Ministère des Enseignements Secondaire,
Supérieur et de la Recherche Scientifique
(MESSRS)

Université Polytechnique de Bobo-Dioulasso
(UPB)

Ecole Supérieure d'Informatique (ESI)
URL:// esi.bf.refer.org
E mail : esi@bf.refer.org

Cycle des Ingénieurs de Travaux
Informatique (CITI)

Option : Analyse et Programmation

Année Académique 2006-2007

SOCIETE NATIONALE
DE TRANSIT DU BURKINA
SNTB

SDV

SETO
SOCIETE D'EXPLOITATION DU
TERMINAL DE OUAGADOUGOU

Projet de fin de cycle

Période : du 20 Août au 19 Novembre 2007

GESTION DE LA HOTLINE :
Gestion du Parc Informatique et Gestion des Incidents

Présenté par :

FARMA Amedée Abdramane

SANOU Zézouma Alfred

Superviseur :
Mr COULIBALY Charles
Informaticien à la BCEAO

Maître de stage :
Mr AOUENDE Karim
Ingénieur de conception en Informatique

SOMMAIRE

REMERCIEMENTS

- Nous adressons nos plus vifs remerciements à Mr le **Directeur Général** et à Mr **le Directeur Administratif et Financier** pour nous avoir accepté dans leur entreprise;

- Nous remercions tout particulièrement le chef du service informatique, Mr **SANON Lamine** pour avoir placé sa confiance en nous et pour avoir mis personnellement à notre disposition les moyens humains et matériels nécessaires au bon déroulement de notre stage ;

- Nous adressons également des remerciements spéciaux à Mr **AOUENDE Karim** qui a su nous apporter un suivi et un soutien sans faille lors de notre stage ;

- A tout le personnel du service Informatique, nous disons un grand merci pour leur disponibilité et leur accueil ;

- Nous tenons à témoigner toute notre gratitude à l'ensemble du personnel de la SDV, de la SNTB et de la SETO qui nous ont permis d'effectuer notre stage dans d'excellentes conditions ;

- Nous souhaitons remercier aussi notre Superviseur, Mr **COULIBALY Charles** pour son suivi quotidien de l'évolution du travail et l'ensemble du corps professoral ainsi que la Direction de l'Ecole Supérieure d'Informatique (**ESI**) qui nous ont fourni une formation de qualité ;

- Nous voudrions également remercier nos parents pour leur soutien moral et financier tout au long de notre cycle ;

- Nous remercions au même titre que ceux déjà énumérés nos amis et nos camarades d'école pour leur soutien constant.

INTRODUCTION GENERALE

Dans un souci d'offrir une formation de qualité répondant aux multiples exigences professionnelles du monde informatique, L'**E**cole **S**upérieure d'**I**nformatique (**ESI**) intègre dans le cursus de formation de ses étudiants du **C**ycle des **I**ngénieurs de **T**ravaux **I**nformatiques (**CITI**) option **Analyse et Programmation**, deux stages pratiques en entreprise.

Le premier vise la mise en pratique des connaissances en programmation et a lieu à la fin de la deuxième année. Le deuxième, offre aux étudiants, en fin de cycle, l'occasion de mener une étude complète d'analyse et de conception informatique. Ce stage dure trois (03) mois et fait l'objet d'une soutenance. C'est dans ce cadre que nous avons été accueillis par **SNTB** pour mener une étude sur le thème : **« Gestion de la Hotline Locale : Gestion des incidents et du Parc Informatique ».**

Ce présent rapport présente l'essentiel du travail que nous avons eu à effectuer durant ces trois (03) mois de stage.

CHAPITRE I : GENERALITES

INTRODUCTION

Ce chapitre a pour objectif principal la présentation de notre structure d'accueil et du thème de notre étude. Nous allons également faire cas de notre démarche d'analyse pour mener à bien le projet. Enfin, nous présenterons les différents acteurs du projet et nous donnerons un planning prévisionnel du déroulement des différentes phases de l'analyse.

I. Présentation de la structure d'accueil

I.1. Présentation de la SDV-Burkina

Créée en 1962, **SCAC-DELMAS-VIELJEUX**, plus connu sous le nom de **SDV** est une société anonyme à caractère commercial, affiliée au célèbre groupe **BOLLORE.**

Elle est une des premières sociétés de transit du Burkina Faso dont la dénomination était **SOCOPAO** (Société Commerciale des Ports d'Afrique Occidentale) jusqu'à 1992. A partir de cette date, elle fut fusionnée avec la compagnie maritime DELMAS-VIELJEUX et prit le nom de SDV-Burkina.
La SDV-Burkina est présente sur le territoire Burkinabé principalement à Ouagadougou et à Bobo-Dioulasso, la filiale de la société mère étant implantée en Europe.

Son Capital social est de **cinq cent vingt et un millions deux cent mille francs CFA (521.200.000 FCFA).** Son siège social est situé à **Ouagadougou au 474-rue Ilboudo Waogyandé.**

I.2. Présentation de la SNTB

La SNTB a été créée dans les années 47 par les anciens colons français sous le nom de **TRANSAFRICAINE**. En 1971, dans le cadre de la nationalisation des sociétés privées, elle prendra le nom de **Société Voltaïque de Groupage (SOVOG)**.

Sa dénomination actuelle **« Société Nationale de Transit du Burkina (SNTB) »** résulte de l'avènement de la révolution qui a initié des changements à travers la prise en compte de la majorité de son capital par l'Etat Burkinabé et ses démembrements. Elle devient alors à la date du 29 /01/1985 une société d'économie mixte par décret N°85-039/CNPR/PRESS/MCC.

Dans le cadre du désengagement de l'Etat du capital des sociétés, la SNTB a été privatisée et a ensuite absorbé la Société Africaine de Groupage (SAG) en juin 1995. Elle redevient alors une Société Anonyme au **capital de trois cent quatre vingt dix millions de francs CFA (390 000 000 FCFA)**.

La SNTB est représentée dans le monde entier à travers le **Groupe SAGA**. Son chiffre d'affaire est de **deux milliards six cent soixante seize millions de francs CFA (2 676 000 000 FCFA).** Elle fait aussi partie du groupe Bolloré.

I.3. Présentation de la SETO

La **SETO (Société d'Exploitation du Terminal de Ouagadougou)**, est l'une des trois sociétés du groupe BOLLORE qui sont prises en compte dans notre étude. C'est une société anonyme qui a été créé le 1er septembre 1999. Elle a pour principale activité la manutention des conteneurs. Son chiffre d'affaire en 2006 s'élevait à **cent quatre vingt treize millions six cent trente huit mille cent douze francs CFA (193 638 112 FCFA)** et son capital est de **deux cent millions de francs CFA (200 000 000 FCFA)**. Ses principaux actionnaires sont :

- SNTB et SDV-Burkina qui interviennent chacune à hauteur de **vingt sept virgule cinquante pour cent (27,50%)** du capital ;
- CCIA et MAERSK B qui interviennent elles, à hauteur de **vingt pour cent (20%)** du capital chacune ;
- CBC qui détient quant à lui, **cinq pour cent (5%)** du capital.

I.4. Structuration des Sociétés

La SDV-Burkina, la SNTB et la SETO sont organisées de la manière suivante :

- **Le Conseil d'Administration :** situé au sommet de la hiérarchie de l'organigramme, le Conseil d'Administration est le garant (moral et technique) de la bonne gestion de la société. Il se réunit en Assemblée Générale Ordinaire et Extraordinaire ;

- **La Direction Générale :** elle est en charge de la gestion quotidienne de la société et a à sa tête un Directeur Général. Sous l'autorité du Conseil d'Administration, la Direction Générale est le sommet de l'entreprise et a sous sa tutelle le siège social de Ouagadougou et l'agence de Bobo-Dioulasso ;

- **La Direction Administrative et Financière :** elle est sous l'autorité de la Direction Générale et le Directeur Administratif et Financier en est le responsable. Outre la gestion des affaires de la société en matière d'administration et de finance, la Direction Administrative et Financière est investie des pouvoirs qui lui permettent d'élaborer le budget annuel, de seconder et d'assurer l'intérim de la Direction Générale ;

- **Le service Exploitation :** Il est directement rattaché à la Direction Générale et s'occupe des activités de la société à savoir le transport, la consignation, le transit et la manutention ;

- **Le service Commercial et Shipping :** Ce service s'occupe de l'import et de l'export multimodal (maritime, routier, ferroviaire). Il est dirigé par un responsable de service qui coordonne et contrôle les activités du service. Le Chef de service veille surtout à l'exécution des taches dans les plus brefs délais ;

- **Le service Transit :** Il comprend les services Transit maritime et Transit aérien. Chacun de ces services se scinde en quatre (04)

sections (Ouverture de dossier, Déclaration, Passeur en douane et Livraison).

NB : Voir organigrammes annexe 3, 4 et 5 à partir de la page 112

I.5. Le Service Informatique

Ce service se trouve au centre des trois sociétés ci-dessus citées. En effet, c'est ce même Service Informatique qui gère l'ensemble des ressources informatiques de ces trois sociétés.

Il s'occupe de l'administration dans le domaine informatique en deux volets : le volet micro-informatique, permettant l'application de la bureautique et le volet mini-informatique c'est-à-dire l'informatique reliée au serveur AS400[1] (écran passif). Tous ces deux volets sont en réseau mondial (reliant tout le groupe) et en réseau national (reliant l'ensemble des sites du pays).

Le service informatique s'occupe également de l'assistance quotidienne des utilisateurs (réparation des pannes, installation des logiciels et des anti-virus, administration du réseau etc.).

II. Description du thème et Problématique

II.1. Description du thème

La mise en place d'une hotline locale informatique pour assurer une gestion des problèmes informatiques de la SNTB, la SDV et la SETO au niveau du Burkina Faso, est une demande de la Direction des Systèmes Informatique du groupe BOLLORE depuis juin 2000. Cela pour répondre aux trois objectifs suivants :

- Avoir une meilleure maîtrise des problèmes informatiques sur l'ensemble des trois sociétés citées plus haut ;
- Une célérité dans la résolution des incidents informatiques ;
- Une meilleure gestion du patrimoine informatique des trois sociétés.

[1] AS/400 est une gamme de mini-ordinateurs IBM apparue début février 1987. C'est un serveur mini fonctionnant avec le système d'exploitation OS400.

D'où le thème que nous avons à traiter : « **GESTION DE LA HOTLINE :**

- Gestion des incidents

- Gestion du parc informatique. »

En outre, la hotline qui sera mise en place prendra en charge les sites des villes de Ouagadougou et Bobo-Dioulasso.

II.2. Problématique

L'ensemble des trois sociétés que sont la SNTB, la SDV et la SETO dispose d'un important lot de matériels informatiques qui sont gérés par un même service informatique. A l'heure actuelle, il n'existe pas de mécanisme fiable pour assurer un suivi du mouvement des équipements informatiques. Il est également difficile pour le service informatique de garder une trace des incidents qui surviennent. Toutes ces difficultés sont liées au fait que la gestion est toujours manuelle.

Il nous revient alors de proposer une solution informatique à même de résoudre les difficultés que connaît le service informatique dans la gestion des incidents et du parc informatique. Pour ce faire, nous allons développer une application qui permettra d'automatiser certains traitements liés à la gestion de la hotline et ce logiciel devra permettre de :

- Faire le point du parc informatique à tout moment ;
- Identifier de manière unique chaque équipement du parc ;
- Localiser de façon précise chacun de ces équipements ;
- Suivre tous les incidents matériels et logiciels ;
- Faire des statistiques des différents incidents ;
- Mémoriser la description de chaque incident.

III. Méthode d'analyse et de conception

La conception d'un logiciel passe par l'emploi d'une méthode qui s'appuie sur un langage de modélisation. La méthode de développement est une succession d'étapes qui permettent de maîtriser le déroulement du projet et qui donnent aux utilisateurs une meilleure visibilité de certains résultats produits et garantissent ainsi l'adéquation du résultat final. Chaque étape de la méthode employée est

ponctuée par la création d'un modèle du système étudié. Un modèle est une représentation abstraite selon un certain formalisme de la réalité qui exclut certains détails du monde réel.

Il existe deux approches principales dans la conception des systèmes d'informations : l'approche fonctionnelle et l'approche orientée objet. L'approche fonctionnelle perçoit le système comme une entité réalisant une fonction globale que l'on peut décomposer en sous-fonctions. L'approche objet, elle, appréhende le système comme étant un ensemble d'objets qui interagissent entre eux.

Dans le cadre de notre étude nous avons opté pour l'approche par objet qui assure une évolution du logiciel et une réutilisation des objets, ce qui n'est pas garanti avec l'approche fonctionnelle. Pour ce faire nous avons choisi **UML** comme langage de modélisation et le processus **2TUP** comme méthode de développement.

III.1. Le langage UML

UML ou Unified Modeling Language, est un langage de modélisation qui est né au milieu des années 90 de la fusion de trois méthodes objets : OMT (Object Modeling Technique), BOOCH[1] et OOSE (Object Oriented Software Engineering). L'idée de cette fusion est partie du constat qu'à l'époque il existait plusieurs méthodes objets liées par un consensus autour d'idées communes : objets, classes, sous-systèmes etc.
C'est à partir de 1997 que l'OMG[2] (Object Management Group) qui standardise les technologies de l'objet, s'est attachée à la définition d'un langage commun unique, utilisable par toutes les méthodes objets dans toutes les phases du cycle de vie et compatible avec les techniques de réalisation du moment. D'où la naissance d' UML. UML offre des éléments pour décrire les différents aspects d'un système : les diagrammes. Les diagrammes utilisés dans l'ensemble de notre analyse sont :

- Le diagramme des cas d'utilisation
- Le diagramme de classes d'objets
- Le diagramme de package

[1] BOOCH : la méthode BOOCH est avec OMT et OOSE, l'une des méthodes d'analyse et de conception orientée objet à l'origine d'uml. Son nom vient de celui de son concepteur, Grady Booch.
[2] L'OMG est un organisme à but non lucratif, créé en 1989 à l'initiative de grandes sociétés (**HP**, **Sun**, Unisys, American Airlines, Philips...). Il a pour but de mettre au point des standards garantissant la **compatibilité** entre des applications programmées à l'aide de langages objet et fonctionnant sur des réseaux **hétérogènes** (de différents types)

- Le diagramme de séquences
- Le diagramme d'activité

NB : pour plus de détails concernant les concepts et les formalismes d'UML, se référer à l'annexe1 à la page 95.

III.2. Le Processus 2TUP

Le **Two Track Unified Process (2TUP)**, est une variante du processus unifié (UP). Un processus unifié est un processus de développement logiciel construit sur UML. Il est itératif et incrémental, conduit par les cas d'utilisation et piloté par les risques. La gestion d'un tel processus est organisée selon 5 phases :

- Pré étude, capture des besoins et étude
- Analyse et conception préliminaire
- Conception et tests
- Déploiement et tests
- Bilan du cycle de développement.

Le processus unifié n'est pas un processus universel mais une trame commune des meilleures pratiques de développement.
L'objectif du processus 2TUP est de capitaliser le travail effectué sur la branche fonctionnelle d'une part et technique d'autre part avant même de commencer la conception du système souhaité.

Le 2TUP propose un cycle de développement en **Y**, qui dissocie les aspects techniques des aspects fonctionnels. Illustré par la figure1 ci-dessous, le processus en Y s'articule autour de 3 phases :

- une **branche technique** : qui a pour objectif de recenser les besoins techniques et d'élaborer une architecture logicielle et applicative ;
- une **branche fonctionnelle** : elle a pour but de dégager les grandes fonctionnalités et les frontières du système ;
- et une **phase de réalisation** : qui consiste en la conception et en la mise en œuvre du système.

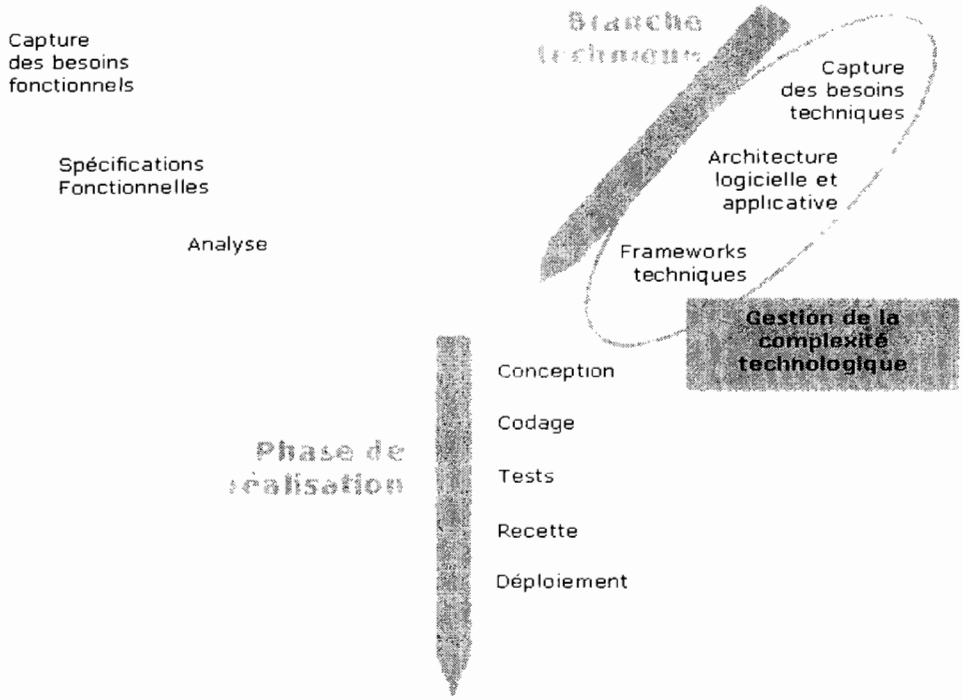
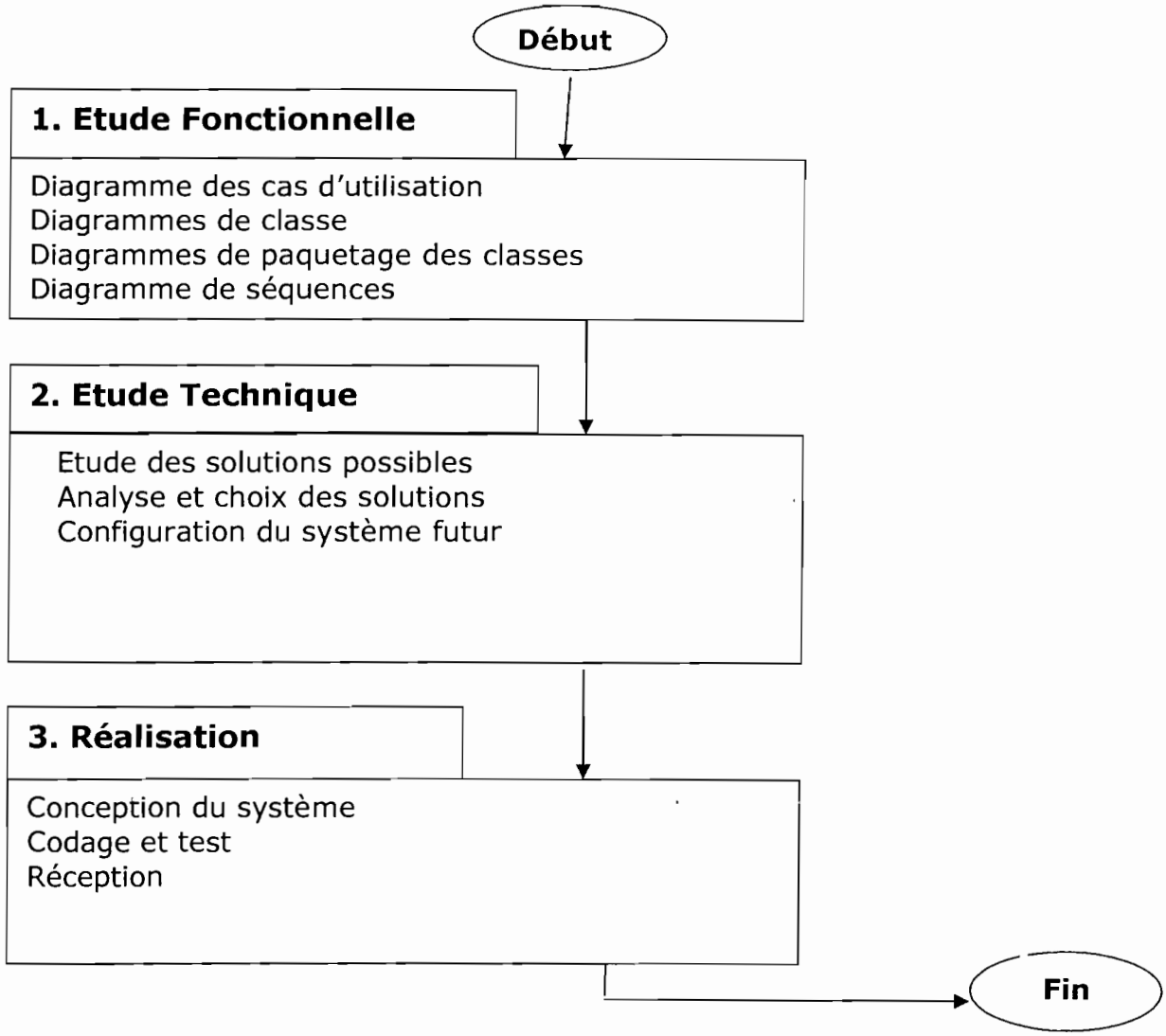


Figure 1 : cycle de développement en Y

IV. Démarche d'Analyse



V. Les Acteurs du Projet

	Rôles	Membres
Groupe de pilotage	- prend les décisions relatives aux objectifs ; - fixe les orientations générales et les délais à respecter ; - définit les moyens à mettre en place pour la réalisation du projet.	- M. Lamine SANON - M. A Abdoul Karim
Groupe de projet	- charger de l'exécution du projet ; - charger de la conception et éventuellement de la réalisation du projet.	- M. Z. Alfred SANOU - M. A. Amédée FARMA
utilisateurs	- fournir toutes les informations nécessaires à la bonne conduite du projet ; - intervient dans la validation des dossiers d'étude produits par le groupe de projet.	Tous les membres du service informatique.

Tableau1: les acteurs du projet

VI. Planning Prévisionnel

Etape	Dossier à produire	Période	Durée
Lancement	Note de lancement	20 août au 02 septembre 2007	14 jours
Etude fonctionnelle	Dossier de l'étude fonctionnelle	03 au 23 septembre 2007	21 jours
Etude technique	Dossier de configurations du système	24 septembre au 14 octobre 2007	21 jours
Conception	Dossier de conception	15 au 28 octobre 2007	14 jours
Codage et test	Dossier de programmation et guide utilisateur	1er novembre 2007 au 30 avril 2008	6 mois

Tableau 2 : planning prévisionnel

CONCLUSION

Cette première étape nous a permis de nous familiariser avec notre structure d'accueil et aussi de bien nous imprégner du thème d'étude. Nous avons pu définir une démarche d'analyse qui va nous permettre de réaliser notre projet. Dans la suite du travail, nous nous emploierons à suivre les différentes étapes de notre démarche.

CHAPITRE II : LES ETUDES FONCTIONNELLE ET TECHNIQUE

INTRODUCTION

Dans le chapitre précédent, nous avons décrit le processus ***2TUP*** que nous allons utiliser pour réaliser le projet. Ce processus est décomposé en deux branches d'étude : la branche fonctionnelle et la branche technique. Ce chapitre nous présentera l'étude du système selon ces deux axes.

I. Etude de l'existant

L'objectif de ce paragraphe est de permettre au groupe de projet, d'une part, de s'imprégner du fonctionnement actuel de la hotline et d'autre part, de recueillir les informations auprès du groupe de pilotage et de référencer les souhaits des utilisateurs.

A notre arrivée, le service informatique ne disposait pas d'un système informatisé de gestion de la hotline. En effet, pour la gestion des incidents et du parc informatique, il avait recours au papier ou au tableur MS Excel pour mémoriser les incidents et les affectations de matériels. Après chaque intervention, l'agent qui a intervenu saisit ses références personnelles et ceux concernant l'incident dans un formulaire conçu en Excel (confert Annexe 6 page : 116). Ce formulaire est imprimé et mis dans les archives du chef informatique.

Cette méthode de gestion révèle de nombreuses insuffisances telles que :

- ✓ La difficulté de faire des recherches sur des incidents et des matériels ;
- ✓ La difficulté d'établir des inventaires ;
- ✓ La contrainte des agents du service informatique de saisir les mêmes informations à chaque fois ;
- ✓ Etc.

Tous ces problèmes sont liés à l'absence d'une base de données fiable. Il est donc plus que nécessaire pour le service informatique de se doter d'une solution informatique capable de répondre aux exigences qui deviennent de plus en plus nombreuses et complexes.

II. Etude fonctionnelle

L'étude fonctionnelle à pour but de définir les contraintes fonctionnelles c'est-à-dire ce que notre système doit réaliser en terme de métier. Elle comprend la capture des besoins fonctionnels et l'analyse et la spécification

II.1. Capture des besoins fonctionnels

La capture des besoins permet d'identifier les acteurs et les fonctionnalités réalisées par le système. Elle produit un modèle des besoins focalisé sur le métier des acteurs. La technique des cas d'utilisation d'UML est au centre de cette phase du processus 2TUP. La démarche ici consiste en :

- ✓ L'identification des acteurs ;
- ✓ L'identification des cas d'utilisation ,
- ✓ La description des cas d'utilisation.

II.1.1. Identification des acteurs

Un acteur est une personne ou un système qui interagit avec le système étudié, en échangeant de l'information (en entrée et en sortie). On trouve les acteurs en observant les utilisateurs directs du système, les responsables de sa maintenance, ainsi que les autres systèmes qui interagissent avec lui. Les acteurs de notre système ici sont :

- Les **agents informatiques** : il s'agit du chef du service informatique et des autres membres du service informatique. Ce sont les acteurs principaux du système car ils interagissent directement avec le système (consultation et mise à jour des données du système);

- **Les utilisateurs :** il s'agit des personnes qui utilisent le matériel informatique. Ce sont des acteurs secondaires car ils n'ont pas d'interaction

directe avec le système. Mais ils peuvent déclencher des évènements qui vont modifier l'état du système ;

- **Les services extérieurs :** il s'agit de la DSI[1] et des prestataires extérieurs. Ce sont aussi des acteurs secondaires et ils n'interviennent que dans la résolution des incidents.

II.1.2. Identification des cas d'utilisation

Les cas d'utilisation sont une technique de description du système étudié privilégiant le point de vue de l'utilisateur. Ils délimitent le système, ses fonctions (ses cas), et ses relations avec son environnement. Ils constituent un moyen de déterminer les besoins du système. Ils permettent d'impliquer les utilisateurs dès les premiers stades du développement pour exprimer leurs attentes et leurs besoins (analyse des besoins). Ils constituent un fil conducteur pour le projet et la base pour les tests fonctionnels.

