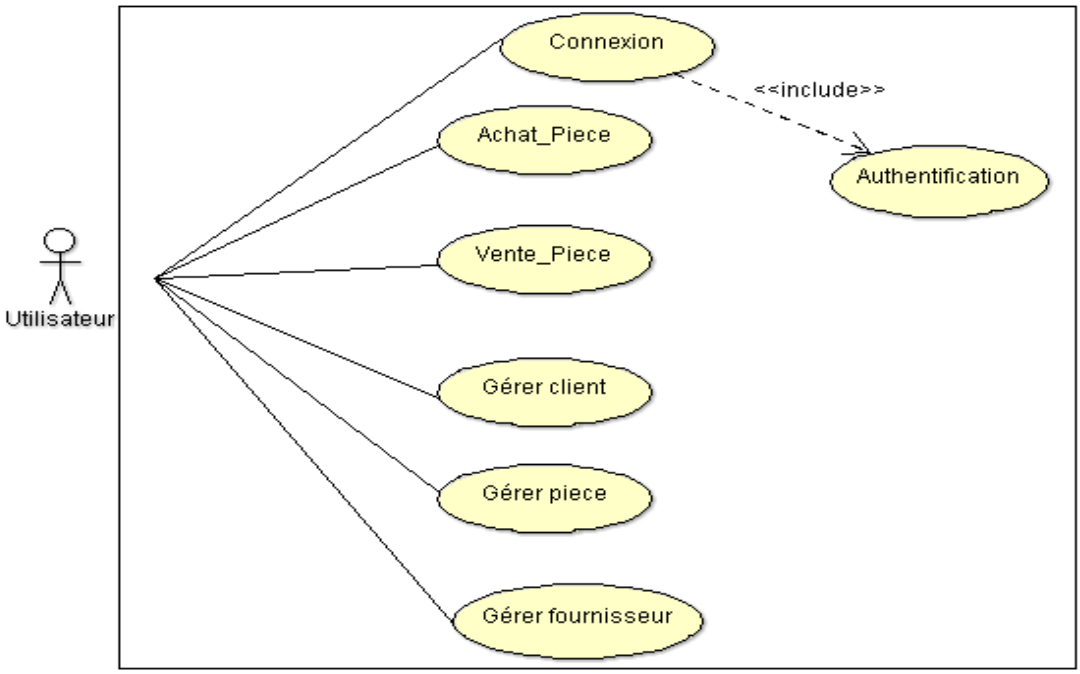

Figure III.1 : Diagramme de cas d'utilisation « gestion d'achat et vente de pièces de rechanges ».

- **Identification d'acteur « Utilisateur »**

L'utilisateur joue un rôle très important. Cet acteur est une personne qui travaille au magasin, dans notre cas c'est la secrétaire.

L'acteur interagit avec le système GMPR :

- Gestionnaire d'un magasin
- Commerçant de pièces de rechange

Chaque scénario sera détaillé dans ce qui suit.

b- Les diagrammes de séquence

- **Le scénario de cas d'utilisation « Connecter »**

Le système vérifie le nom d'utilisateur et le mot de passe.

- Si les informations ne sont pas correctes le système affiche un message d'erreur sinon le système affiche le menu de l'application.

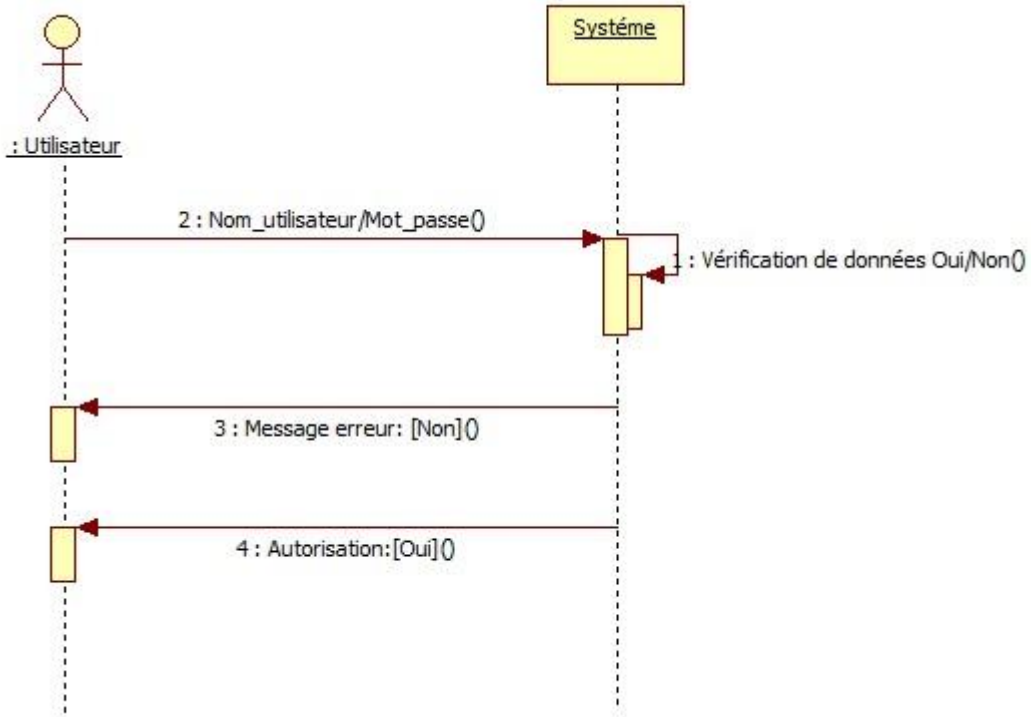

Figure III.2 : Diagramme de séquence « Connecter ».

- **Les scénarios de cas d'utilisation « Acheter pièces »**

✓ **Gérer commande fournisseur**

Si l'utilisateur a un manque de pièces, il doit crée un bon de commande au fournisseur.

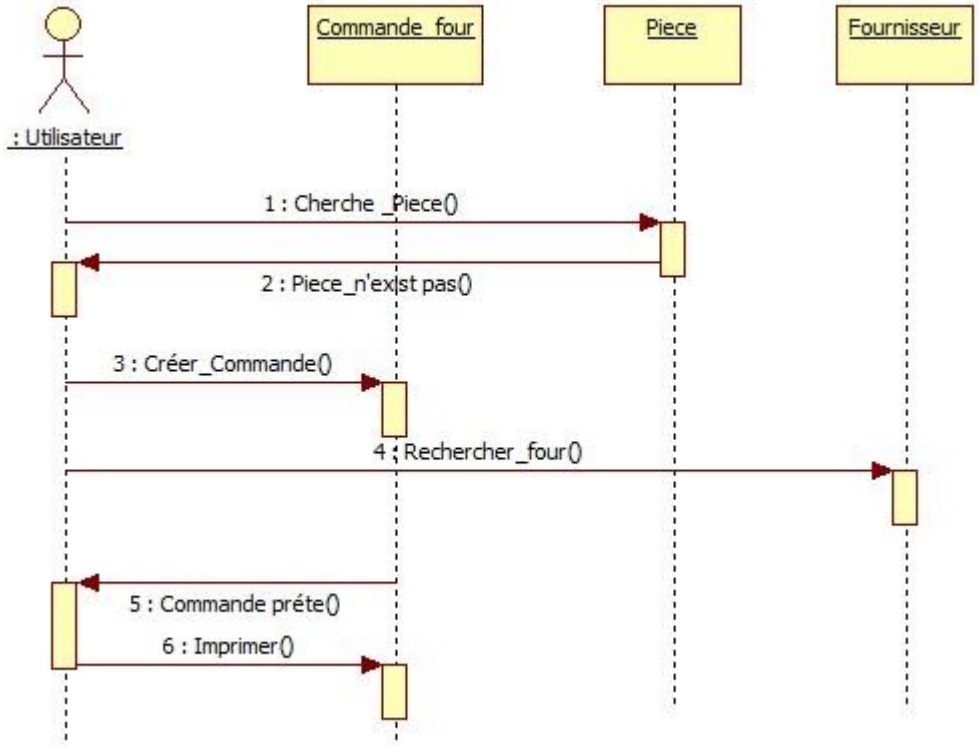

Figure III.3 : Diagramme de séquence « gérer commande fournisseur ».