Pour le cas de notre système les cas d'utilisation (CU) que nous pouvons retenir sont :

CU 01 : **Authentification ;**

CU 02 : **Affectation Provisoire ;**

CU 03 : **Affectation Définitive ;**

CU 04 : **Traiter Incident.**

[1] DSI (Direction des Systèmes d'Information) : tutelle technique au niveau du groupe Bolloré de l'équipe locale

II.1.3. Diagramme des cas d'utilisation

Le diagramme des cas d'utilisation de notre système est représenté par la figure ci-dessous :

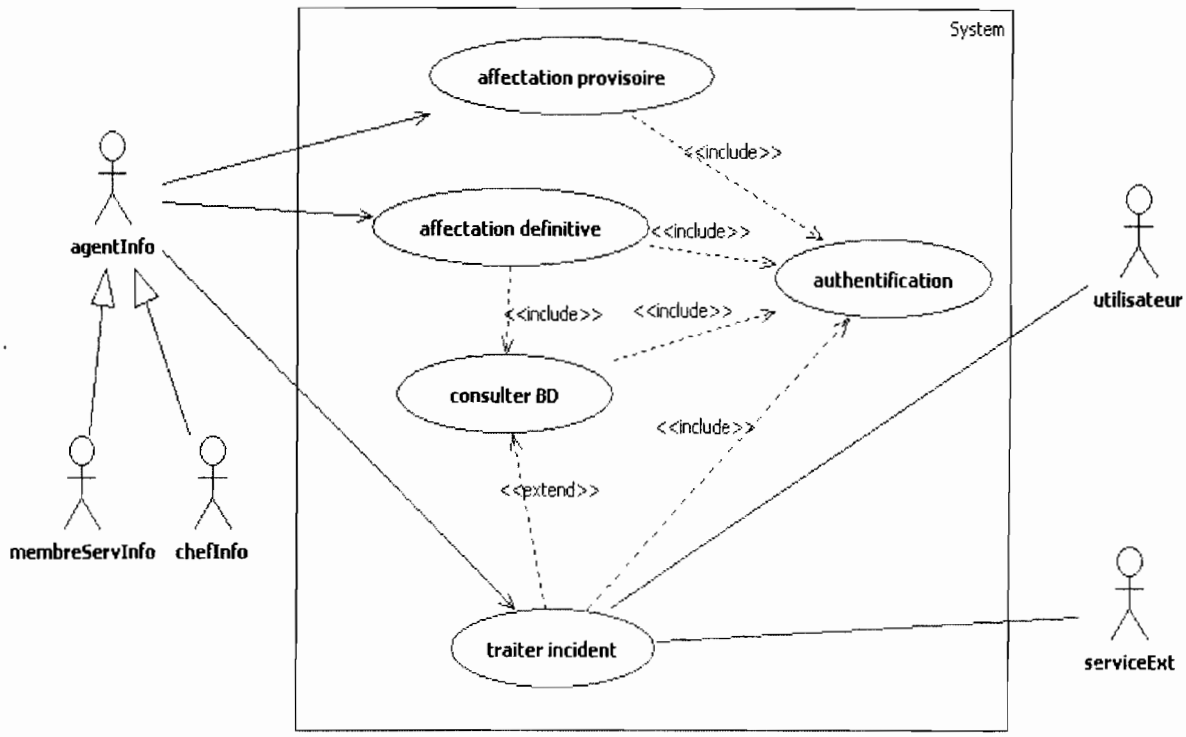


Figure 2 : diagramme des cas d'utilisation de notre système

II.1.4. Description textuelle des cas d'utilisation

UML ne propose pas une forme particulière pour décrire les cas d'utilisation, mais la forme textuelle est la meilleure façon de les décrire. Pour notre part, nous avons choisi d'expliciter les cas d'utilisations de notre système à travers le formalisme suivant :

Cas d'utilisation N° : **« nom cas d'utilisation »**
Résumé : vérification de l'identité des utilisateurs du système.
Acteurs : « acteurs participants au cas d'utilisation »
Pré condition : « condition devant être remplis avant le début du cas d'utilisation »
Scénario nominal : « séquence d'actions normales associées à un cas d'utilisation »
Alternative : « séquence d'action alternatif pouvant conduire également à un succès» **Exception :** « séquence d'action conduisant à un échec »

Tableau 3 : formalisme de description des cas d'utilisation proposé

Cas d'utilisation 1 : **Authentification.**
Résumé : vérification de l'identité des utilisateurs du système.
Acteurs : agent informatique
Pré condition : application active.
Scénario nominal : **[Début]** 1. le système affiche la page de connexion ; 2. l'agent saisit son login et son mot de passe ; 3. le système vérifie la validité des informations fournies ; **(A1) (E1)** 4. le système donne l'accès au menu de l'application. **[Fin]**
Alternative : les informations fournies sont incorrectes **A1 :** le système réaffiche la page de connexion et attend que l'utilisateur saisisse ses informations d'authentification et cela trois (03) fois de suite. **Exception :** l'utilisateur a fourni trois fois des informations erronées **E1 :** l'application se ferme

Tableau 4 : description du cas d'utilisation *authentification*

Cas d'utilisation 2 : traiter incident
Résumé : manière dont les incidents sont résolus
Acteurs : agent informatique.
Pré condition : application active, appel effectué par un utilisateur
Scénario nominal : **[Début]** 1. inclusion du cas d'utilisation << **authentification** >> ; 2. l'agent saisie les informations concernant l'appel et l'incident (localisation, service, identité de utilisateur, description de l'incident, origine…) ; 3. l'agent résout le problème par téleintervention ; **(A1)** 4. il enregistre la procédure de résolution. **(A2)** **[Fin]**
Alternative : **A1 :** l'incident n'est pas répertorié dans la BD ou l'agent n'a pas résolu le problème, et une équipe est envoyée ; **[Début]** 1. l'agent saisit les informations concernant l'équipe d'intervention (identité des membres de l'équipe, date d'intervention…) ; 2. il enregistre ces informations ; 3. l'enchaînement reprend à partir du point **4** du scénario nominal. **[Fin]** **A2 :** l'équipe et l'agent n'ont pas pu résoudre le problème et l'incident est enregistré comme non résolu, et un service extérieur est contacté. **[Début]** 1. inclusion du cas d'utilisation << **authentification** >> ; 2. le service informatique recherche le service compétant pour ce type d'incident ; 3. le chef adresse une correspondance au service en question ; 4. enregistrer intervention. (service intervenant, date d'intervention, nature de l'intervention…) **(A3)**. **[Fin]** **A3 :** le service extérieur n'a pas pu résoudre le problème, alors l'incident est enregistré comme non résolu.

Tableau 5 : description du cas d'utilisation *traiter incident*

Cas d'utilisation 3 : Affectation définitive
Résumé : confirmation d'une affectation.
Acteurs : agent informatique.
Pré condition : matériel à la disposition de son utilisateur et application active.
Scénario nominal : **[Début]** 1. inclusion du cas d'utilisation << **authentification** >> ; 2. l'agent recherche le matériel et son utilisateur initial dans la BD ; 3. le système affiche les références du matériel et de son utilisateur ; 4. l'agent valide l'affectation. **(A1)** **[Fin]**
Alternative **A1 :** le matériel a changé d'utilisateur et/ou de destination **[Début]** 1. l'agent modifie l'utilisateur et la destination du matériel ; 2. il valide sa saisie. **[Fin]**

Tableau 6 : description du cas d'utilisation *affectation définitive*

Cas d'utilisation 4: Affection provisoire
Résumé : mémorisation d'un nouveau matériel et de son futur destinataire.
Acteurs : agent informatique.
Pré condition : arrivée de nouveaux matériels, application active.
Scénario nominal: **[Début]** 1. inclusion du cas d'utilisation << **authentification** >> ; 2. l'agent saisit les informations sur le matériel (caractéristiques du matériel, date d'arrivée…); 3. l'agent saisit les informations sur la destination du matériel (service bénéficiaire, localisation du service, l'utilisateur…); 4. il valide la saisie. **[Fin]**

Tableau 7 : description du cas d'utilisation *affectation provisoire*

II.2. Analyse et spécification des besoins

C'est la dernière étape de l'étude fonctionnelle. Elle a pour objectif de générer le squelette du domaine dans ses grandes lignes mais pas encore en détail. Le modèle d'analyse du domaine définit la structure et le comportement des objets connus dans le métier des utilisateurs du système. Le modèle du domaine sera représenté selon une vue statique et une vue dynamique.

II.2.1. Modèle statique du domaine

Dans ce paragraphe nous définirons les différentes classes candidates des cas d'utilisation que nous avons énumérés plus haut. Cela nous permettra de produire le diagramme des classes participantes à la réalisation du système que nous allons ensuite regrouper en package.

II.2.1.2. Identification des classes participantes au cas d'utilisation

Il s'agit de représenter les classes qui concourent à la réalisation de chaque cas d'utilisation défini plus haut, ainsi que les associations qui les lient.

Classes participantes au cas d'utilisation « traiter incident » :

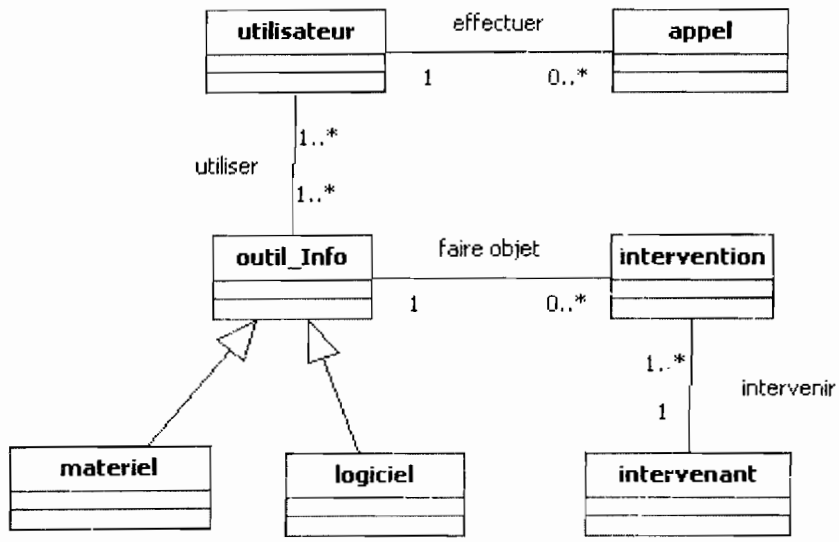


Figure 3 : diagramme des clases participantes au cas d'utilisation "traiter incident"

Classes participantes aux cas d'utilisation « Affectation provisoire » et « Affectation définitive » :

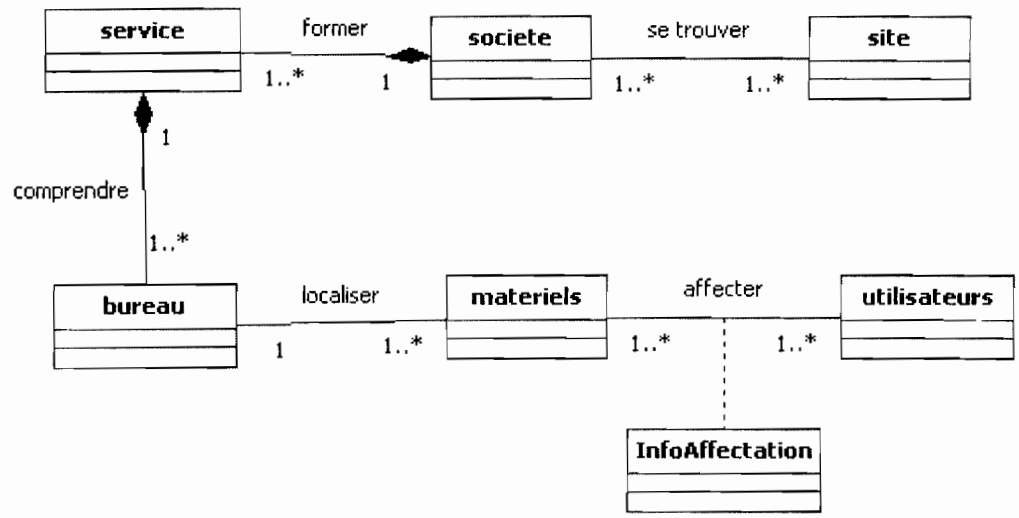


Figure 4 : diagramme des clases participantes aux cas d'utilisation " affectation provisoire" et "affectation definitive"

II.2.1.3. Le diagramme des classes participantes

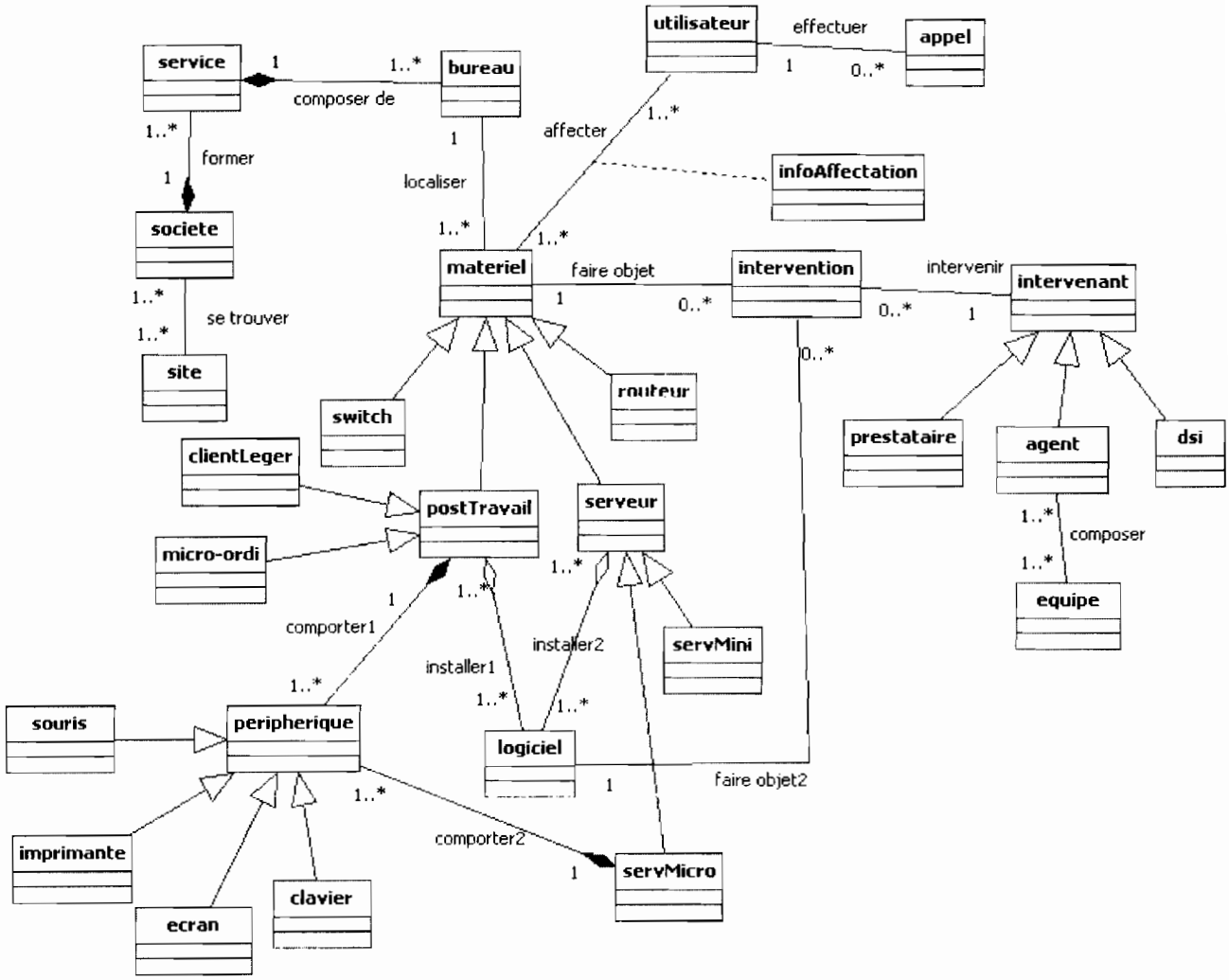


Figure 5 : diagramme des classes participantes

II.2.1.4. Le diagramme de package

Figure 6 : diagramme de packages

I.2.2. Modèle dynamique du domaine

Ce modèle décrit les scénarios qui mettent en jeu les messages échangés entre les acteurs et leurs interactions avec le système. Un scénario est l'exécution d'un ou de plusieurs enchaînements entre le début et la fin normale ou non d'un cas d'utilisation. Les scénarios sont généralement décrits à l'aide d'un diagramme de séquence. Pour notre cas nous présenterons le diagramme de séquence du déroulement normal des cas d'utilisation de notre système.

Les diagrammes de séquence

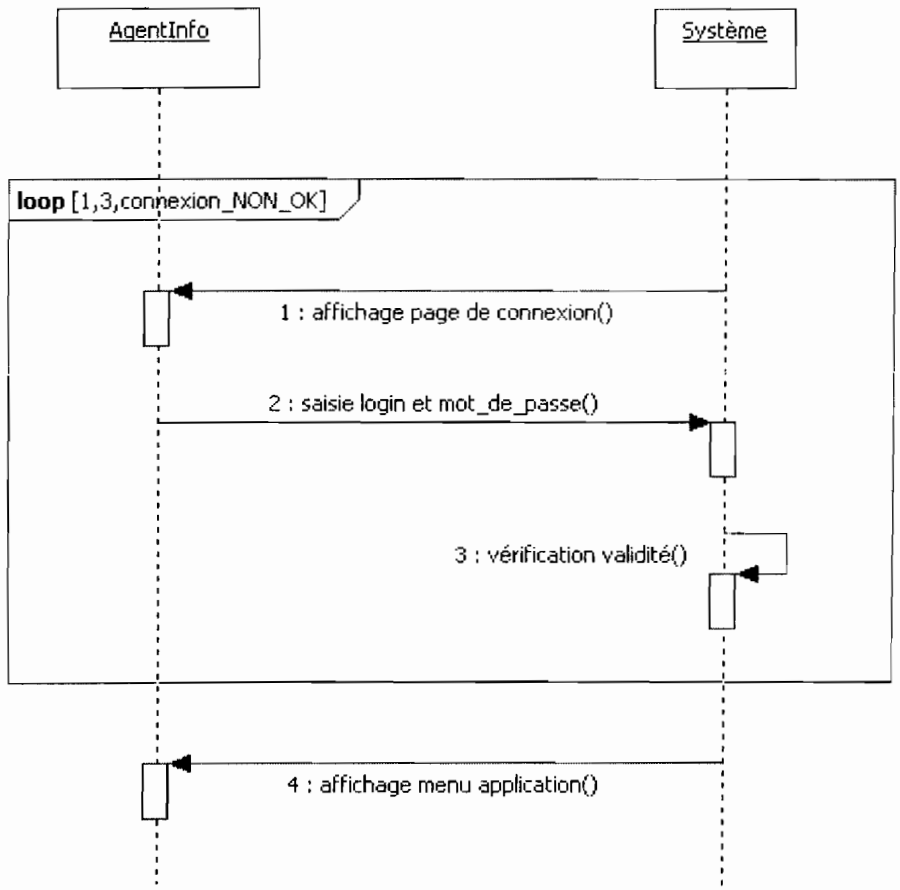


Figure 7 : Diagramme de séquence du CU[1] authentification

[1] CU : Cas d'Utilisation

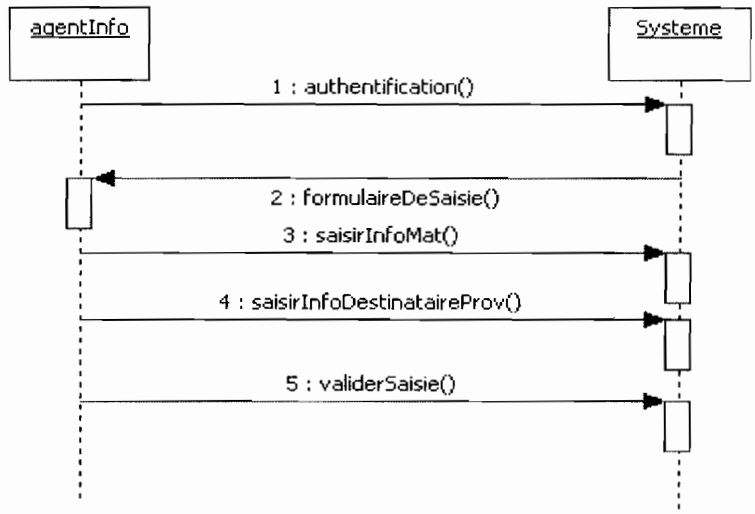


Figure 8 : Diagramme de sequence du CU affectation provisoire

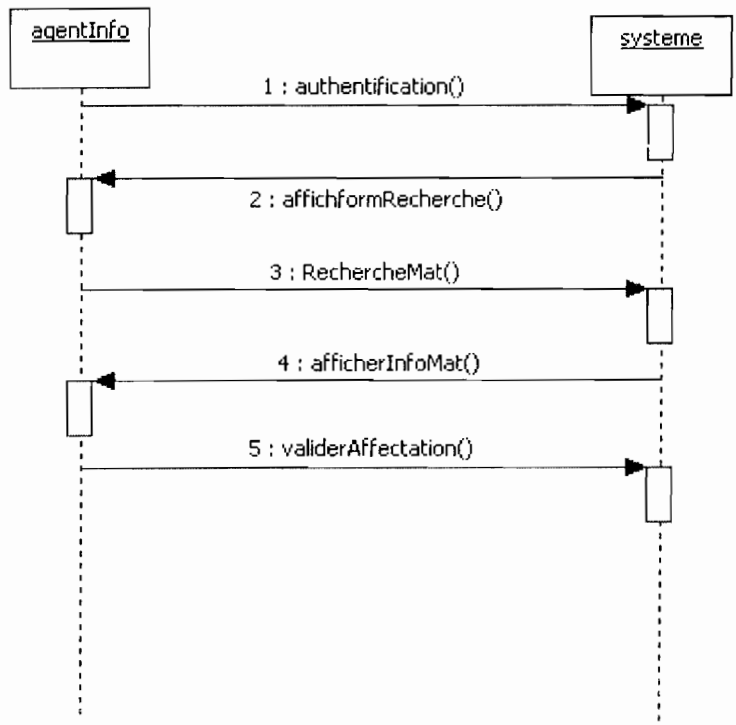


Figure 9 : Diagramme de sequence du CU affectation définitive

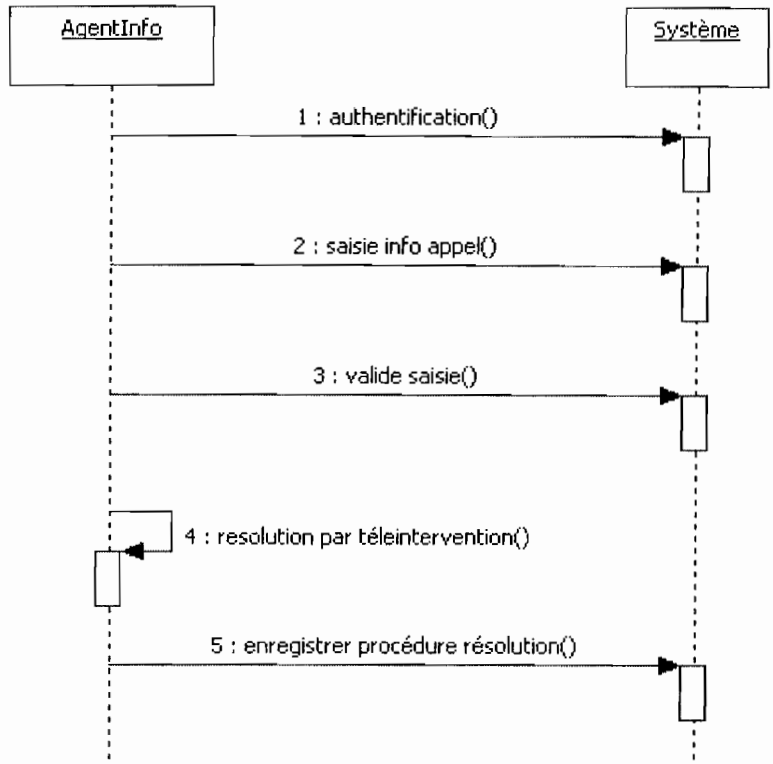


Figure 10 : Diagramme de séquence du CU traiter incident

III. Etude Technique

C'est l'étude suivant la branche de gauche du cycle de développement en « Y ». Elle permet de prendre en compte les contraintes techniques et logicielles du système étudié. C'est au cours de cette phase que nous allons faire le choix de la configuration matérielle et du modèle de développement logique de notre système.

III.1. Présentation des différentes couches de développement

- **le modèle en 5 couches :** c'est une vue logique de l'architecture du système qui préconise un développement en cinq niveaux :

- **Couche présentation :** elle contient l'interface graphique de l'application et assure les interactions entre le système et les utilisateurs ;
- **Couche application :** elle regroupe la logique fonctionnelle de l'application telle qu'elle est définie dans les spécifications fonctionnelles détaillées. La couche Application utilise les services de la couche métier pour réaliser les

fonctionnalités du système, et présente ces fonctionnalités sous la forme de services ;

- **Couche métier :** cette couche correspond aux objets structurant de l'entreprise. Ces objets n'intègrent aucune notion fonctionnelle. Elle regroupe des objets transversaux à toutes les applications. De plus, la couche métier propose des services d'accès à ces objets au travers de méthodes de création, recherche, modification, suppression. Ces méthodes contiennent les règles de gestion associées aux différentes opérations ;
- **Couche accès aux données :** elle sert à relier la couche métier à la couche Physique ;
- **Couche stockage des données** ou **couche physique :** elle correspond à la structure physique des données (base de données).

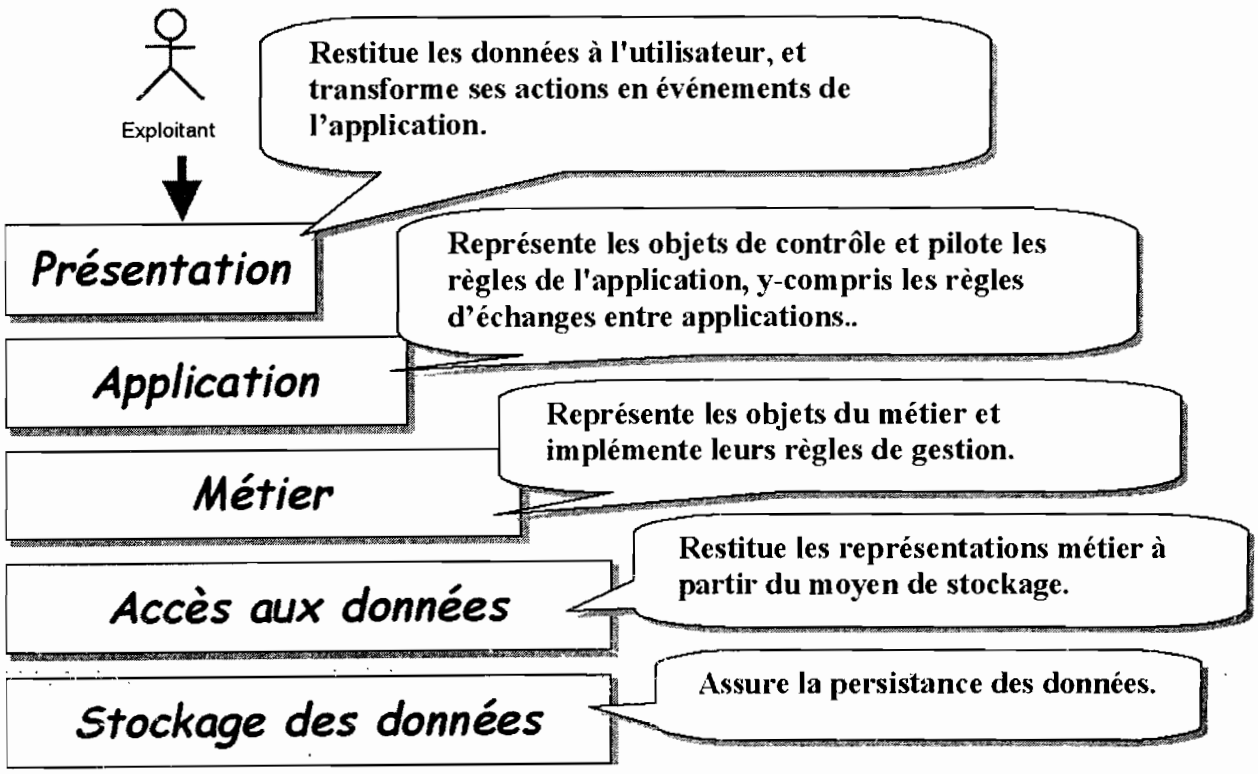


Figure 11 : modèle en 5 couches

- **le modèle MVC :** cette architecture consiste à distinguer trois entités distinctes qui sont : le **modèle**, la **vue** et le **contrôleur** ayant chacune un rôle précis dans l'interface.

- **Le modèle** : le modèle contient les données manipulées par le programme. Il assure la gestion de ces données et garantit leur intégrité. C'est lui qui supporte la base de données.
- **La vue** : la vue sert d'interface avec l'utilisateur. Elle permet d'afficher les données qu'elle a récupérées auprès du modèle et de recevoir toutes les actions de l'utilisateur (clic de souris, sélection d'une entrées, boutons, ...). Ces différents événements sont envoyés au contrôleur.
- **Le contrôleur** : il est chargé de la synchronisation du modèle et de la vue. Il reçoit tous les événements de l'utilisateur et enclenche les actions à effectuer. Si une action nécessite un changement des données, le contrôleur demande la modification des données au modèle et ensuite avertit la vue que les données ont changé pour que celle-ci se mette à jour. Certains événements de l'utilisateur ne concernent pas les données mais la vue. Dans ce cas, le contrôleur demande à la vue de se modifier.

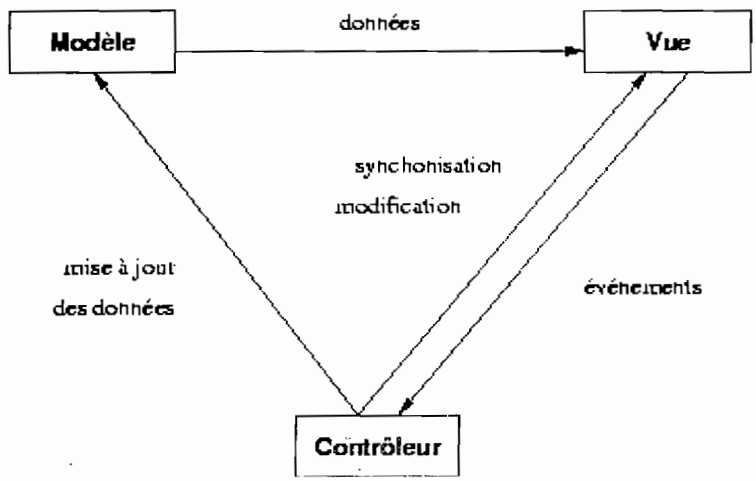


Figure 12 : Interactions entre le modèle, la vue et le contrôleur

III.2. Description des scénarii

III.2.1. Description du Scénario 1

Le premier scénario que nous proposons est assez simple dans son architecture réseau et dans sa mise en œuvre. Ce scénario consistera à mettre en place une application fonctionnant selon une **architecture 2-tiers**[1] encore appelée architecture **client lourd/serveur**. Ce style d'architecture met en œuvre deux niveaux environnementaux et organisationnels. En effet, on distingue d'une part le **« client lourd »** demandeur de service et d'autre part le **« serveur de données »** qui fournit le service. Du point de vue logique, nous avons choisi d'associer l'architecture logicielle en **cinq (05) couches** à cette architecture physique. De ce fait, les couches présentation, application et métier seront développées du côté **« client »** et les couches accès aux données et physique du côté **« serveur ».**

Nous aurons donc la base de données qui sera délocalisée sur un serveur dédié, le serveur de données qui fournira les données à exploiter.
Les utilisateurs que sont les membres du service informatique, pourront avoir accès à ces données à travers le code applicatif qui sera installé sur leurs postes de travail respectifs et ce, via le réseau local d'entreprise.

[1] «2- tiers » : *tier* signifie « partie » en anglais

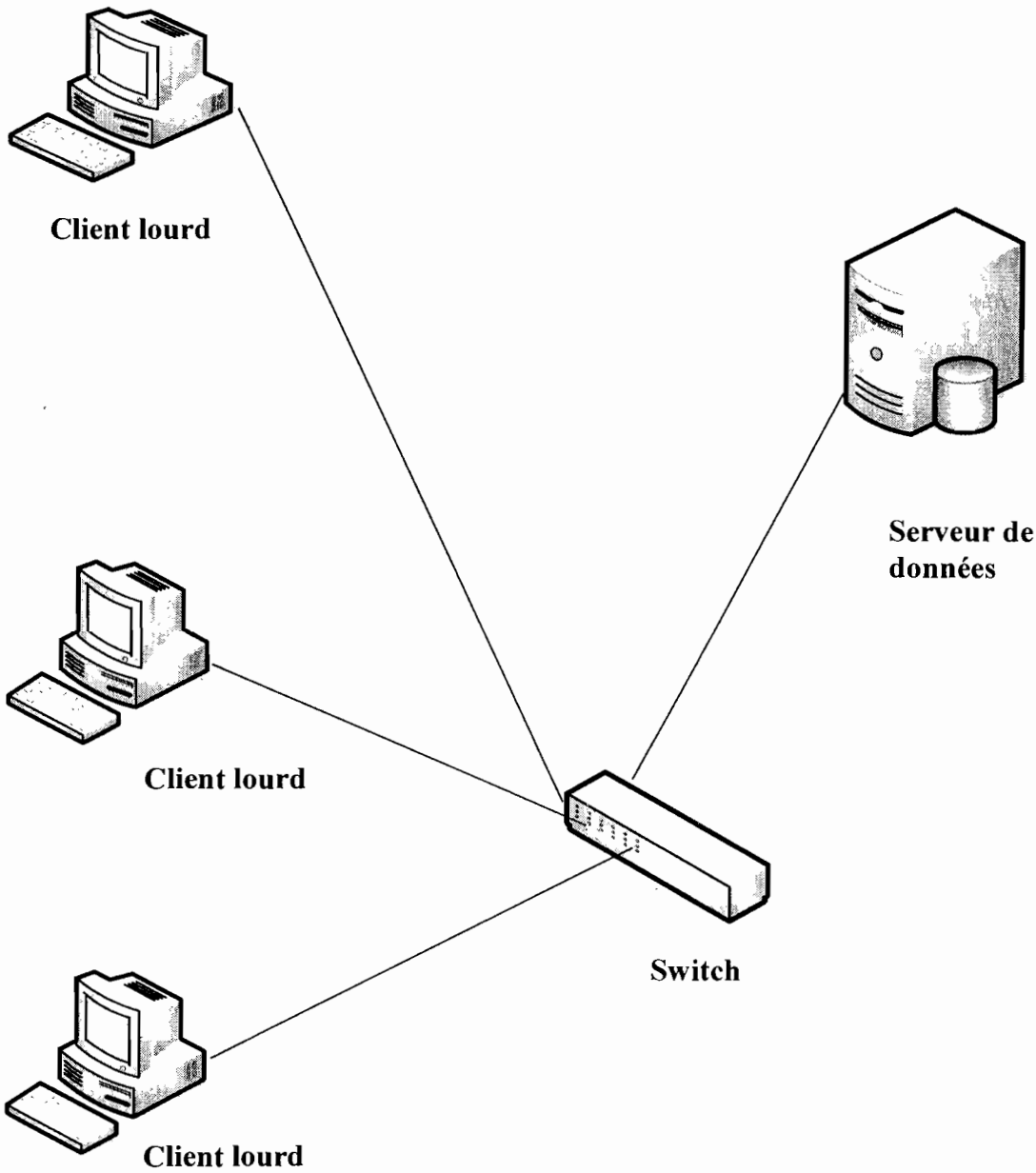


Figure 13 : architecture du scénario 1

Matériels et logiciels requis :

Le service informatique dispose déjà d'un réseau local avec des serveurs de grande capacité. Il dispose également des matériels nécessaires (switchs, postes de travail, routeurs ...) pour la mise en œuvre de l'application. La mise en œuvre de notre système ne va donc pas entraîner un coût matériel.

III.2.2. Description du Scénario 2

Ce scénario est assez semblable au premier, mais en plus des « **clients** » et du **serveur de données** évoqués plus haut, il fait intervenir un troisième tiers : un **serveur d'application**. Ce type d'architecture physique est appelé **architecture 3-tiers.** A la différence du scénario précédent, les machines clientes, appelées aussi « **clients légers** » ne contiennent que l'interface de l'application afin d'assurer les interactions avec le système. Elles sont ainsi déchargées de tout traitement comme c'était le cas au niveau du premier scénario. La logique applicative est maintenant dédiée au second tiers, le serveur d'application, qui sert de liaison entre l'interface applicative et les données toujours localisées au niveau du serveur de données. Cette architecture sur le plan logique, sera mise en œuvre suivant le **modèle MVC** (Modèle Vue Contrôleur). Le modèle c'est-à-dire la base de données sera déployée au niveau du serveur de données, la Vue (interface) au niveau client et le contrôleur (code applicatif) au niveau du serveur d'application. Les utilisateurs, à partir de n'importe quel poste de travail relié au serveur d'application pourront accéder à l'application.

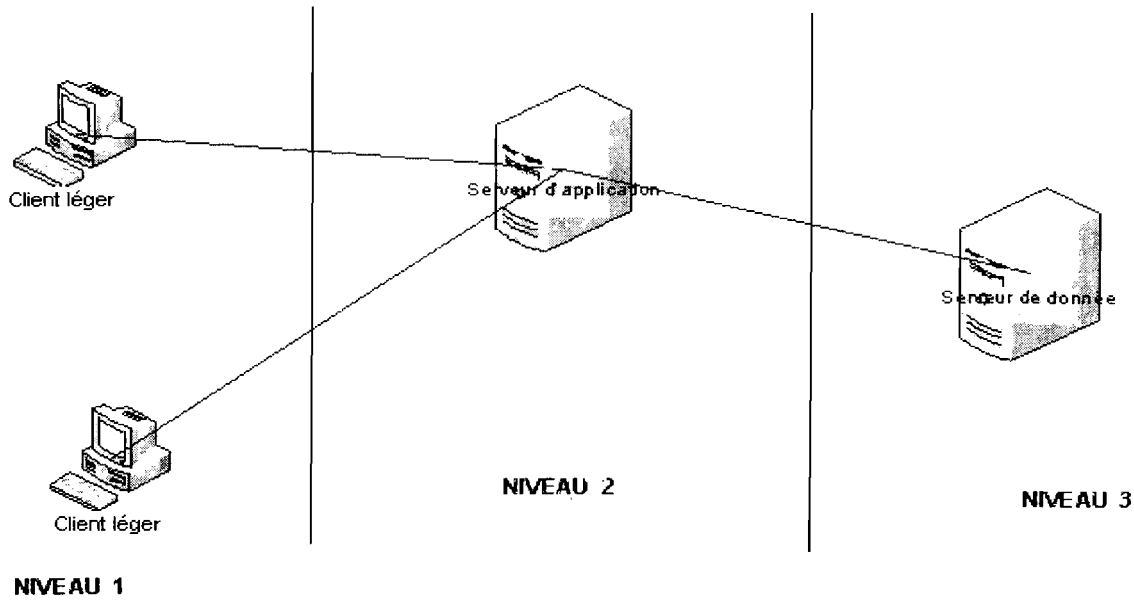


Figure 14 : architecture du scénario 2

Matériels et logiciels requis :

Ce scénario n'entraînera pas aussi de coût matériel. En effet, les équipements matériels nécessaires existent déjà.

III.2.3. Etude comparative des différents scénarii

Il s'agit pour nous dans ce paragraphe, de faire un choix entre les différentes solutions présentées, en se basant sur les points forts et les points faibles de chacune d'elles. L'étude comparative des scénarii proposées qui suivra, nous aidera à opérer le choix le plus judicieux.

- **Avantages et Inconvénients du Premier scénario**

Avantage :

Facile à réaliser ;

Sécurité renforcée à cause de la réduction de points d'entrée ;

Permet un dialogue direct entre le client et le serveur ;

Inconvénient :

Performance réduit ;

Nécessité d'installer les pilotes applicatifs et les pilotes de données sur chaque poste ;

- **Avantages et Inconvénients du Deuxième scénario**

Avantages :

Possibilité de modifier une couche sans répercussion sur les autres ;

Déploiement et maintenance peu coûteux ;

Inconvénients :

Complexité de la mise en œuvre ;

Nombre de points d'entrées élevés ;

❖ **Choix du scénario**

Le premier scénario, à la différence du deuxième, propose une architecture physique et logicielle assez simple à mettre en place. En effet, il fait l'économie de l'usage d'un second serveur (serveur d'application) et se trouve moins coûteux. De plus en terme de sécurité, il s'avère plus avantageux car avec l'architecture 2-tiers, les points d'accès au système sont considérablement réduits. De ce fait, il n'exige pas une politique de sécurité complexe.

Le second scénario quant à lui, est moins coûteux en terme de déploiement et de maintenance.

Au sortir de cette étude comparative, notre choix en conformité avec le groupe de pilotage s'est porté sur le premier scénario qui est le plus optimal pour la mise en œuvre de notre système.

CONCLUSION

L'étude de l'existant a montré la nécessité d'informatiser le système de gestion de la hotline locale au regard des insuffisances qui ont été évoquées. A travers les études fonctionnelle et technique, nous avons pu d'une part, dégager les grandes fonctionnalités et d'autre part l'architecture physique et logicielle de notre futur système. Dans le chapitre suivant, nous aborderons l'étude conceptuelle du système qui consistera à définir la manière dont ces contraintes fonctionnelles et techniques seront implémentées.

CHAPITRE III : PHASE DE CONCEPTION

INTRODUCTION

Après les études fonctionnelle et technique que nous avons réalisées, nous allons maintenant passer à la phase de conception. Le processus de conception permet de passer de l'analyse à la fabrication de la solution en langage objet. C'est à cette étape du processus 2TUP que s'effectue la fusion des études fonctionnelle et technique. Notre étude conceptuelle se fera au tour des activités suivantes :

- ✓ L'application du micro-processus de conception logique ;
- ✓ La conception des différentes couches logicielles du système ;
- ✓ La spécification des procédures transitoires et de la politique de sécurité à mettre en place pour un bon fonctionnement du système.

I. Modèle de Déploiement et d'Exploitation

Le développement du modèle de déploiement permet de spécifier les différents postes de travail. Un poste de travail représente un ou plusieurs acteurs pouvant être localisés sur une machine d'un type particulier et remplissant une fonction identifiée dans l'entreprise [Roques 04]. En général chaque acteur de l'analyse correspond à un poste de travail.

En rappel, notre application vise à assurer la gestion des incidents et du parc informatique. Elle sera développée selon l'architecture ***« client lourd/serveur».*** La base de données sera déployée sur un serveur de données de l'entreprise et la logique applicative sur les machines des membres du service informatique. Ces derniers pourront à partir de leur machine, exécuter des requêtes en vue de visualiser ou de mettre à jour la base de données.

II. Outils de développement et langages de programmation

Dans le cadre de notre projet, nous devons développer une application de gestion permettant d'atteindre les objectifs que nous nous sommes fixés. Pour cela, nous avons opté pour une programmation orientée objet afin de pouvoir implémenter certains concepts comme l'héritage et la réutilisation. Dans ce paragraphe, nous ferons une étude comparative de quelques outils et langages de programmation avant d'opérer notre choix.

II.1. Les Environnements de Développement Intégré (IDE)

- **Visual Studio.NET :** c'est une plate forme de développement conçue par Microsoft qui intègre un ensemble d'environnements de développement intégré, de compilateurs (visual basic.net, C#, J#, etc.) et de débogueurs. C'est la réponse de Microsoft face à la montée des autres outils de programmation orienté objet. Visual Studio.NET prône le RAD (Développement Rapide d'Application) rendu possible grâce aux nombreux outils disponibles.

AVANTAGES :

- Dispose de très nombreux outils et pouvant interagir ;
- Design WYSWYG[1] des fenêtres des applications graphiques et des applications Web ;
- Intégration en local des MSDN[2] (F1) et accès dynamique ;
- Extensibilité (possibilité de créer vos propres plug-in[3]) ;
- IDE très optimisé (l'accès à la plupart des fonctionnalités est immédiat).

[1] WYSIWYG : What You See Is What You Get, pour définir des interfaces graphiques hautement intuitives
[2] MSDN : documentation sur l'aide en tapant sur la touche F1.
[3] Plug-in : c'est un programme qui interagit avec un logiciel principal, appelé *programme hôte*, pour lui apporter de nouvelles fonctionnalités.

INCONVENIENTS :

- Son prix est exorbitant : la version la plus complète est de l'ordre de trois milles euros (environ deux millions de francs CFA) ;
- Le nombre de fenêtres est élevé et nécessite une haute résolution ;
- Il est très gourmand en ressources (au moins 256Mo de mémoire RAM pour travailler correctement).

- **Eclipse-SDK-3.1.2 :** éclipse est une plate-forme de développement né du travail d'un consortium de grandes entreprises (IBM, Borland, Rational Rose, HP...). C'est un IDE performant et Open Source[1] qui a su trouver sa place parmi les grosses pointures du marché. Il supporte de nombreux outils de développement de haut niveau très complets : un IDE complet Java tel que le JDT, un environnement de création de plug-in et un ensemble de frameworks[2] de fondations qui garantissent une bonne interopérabilité des plug-in.

AVANTAGES :

- L'ouverture de son noyau permet l'ajout de nouvelles fonctionnalités ;
- Moins gourmand en espace mémoire par rapport à Visual Studio.Net et NetBeans ;
- Très Facile à installer.

INCONVENIENTS :

- Difficile d'utilisation pour les débutants.

- **NetBeans :** il a été créé à l'initiative de Sun MicroSystems[3]. De licence Open Source, NetBeans permet de développer et déployer rapidement et gratuitement les applications graphiques Swing, des Applets, des JSP/servlets dans des environnements personnalisables.

[1] Open source : code source ouvert au public
[2] Framework : C'est un ensemble de bibliothèques, d'outils et de conventions permettant le développement d'applications.
[3] Sun MicroSystems : constructeur d'ordinateurs et éditeur de logiciels américain.

AVANTAGES :

- Facilité de prise en main pour les débutants ;
- Facilité d'ajout de nouvelles fonctionnalités via un système de module téléchargeable que l'on peut installer depuis l'interface du programme.

INCONVENIENTS :

- Très gourmand en espace mémoire.

II.2. Langages de programmation

- **Java :** java a été créé par James Gosling et Patrick Naughton chez Sun MicroSystems avec le soutien de Bill Joy. C'est à la fois un langage de programmation et une plate-forme d'exécution, ce qui garantit la portabilité des applications développées en java. Quatre caractéristiques militent fortement en faveur de java depuis sa création et lui confèrent une notoriété dans le monde des langages de programmation :
    - Il est purement orienté objet ;
    - Il est indépendant de la plate-forme du client ;
    - Il contient des aides et des bibliothèques pour le réseau informatique ;
    - Il a la capacité d'exécuter du code source extérieur de façon sécurisée.

    AVANTAGES :

    - Orienté objet pur ;
    - Interprété et compilé ;
    - Robuste, portable et dynamique ;
    - Simple et familier.

    INCONVENIENTS :

    - Demande beaucoup en ressources matérielles

- **C#** (prononcer « ***ci charp*** ») : c'est un langage de programmation orienté objet créé par Microsoft. Il est proche du langage java dont il reprend les concepts mais la syntaxe est relativement semblable à celle des langages C et

C++. Le CLR (Common Language Runtime) est obligatoire pour exécuter les applications écrites en C# comme l'est la JVM (Java Virtual Machine ou Machine Virtuelle Java) pour les applications java.

AVANTAGES :

- Support complet des classes de programmation orientées objet avec interface et héritage d'implémentation, les fonctions virtuelles et la surcharge d'opérateurs ;
- Le nettoyage automatique de la mémoire allouée dynamiquement ;
- Accès complet à la bibliothèque .Net des classes de base, accès aisé à l'API[1] windows.

INCONVENIENTS :

- Difficulté de faire face à des temps d'exécution critique et assurer des performances très élevées du code (le type de situation où il est important de savoir si une boucle exige 1000 ou 1050 cycles de machine pour s'exécuter et où il faut nettoyer les ressources dans la milliseconde où elles deviennent inutiles) ;
- Impossibilité de spécifier des fonctions en ligne et des destructeurs garantissant leur exécution à certains points précis du code.

- **Visual Basic.Net :** c'est aussi un langage de programmation orienté objet pour la plate-forme .Net de Microsoft. Il très proche du langage Visual Basic et n'a pas de différence fondamentale avec le C#.

AVANTAGES :

- Il offre un environnement graphique de développement ;
- Il est orienté objet ;

[1] API : interface de programmation (en anglais Application Programming Interface) qui définit la manière dont un composant informatique peut communiquer avec un autre.

- Il offre des composants logiciels (ActiveX[1]) et des librairies (préprogrammées) très puissants et prêts à être intégrés et utilisés dans votre application ;
- Il donne la possibilité d'intégrer aisément de nouveaux composants ActiveX développés dans le commerce ;
- Il offre une grande facilité pour développer de nouveaux composants ActiveX.

INCONVENIENTS :

- Il assez gourmand en ressource mémoire (il faut au moins 256Mo de RAM) ;
- Difficulté de faire face à des temps d'exécution critique et assurer des performances très élevées du code.

Pour le développement de notre application, nous avons opté pour la plate forme de développement Visual Studio.Net car c'est un environnement de développement graphique, offrant ainsi la possibilité de créer facilement des interfaces graphiques contrairement à Eclipse et NetBeans. De plus c'est un IDE très optimisé car l'accès à la plupart des fonctionnalités est immédiat. Enfin, c'est un IDE homologué par le groupe **BOLLORE.**

Comme langage de programmation, nous avons retenu **Visual Basic.Net**. Notre choix se justifie par le fait que ce langage est pris en charge par la plate forme .Net. Il offre également des composants logiciels et des librairies très puissants faciles à intégrer dans notre application.

[1] ActiveX : composant logiciel créé par Microsoft utilisé en programmation pour permettre le dialogue entre programmes.