✓ **Facture Fournisseur**

A la réception de la facture d'un fournisseur, l'Utilisateur

1. Cherche le fournisseur (par nom ou numéro).
2. Enregistre la facture.
3. Fait le mis à jour du fichier de la pièce avec la quantité mentionnée dans le bon de commande et le bon de livraison.

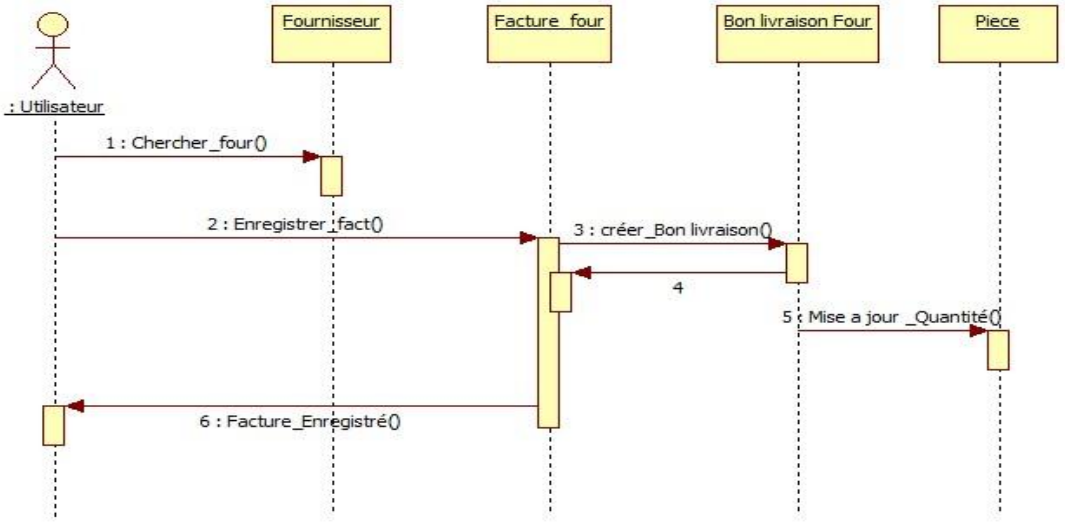

Figure III.4 : Diagramme de séquence « Facture fournisseur ».

- **Les scénarios de cas d'utilisation « Vente pièces »**

  - ✓ **Traitement des bons de commande**

L'utilisateur enregistre le bon de commande reçu de client.

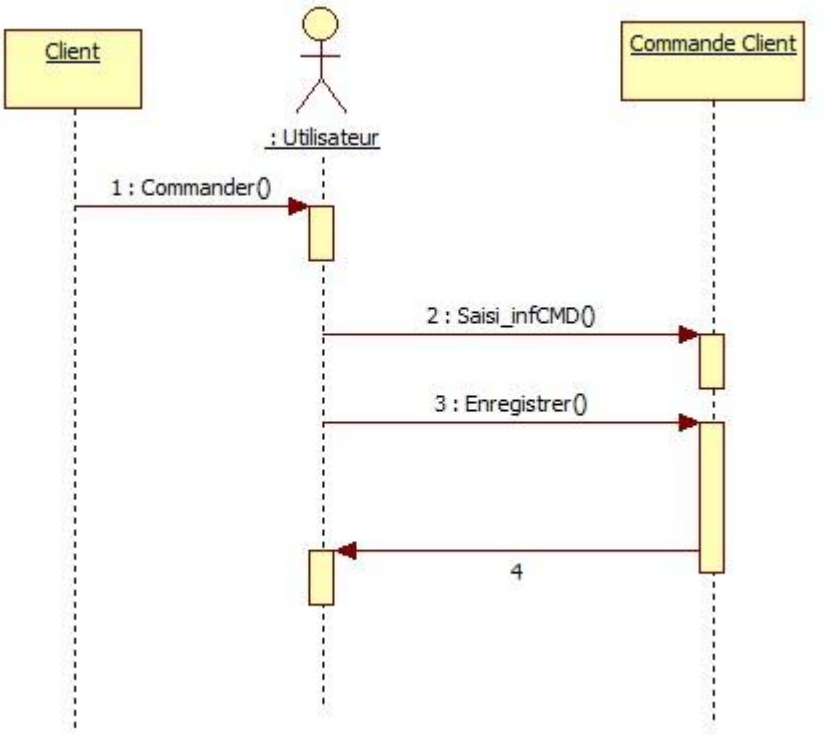

Figure III.5 : Diagramme de séquence « Traitement des bons de commande».

✓ **Facture Client**

A la réception d'un bon de commande de la part d'un client, l'Utilisateur :

1. Recherche le client dans la base de données si le client a été trouvé alors il édite une facture.
2. Ajoute les pièces avec leur quantité demander à partir de bon de commande de client, pour chaque pièces ajouter le système automatiquement le total montant par pièces ainsi que le total acheter et le TTC.

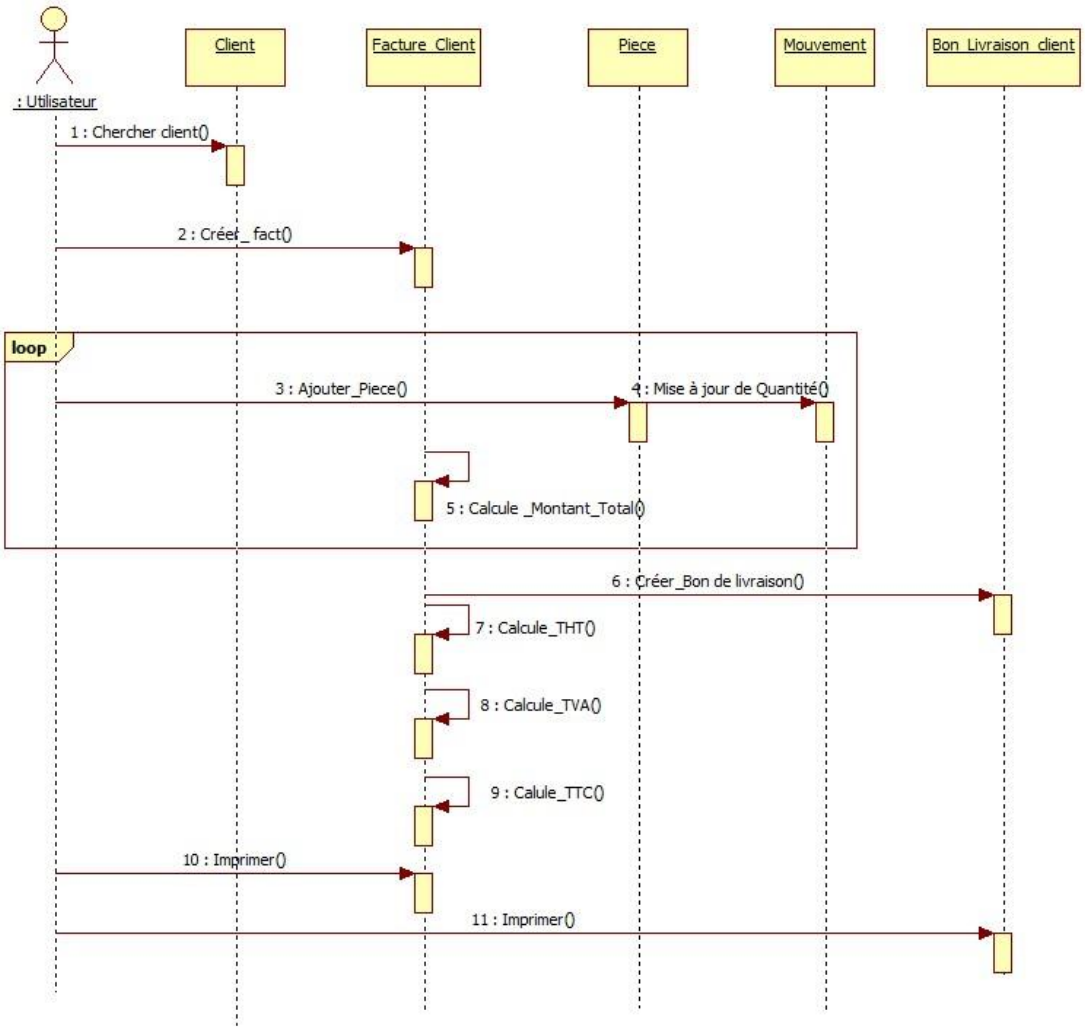

Figure III.6 : Diagramme de séquence « Facture Client ».