III. Modélisation du futur système

III.1. Les diagrammes d'activité du futur système

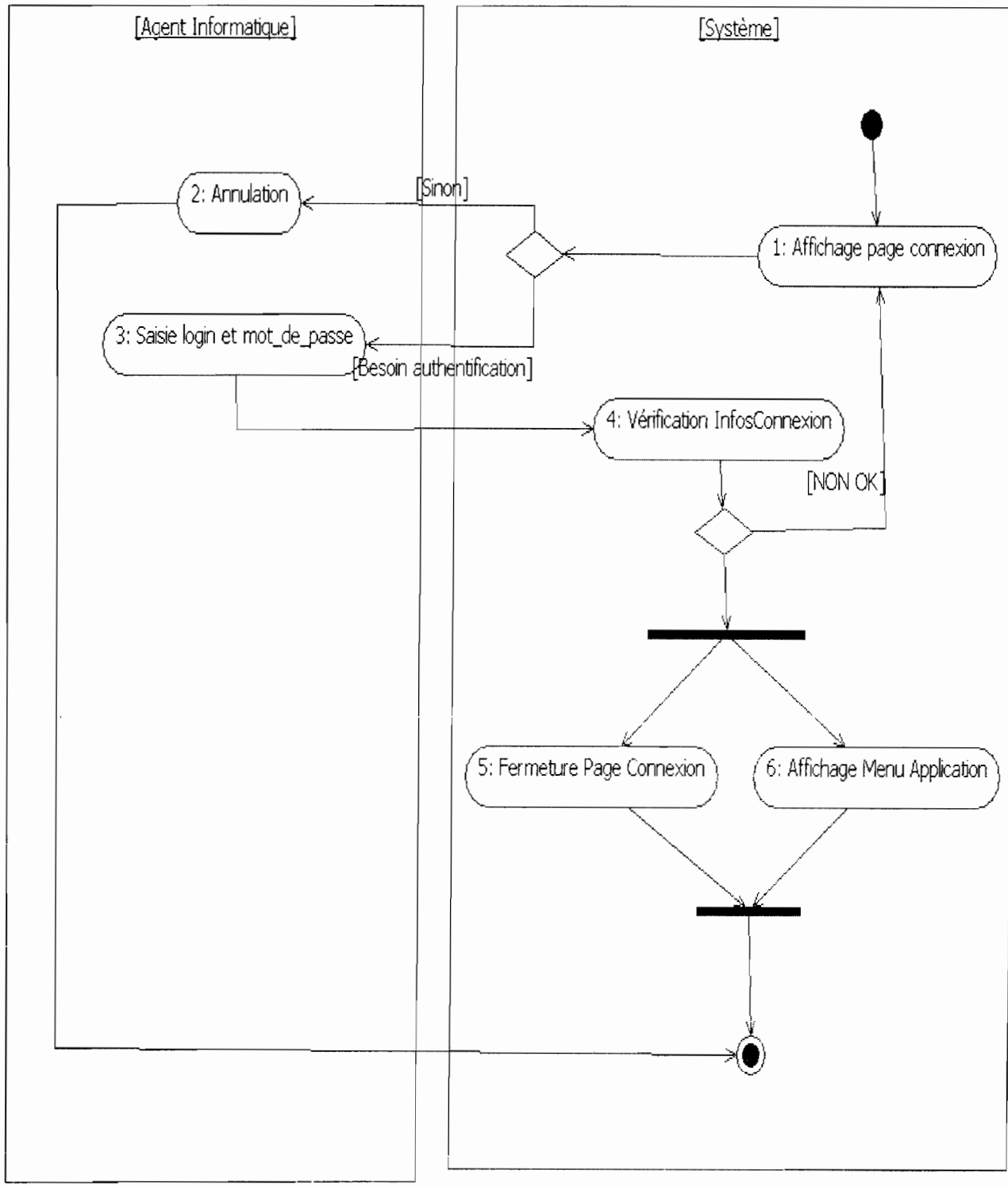


Figure 15 : Diagramme d'activité du cas d'utilisation « authentification »

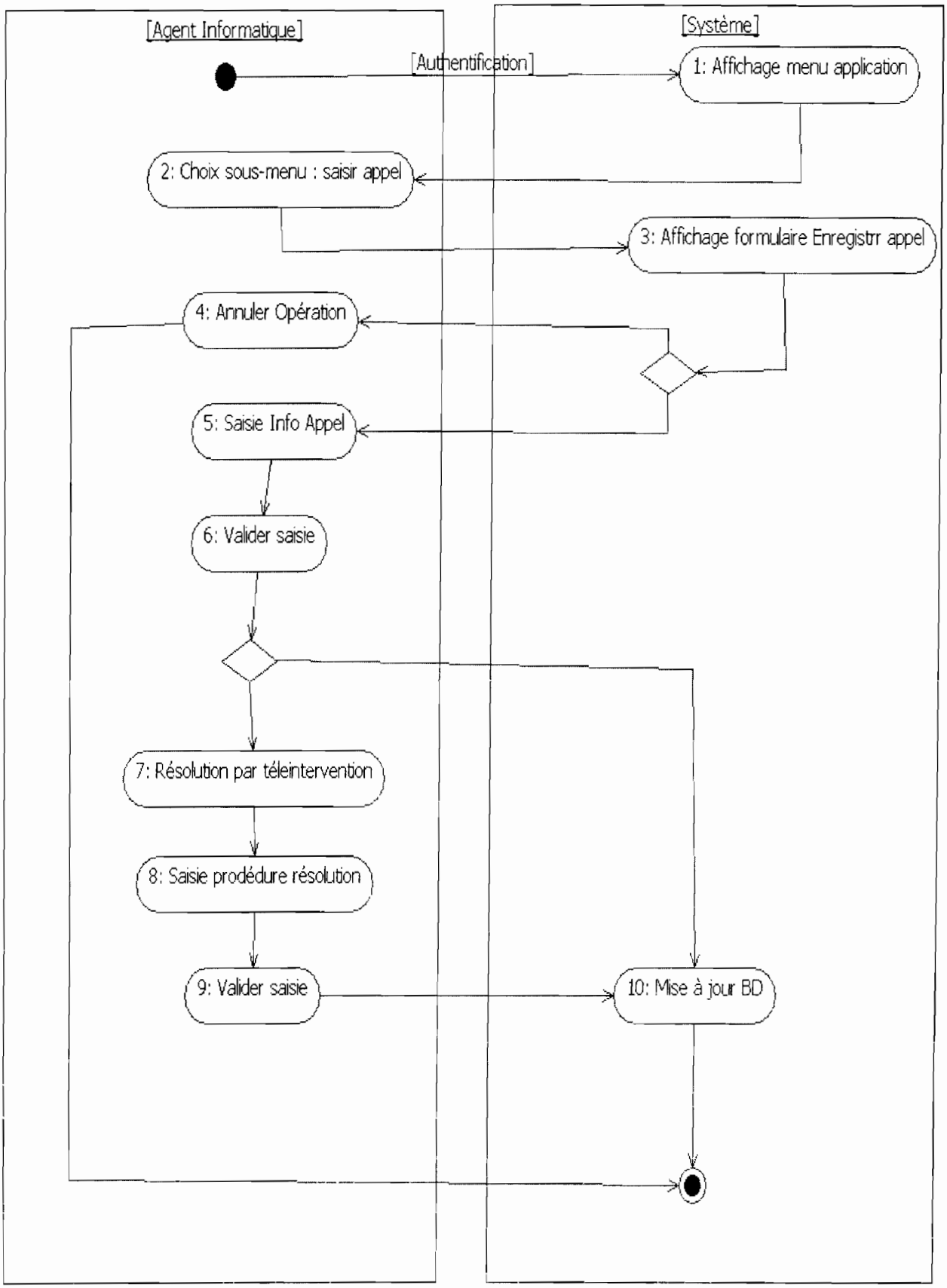


Figure 16 : Diagramme d'activité du cas d'utilisation « traiter incident »

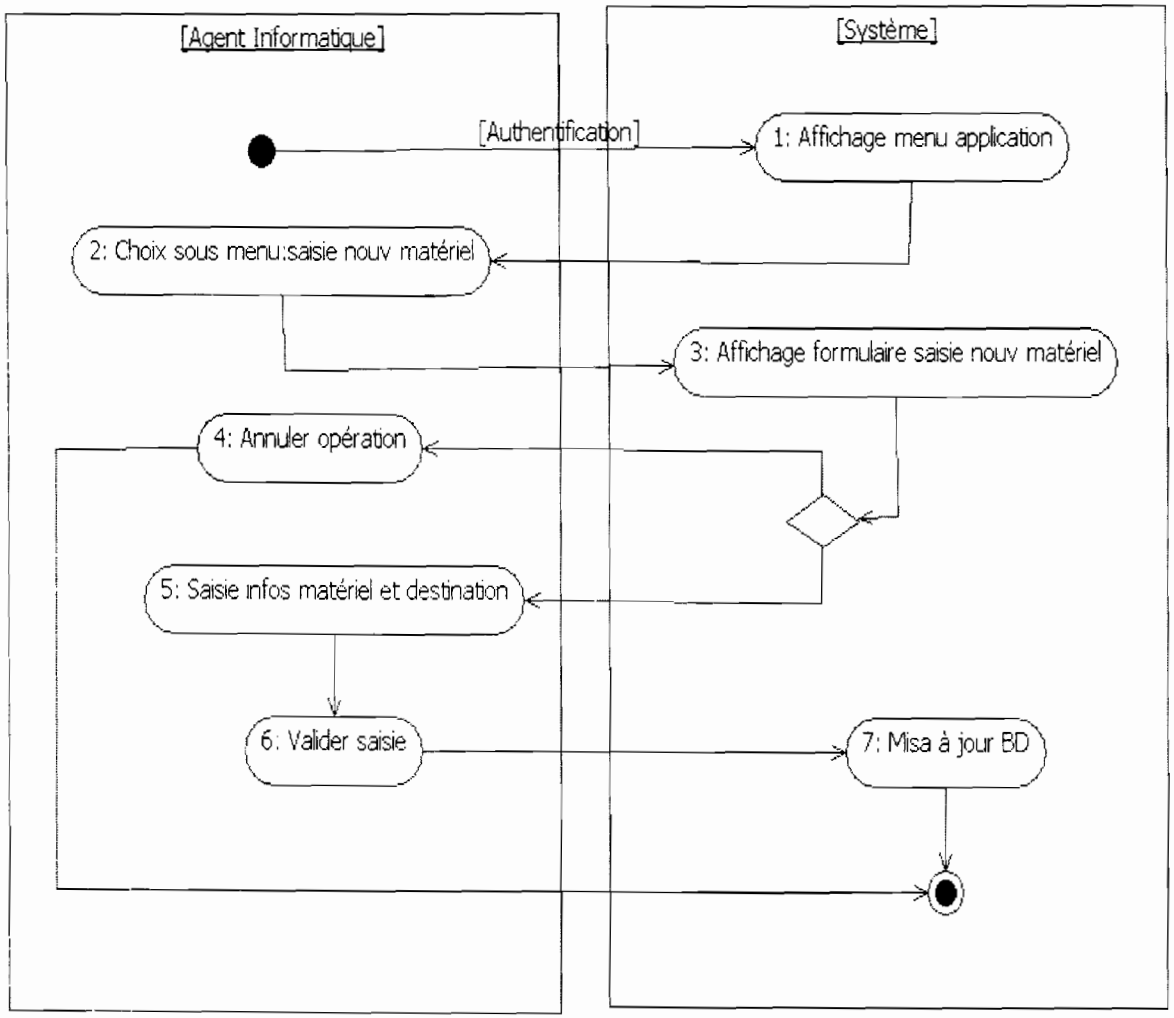


Figure 17 : Diagramme d'activité du cas d'utilisation « affectation provisoire »

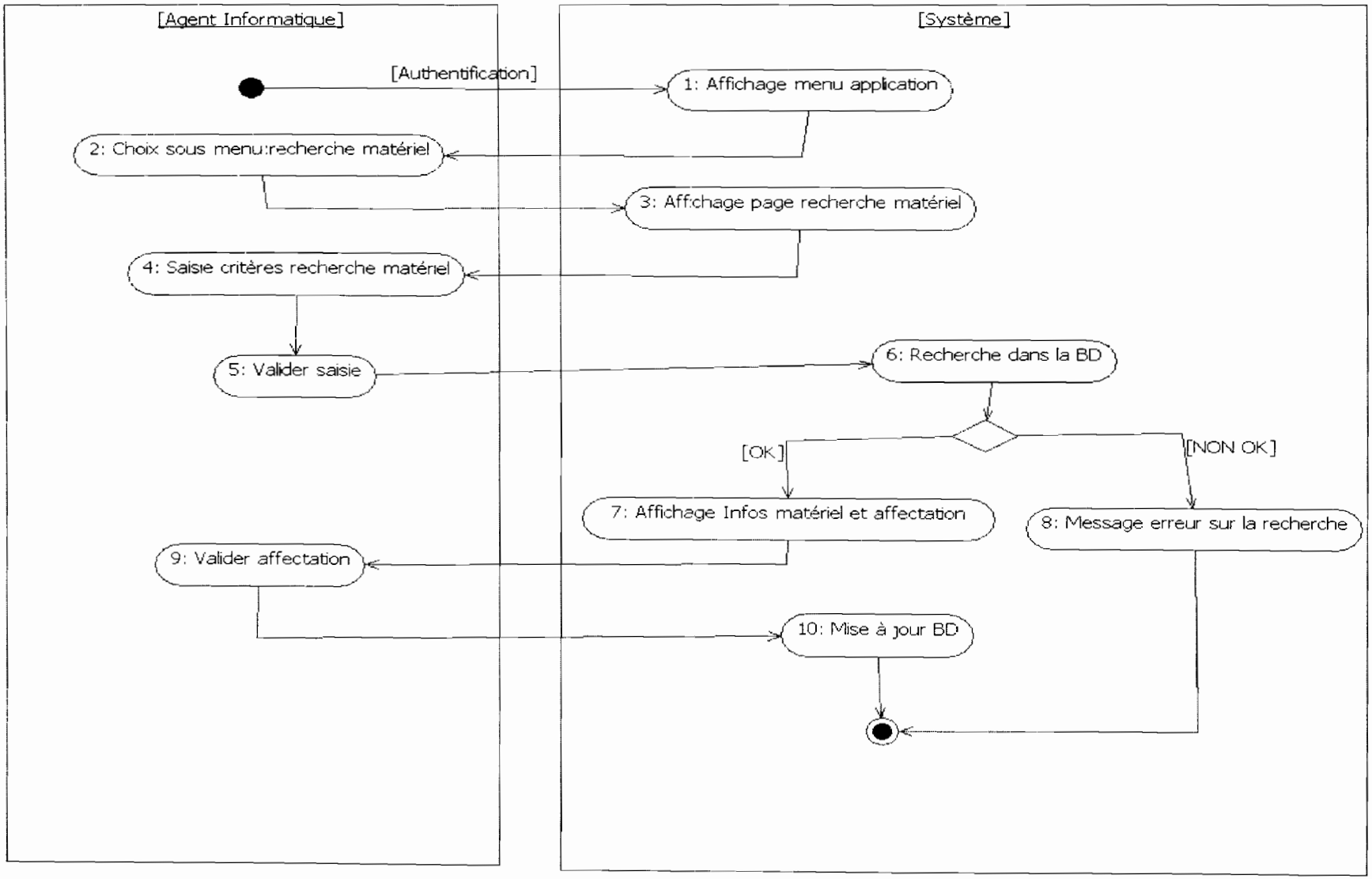


Figure 18 : Diagramme d'activité du cas d'utilisation « affectation définitive »

III.2. Classes Métiers Détaillées

A partir du modèle logique de conception et des interfaces, le micro-processus de conception de la méthode 2TUP-UML permet de définir les classes à implémenter. Cette activité consiste à :

- Concevoir les classes : transformer les classes et métaclasses provenant de l'analyse en codage dans un langage de développement.
- Concevoir les associations : définir la façon d'exploiter chaque association et les techniques qui seront employées dans le codage.
- Concevoir les attributs : identifier les structures de données, les itérations et d'autres types complexes permettant de représenter les attributs d'analyse dans le langage utilisé.
- Concevoir les opérations : déterminer le contenu des méthodes complexes.

↓ Conception des classes

Les classes qui proviennent de l'analyse ne sont pas toujours conformes aux possibilités du langage d'implémentation. Dans certains cas, une analyse orientée objet est réalisée dans un langage non objet. La transformation des classes en codage est alors particulièrement importante pour conserver la trace du passage de l'analyse au code. Visual Basic.net offre bien entendu une transformation beaucoup plus directe. La conception des classes consistera à expliciter les concepts d'attributs, de méthode et de visibilité.

↓ Conception des Associations

Le concept d'association n'est pas directement disponible dans les langages orientés objets en général. Une association se transforme en attribut ou collection d'attributs suivant sa multiplicité. Des opérations sont ensuite rajoutées aux classes pour assurer la navigabilité des associations. Ce principe de conception suffit à décrire les associations de notre application. La figure suivante illustre l'application de ce principe aux classes ***Materiel*** et ***Intervention***.

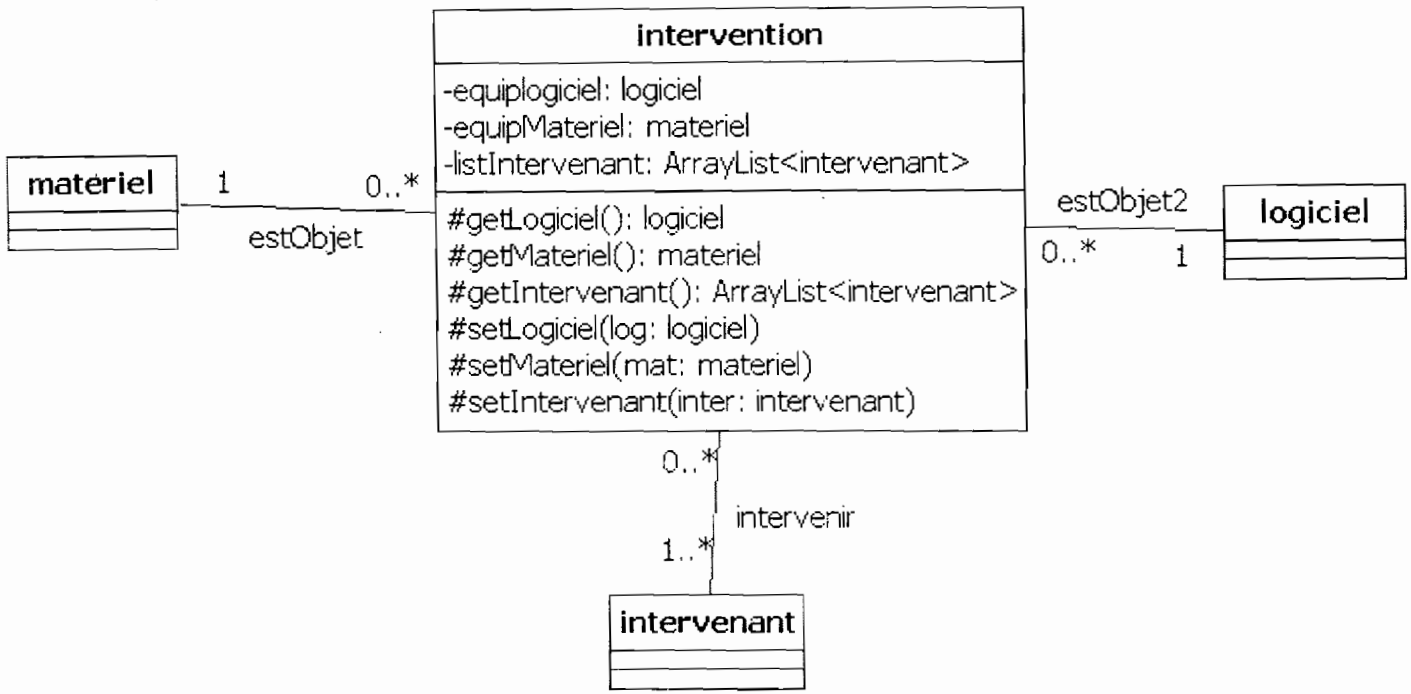


Figure 19 : Conception des associations de la classe « intervention »

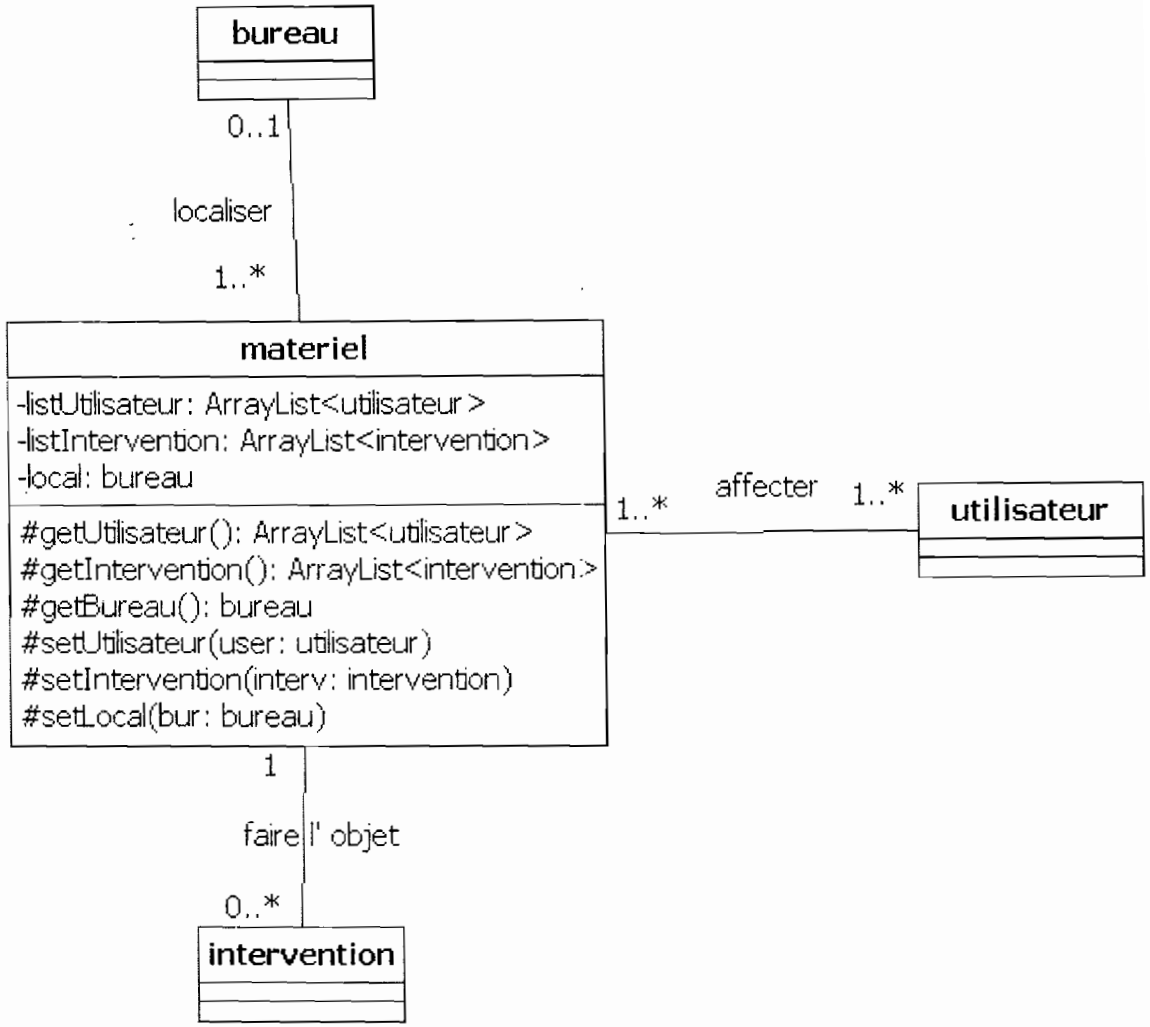


Figure 20 : Conception des associations de la classe « materiel »

Conception des Attributs

La conception des attributs consiste principalement à préciser le type des attributs identifiés dans la phase d'analyse. Les attributs identifiés pour notre système sont tous pris en charge par les types de base du langage visual basic.net. Il reste alors à préciser la visibilité des attributs ainsi que les modes d'accès. Par défaut tous les attributs sont privés (private). En visual basic.net la notion de propriété (Property) permet de définir les accesseurs pour les attributs des classes. Une propriété définit deux méthodes « **get** » et « **set** » qui permettent respectivement de retourner et de renseigner la valeur d'un attribut. Pour un attribut en lecture seule la méthode « *set* » n'est pas définie.

Diagramme de classes et de packages

Règles de gestion :

RG1 : un matériel se trouve dans au plus un bureau ;
RG2 : dans un bureau se trouvent un ou plusieurs matériels ;
RG3 : un matériel est utilisé par un ou plusieurs utilisateurs ;
RG4 : un utilisateur utilise un ou plusieurs matériels ;
RG5 : un matériel ou un logiciel peut ne pas faire l'objet d'une intervention ;
RG6 : une intervention est effectuée par un ou plusieurs intervenants ;
RG7 : un intervenant peut effectuer une ou plusieurs interventions ;
RG8 : une intervention concerne un et un seul matériel ou un et un seul logiciel;
RG9 : un intervenant peut être un prestataire, la DSI ou un agent informatique;
RG10 : une équipe est constituée d'un ou de plusieurs agents ;
RG11 : un agent peut faire partie de plusieurs équipes ;
RG12 : un matériel peut être un poste de travail, un switch, un serveur ou un routeur ;
RG13 : un poste de travail peut être un client léger ou un micro ordinateur;
RG14 : un poste de travail ou un serveur micro comporte un ou plusieurs périphériques ;
RG15 : un périphérique est comporté par un et un seul poste de travail ou un seul serveur micro ;
RG16 : un périphérique peut être un clavier, un écran, une souris ou une imprimante ;

RG17 : sur un poste de travail ou sur un serveur sont installés un ou plusieurs logiciels ;

RG18 : un logiciel peut être installé sur au moins un poste de travail ou un serveur ;

RG19 : un utilisateur peut ne pas effectuer d'appel ;

RG20 : un appel est effectué par un et un seul utilisateur ;

RG21 : un bureau se situe dans un et un seul service ;

RG22 : un service est composé d'un ou plusieurs bureaux ;

RG23 : une société est formée d'un ou plusieurs services ;

RG24 : un service est dans une et une seule société ;

RG25 : une société est localisée sur un ou plusieurs sites ;

RG26 : sur un site se trouvent un ou plusieurs sociétés ;

RG27 : un serveur peut être un serveur micro ou un serveur mini ;

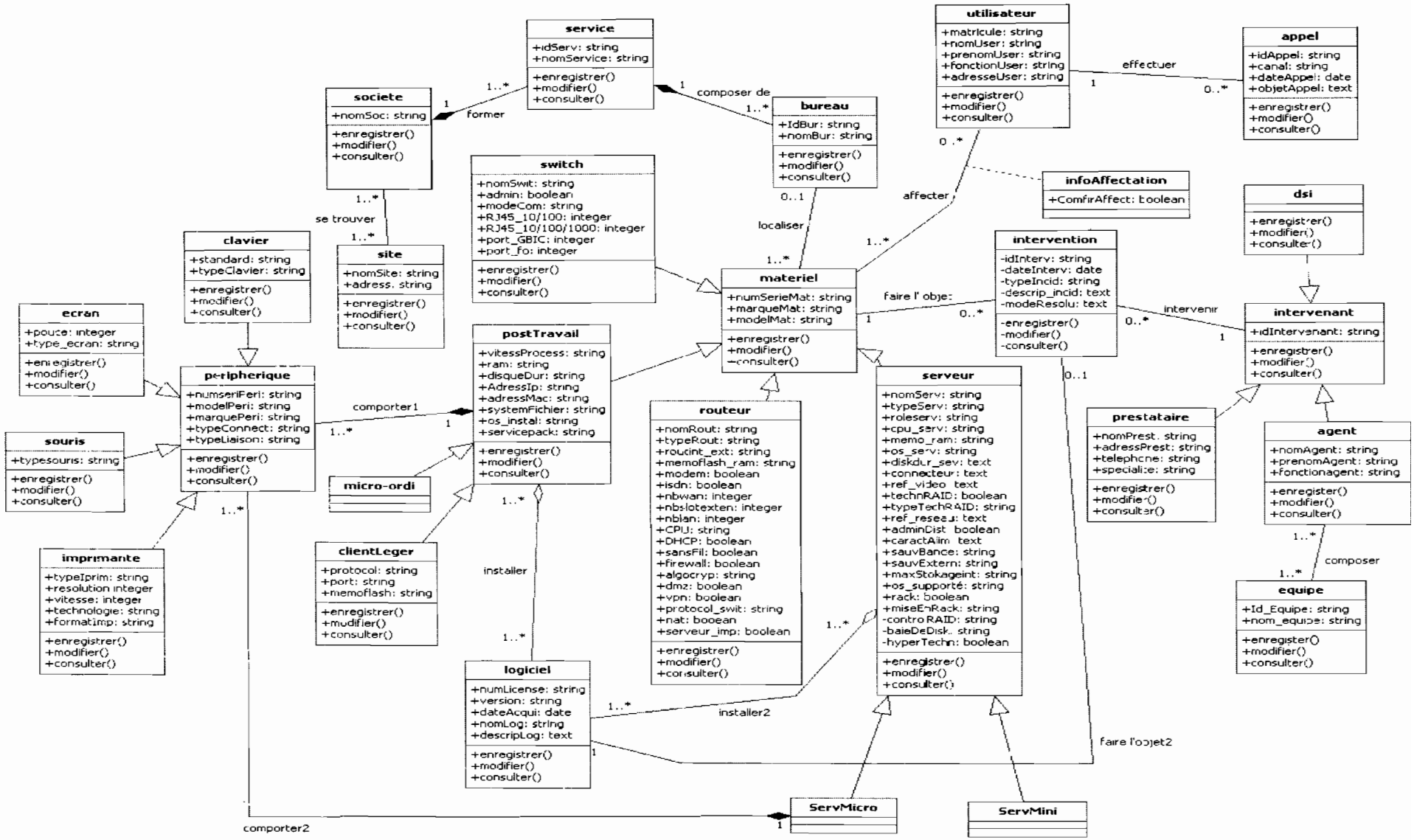


Figure 21 : **Diagramme des classes métiers**

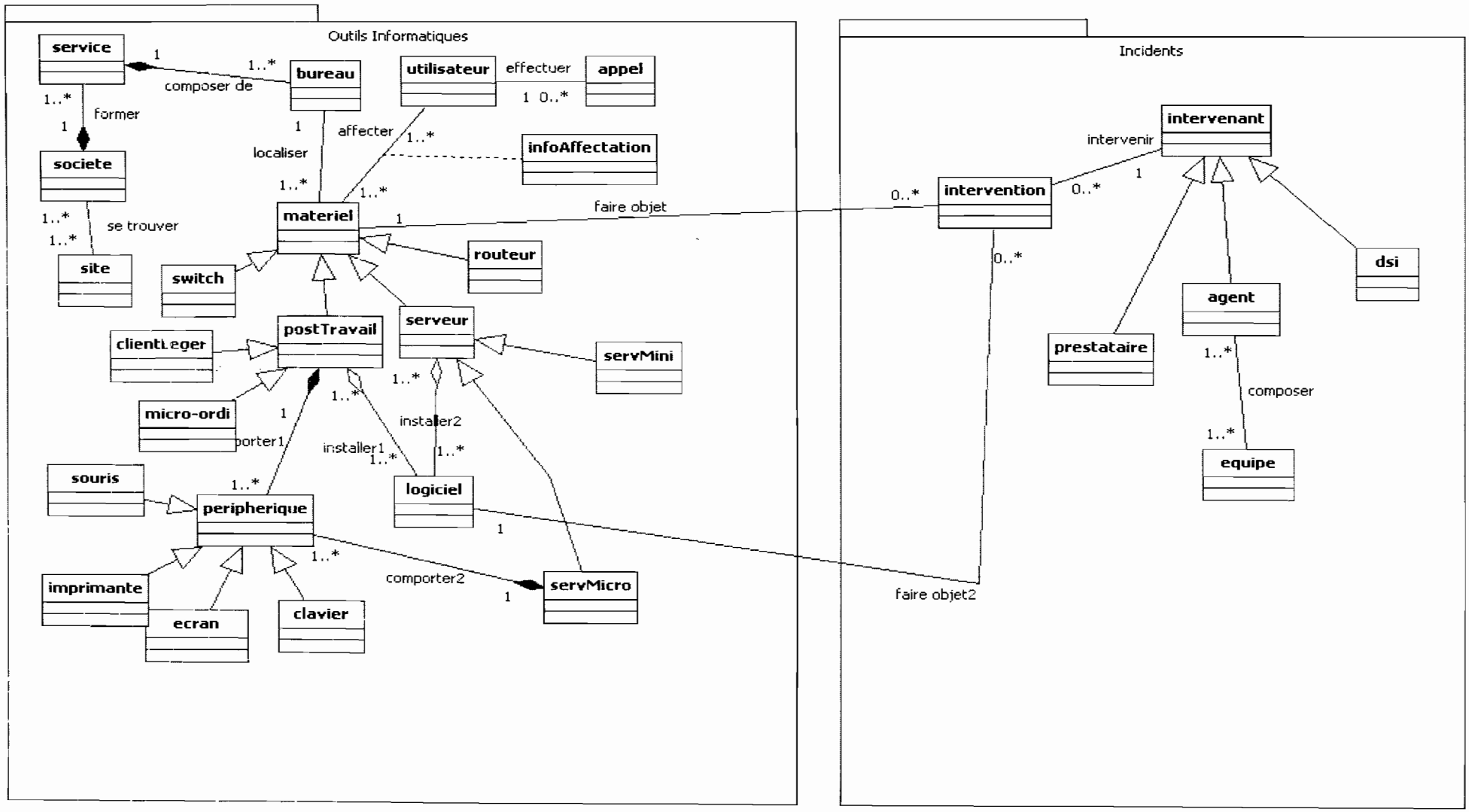


Figure 22 : **Diagramme de Package**

Description des classes

Classe : MATERIEL classe généralisante des entités : Poste de travail, switch, routeur et serveur				
	VISIBILITE	**NOM**	**DESCRIPTION**	**TYPE**
Attributs	Public	NumSericMat	Numéro de série du matériel	Texte
	Public	MarqueMat	Marque du matériel	Texte
	Public	ModelMat	Model du matériel	Texte
Méthodes		**NOM**	**DESCRIPTION**	
	Public	Enregistrer ()	Permet l'enregistrement de matériels	
	Public	Modifier ()	Permet de modifier les caractéristiques d'un matériel	
	Public	Consulter ()	Permet la consultation de matériels	

Tableau 8: Description de la clase MATERIEL

Classe : POSTE DE TRAVAIL classe généralisante des entités client léger et micro-ordi				
	VISIBILITE	**NOM**	**DESCRIPTION**	**TYPE**
Attributs	Public	VitesseProcess	Vitesse du processeur	Texte
	Public	Ram	Mémoire Ram	Texte
	Public	DisqueDur	Capacité du disque dur	Texte
	Public	AdresseIP	Adresse IP du poste de travail	Texte
	Public	AdresseMAC	Adresse MAC du poste de travail	Texte
	Public	SystemFichier	Système de fichier utilisé	Texte
Méthodes		**NOM**	**DESCRIPTION**	
	Public	Enregistrer ()	Permet l'enregistrement des postes de travail	
	Public	Modifier ()	Permet de modifier les caractéristiques d'un poste de travail	
	Public	Consulter ()	Permet la consultation des postes de travail	

Tableau 9: Description de la classe POSTE DE TRAVAIL

Classe : SWITCH classe spécialisante de l'entité MATERIEL				
	VISIBILITE	**NOM**	**DESCRIPTION**	**TYPE**
Attributs	Public	NomSwitch	Nom du switch	Texte
	Public	ModeCom	Mode communication	Texte
	Public	RJ4510/100	Port RJ45 10/100 Mbps	Booléen
	Public	RJ4510/100/1OOO	Port RJ45 10/100/1000 Mbps	Booléen
	Public	Port_GBIC	Port GBIC	Booléen
	Public	Port_FO	Port pour FO	Booléen
Méthodes		**NOM**	**DESCRIPTION**	
	Public	Enregistrer ()	Permet l'enregistrement des switchs	
	Public	Modifier ()	Permet de modifier les caractéristiques d'un switch	
	Public	Consulter ()	Permet la consultation des switchs	

Tableau 10 : Description de la classe SWITCH

Classe : ROUTEUR classe spécialisante de l'entité MATERIEL				
	VISIBILITE	**NOM**	**DESCRIPTION**	**TYPE**
Attributs	Public	NomRouteur	Nom du routeur	Texte
	Public	TypeRouteur	Type du routeur	Texte
	Public	CPU	Vitesse du routeur	Texte
	Public	DHCP	Support DHCP	Booléen
	Public	SansFil	Réseau sans fil	Booléen
	Public	Firewall	Firewall	Booléen
Méthodes		**NOM**	**DESCRIPTION**	
	Public	Enregistrer ()	Permet l'enregistrement des routeurs	
	Public	Modifier ()	Permet de modifier les caractéristiques d'un routeur	
	Public	Consulter ()	Permet la consultation des routeurs	

Tableau 11: Description de la classe ROUTEUR

Classe : ECRAN classe spécialisante de l'entité PERIPHERIQUE				
	VISIBILITE	**NOM**	**DESCRIPTION**	**TYPE**
Attributs	Public	Pouce	Le pouce de l'écran	Entier
	Public	FormeEcran	La forme de l'écran	Texte
Méthode		**NOM**	**DESCRIPTION**	
	Public	Enregistrer ()	Permet l'enregistrement des écrans	
	Public	Modifier ()	Permet de modifier les caractéristiques d'un écran	
	Public	Consulter ()	Permet la consultation des écrans	

Tableau 12 : Description de la classe ECRAN

Classe : PERIPHERIQUE classe généralisante des entités : souris, clavier, ecran, imprimante				
	VISIBILITE	**NOM**	**DESCRIPTION**	**TYPE**
Attributs	Public	NumSeriPeri	Le numéro de série du périphérique	Texte
	Public	ModelPeri	Le model du périphérique	Texte
	Public	TypeConnect	Le type de connexion	Texte
	Public	TypeLiaison	Le type de liaison	Texte
Méthodes		**NOM**	**DESCRIPTION**	
	Public	Enregistrer ()	Permet l'enregistrement des périphériques	
	Public	Modifier ()	Permet de modifier les caractéristiques d'un périphérique	

Tableau 13 : Description de la classe PERIPHERIQUE

Classe : CLAVIER classe spécialisante de l'entité PERIPHERIQUE				
	VISIBILITE	**NOM**	**DESCRIPTION**	**TYPE**
Attributs	Public	Standard	Le standard du clavier	Texte
	Public	TypeClavier	Le type du clavier	Texte
Méthodes		**NOM**	**DESCRIPTION**	
	Public	Enregistrer ()	Permet l'enregistrement des claviers	
	Public	Modifier ()	Permet de modifier les caractéristiques d'un clavier	
	Public	Consulter ()	Permet la consultation des claviers	

Tableau 14: Description de la classe CLAVIER

Classe : souris classe spécialisante de l'entité PERIPHERIQUE				
	VISIBILITE	**NOM**	**DESCRIPTION**	**TYPE**
Attributs	Public	TypeSouris	Le numéro de série du périphérique	Texte
Méthodes		**NOM**	**DESCRIPTION**	
	Public	Enregistrer ()	Permet l'enregistrement des souris	
	Public	Modifier ()	Permet de modifier les caractéristiques d'une souris	
	Public	Consulter ()	Permet la consultation des souris	

Tableau 15 : Description de la classe SOURIS

Classe : LOGICIEL				
	VISIBILITE	**NOM**	**DESCRIPTION**	**TYPE**
Attributs	Public	NumLicence	Le numéro de licence du logiciel	Texte
	Public	Version	La version du logiciel	Texte
	Public	NomLog	Le nom du logiciel	Texte
	Public	TypeLog	Le type du logiciel	Texte
	Public	Date_acqui	La date d'acquisition du logiciel	Date
Méthodes		**NOM**	**DESCRIPTION**	
	Public	Enregistrer ()	Permet l'enregistrement des logiciels	
	Public	Modifier ()	Permet de modifier les caractéristiques d'un logiciel	
	Public	Consulter ()	Permet la consultation des logiciels	

Tableau 16 : Description de la classe LOGICIEL

Classe : INTERVENANT classe généralisante des entités : prestataires, agent et DSI				
	VISIBILITE	**NOM**	**DESCRIPTION**	**TYPE**
Attributs	Public	Id_Intervenant	L'identifiant de l'intervenant	Texte
Méthodes		**NOM**	**DESCRIPTION**	
	Public	Enregistrer ()	Permet l'enregistrement des intervenants	
	Public	Modifier ()	Permet de modifier les caractéristiques des intervenants	
	Public	Consulter ()	Permet la consultation des intervenants	

Tableau 17 : Description de la classe INTERVENANT

Classe : PRESTATAIRE classe spécialisante de l'entité INTERVENANT				
	VISIBILITE	**NOM**	**DESCRIPTION**	**TYPE**
Attributs	Public	NomPrest	Le nom du prestataire	Texte
	Public	AdrPrest	L'adresse du prestataire	Texte
	Public	Specialite	La spécialité du prestataire	Texte
Méthodes		**NOM**	**DESCRIPTION**	
	Public	Enregistrer ()	Permet l'enregistrement des prestataires	
	Public	Modifier ()	Permet de modifier les caractéristiques des prestataires	
	Public	Consulter ()	Permet la consultation des prestataires	

Tableau 18 : Description de la classe PRESTATAIRE

Classe : AGENT				
	VISIBILITE	**NOM**	**DESCRIPTION**	**TYPE**
Attributs	Public	NomAgent	Le nom de l'agent	Texte
	Public	PrenomAgent	Le prénom de l'agent	Texte
	Public	FonctionAgent	La fonction de l'agent	Texte
Méthodes		**NOM**	**DESCRIPTION**	
	Public	Enregistrer ()	Permet l'enregistrement des agents	
	Public	Modifier ()	Permet de modifier les caractéristiques des agents	
	Public	Consulter ()	Permet la consultation des agents	

Tableau 19 : Description de la classe AGENT

Classe : EQUIPE classe spécialisante de l'entité INTERVENANT				
	VISIBILITE	**NOM**	**DESCRIPTION**	**TYPE**
Attributs	Public	NomEquipe	Nom de l'équipe ayant intervenu	Texte
Méthodes		**NOM**	**DESCRIPTION**	
	Public	Enregistrer ()	Permet l'enregistrement des équipes	
	Public	Modifier ()	Permet de modifier les caractéristiques des équipes	
	Public	Consulter ()	Permet la consultation des équipes	

Tableau 20 : Description de la classe EQUIPE

Classe : INTERVENTION				
	VISIBILITE	**NOM**	**DESCRIPTION**	**TYPE**
Attributs	Public	Id_Interv	L'identifiant de l'intervention	Texte
	Public	Date_Interv	La date de l'intervention	Date
	Public	Type_Interv	Le type d'intervention	Texte
	Public	ModeResolu	Le mode de résolution	Texte
Méthodes		**NOM**	**DESCRIPTION**	
	Public	Enregistrer ()	Permet l'enregistrement des interventions	
	Public	Modifier ()	Permet de modifier les caractéristiques des interventions	
	Public	Consulter ()	Permet la consultation des interventions	

Tableau 21 : Description de la classe INTERVENTION

Classe : UTILISATEUR				
	VISIBILITE	**NOM**	**DESCRIPTION**	**TYPE**
Attributs	Public	Matricule	Le matricule de l'utilisateur	Texte
	Public	NomUser	Le nom de l'utilisateur	Texte
	Public	PrenomUser	Le prénom de l'utilisateur	Texte
	Public	FonctionUser	La fonction de l'utilisateur	Texte
	Public	AdresseUser	L'adresse de l'utilisateur	Texte
Méthodes		**NOM**	**DESCRIPTION**	
	Public	Enregistrer ()	Permet l'enregistrement des utilisateurs	
	Public	Modifier ()	Permet de modifier les caractéristiques des utilisateurs	
	Public	Consulter ()	Permet la consultation des utilisateurs	

Tableau 22 : Description de la classe UTILISATEUR

Classe : InfoAffectation				
	VISIBILITE	**NOM**	**DESCRIPTION**	**TYPE**
Attributs	Public	ConfirAffect	La confirmation de l'affectation	Booléen
Méthodes		**NOM**	**DESCRIPTION**	

Tableau 23 : Description de la classe InfoAffectation

Classe : APPEL				
	VISIBILITE	**NOM**	**DESCRIPTION**	**TYPE**
Attributs	Public	ID_Appel	L'identifiant de l'appel	Entier
	Public	CanalAppel	Le canal utilisé pour l'appel	Texte
	Public	DateAppel	La date de l'appel	Date
	Public	ObjetAppel	L'objet de l'appel	Texte
Méthodes		**NOM**	**DESCRIPTION**	
	Public	Enregistrer ()	Permet l'enregistrement des appels	
	Public	Modifier ()	Permet de modifier les informations des appels	
	Public	Consulter ()	Permet la consultation des appels	

Tableau 24 : Description de la classe APPEL

Classe : BUREAU				
	VISIBILITE	**NOM**	**DESCRIPTION**	**TYPE**
Attributs	Public	ID_Bur	L'identifiant du bureau	Texte
	Public	Nom_Bur	Le nom du bureau	Texte
Méthodes		**NOM**	**DESCRIPTION**	
	Public	Enregistrer ()	Permet l'enregistrement des bureaux	
	Public	Modifier ()	Permet de modifier les informations des bureaux	
	Public	Consulter ()	Permet la consultation des bureaux	

Tableau 25 : Description de la classe BUREAU

Classe : SERVICE				
	VISIBILITE	**NOM**	**DESCRIPTION**	**TYPE**
Attributs	Public	ID_Service	L'identifiant du service	Texte
	Public	Nom_Service	Le nom du service	Texte
Méthodes		**NOM**	**DESCRIPTION**	
	Public	Enregistrer ()	Permet l'enregistrement des services	
	Public	Modifier ()	Permet de modifier les informations des services	
	Public	Consulter ()	Permet la consultation des services	

Tableau 26 : Description de la classe SERVICE

Classe : SOCIETE				
	VISIBILITE	**NOM**	**DESCRIPTION**	**TYPE**
Attributs	Public	ID_Soc	L'identifiant de la société	Texte
	Public	Nom_Soc	Le nom de la société	Texte
	Public	Sigle_Soc	Le sigle de la société	Texte
Méthodes		**NOM**	**DESCRIPTION**	
	Public	Enregistrer ()	Permet l'enregistrement des sociétés	
	Public	Modifier ()	Permet de modifier les informations des sociétés	
	Public	Consulter ()	Permet la consultation des sociétés	

Tableau 27 : Description de la classe SOCIETE

Classe : SITE				
	VISIBILITE	**NOM**	**DESCRIPTION**	**TYPE**
Attributs	Public	ID_Site	L'identifiant du site	Texte
	Public	Nom_Site	Le nom du site	Texte
	Public	Adress_Site	L'adresse du site	Texte
Méthodes		**NOM**	**DESCRIPTION**	
	Public	Enregistrer ()	Permet l'enregistrement des sites	
	Public	Modifier ()	Permet de modifier les informations des sites	
	Public	Consulter ()	Permet la consultation des sites	

Tableau 28 : Description de la classe SITE

Classe : CLIENTLEGER classe spécialisante de l'entité POSTE de TRAVAIL				
	VISIBILITE	**NOM**	**DESCRIPTION**	**TYPE**
Attributs	Public	Protocol	Le Protocole utilisé par le client léger	Texte
	Public	Port	Le port utilisé par le client léger	Texte
	Public	MemoFlash	Capacité de la mémoire flash	Texte
Méthodes		**NOM**	**DESCRIPTION**	
	Public	Enregistrer ()	Permet l'enregistrement des clients légers	
	Public	Modifier ()	Permet de modifier les caractéristiques d'un client léger	
	Public	Consulter ()	Permet la consultation des clients léger	

Tableau 29 : Description de la classe CLIENT LEGER

Classe : SERVEUR classe spécialisante de l'entité MATERIEL et classe généralisante les entités : ServMini et ServMicro				
	VISIBILITE	**NOM**	**DESCRIPTION**	**TYPE**
Attributs	Public	Nom_Serv	Nom du serveur	Texte
	Public	Type_Serv	Le type du serveur	Texte
	Public	Role_Serv	Rôle du serveur	Texte
	Public	CPU_Serv	Vitesse du serveur	Texte
	Public	Ram_Serv	Mémoire Ram du serveur	Texte
	Public	OS_Serv	Les OS installés sur le serveur	Texte
	Public	DD_Serv	La capacité du disque dur	Texte
	Public	Connecteurs	Les différents connecteurs	Texte
	Public	Ref_carte_Serv	Référence de la carte réseau	Texte
	Public	TechnRAID	Technologie RAID	Booléen
	Public	TypeRAID	Le type de RAID	Texte
	Public	Admin_dist	Administration à distance	Booléen
	Public	Caract_alim	Caractéristiques alimentation	Texte
	Public	Sauv_bande	Option de sauvegarde sur bande	Texte
	Public	Sauv_extern	Option de sauvegarde externe	Texte
	Public	Max_stockage	La taille maximale de stockage	Texte
	Public	Rack	Possibilité de mise en Rack	Booléen
	Public	Mise_rack	Les possibilités de mise en RACK	Texte
	Public	Control_RAID	Les contrôleurs RAID	Texte
	Public	Baie_disque	Les baies de disques disponibles	Texte
	Public	Hyper_tech	Technologie Hyper Transport	Booléen
Méthodes		**NOM**	**DESCRIPTION**	
	Public	Enregistrer ()	Permet l'enregistrement des serveurs	
	Public	Modifier ()	Permet de modifier les caractéristiques d'un serveur	
	Public	Consulter ()	Permet la consultation des serveurs	

Tableau 30 : Description de la classe SERVEUR

Classe : IMPRIMANTE classe spécialisante de l'entité PERIPHERIQUE				
	VISIBILITE	**NOM**	**DESCRIPTION**	**TYPE**
Attributs	Public	TypeImprim	Le type de l'imprimante	Texte
	Public	Resolution	La résolution maximale de l'imprimante	Entier
	Public	Vitesse	La vitesse d'impression	Entier
	Public	Technologie	La technologie utilisée	Texte
	Public	Format	Le format d'impression	Texte
Méthodes		**NOM**	**DESCRIPTION**	
	Public	Enregistrer ()	Permet l'enregistrement des périphériques	
	Public	Modifier ()	Permet de modifier les caractéristiques d'un périphérique	
	Public	Consulter ()	Permet la consultation des périphériques	

Tableau 31: Description de la classe IMPRIMANTE

IV. Conception des Différentes Couches Logicielles

IV.1. Conception de la Couche Présentation

La couche de présentation ou IHM (Interface Homme Machine) est la partie visible de l'application. Le comportement interactif des IHM avec l'utilisateur est piloté par des classes contrôleurs. Dans le cas de notre application, les classes contrôleurs comme ***« Intervention », « matériel »*** définies au niveau de la conception des associations, sont chargées d'établir la jonction entre les classes et les fenêtres.

✦ Enumération des interfaces utilisateurs

L'utilisation d'une application est rendue possible par ses interfaces utilisateur. L'énumération des vues d'IHM permet de développer la structure des classes de la couche de présentation. Le tableau suivant donne la liste des IHM de l'application.

Fenêtres	**Description des Fenêtres**
FenLogin	Fenêtre d'authentification
FenAjoutAppel	Fenêtre d'enregistrement des appels
FenConsultAppel	Fenêtre de consultation de la liste des appels
FenModifAppel	Fenêtre de modification des informations d'un appel
FenRechAppel	Fenêtre de recherche d'un appel
FenAjoutIncident	Fenêtre d'enregistrement d'un incident
FenConsultIncident	Fenêtre de consultation de la liste des incidents
FenAjoutProcResolIncident	Fenêtre d'enregistrement de la procédure de résolution d'un incident

Fenêtres	Description des Fenêtres
FenModifIncident	Fenêtre de modification des informations d'un incident
FenAjoutMat	Fenêtre d'enregistrement des caractéristiques d'un nouveau matériel et des références de son utilisateur
FenValidAffect	Fenêtre de validation d'une affectation
FenRechMat	Fenêtre de recherche d'un matériel
FenModifMat	Fenêtre de modification des caractéristiques d'un matériel et/ou des références de son utilisateur
FenConsultMat	Fenêtre de consultation de la liste des matériels
FenAjoutEquipe	Fenêtre d'enregistrement des équipes
FenRechEquipe	Fenêtre de recherche d'une équipe
FenConsultEquipe	Fenêtre de consultation des différentes équipes
FenModifEquipe	Fenêtre de modification des informations sur les équipes
FenAjoutPrest	Fenêtre d'enregistrement des prestataires extérieurs
FenConsultPrest	Fenêtre de consultation de la liste des différents prestataires extérieurs
FenModifPrest	Fenêtre de modification des informations sur les prestataires extérieurs
FenRechPrest	Fenêtre de recherche d'un prestataire

Tableau 32: Liste descriptive des vues d'IHM de notre Application

- **Organisation de la couche présentation**

L'organisation de la couche de présentation permet de préciser les relations entre les classes de l'interface utilisateur et la manière dont ces classes communiquent avec les classes métier. La structuration du système en 5 couches logicielles fait que les classes de l'interface utilisateur communiquent avec les classes métier à travers des classes contrôleurs appartenant à la couche application. Le diagramme UML suivant montre la hiérarchisation des classes de la couche de présentation.

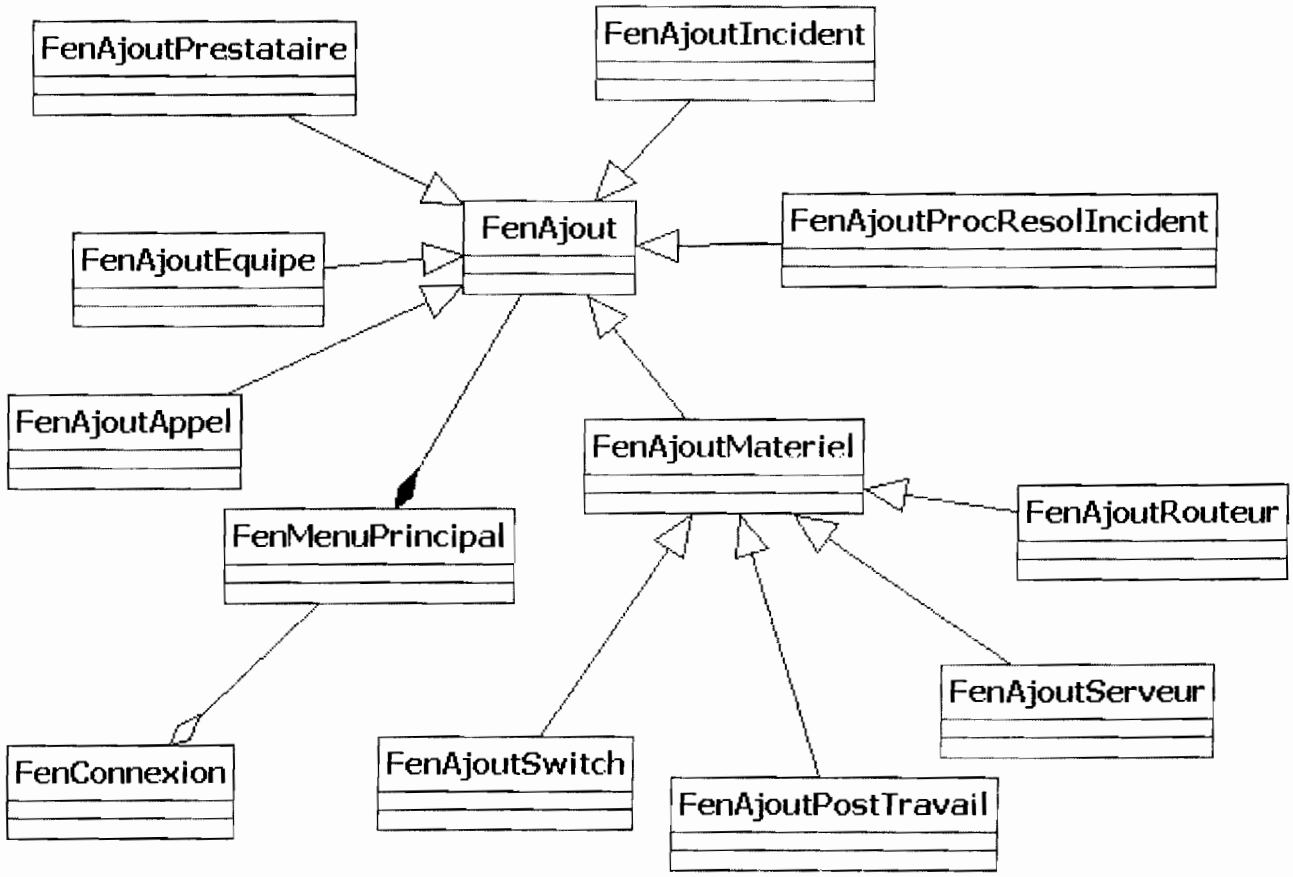


Figure 23 : Extrait du Diagramme hiérarchique des fenêtres

IV.2. Gestion de la Persistance des Données

La gestion de la persistance objet est une problématique complexe, qui est en passe de devenir l'un des volets stratégiques des architectures logicielles. Elle traite de la manière dont on peut sauvegarder et restaurer l'état des objets manipulés au sein des applications, dans le but de prolonger leur durée de vie au-delà de la durée de vie d'une session applicative.