- **Les scénarios de cas d'utilisation « Gérer Client »**

  - ✓ **Ajouter Client**

Si l'utilisateur a un nouveau client, il doit l'ajouter.

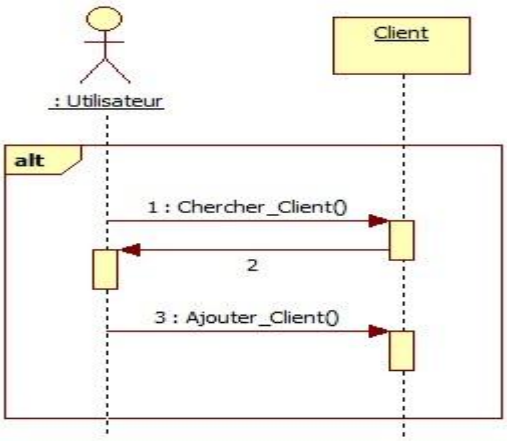

Figure III.7 : Diagramme de séquence « Ajouter Client ».

- ✓ **Modifier Client**

Si l'utilisateur a des modifications dans les informations des clients (changement de leur adresse ou numéro de téléphone), il va les modifier.

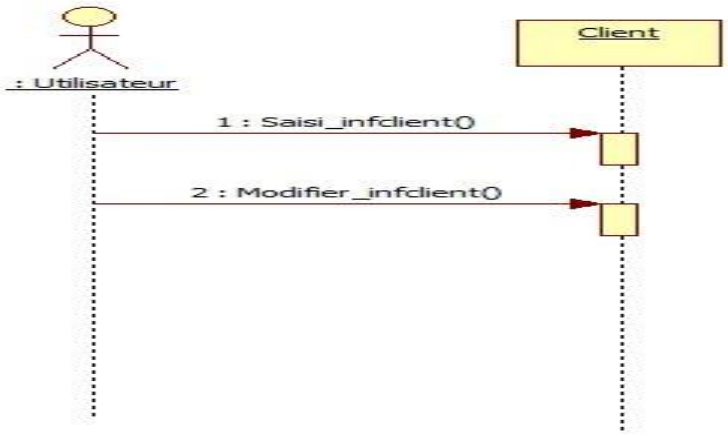

Figure III.8 : Diagramme de séquence « Modifier Client ».

✓ **Supprimer Client**

S'il y a un client qui s'arrête de commander des pièces pendant une longue durée, l'utilisateur doit le supprimer.

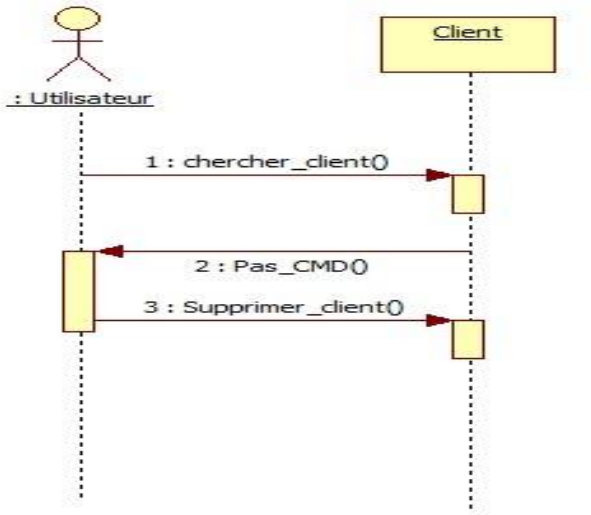

Figure III.9 : Diagramme de séquence « Supprimer Client ».

- **Les scénarios de cas d'utilisation « Gérer Pièces »**

✓ **Ajouter Pièce**

Si l'utilisateur a une nouvelle pièce, il va créer une nouvelle pièce.

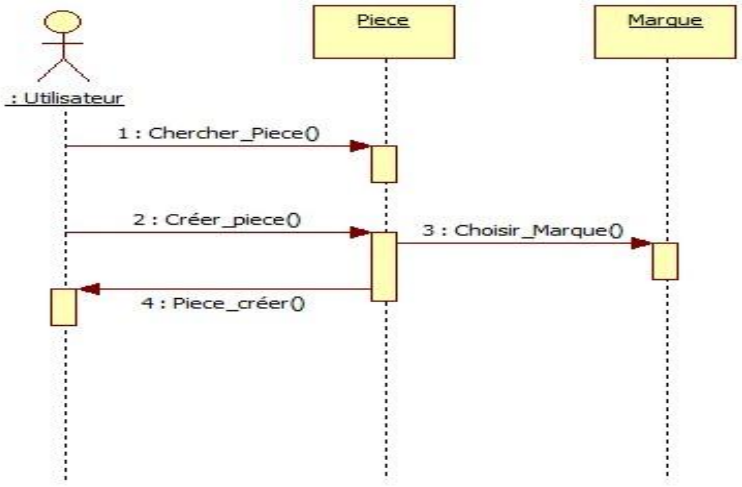

Figure III.10 : Diagramme de séquence « Ajouter pièce ».

✓ **Modifier Pièce**

Si la quantité de la pièce est changée, l'utilisateur va la modifier

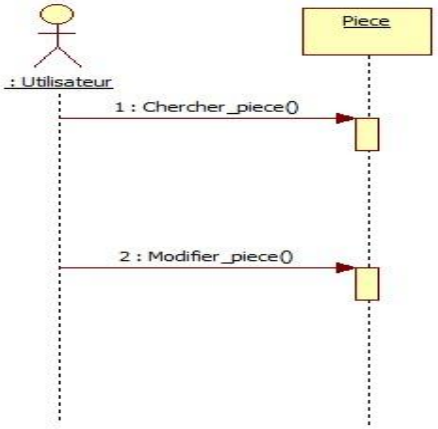

Figure III.11 : Diagramme de séquence « Modifier Pièce ».

✓ **Supprimer Pièce**

Si la pièce dépasse la date d'expérimentation, l'utilisateur doit la supprimer.

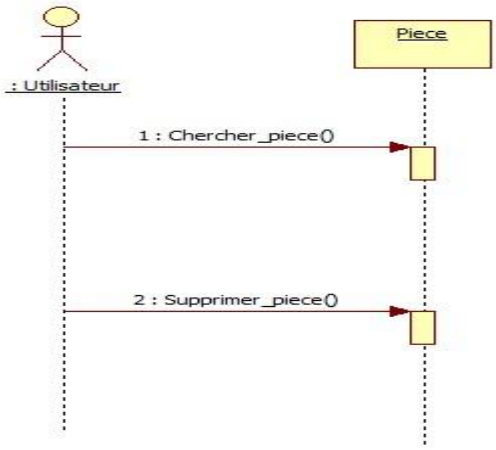

Figure III.12 : Diagramme de séquence « Supprimer Pièce ».

- **Les scénarios de cas d'utilisation « Gérer fournisseur»**

✓ **Ajouter Fournisseur**

Si l'utilisateur a un nouveau fournisseur, il doit l'ajouter.

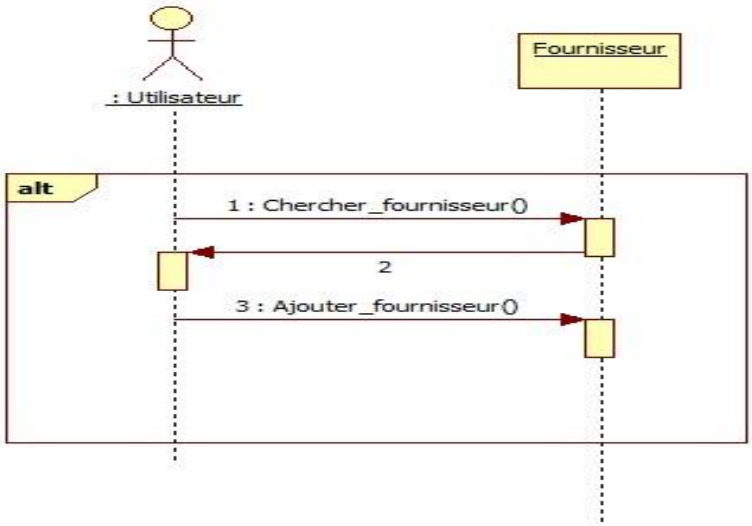

Figure III.13 : Diagramme de séquence « Ajouter Fournisseur ».