Les Différents Modes de Stockage

La conception du stockage des données consiste à étudier sous quelle forme les instances seront sauvegardées sur un support physique. La réalisation d'un stockage des instances varie suivant le mode de stockage retenu. Dans tous les cas, la réalisation d'un modèle objet facilite la maintenance des données stockées. Il existe aujourd'hui plusieurs modes de stockage possibles.

- ✓ Le système de fichiers est actuellement le moyen le plus rudimentaire de stockage. Il ne permet que de lire ou d'écrire une instance par des moyens externes à l'application et il n'a aucune capacité à administrer ou à établir des requêtes complexes sur les données.
- ✓ La base de données relationnelle ou SGBDR est un moyen déjà plus sophistiqué. Il existe aujourd'hui une large gamme de SGBDR répondant à des besoins de volume, de distribution et d'exploitation différents. Le SGBDR permet d'administrer les données et d'y accéder par des requêtes complexes. C'est la technique la plus répandue.
- ✓ La base de données objet ou SGBDO[1] constitue la méthode la plus élaborée de toutes. Cette technique élude la conception d'un stockage des données puisqu'elle permet de stocker et d'administrer directement des instances de classe. Cette technique n'a pourtant pas connu un grand succès sur le marché des bases de données.
- ✓ La base de données XML ou SGBDX[2] ou est un concept émergeant qui répond au besoin croissant de stocker des documents XML sans risque d'altération de ces derniers. Dans le cadre de développement orienté objet qui nous occupe, cela signifierait une translation intermédiaire entre nos objets et un format XML.

Malgré le fait qu'il existe des systèmes dédiés à la persistance objet (bases de données objet), nous avons choisi le mode de stockage relationnel et les raisons en sont que :

- ➢ **l'interopérabilité avec l'existant :** à l'heure actuelle en effet, la plupart des données informatiques sont stockées au sein de bases de données

[1] **SGBDO** : Système de Gestion de Base de Données Objet
[2] **SGBDX** : Système de Gestion de Données au format XML

relationnelles. L'utilisation d'un SGBD pour la persistance objet permet ainsi d'accéder aux données existantes de l'entreprise ;

- **la possibilité de tirer parti de fonctionnalités sophistiquées :** les SGBDs relationnels allient performance et fiabilité, à travers notamment des fonctionnalités de recherche puissantes (langage de requête, indexation) et le support de la concurrence d'accès (transactions). L'utilisation d'un SGBDR permet également aux entreprises de mettre à profit l'expertise acquise dans ce domaine.

Choix du système de gestion de base de données

- **MySQL :** c'est un Système de Gestion de Base de Données Relationnel développé par la société MySQL AB. Il est compatible avec plusieurs systèmes d'exploitations comme : Windows, Linux, Mac OSX ou Unix.

AVANTAGES :

- Solution très courante en hébergement public ;
- Open Source, bien que les critères de licence soit de plus en plus difficiles à supporter ;
- Facilité de déploiement et de prise en main.

INCONVENIENTS :

- Ne supporte qu'une faible partie des standards SQL ;
- Très peu de richesse fonctionnelle;
- Pas d'héritage de tables.

- **Ms Access : Microsoft Access** ou **MS Access** (officiellement **Microsoft Office Access**) est un système de gestion de base de données relationnelle (SGBDR) édité par Microsoft. MS Access fait partie de la suite bureautique Microsoft Office Pro. MS Access est donc un SGBDR présentant une approche bureautique et n'est pas conçu pour supporter de très grandes bases de données opérationnelles sur de vastes réseaux, ces utilisations étant confiées dans la gamme Microsoft à Microsoft SQL server. Une base de données Access ne peut dépasser deux gigaoctets.

AVANTAGES :

- Facile à utiliser et approprié pour les débutants ;
- Peut servir de client pour un serveur de bases de données comme SQL Serveur, ORACLE, MySQL...

INCONVENIENTS :

- Limité en nombre d'utilisateurs ;
- Gère des données de petite taille (moins de deux gigaoctets);

Pour la manipulation des données de notre système, nous avons choisi le SGBD MS Access et ce, pour plusieurs raisons :

- Les données utilisées par notre système sont de petite taille et le nombre d'utilisateurs est très limité (seulement les agents du service informatique);
- Si la taille des données devient très importante, MS Access peut servir de client pour un serveur de bases de données comme MySQL, SQL Serveur, ORACLE ;
- Parmi les deux SGBDR présentés, MS Access est le seul à être homologué par le groupe BOLLORE.

Le choix de MS Access comme SGBDR et de la plate forme Visual Studio.Net comme environnement de développement n'a aucune influence sur le coût de réalisation du projet car le service informatique dispose déjà de ces outils.

↓ Fonctionnalités Attendues d'une Couche d'Accès aux Données dans le Contexte du SGBD

La couche d'accès aux données doit permettre de communiquer avec le serveur de base de données et de renvoyer les objets métiers au programmeur. Le serveur de base de données et la structure de la base de données doivent rester transparents.

Les fonctionnalités attendues d'une couche d'accès aux données sont :

- implémenter les principales méthodes de manipulation des données (create, update, insert, delete) ;
- garantir l'indépendance vis-à-vis du serveur de bases de données ;

- garantir la cohérence des objets dans un contexte client/serveur avec des accès concurrents.

Si plusieurs utilisateurs instancient un objet de notre système et font des modifications sur celui-ci en même temps, la couche d'accès aux données doit veiller à la cohérence de l'objet.

✦ Les Différentes Approches

Différentes approches existent pour mettre en place une couche d'accès aux données dans un contexte de bases de données relationnelles :

- **la méthode force brute :** elle consiste à intégrer du code SQL[1] ou OQL[2] dans les classes métier de façon à ce que celles-ci puissent accéder directement à la base de données. Cette stratégie souvent utilisée en raison de sa simplicité est bien adaptée en début de projet quand l'accès aux données n'est pas encore critique ou dans le cas où les objets métiers sont peu nombreux et faciles à gérer. Mais cette méthode est inadaptée dans la plupart des cas à cause du couplage fort entre objets métier et serveur de données.
- **Utilisation des DAO (Data Access Objects) :** les DAO séparent la logique d'accès aux données des classes métiers. La stratégie typique consiste à implémenter une classe d'accès aux données pour chaque classe métier, par exemple, à la classe métier « utilisateur », correspondra une classe DAO utilisateur_Data encapsulant la logique d'accès aux données. L'avantage de cette approche par rapport à la méthode force brute est que les classes métier ne sont plus directement couplées avec la base de données. En revanche les classes d'accès aux données le sont. Exemple de DAO : ActiveX Data Objects.Net (ADO.Net) de Microsoft, Java Data Objects (JDO).
- **Les frameworks de persistance :** un framework de persistance ou encore couche de persistance sépare complètement la logique d'accès à la base de données des objets métiers. Au lieu d'écrire du code implémentant la logique d'accès aux données, la couche de persistance définit des

[1] Structure Query Language
[2] Object Query Language

metadonnées décrivant les relations entre les objets métier, les associations entre objets et les tables correspondantes dans la base de données. Ces metadonnées seront utilisées pour générer automatiquement tout le code nécessaire à la gestion de la persistance.

- **Les services :** un service est une opération offerte par un composant qui peut être utilisé par d'autres composants. Les services permettent à une application d'utiliser les fonctionnalités d'une autre application et aussi d'accéder à des données. Il existe une variété de services tels CORBA (Common Object Request Broker Architecture), DCOM (Distributed Component Object Mode), les procédures stockées, les Web Services.

Les Web Services sont les plus utilisés car ils facilitent la réutilisation à travers l'intégration des systèmes.

IV.3. Gestion de la Persistance avec la Plate Forme .NET

L'architecture .NET comprend une API[1] standard d'accès aux données, appelée ADO.NET[2]. Bien que ADO.NET facilite l'accès aux SGBD, cette API nécessite l'écriture de code et elle ne permet pas directement de résoudre la problématique de gestion de la persistance des objets de façon transparente. En effet, cette API ne permet de manipuler que des ensembles de données tabulaires issues de requêtes SQL, et non les objets métier utilisés par l'application. Nous allons implémenter la liaison entre nos objets métiers et l'interface de programmation ADO.NET.

IV.4. Le Choix de la Solution Framework de Persistance

La solution framework de persistance a été retenue pour gérer la persistance objet dans le cadre de notre système. Ce choix est principalement guidé par le souci de doter l'application d'une couche de persistance répondant aux problèmes de la persistance objet à savoir :

- des interfaces de programmation intuitives ;
- le Mapping objet-relationnel ;
- la gestion de différentes bases de données ;
- la gestion de la concurrence ;

[1] Application Programming Interface
[2] Active X Data Object for .Net

- la gestion des transactions ;
- gestion des types de données évolués ;
- l'encryptage des données ;
- la gestion du cache objet.

Il existe plusieurs frameworks de persistance sur le marché. On peut citer ObjectSpaces de Microsoft, Data Tier Modeler de Evaluant, Deklarit, Pragmatier... L'inconvénient majeur de ces outils est qu'il s'agit pour la plupart de solutions commerciales assez coûteuses.

Parmi les frameworks de persistance disponibles pour l'environnement Microsoft.Net, le Data Tier Modeler (DTM) de la société Evaluant a été retenu dans le cadre de notre application. Cet outil présente le double avantage de s'intégrer harmonieusement à l'environnement de développement Microsoft.Net et d'être performant.

IV.5. Passage du modèle objet au modèle relationnel

Le principe de mapping objet/relationnel utilisé par DTM consiste à :

- faire correspondre à chaque classe métier une table dans la base de données. La clé primaire de cette table est générée automatiquement et ses attributs correspondent aux attributs de la classe métier.
- traduire les associations entre classes métiers sous forme de contraintes référentielles en tenant compte des rôles des classes dans ces associations.
- créer une table pour chaque association de type « plusieurs à plusieurs ».
- créer une seule table pour les classes liées par une relation d'héritage.

V. Coût de réalisation du projet

Nous allons calculer le coût de développement par le mode semi détaché de la méthode COCOMO (Voir Annexe 2 page : 110, pour les détail) qui est le plus approprié dans notre cas présent.

Le coût total de développement dans notre cas sera estimé à **effort*X** où **X** représentant le salaire moyen d'un informaticien au Burkina Faso.

Coût de développement :

. **Formule de calcul :**

Effort de développement requis : effort = 3 (KLSL) $^{1.12}$

Temps de développement : TDEV = 2.5 (*effort*) $^{0.35}$

Nombre de développeurs : N = effort/TDEV.

Coût total de développement : CTD = effort*X

Application numérique :

Nous estimons à **3000** le nombre de lignes de code source nécessaire pour la réalisation de notre application.

Nous estimons le salaire moyen d'un informaticien BAC + 3 au Burkina Faso à 200 000 FCFA. **Soit : X = 200 000 FCFA**

effort = 3 (3000/1000)$^{1.12}$

= **10.27 Homme-mois**

TDEV = 2.5 (10.27)$^{0.35}$

= 5.65 soit 6 mois

CTD = 10.27*200 000

= **2 054 000 Fcfa**

Coût de formation des utilisateurs :

Prix de l'horaire (FCFA)	Nombre d'heures par utilisateur	Nombre d'utilisateurs	Montant (FCFA)
2500	05	6	**75 000**

Montant = (prix de l'horaire * nombre d'heures par utilisateur)*nombre utilisateurs

Coût total de réalisation :

Désignation	Prix (FCFA)
Coût matériel et logiciel	0
Coût de développement	2 054 000
Coût de formation	75 000
Coût total	**2 129 000**

VI. Procédure Transitoire

La procédure transitoire est un ensemble de tâches consécutives à exécuter pour passer du système actuel au futur système. Ces tâches correspondent tout d'abord à la mise en place du nouveau système et son fonctionnement en parallèle avec l'existant pendant une période déterminée, afin de déceler d'éventuelles erreurs et de s'assurer de son efficacité en terme de service rendu :

- Une fois l'application réalisée, nous préconisons qu'elle fonctionne d'abord en mono poste c'est-à-dire qu'elle soit déployée sur un poste de travail isolé afin qu'elle subisse une série de tests. Ces tests ont pour but de s'assurer de la qualité de l'application en terme d'efficacité, de robustesse, de performance etc. Au cours de ces tests nous pourrons alors corriger les éventuelles défaillances.

- Après cette phase de correction d'erreurs, l'on passera à la formation des utilisateurs et à la mise en réseau de l'application. Et là, nous préconisons l'usage de la méthode manuelle pour mémoriser les données, en parallèle avec le nouveau système pendant une période de six mois.

- Au bout des six mois d'essai, les dernières modifications seront effectuées et nous aurons alors une version stable de notre application qui garantit la fiabilité et la cohérence du système.

VII. Procédure de secours

Les procédures de secours sont des procédures organisationnelles à appliquer lors d'une indisponibilité des ressources informatiques indispensables au bon fonctionnement du système.

Ces procédures permettent d'offrir un minimum de services conformément aux exigences des utilisateurs. Elles seront exécutées lors du fonctionnement en mode dégradé du système. Le mode dégradé est une situation où le système n'est pas en mesure d'offrir toutes les fonctionnalités aux utilisateurs.

Ce système peut être entièrement incapable de fonctionner. Diverses situations peuvent être à l'origine du mode dégradé du système.

❖ Indisponibilité d'un poste de travail

La panne d'un ordinateur ne saurait arrêter totalement les traitements effectués sur le poste de travail correspondant. Au vu des possibilités offertes par le système à mettre en place, les utilisateurs de ce système pourront effectuer des traitements de concert avec les utilisateurs d'autres postes afin d'éviter un blocus dans le circuit des traitements en attendant la réparation du poste ou son remplacement.

❖ Plantage du logiciel

En cas de plantage du logiciel, il est conseillé de réinitialiser le programme. En cas de persistance de la panne, il faudra contacter les développeurs pour une maintenance.

❖ Indisponibilité totale du système

En cas de panne généralisée du système, nous préconisons un fonctionnement en mode manuel jusqu'à ce que le problème soit résolu.

VIII. Politique de sécurité

❖ Surveillance du réseau

Il est important pour le service informatique de se doter d'un outil de surveillance du réseau, afin d'assurer un bon fonctionnement du réseau et de prévenir un certains nombres d'incidents qui pourraient entraver la disponibilité du réseau.
Nous proposons alors le logiciel **WhatsUp Gold**, qui est un logiciel de surveillance très performant et qui est adapté à toutes les tailles d'entreprises possédant un réseau. Il offre une visibilité temps réel sur l'état des réseaux distribués sur les différents sites que sur le réseau local. Les agents du service informatique ont ainsi la possibilité d'identifier les problèmes et de mettre en place des contre-mesures a fin d'éviter toute perte de disponibilité et d'assurer la continuité de l'activité. Ce logiciel est déjà utilisé par le service informatique et il n'est donc pas nécessaire de l'acquérir.

❖ Protection contre les virus informatiques

L'une des plus grandes sources de danger d'un système informatique et des données qu'il renferme sont les virus informatiques. Ces petits programmes informatiques s'auto-reproduisent et se répandent sur le réseau à travers les supports amovibles comme les clés USB, les CDROM etc. Cela peut avoir pour conséquence la déstabilisation du système et la destruction des données. En vue d'éviter tout désagrément et de renforcer les mesures de sécurité nous préconisons d'installer un anti-virus sur tous les postes de travail du réseau local du service informatique et sur les serveurs.

❖ Confidentialité des données

La confidentialité des données requiert la définition des droits d'accès. Ceci se traduit par l'utilisation de mots de passe et de noms de connexion pour l'accès aux données de la base de données. De cette façon l'accès à la base de données sera restreint aux personnes qui sont autorisées tout en contrôlant qui peut afficher et modifier les informations de la base de données

❖ Protection contre les catastrophes

Les catastrophes naturelles susceptibles d'endommager les installations sont les incendies, les orages, les inondations, les coupures électriques et la foudre. Le service informatique dispose déjà d'une politique de protection contre les catastrophes naturelles qui répond à nos attentes.

❖ La politique de sauvegarde

Pour prévenir le système contre d'éventuels sinistres, il est conseillé de définir une politique de sauvegarde des données. Celle que nous proposons consiste en :

- Des sauvegardes journalières qui ont une durée d'une semaine ;
- Des sauvegardes hebdomadaires qui ont une durée d'un mois ;
- Des sauvegardes mensuelles qui ont une durée de six mois ;
- Des sauvegardes annuelles qui seront conservées définitivement.

Par ailleurs, il est conseillé que chacune de ces sauvegardes soit en double.

CONCLUSION

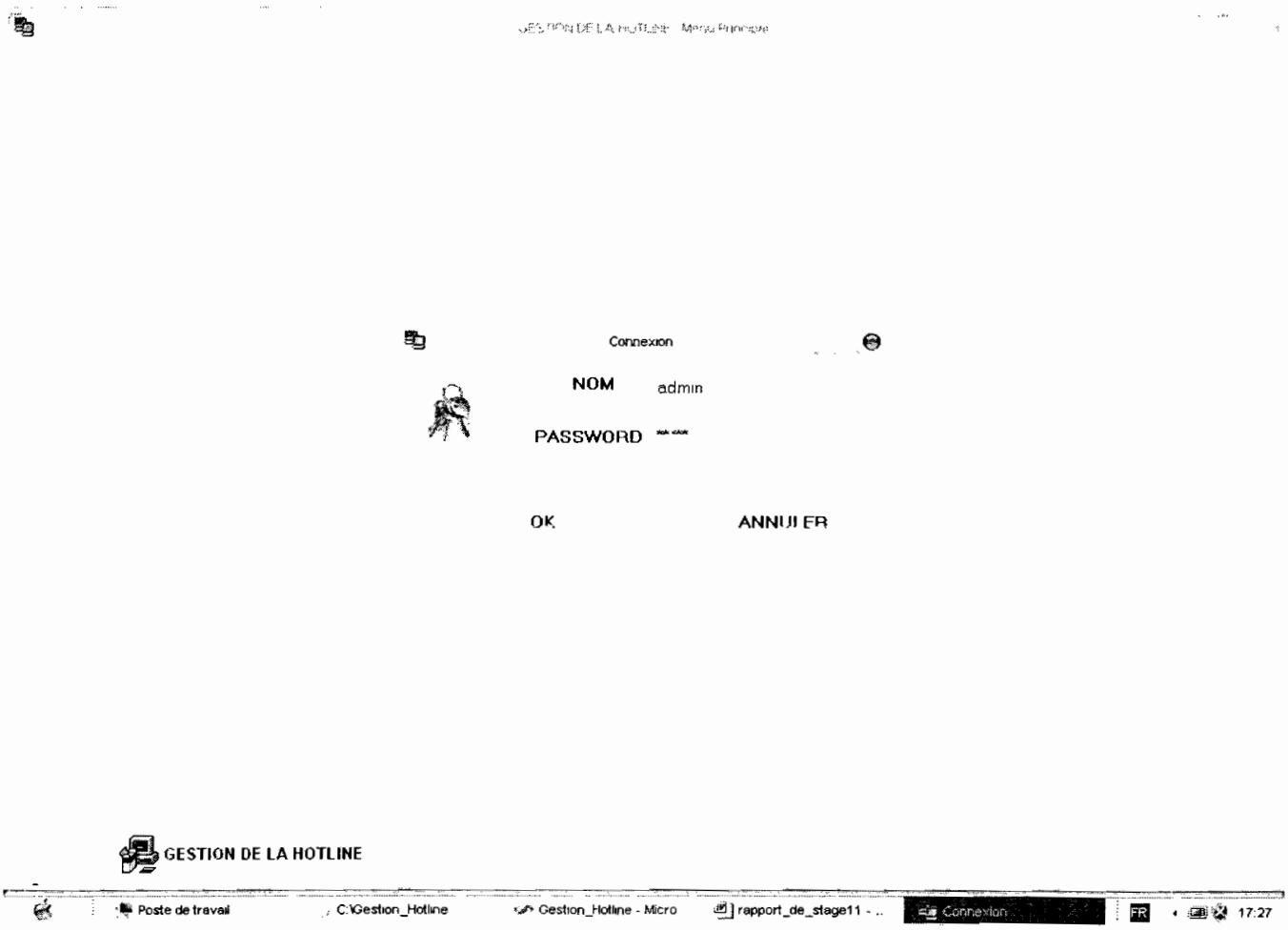
Cette phase conception qui vient de s'achever précède la phase de codage. À ce niveau, toutes les questions relatives à l'agencement et aux détails de la solution ont été modélisées. Ainsi, les interrogations restantes concernent exclusivement la bonne utilisation des langages et des outils de développement spécifiés à savoir visual basic.net et Ms Access.

CONCLUSION GENERALE

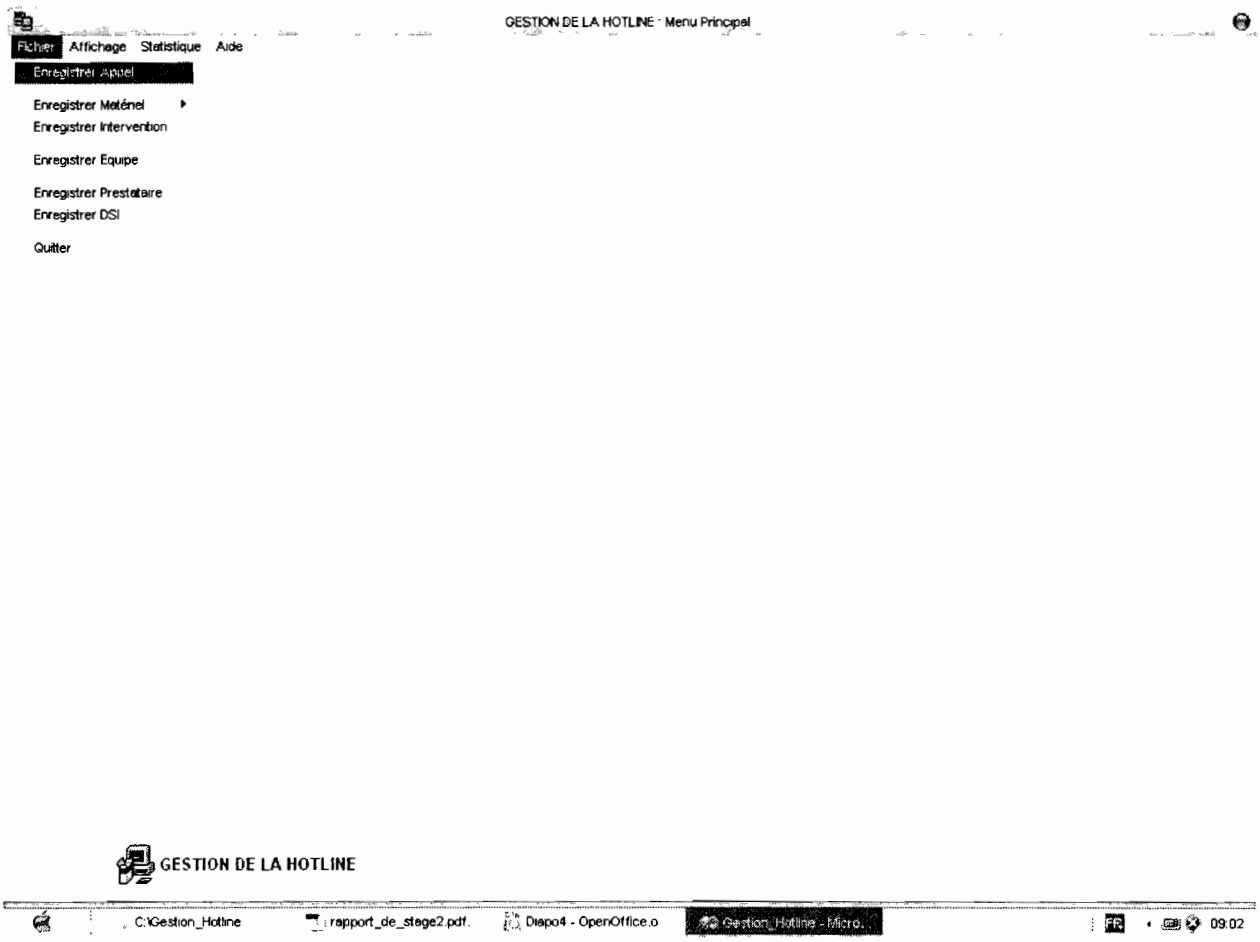
L'étude qui nous a été demandée durant nos trois mois stage, s'est articulée au tour du thème : « Gestion de la Hotline : Gestion du parc informatique et Gestion des incidents ». Au cours cette étude, conformément au processus 2TUP, nous sommes partis de l'analyse fonctionnelle et de l'analyse technique de notre futur système pour dégager une solution respectant l'ensemble des contraintes et conforme aux souhaits des utilisateurs.

Notre stage qui s'est déroulée en parfaite harmonie avec le groupe de pilotage, nous a apporté beaucoup sur le plan académique et professionnel.
Nous souhaitons que le travail que nous avons entrepris connaisse son achèvement par la mise en place effective de notre système afin que nos efforts soient couronnés.

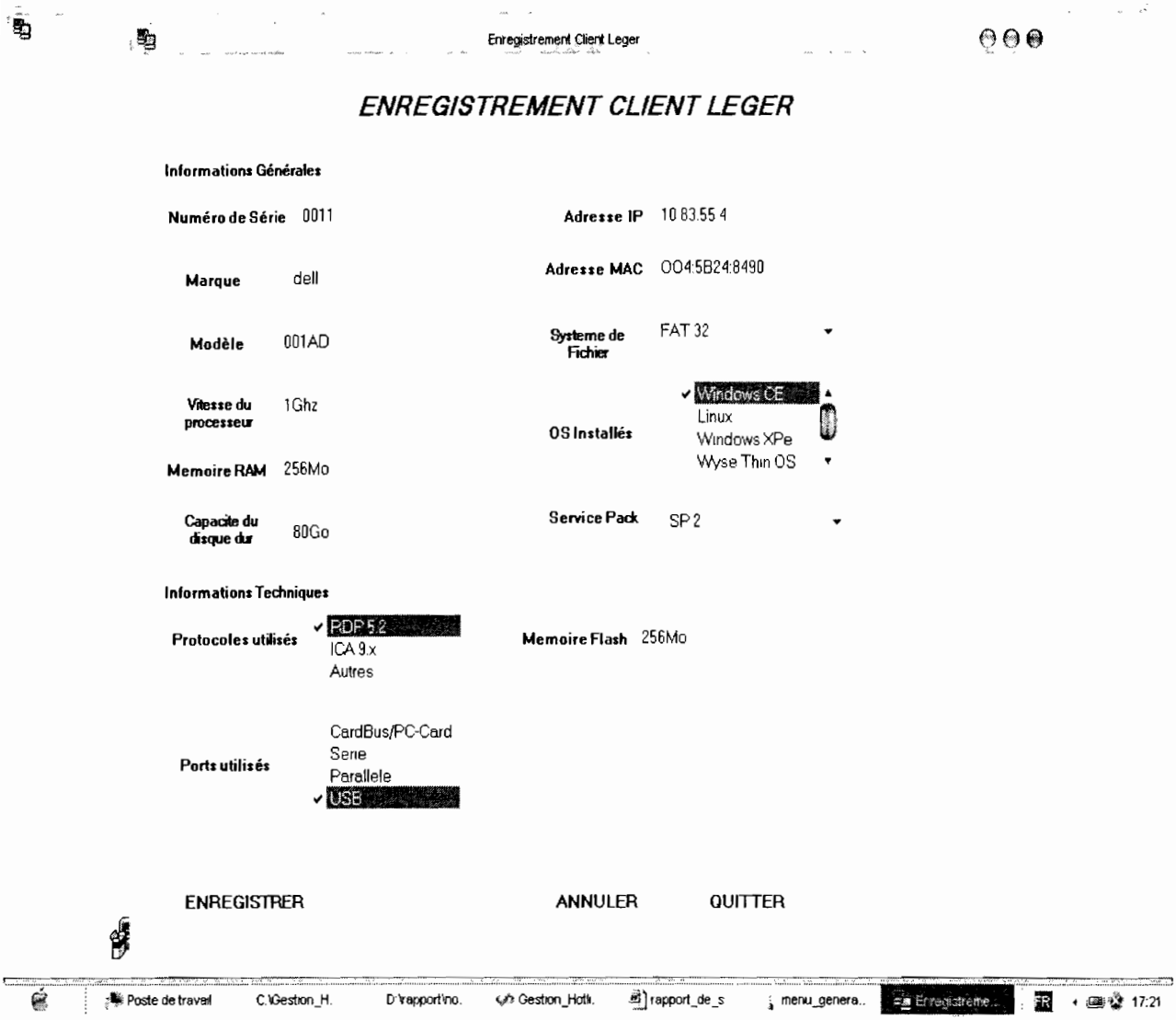
PRESENTATION DE QUELQUES MAQUETTES



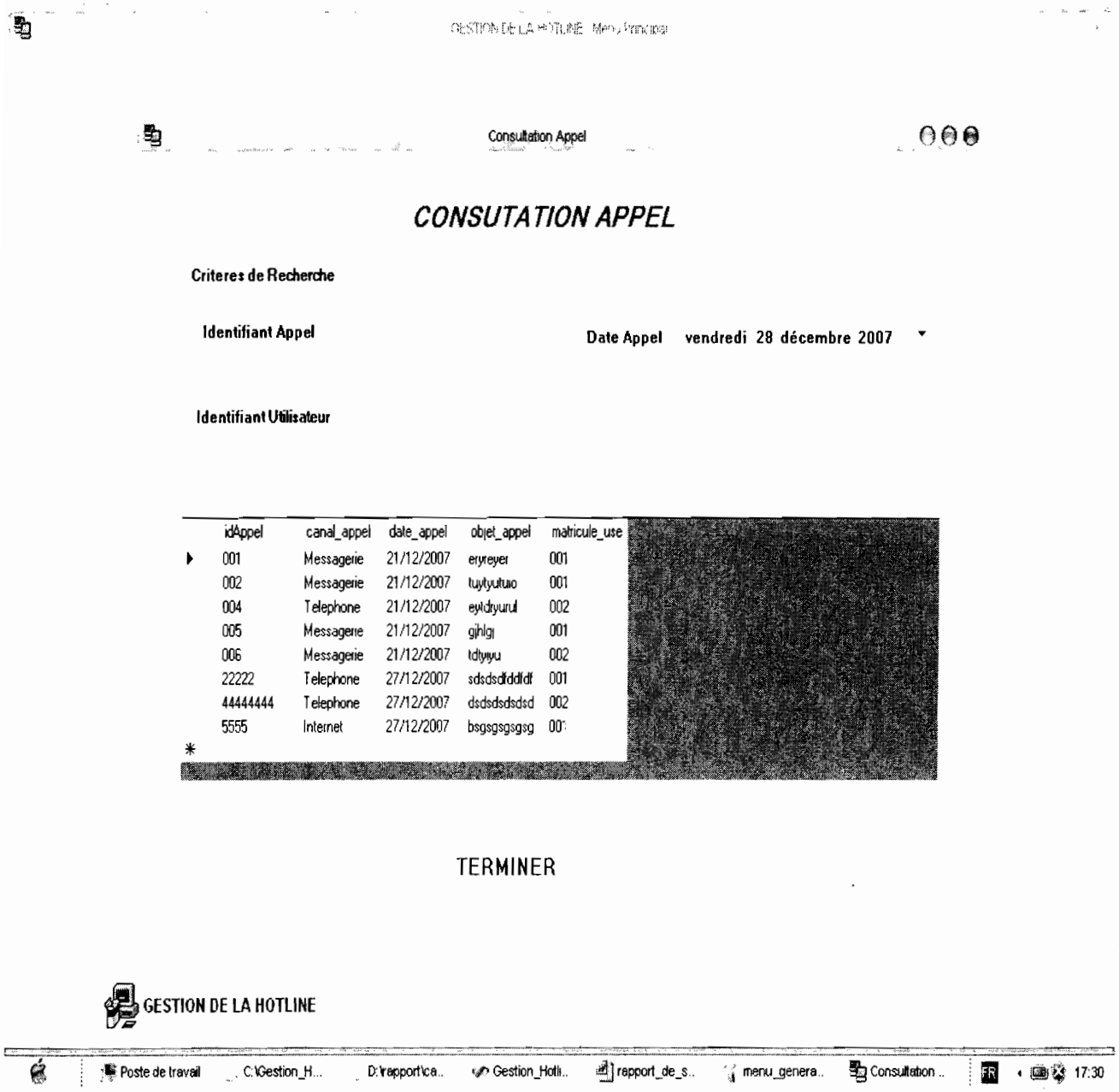
Fenêtre de connexion à l'application. Cette fenêtre s'affiche au démarrage de l'application



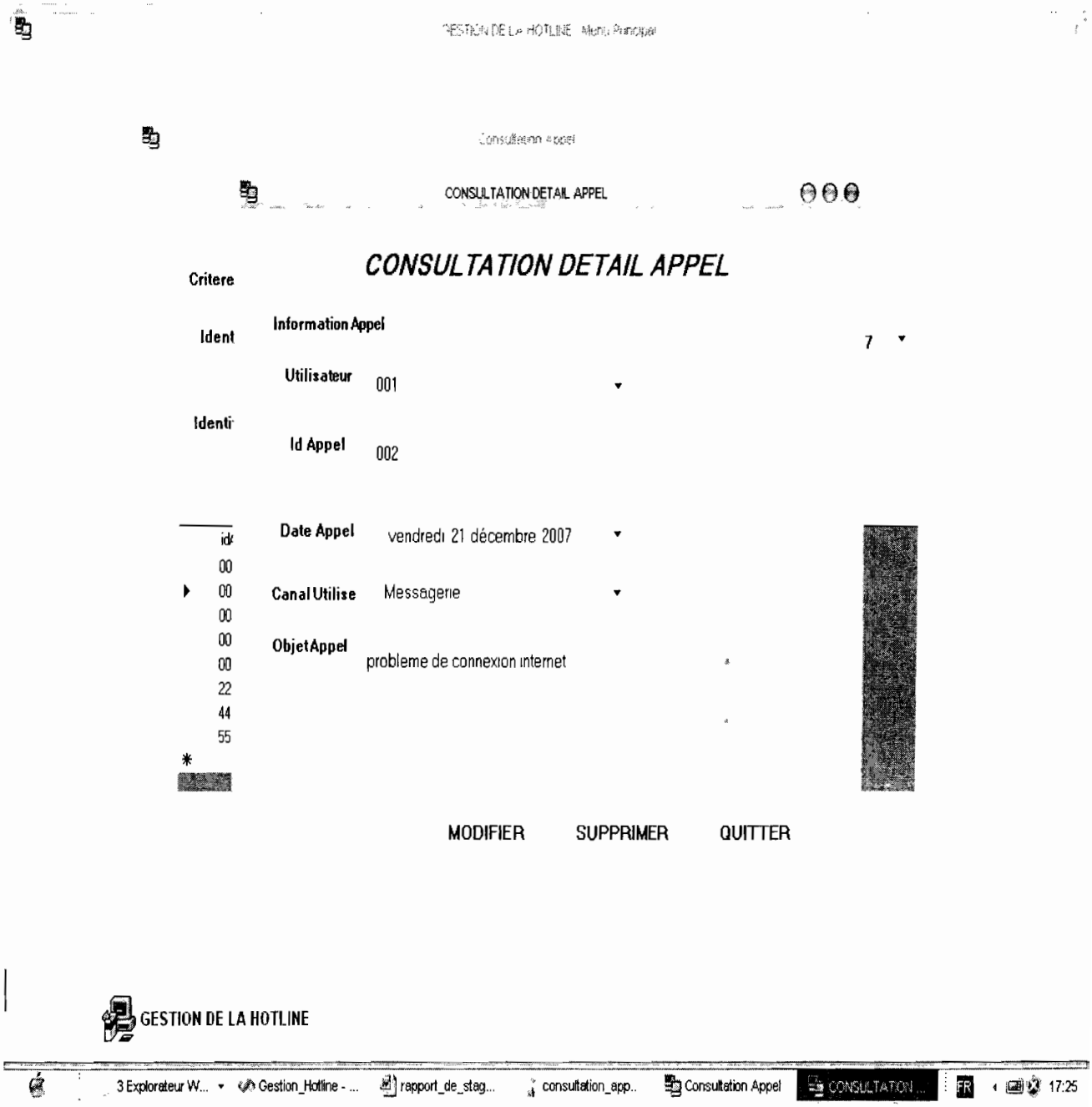
Le menu principal de l'application. Après une indentification réussie, l'utilisateur à accès à cette fenêtre qui lui donne la possibilité d'accéder aux fonctionnalités du système.



Fenêtre d'enregistrement d'un client léger dans la base de donnée.



Fenêtre de consultation et de recherche des appels enregistrés



Fenêtre de modification et de suppression d'un appel enregistré dans la base de donnée

BIBLIOGRAPHIE

- Pascal Roques, Franck Vallé, **« UML en Action de l'Analyse des besoins à la conception en Java »**, 2e édition, Groupe Eyrolles, 2003, ISBN 12-212-11213-0 ;
- Pascal Roques, Franck Vallé, **« UML 2 en Action de l'Analyse des besoins à la conception en J2EE »**, 3e édition, Groupe Eyrolles, 2000, 2003, 2004, ISBN 12-12- 1462-Alexandre Brillant (2006), Java 5, ISBN 2-7460-3170-1 ;
- Jacques Lonchamp, deuxième partie, **« Les techniques de spécification »**
- Alexandre Durain, **« UML 2.0 et le processus 2TUP »**;
- Zango Abdoulaye, Ouattara Yacouba 2005-2006, Rapport de fin de cycle **« Migration vers une architecture 3-tier (modèle MVC) d'une application web de gestion des règlements pécuniaires des Avocats du Burkina (GESTCARPA) » ;**
- Ki Emmanuel, Ouédraogo Yacouba 2005-2006, Rapport de fin de cycle **« mise en oeuvre d'un système de gestion des Micro crédits financés par des bailleurs de fonds ».**

Sites de référence :

- http://www.microsoft.com/france/msdn/net/decouvrez/fiche-vbnet.mspx;
- http://www.application-servers.com;
- http://www.uml.org/#UML2.0.

ANNEXES

ANNEXE 1 : CONCEPTS ET FORMALISMES DES DIAGRAMMES D'UML

UML est un langage graphique de modélisation objet permettant de spécifier, de construire, de visualiser et de décrire les détails d'un système logiciel. Il comprend un vocabulaire et un ensemble de règles centrées sur la représentation conceptuelle et physique d'un système logiciel. Ses domaines d'utilisation sont :

- Visualisation d'un système ;
- Spécification d'un système ;
- Construction d'un système ;
- Documentation d'un système.

Le modèle conceptuel d'UML comprend les notions de base génériques du langage. Il défini trois sortes de « briques » de base :

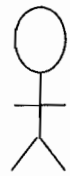
- Les éléments, qui sont des abstractions essentielles à un modèle. Il existe quatre types d'éléments dans UML :
  - Les éléments structurels (classes, interface, collaboration, ...) ;
  - Les éléments comportementaux (interaction, automate à états finis) ;
  - Les éléments de regroupement (package) ;
  - Les éléments d'annotation (note).
- Les relations, qui constituent des liens entre ces éléments. Il existe quatre types de relations dans UML :
  - La dépendance ;
  - L'association ;
  - La généralisation ;
  - La réalisation.
- Les diagrammes, qui sont la représentation graphique d'un ensemble d'éléments et de relations qui constituent un système. UML définit neuf types de diagrammes divisés en deux catégories :
  - Les diagrammes statiques (appelés aussi diagrammes structurels) : diagrammes de classe, d'objets, de composants, de déploiements et de cas d'utilisation ;
  - Les diagrammes dynamiques (appelés aussi diagrammes comportementaux) : diagrammes d'activités, de séquences, d'états transitions et collaboration.

I. Concepts et formalismes du diagramme des cas d'utilisation

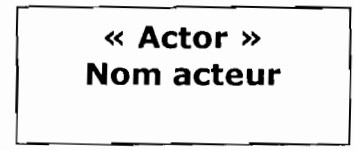
Les acteurs : un acteur représente un rôle joué par un utilisateur qui interagit avec le système. La même personne physique peut jouer le rôle de plusieurs acteurs. D'autre part, plusieurs personnes peuvent jouer le même rôle, et donc agir comme un même acteur. On distingue deux types d'acteurs :

- ✓ Les acteurs primaires, ceux pour qui un service est directement rendu par le système ;
- ✓ Et les acteurs secondaires, ceux qui participent à rendre un service.

Le formalisme d'UML des acteurs est le suivant :



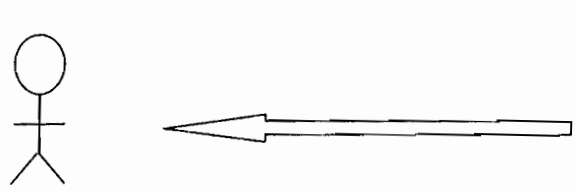
Acteur physique



Acteur non physique

Figure A2 : représentation des acteurs

Un acteur peut également participer à des relations de généralisation/spécialisation :



Un acteur **Un autre acteur**

Figure A2 : Généralisation entre acteurs

Les cas d'utilisation : ce sont une modélisation d'une fonctionnalité ou d'une classe. Les cas d'utilisation se déterminent en observant et en précisant, acteur par acteur, les séquences d'interaction du point de vue de l'utilisateur. Ils se décrivent en termes d'informations échangées et d'étapes dans la manière

d'utiliser le système. Un cas d'utilisation regroupe une famille de scénarios d'utilisation selon un critère fonctionnel. Ils décrivent des interactions potentielles, sans entrer dans les détails de l'implémentation.

Le formalisme d'UML des cas d'utilisation est le suivant :

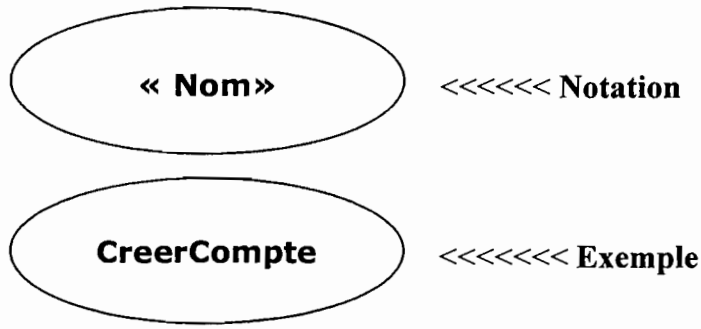


Figure A3 : représentation des cas d'utilisation

Il existe trois types de relations entre les cas d'utilisation :

- La relation de **généralisation** : le cas d'utilisation enfant est une spécialisation du cas d'utilisation parent ;
- La relation **d'inclusion** : le cas d'utilisation source comprend également le comportement de son cas d'utilisation destination. Cette relation a un caractère obligatoire (à la différence de la généralisation) et permet ainsi de décomposer des comportements partageables entre plusieurs cas d'utilisation différents ;
- Le relation **d'extension** : le cas d'utilisation source ajoute son comportement au cas d'utilisation destination. L'extension peut-être soumise à condition. Cette relation permet de modéliser des variantes de comportement d'un cas d'utilisation.

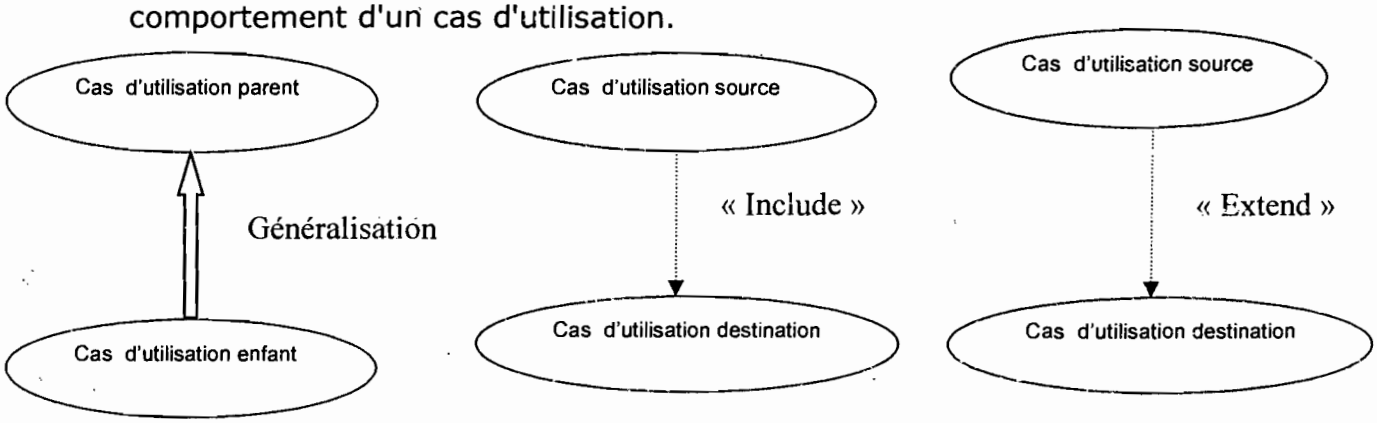


Figure A4 : les trois relations entre les cas

Le diagramme des cas d'utilisation : il représente les cas d'utilisation identifiés et les acteurs associés à chacun d'eux. Ils permettent de représenter les processus d'un domaine. Le formalisme proposé par UML est le suivant :

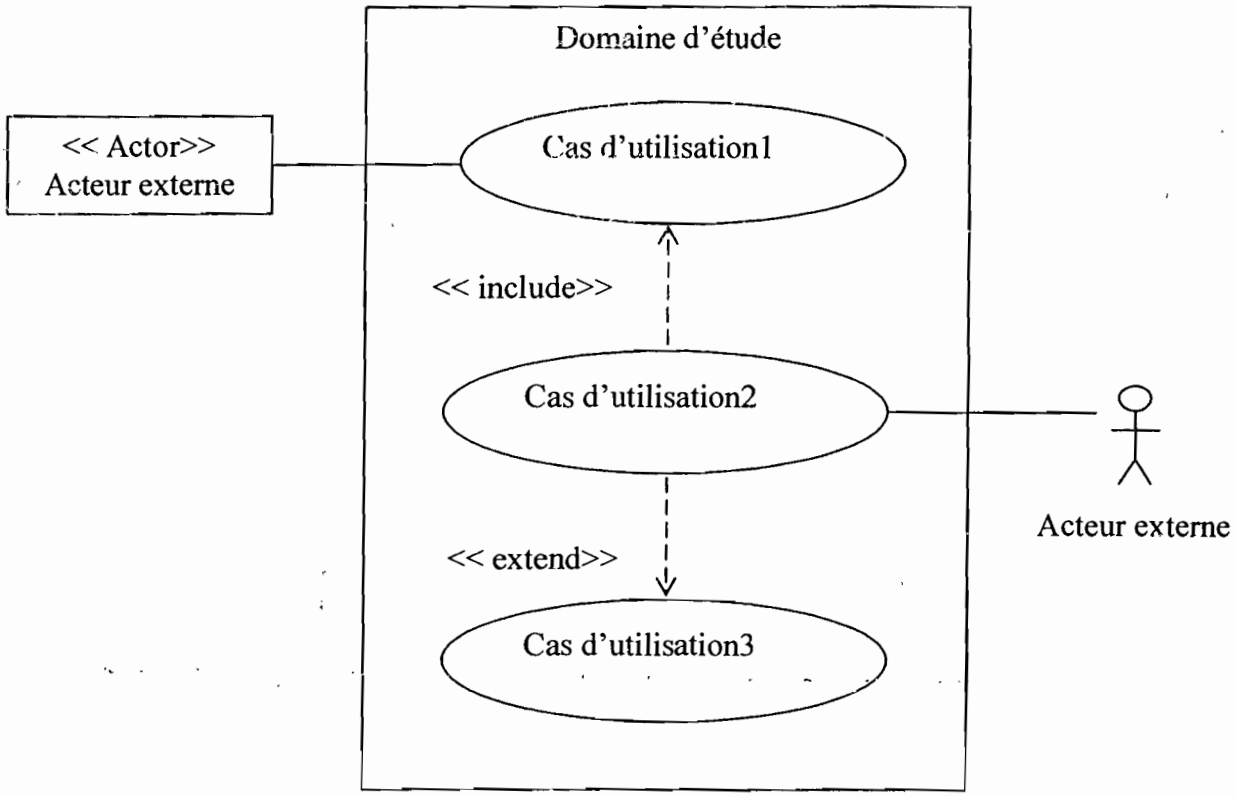


Figure A5 : formalisme du diagramme des cas d'utilisation

II. Concepts et formalismes du diagramme de classes

Concept de classe : une classe est un type abstrait caractérisé par des propriétés (attributs et méthodes) commune à un ensemble d'objets et permettant de créer des objets ayant ces propriétés.

Classe = attributs + méthodes + instanciation

Un attribut est une information élémentaire composant une classe. Il peut permettre d'identifier la classe. Il est typé (integer, real, string, etc.).
Une méthode ou opération est une fonctionnalité assurée par la classe.

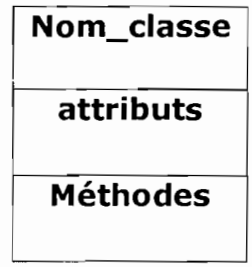


Figure A6 : représentation d'une classe

Exemple :

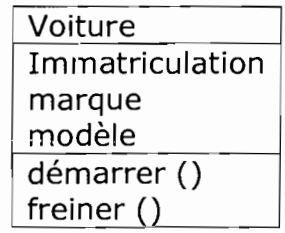


Figure A7 : exemple de classe

Concept d'attribut : Un attribut est une information élémentaire composant une classe. Un attribut peut permettre d'identifier la classe. Il est typé (integer, real, string...).

La syntaxe complète des attributs est :

Visibilité nom [multiplicité] : Type = valeur_initiale {propriétés}

La multiplicité est le nombre d'occurrences possible de l'attribut.

Concept de méthode : une méthode ou opération est une fonctionnalité assurée par la classe.

La syntaxe d'une méthode est la suivante :

Visibilité nom (liste de paramètres) : type de retour {propriétés}

Visibilité d'une caractéristique *:* détermine si d'autres éléments peuvent l'utiliser. Trois niveaux de visibilité sont possibles pour les attributs et les opérations :

- « + » : **visibilité public,** la caractéristique peut être utilisée par n'importe quelle Instance ayant une visibilité sur les instances de la classe complète.
- « - » **: visibilité private,** la caractéristique ne peut être utilisée que par des instances de la classe elle même.
- « # » **: Visibilité protected**, la caractéristique ne peut être utilisée que par des instances de la classe elle-même ou bien par les descendants directs de cette classe.

Concept d'association : une association exprime une connexion sémantique bidirectionnelle entre deux classes.

Exemple :

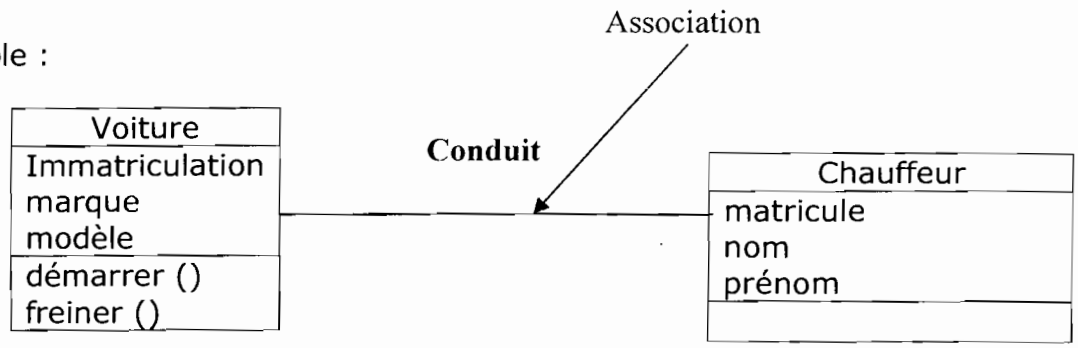


Figure A8 : exemple d'association

Concept de multiplicité : la multiplicité est le nombre d'instances d'une classe impliquée dans une association. Elle est la traduction d'une règle de gestion. En général, on fait apparaître deux nombres entiers représentant le minimum (min) obligatoire et le maximum autorisé (max).

On peut avoir les multiplicités suivantes :

0...i 0...*

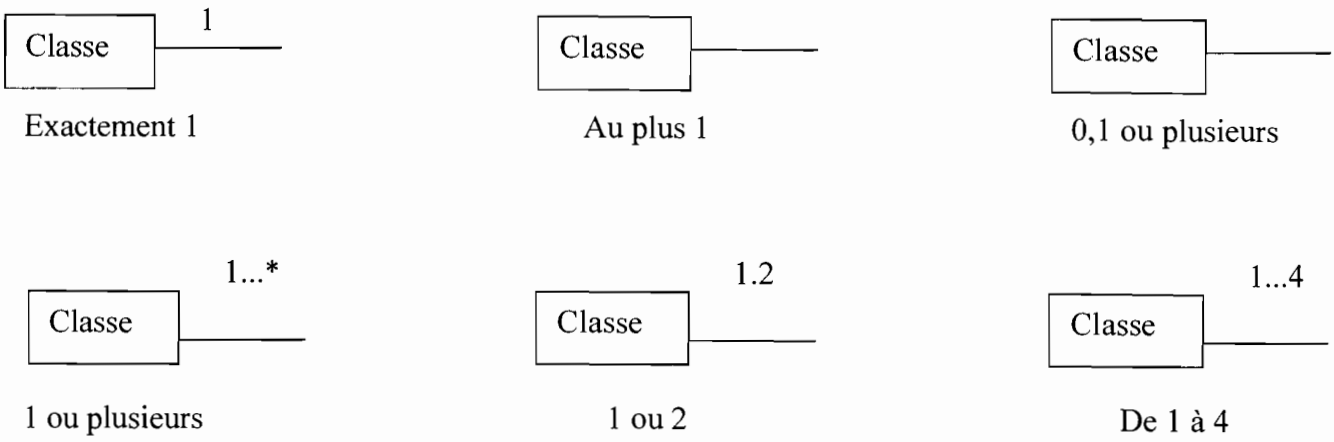


Figure A9 : les différentes multiplicités

Concept de Généralisation / Spécialisation : la généralisation est une relation entre un élément général (superclasse ou classe mère) et un élément dérivé de celui-ci mais plus spécifique désigné par le terme sous-classe ou classe fille.
La spécialisation d'une classe permet de mettre en facteur commun certaines descriptions, soit préciser de nouvelles contraintes sur le modèle de classes.