✓ **Modifier Fournisseur**

Si l'utilisateur a des modifications dans les informations de fournisseur (changement de leur adresse ou leur numéro de téléphone), il les va modifier.

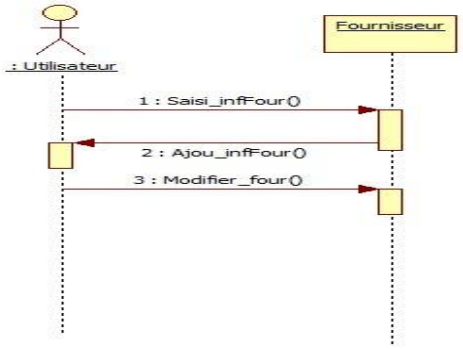

Figure III.13 : Diagramme de séquence « Modifier Fournisseur ».

✓ **Supprimer Fournisseur**

S'il y a un fournisseur qui s'arrête de livrer des pièces pendant une longue durée, l'utilisateur doit le supprimer.

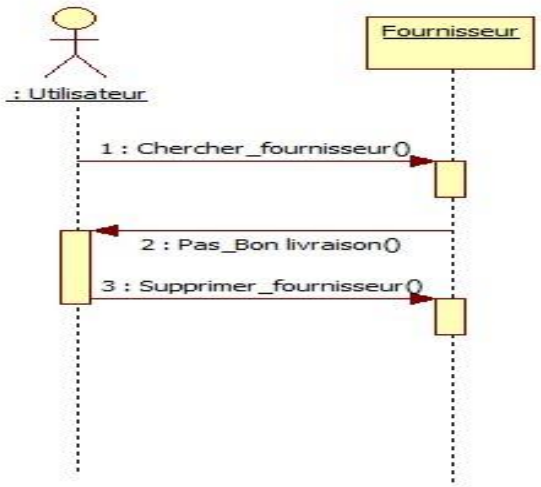

Figure III.14 : Diagramme de séquence « Supprimer Fournisseur ».

c- Diagramme de classe

Le diagramme de classes identifie les classes de notre système et les associations entre elles. Ce diagramme contient neuf classes.

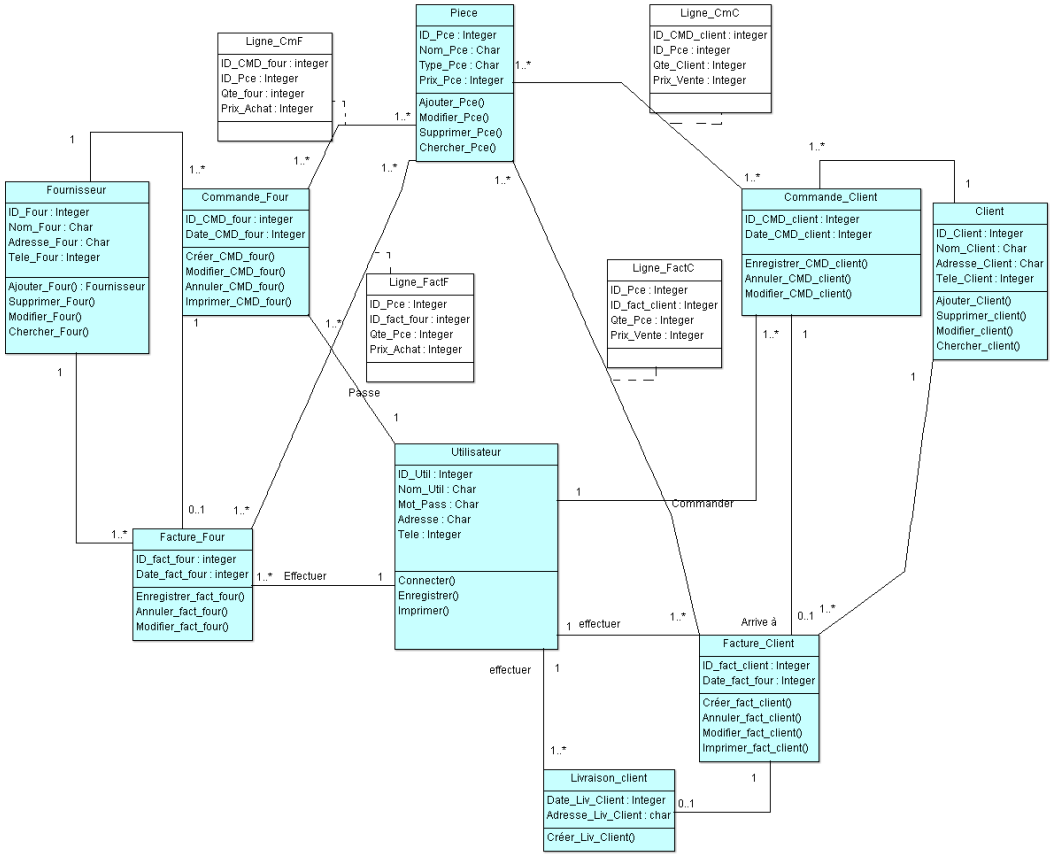

Figure III.15 : Diagramme de classes.

III.3- Le Model Logique du système GMPR

Piece (ID_Pce, Nom_Pce, Type_Pce, Prix_Pce)

Utilisateur (ID_Util, Nom_Util, Mot_Pass, Adresse, Tele)

Commande_Four (ID_CMD_four, Date_CMD_four, ID_Four*, ID_Util*)

Commande_Client (ID_CMD_Client, Date_CMD_Client, ID_Util*, ID_Client*)

Facture_Client (ID_fact_Client, Date_fact_Client, ID_CMD_Client*, ID_Util*, ID_Client*)

Facture_Four (ID_fact_four, Date_fact_four, ID_CMD_four*, ID_Util*, ID_Four*)

Livraison_Client (Date_Liv_Client, Adresse_Liv_Client, ID_Util*, ID_fact_client*)

Client (ID_Client, Nom_Client, Adresse_Client, Tele_Client)

Fournisseur (ID_Four, Nom_Four, Adresse_Four, Tele_Four)

Ligne_CmC (ID_CMD_Client*,ID_Pce*,Qte_Client,Prix_Vente)

Ligne_CmF(ID_CMD_Four*,ID_Pce*,Qte_Four,Prix_Achat)

Ligne_FactF(ID_Pce*,ID_fact_four*,Qte_Pce,Prix_Vente)

Ligne_FactC (ID_Pce*,ID_fact_client,Qte_Pce,Prix_Achat).

III.4- Conclusion

Nous avons traité dans ce chapitre la conception du système GMPR. En utilisant le langage UML, cette conception est une phase nécessaire pour pouvoir créer notre base de données et réaliser l'application, le chapitre suivant détaillera les phases de l'implémentation de notre système.

Chapitre IV

L'Implémentation du Système GMPR

IV.1- Introduction

Ce chapitre est consacré à la traduction de la conception en code source, qui s'appuie sur les données définis au chapitre précédent. Nous allons justifier le choix du langage, donner un bref aperçu sur les outils utilisés, présenter les résultats de notre travail. Cette implémentation est réalisée avec le langage de programmation pascal objet en utilisant le delphi7.

IV.2- Choix du langage de programmation [4] [5]

Il existe en fait deux catégories de langages qui permettent la programmation utilisant les objets :

• Les langages ne permettant rien d'autre que l'utilisation d'objets, dans lesquels tout est objet. Ces langages sont assurément appréciés par certaines personnes mais imposent un apprentissage des plus délicats.

• Les langages permettant l'utilisation des objets en même temps que le style impératif classique. C'est le cas de Pascal Objet, et de nombreux autres langages comme Java et C++. Ces langages sont beaucoup plus souples même si nombre d'entre eux ne donnent pas les mêmes possibilités en termes de programmation objet. Delphi supporte donc un langage orienté objet, ce qui signifie que quelqu'un qui ignore tout des objets peut presque entièrement s'en affranchir, ou les utiliser sans vraiment s'en rendre compte. Limitée dans la version 1 de Delphi, la « couche » (l'ensemble des fonctionnalités) objet de la version6 est réellement digne de ce nom. Mais trêve de blabla, passons aux choses sérieuses.

a- Objectifs de la programmation objet [6]

• Lier les données et les fonctions qui les manipulent afin d'éviter des accès aux données par des fonctions non autorisées.

• Obtenir une meilleure abstraction en cachant l'implémentation des techniques utilisées et en ne rendant visible que des points d'entrée. Ainsi, si l'implémentation change, le code utilisateur n'est pas affecté.

• Réutiliser l'existant dans un souci de productivité.

• Traiter les erreurs localement au niveau des objets sans que cela ne perturbe les autres parties du programme.

• Faciliter la maintenance.

b- Pascal Objet

Le Pascal est un langage fortement typé qui sert de support à Delphi. La version utilisé par Delphi est une version enrichie appelée Pascal Objet qui permet la programmation orienté objet dans l'environnement Windows.

IV.3- Choix du l'environnement de développement (Delphi) :

Delphi est le nom d'un logiciel actuellement largement employé pour créer des logiciels. Delphi permet d'utiliser le langage Pascal. Il faut bien comprendre que Pascal et Delphi ne sont pas une seule et même chose : Pascal est un langage informatique, Delphi est un logiciel destiné à créer des logiciels avec ce langage. Delphi n'est qu'un enrobage, une enveloppe de confort autour de Pascal, c'est-à-dire qu'il simplifie de nombreuses tâches liées à la programmation en langage Pascal. Un autre fait qu'il faut noter est que Delphi est destiné à écrire des programmes fonctionnant exclusivement sous Windows. Comme beaucoup de logiciels, Delphi existe en plusieurs versions. Actuellement des versions numérotées de 1 à 7 existent. Ces versions successives du logiciel ont vu de nombreuses améliorations, tant cosmétiques, qu'au niveau du langage. La version la plus élevée est la plus intéressante, car permet toujours plus de choses.

a- Delphi 7 [7] [8]

La version 7 utilisée pour écrire les exemples est la dernière disponible sur Windows, mais tous les exemples sont écrits avec les fonctionnalités générales de Delphi ce qui permet de les compiler sur n'importe quelle version de Delphi depuis la version 5.

Dans notre application on a choisi le Delphi 7.

b- Vocabulaire

L'informatique, en tant que science, a son propre jargon, qu'il est nécessaire de connaître. Les quelques termes qui suivent sont des incontournables. Lorsque vous les rencontrerez, vous devrez savoir ce qu'ils signifient. Les définitions ci-dessous ne sont valables qu'à l'intérieur du guide (et sont souvent utilisées en dehors) :

« **Programme** » : texte écrit dans un langage informatique, comportant dans notre cas des instructions structurées (organisées en structures). Il est destiné à être « converti » par Delphi en un logiciel utilisable sous Windows.

« **Développer en Delphi** » : écrire des programmes en utilisant le langage Pascal. Par abus, on confond le langage (Pascal) et le logiciel (Delphi) qui le gère, et on parle de développer, plutôt que de programmer.

« **Application** » : Logiciel fonctionnant sous Windows.

« **Projet** » : c'est la base d'une application. Sous Delphi, pour créer une application, on constitue d'abord un projet, constitué de divers morceaux.

« **Code** », « **Code Pascal** », « **Code source** » : morceau de programme, texte d'un programme écrit en Pascal.

« **Interface (utilisateur)** » : la partie d'un logiciel qui est visible par l'utilisateur, à l'opposé du code source, invisible à l'utilisateur.

« **Fiche** » : fenêtre à l'état non compilé. Les fiches sont les alter egos sous Delphi des fenêtres sous Windows.

Le schéma suivant tente de reprendre les termes mentionnés ci-dessus :

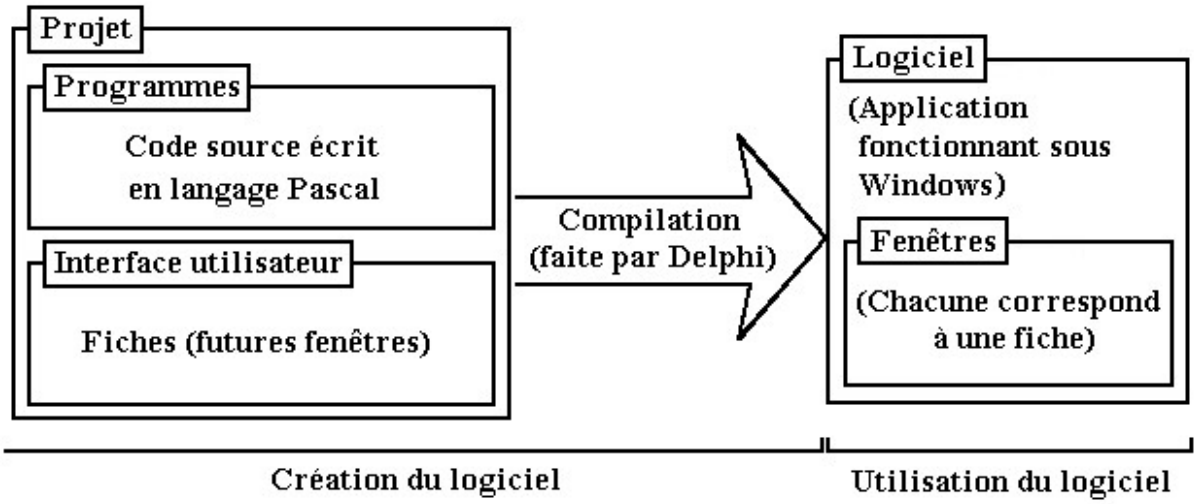

Figure IV.1 : Caractéristiques de Delphi.

IV.4- Description de l'application

Cette partie concerne la description de notre application, la gestion d'un magasin de pièces de rechange.

Elle est composée d'une base de données qui contient les fichiers pièces, clients et fournisseurs.

a- Page d'authentification

Cette forme permet à l'utilisateur de s'introduire au menu de l'application. Il devra saisir le nom d'utilisateur, le mot de passe.

Figure IV.2 : Page d'authentification.

Le système vérifie le nom d'utilisateur et le mot de passe.

Si les informations ne sont pas correctes le système affiche un message d'erreur sinon le système affiche le menu principal.

Figure IV.3 : Page d'authentification « cas d'erreur ».

b- Menu principal

La forme de menu principal contient des différents menus permettant d'accéder aux différents formulaires.

Elle comprend six menus principaux : Pièce, Client, Fournisseur, Traitement, Recherche et Aide.

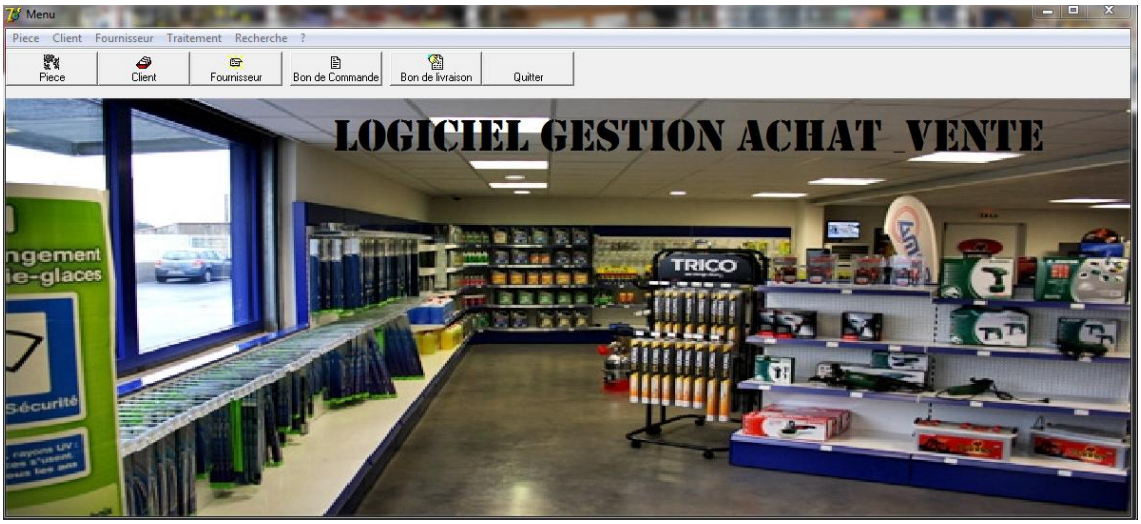

Figure IV.4 : Menu Principal.

c- Formulaire d'Ajouter une pièce

Cette forme permet d'ajouter une nouvelle pièce

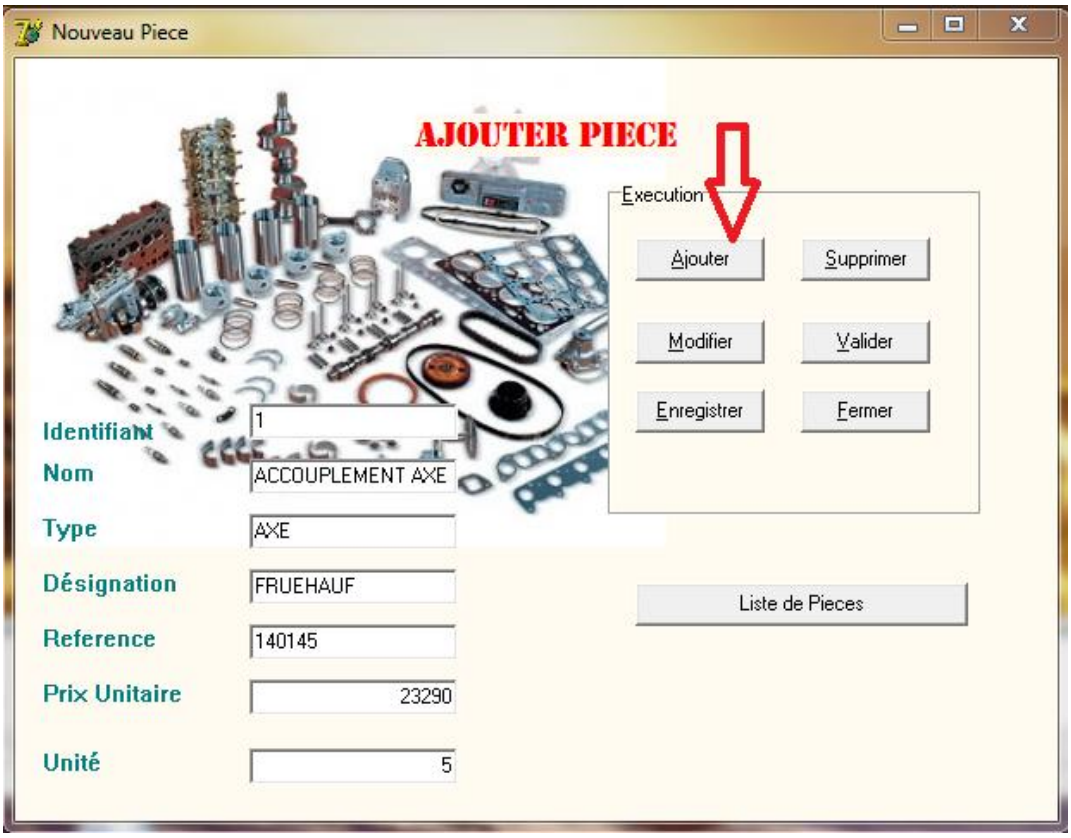

Figure IV.5 : Formulaire d'ajouter une pièce.

d- Formulaire de chercher un Client (par Nom)

Cette forme permet de chercher et afficher si un client existe ou pas par son nom.

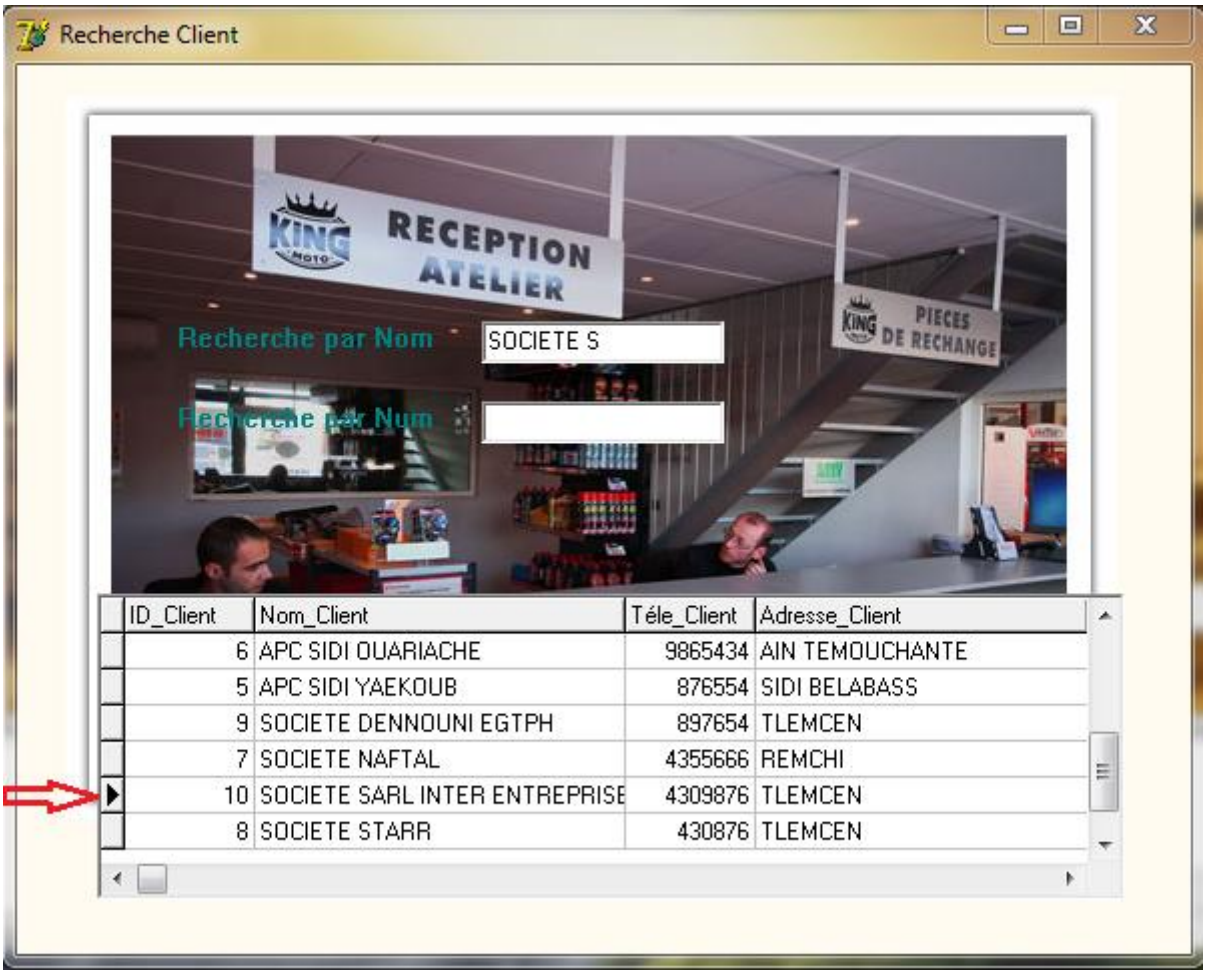

Figure IV.6 : Formulaire de recherche Client.

e- Formulaire de rechercher un Fournisseur (par Num_Four)

Cette forme permet de chercher et d'afficher, si le fournisseur existe ou pas par son identifiant.

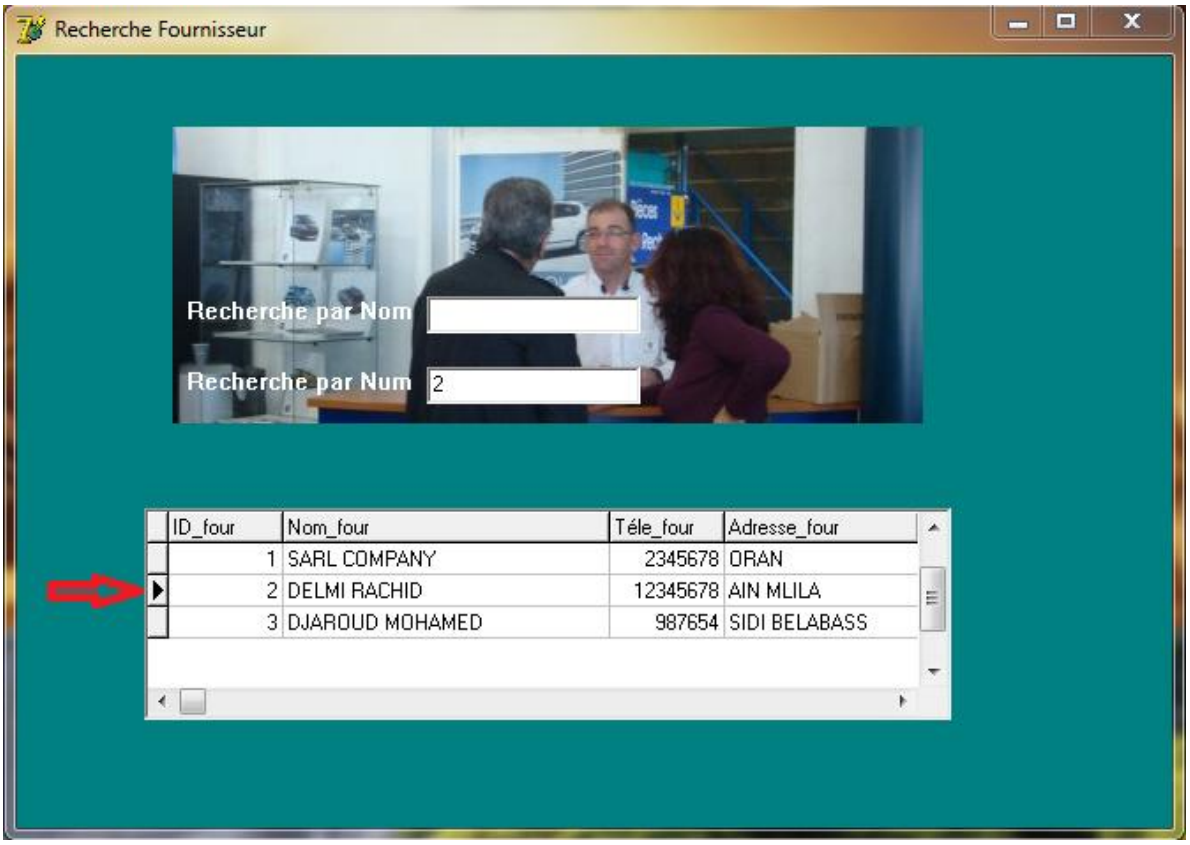

Figure IV.7 : Formulaire recherche Fournisseur.

f- Formulaire Bon de Livraison Client

Cette forme va aider le magasin à connaître si la livraison est conforme à la commande du client.

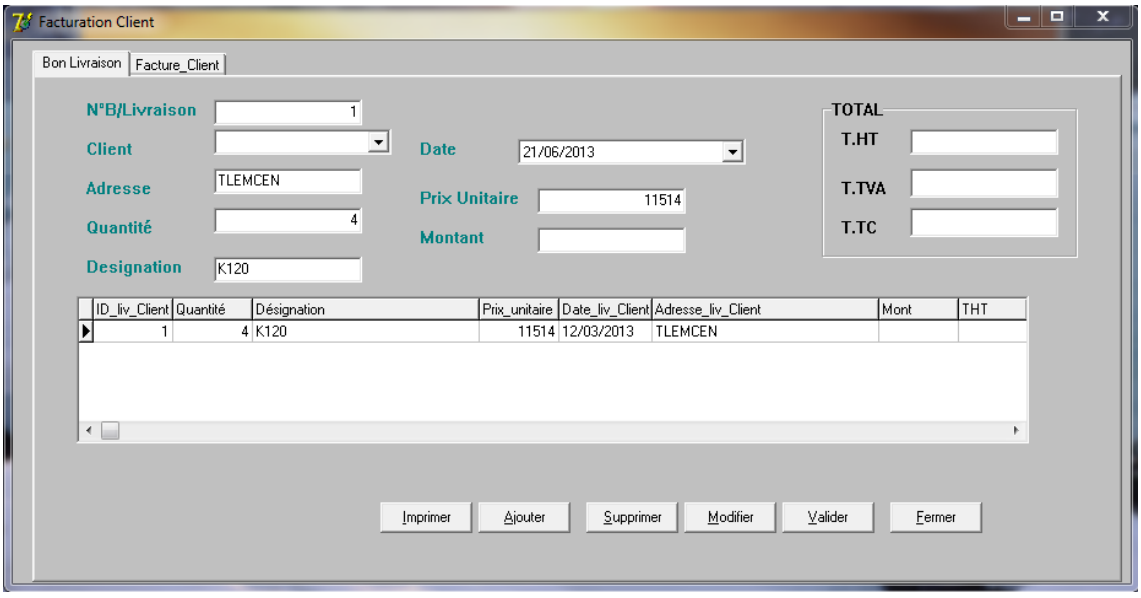

Figure IV.8 : Formulaire de Bon De Livraison.

g- Formulaire Facture Client

Cette fenêtre représente le traitement de facture de client chez le magasin ;

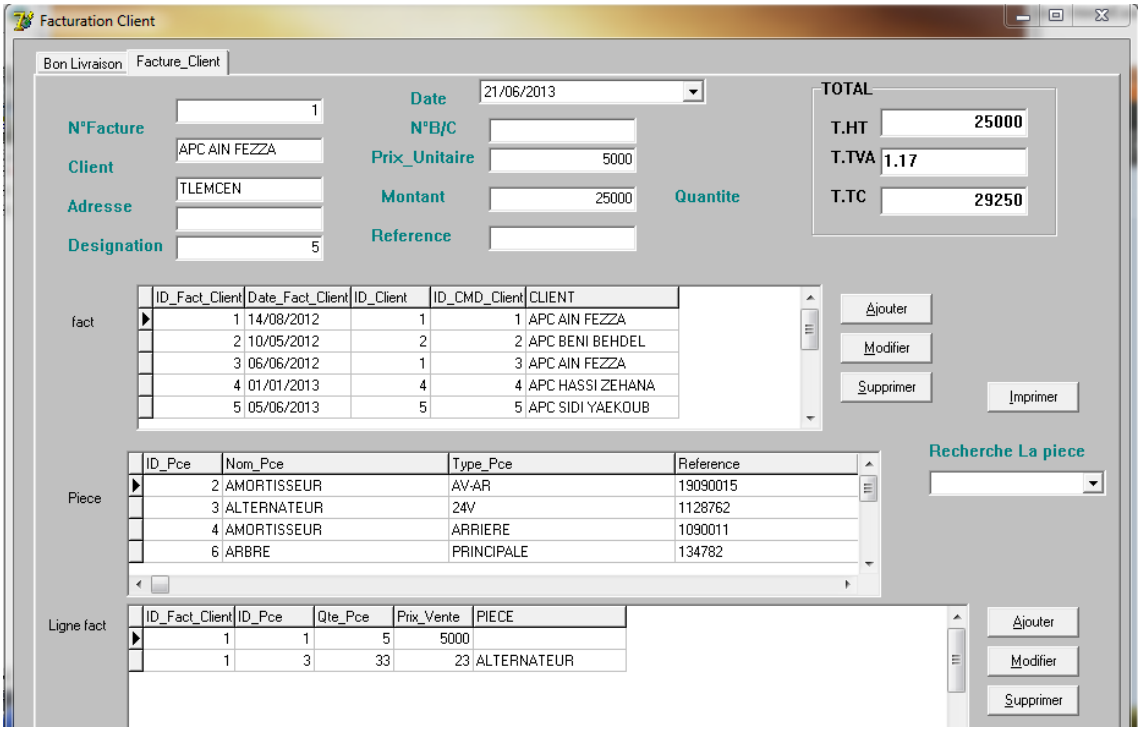

Figure IV.9 : Formulaire Facture Client.

h- Formulaire de Bon de Commande Fournisseur

Cette forme représente le traitement de Bon de Commande imprimé au fournisseur

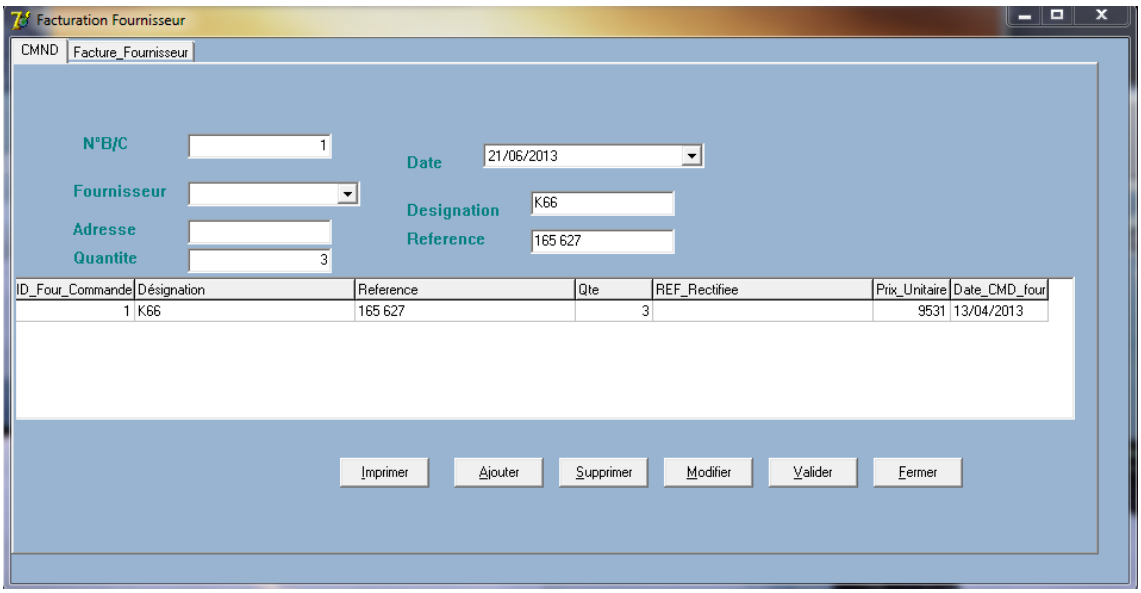

Figure IV.10 : Formulaire Bon de Commande Fournisseur.

i- Formulaire modifier mot de passe

Cette fenêtre permet de modifier le mot passe de l'application.

Figure IV.11: Formulaire modifier mot de passe.

IV.5- Conclusion

Dans ce dernier chapitre, nous avons réalisé une application de la gestion d'un magasin de pièces de rechange en utilisant le langage pascal objet.

La mise en place de cette modeste application nécessite un environnement de développement comme Delphi dont nous avons profité de tous ces avantages qu'il possède. Et enfin, nous avons finis par présenter les interfaces les plus importantes.

Conclusion générale

Généralement quand on commence quelque chose, on attend patiemment le jour où elle va prendre sa fin. Nous avons commencé notre projet intitulé «Application de gestion du magasin de pièces de rechange» et nous arrivons à sa fin.

Notre travail était composé de quatre chapitres repartis comme suit :

- L'introduction générale qui comprend : la problématique, l'intérêt et le choix du sujet, les objectifs du travail, la méthodologie de recherche et la subdivision du travail.
- Le premier chapitre explique le fonctionnement du système d'achat et de vente des pièces de rechanges.
- Le deuxième chapitre explique le langage de modélisation UML et le processus unifié UP.
- Le troisième chapitre comprend la conception de nouveau système GMPR (gestion d'un magasin de pièces de rechange) par le langage UML et le processus UP.
- Le quatrième chapitre présente le nouveau système, ses exigences, les résultats obtenus et les outils utilisés pour mettre en place l'application. L'application est intégrée par la programmation orientée objet (environnement Delphi7).

Cette application va faciliter les activités de gestion d'achat et de vente des pièces de rechange en suivant tous les dossiers.

Ce projet a contribué à améliorer nos connaissances dans plusieurs domaines surtout en conception objet et d'apprendre un nouveau langage de modélisation.

Nous avons utilisé UML pour modéliser le système et le langage Pascal Objet dans l'environnement Delphi pour implémenter l'application.

Le langage UML a permis à notre application d'éclairer les spécifications (limité le domaine de besoins) et d'éviter les remises en cause (le gain du temps dans la réalisation).

Le diagramme de classe obtenu centralise l'organisation des classes de conception, c'est lui qu'on a transformé en code source. Le diagramme de cas d'utilisation nous a montré le fonctionnement du système vis-à-vis des utilisateurs.

Pour maîtriser le développement d'une application sous Delphi, il est indispensable d'aborder les trois sujets suivants :

- le langage Pascal et la programmation orientée objet ;
- l'Environnement de Développement Intégré (EDI) de Delphi ;
- les objets de Delphi et la hiérarchie de classe de sa bibliothèque.

Nous complèterons cette approche par la connexion aux bases de données avec Delphi.

Le but de cette étude est de réaliser et concevoir une application de la gestion du magasin, nous avons insisté à terminer notre travail en diminuant toutes les anomalies existantes, nous espérons avoir fait un travail qui sera pris en considération par les utilisateurs afin de satisfaire leur besoin et qui sera aussi proposé comme guide pour toute personne désirant aborder un problème de gestion.

On espère que notre travail sera complété par d'autres étudiants pour être adapté au besoin de l'utilisateur.

Liste des figures

CHAPITRE I

CHAPITRE II

CHAPITRE III

CHAPITRE IV

Références bibliographiques

- [1] : Olivier Sigaud, Introduction à la modélisation orientée objets avec UML Edition 2005-2006.
- [2] : Laurent Piechocki, cours UML.
- [3] : SMAHI Mohammed Ismail, conception et réalisation d'un système de gestion de demande de formation pour une entreprise FDFE, projet de fin d'études.
- [4] : Frédéric Beaulieu, guide pascal et Delphi, Edition 2000, Mis à jour 2008.
- [5] : www.Developpez.com.
- [6] : Jérôme Darmont, Programmation sous Delphi, Edition 1999-2000.
- [7] : http://fbeaulieu.developpez.com/guide.
- [8] : http://fr.wikipedia.org.

ملخص

ان الغرض من هذا المشروع هو تنفيذ برنامج لإدارة محل لقطع الغيار ، الذي يمكن من معالجة فواتير و أوصال دفع الزبائن وطباعة أوصال طلبات الموردين .

بدأنا من خلال دراسة النظام القائم عن طريق مقابلات و ملاحظات.

قدم تصميم نظام GMPRعن طريق استعمال UML و استنادا الى عملية UP.

وأخيرا تم انجاز البرنامج باستعمال دالفي7 .

الكلمات المفتاحية : ادارة محل ، UML ، UP ، Delphi7 .

Résumé

L'intérêt de notre projet est de réaliser une application d'un magasin de pièces de rechange qui permet de traiter les factures et les bons de livraison des clients et créer des bons de commandes aux fournisseurs.

On a commencé par l'étude du système existant avec des interviews et des observations.

La modélisation du système GMPR a été faite par le langage UML en s'appuyant sur le processus UP. L'implémentation a été faite avec l'environnement Delphi7.

Mots-clés : Gestion du magasin, UML, UP, Delphi7.

Summary

The interest of this project is to develop an application for a store of spare parts that can process invoices and deliveries to the customers and create the suppliers commands.

We began by studying the existing system with interviews and observations.

GMPR modeling system was made by the UML based on the UP process.

The implementation was done with Delphi7.

Keywords: store management, UML, UP, Delphi7.