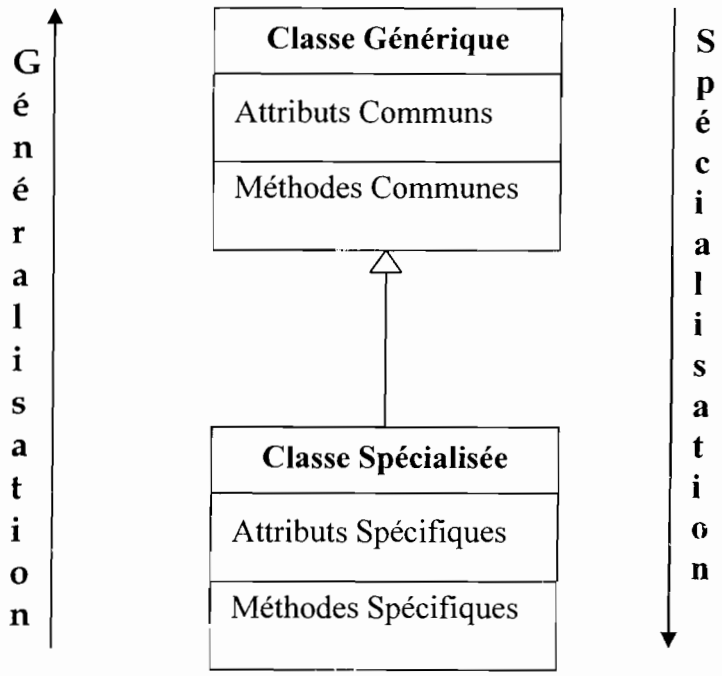


Figure A10 : les relations de généralisation et de

Concept de Agrég: une relation d'inclusion entre une partie et un tout (l'agrégat). Elle met en évidence une classe agrégat et une classe agrégée. L'agrégation est représentée par un losange clair associé à l'agrégat.

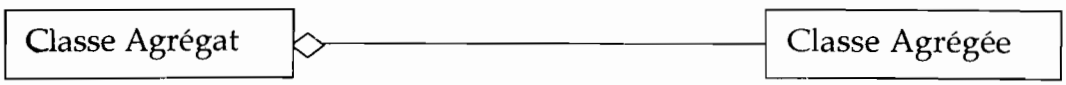


Figure A11 : relation d'agrégation

Concept de Composition : c'est une forme d'agrégation qui véhicule des notions de fortes propriétés. Dans la composition, le tout est responsable de la mise à disposition de ses parties. La composition est représentée par un losange noir.

Figure A12 : relation de composition

NB : la suppression d'un agrégat entraîne la suppression des objets agrégés.

Exemple :

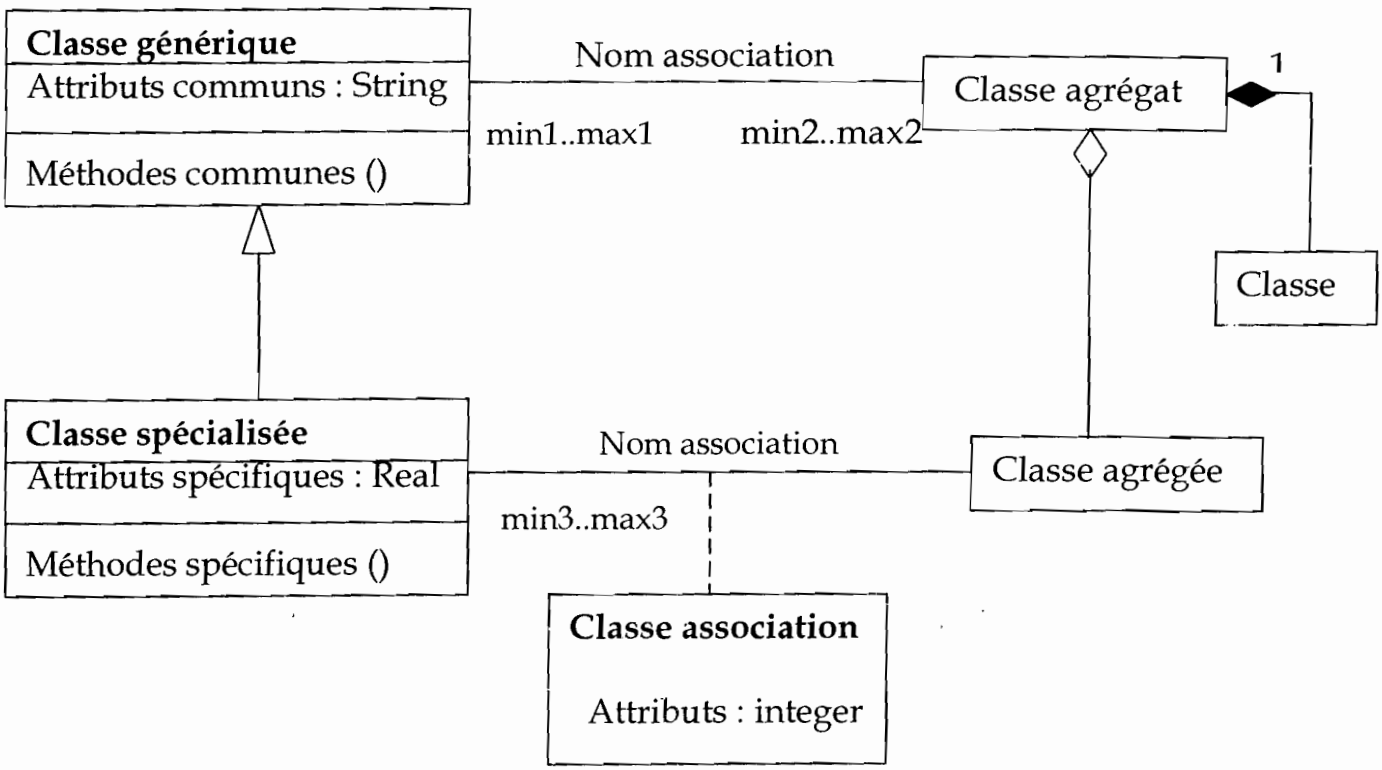


Figure A13 : Exemple de formalisme du diagramme de classes

formés d'éléments ayant entre eux une certaine logique. Leurs caractéristiques sont les suivantes :

- Ils regroupent des éléments de modélisation selon des critères purement logiques ;

- Ils permettent d'encapsuler des éléments de modélisation par l'intermédiaire d'interfaces ;
- Ils permettent de structurer un système en catégories ou sous-systèmes ;
- Ils servent de « briques » réutilisables dans la conception d'un logiciel.
- Chaque package doit avoir un nom différent de celui des autres packages et peut être composé d'autres éléments, y compris d'autres packages. Les éléments contenus sont en fait « possédés » (au sens UML du terme), ce qui signifie que la destruction d'un package implique la destruction de tous ses éléments.

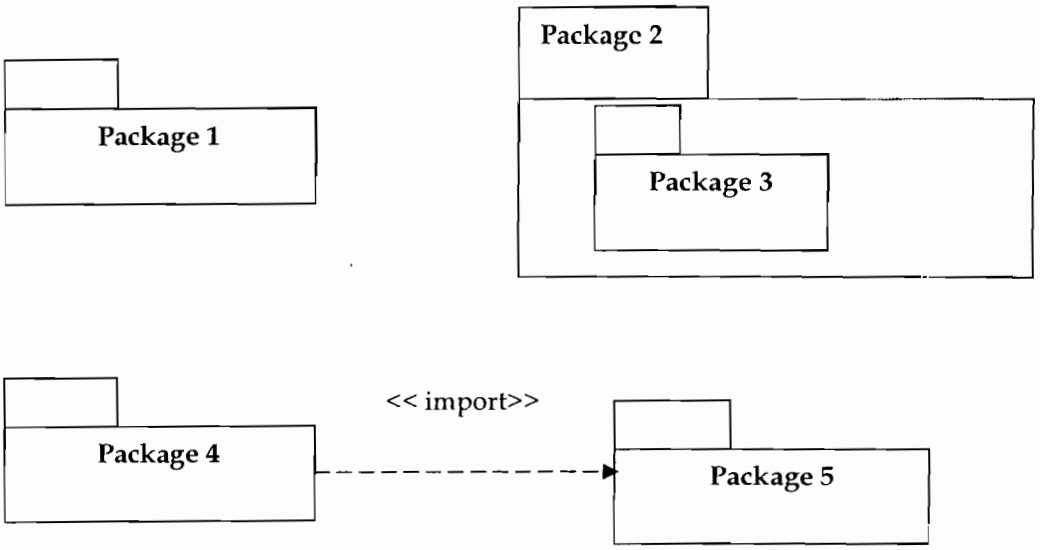


Figure A14 : représentation graphique d'un package

III. Concepts et formalismes du diagramme de séquence

✓ **Les Acteurs** (cf. paragraphe I. p92)

✓ **Les types de message :** un message est un moyen de communication entre objets. Ici, le message caractérise un événement c'est-à-dire une information envoyée à un objet et provoquant en réponse le déclenchement d'actions

associées à cet objet. Comme on peut le voir dans l'exemple ci-dessous, UML propose un certain nombre de stéréotypes graphiques pour décrire la nature du message :

- **Message simple :** message dont on ne spécifie aucune caractéristique d'envoi ou de réception particulière ;
- **Message minuté (timeout) :** bloque l'expéditeur pendant un temps donné (qui peut être spécifié dans une contrainte), en attendant la prise en compte du message par le récepteur. L'expéditeur est libéré si la prise en compte n'a pas eu lieu pendant le délai spécifié ;
- **Message synchrone :** bloque l'expéditeur jusqu'à la prise en compte du message par le destinataire. Le flot de contrôle passe de l'émetteur au récepteur (l'émetteur devient passif et le récepteur actif) à la prise en compte du message ;
- **Message asynchrone :** n'interrompt pas l'exécution de l'expéditeur. Le message envoyé peut être pris en compte par le récepteur à tout moment ou ignoré (jamais traité) ;
- **Message dérobant :** n'interrompt pas l'exécution de l'expéditeur et ne déclenche une opération chez le récepteur que s'il s'est préalablement mis en attente de ce message.

✓ **Formalisme du diagramme de séquence :**

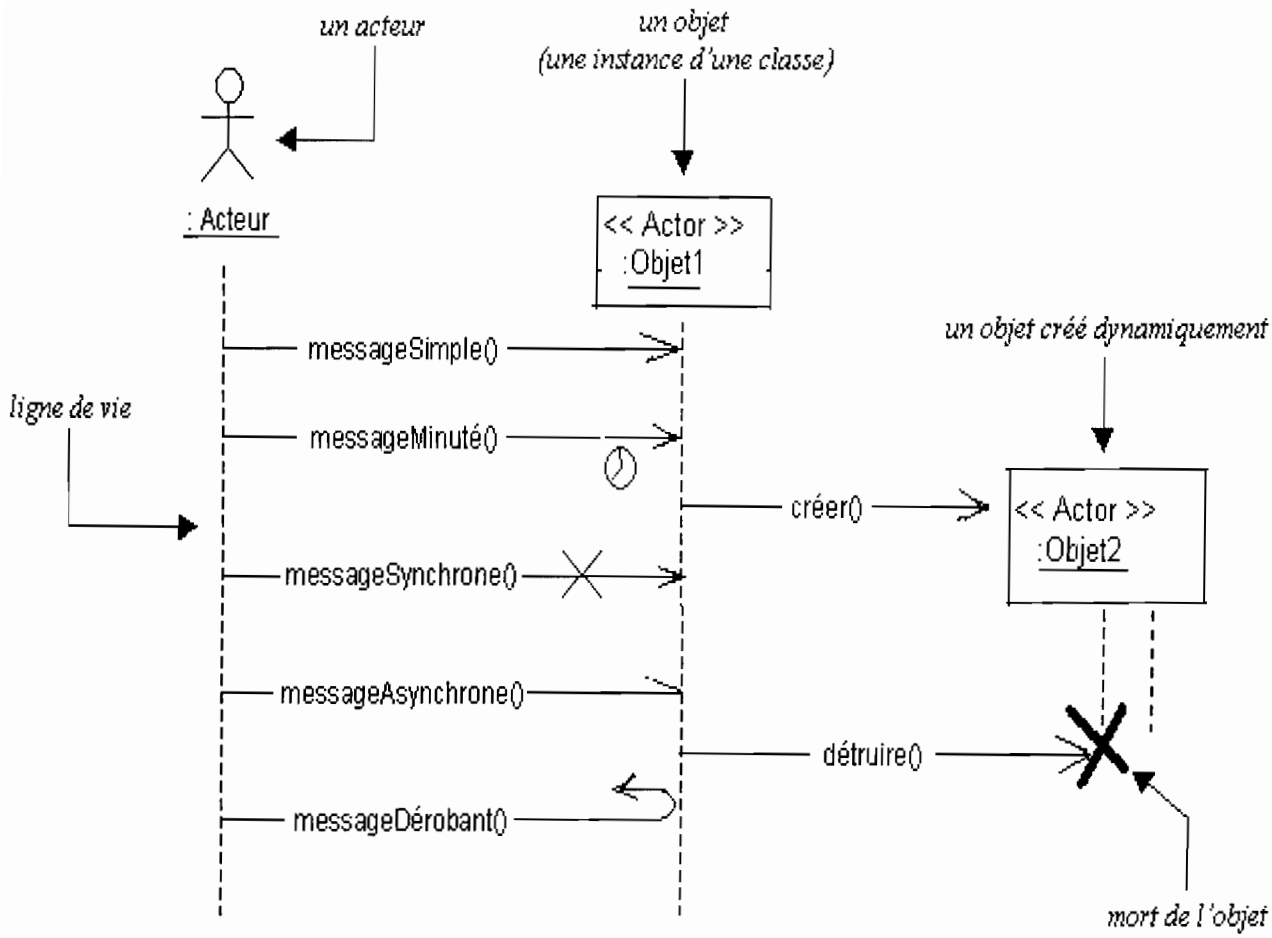


Figure A16 : représentation du formalisme du diagramme de séquence

✓ **Activation d'un objet :** sur un diagramme de séquence, il est aussi possible de représenter de manière explicite les différentes périodes d'activités d'un objet au moyen d'une bande rectangulaire superposée à la ligne de vie de l'objet. Pour représenter de manière graphique une exécution conditionnelle d'un message, on peut documenter un diagramme de séquence avec du pseudo code et représenter des bandes d'activations conditionnelles.

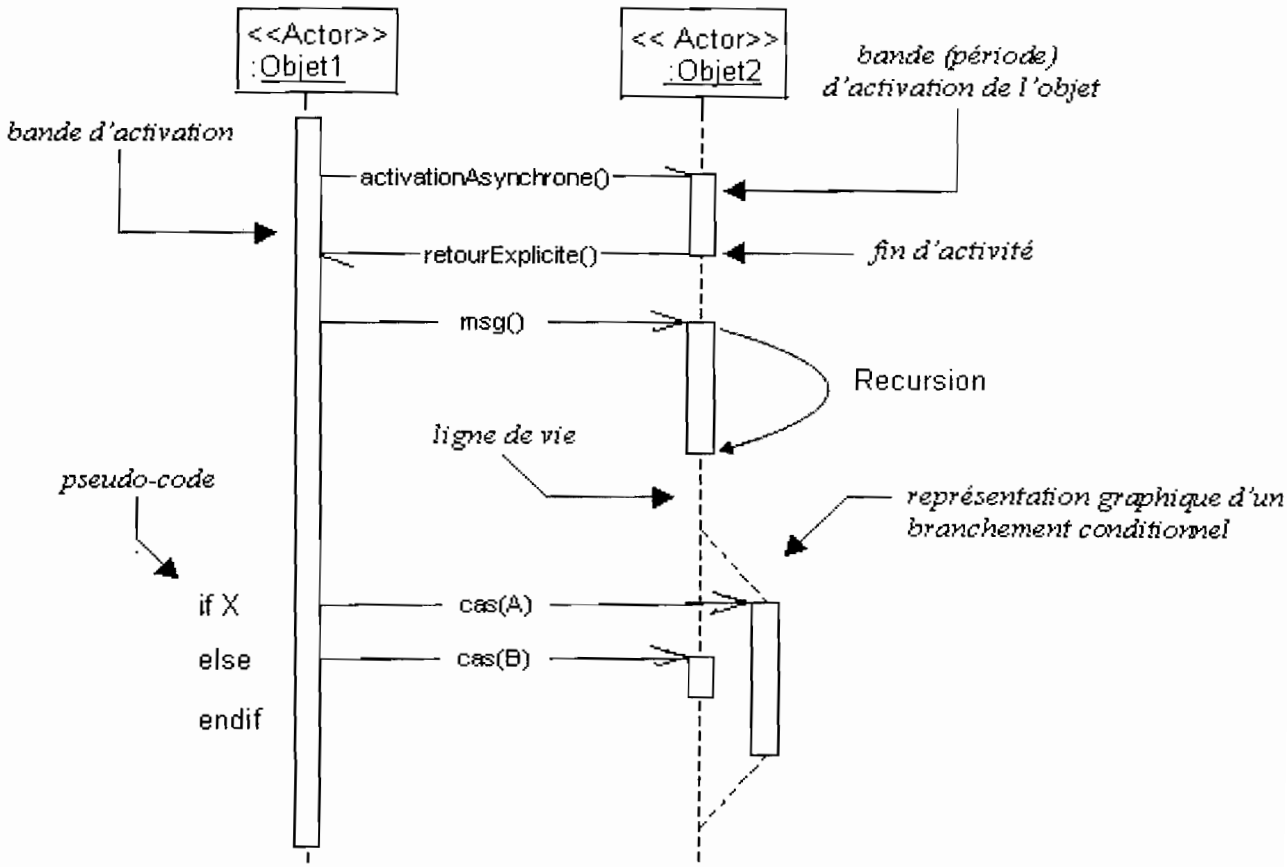


Figure A17 : représentation du formalisme du diagramme de séquence avec activation des objets

✓ **Commentaires :** un objet peut être actif plusieurs fois au cours de son existence (voir exemple ci-dessus).Le pseudo code peut aussi être utilisé pour indiquer des itérations (avec incrémentation d'un paramètre d'un message par exemple). Le retour des messages asynchrones devrait toujours être matérialisé, lorsqu'il existe.

IV. Concepts et formalismes du diagramme d'activité

UML permet de représenter graphiquement le comportement d'une méthode ou le déroulement d'un cas d'utilisation, à l'aide de diagrammes d'activités (une variante des diagrammes d'états-transitions[1]). Une activité représente une exécution d'un mécanisme, un déroulement d'étapes séquentielles. Le passage d'une activité vers une autre est matérialisé par une transition. Les transitions sont déclenchées par la fin d'une activité et provoquent le début immédiat d'une autre (elles sont automatiques). L'enchaînement des activités peut être soumis à des branchements ou à des synchronisations. En théorie, tous les mécanismes

[1] Les diagrammes d'états-transitions permettent de décrire les changements d'états d'un objet ou d'un composant, en réponse aux interactions avec d'autres objets/composants ou avec des acteurs.

dynamiques pourraient être décrits par un diagramme d'activités, mais seuls les mécanismes complexes ou intéressants méritent d'être représentés.

▪ **Transition :** une transition matérialise le passage d'une activité vers une autre. Les transitions sont déclenchées par la fin d'une activité et provoquent le début d'une autre (elles sont automatiques).

- un **événement**, c'est quelque chose qui a une signification pour le domaine et pouvant se produire suffisamment, fréquemment pour que l'on puisse définir a priori le comportement à adopter. L'événement peut être interne (il provient de l'intérieur du domaine), externe (il provient de l'extérieur du domaine) ou temporel (expiration d'un délai ou avènement d'une date).
- une **condition de garde** est une condition devant être vérifiée pour permettre la transition. Elle est optionnelle.
- une **action** est une opération atomique (non interruptible) déclenchée par une transition. Elle est optionnelle.

Notation : activité et transition

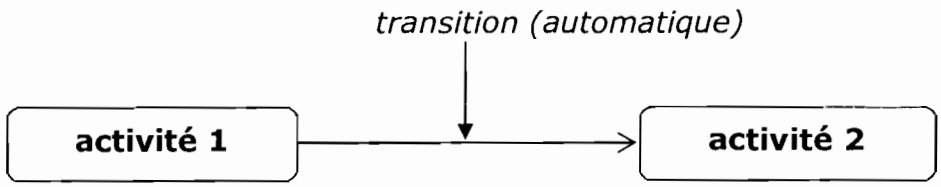


Figure A18 : représentation d'une transition entre deux activités

Pour représenter des **transitions conditionnelles**, utilisez des **gardes** (expressions booléennes exprimées en langage naturel), comme dans l'exemple suivant :

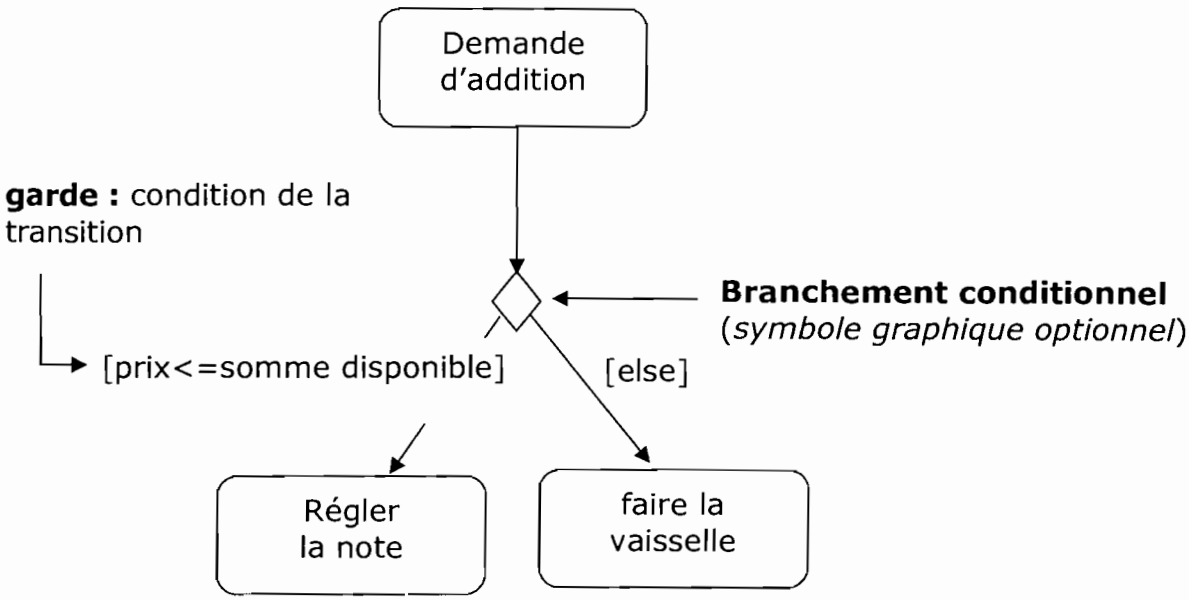


Figure A19 : représentation d'une transition

▪ **Synchronisation :** il est possible de synchroniser les transitions à l'aide des "**barres de synchronisation**" (comme dans les diagrammes d'états-transitions). Une barre de synchronisation permet d'ouvrir et de fermer des branches parallèles au sein d'un flot d'exécution :

- les transitions qui partent d'une barre de synchronisation ont lieu en même temps ;
- on ne franchit une barre de synchronisation qu'après réalisation de toutes les transitions qui s'y rattachent.

L'exemple suivant illustre l'utilisation des barres de synchronisation :

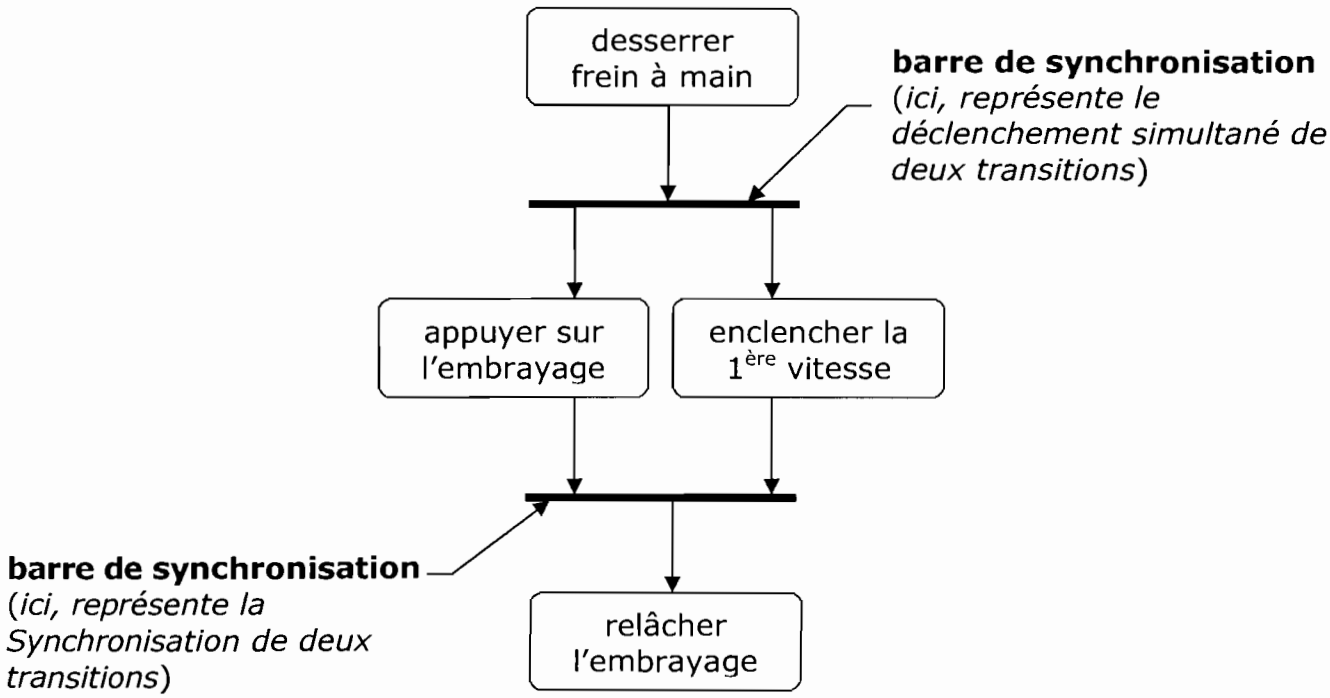


Figure A20 : représentation des barres de synchronisation

▪ **Couloir d'activité :** Afin d'organiser un diagramme d'activités selon les différents responsables des actions représentées, il est possible de définir des "*couloirs d'activités*". Il est même possible d'identifier les objets principaux, qui sont manipulés d'activité en activité et de visualiser leur changement d'état[1].

[1] **Un état** se caractérise par sa durée et sa stabilité, il représente une conjonction instantanée des **valeurs** des attributs d'un objet.

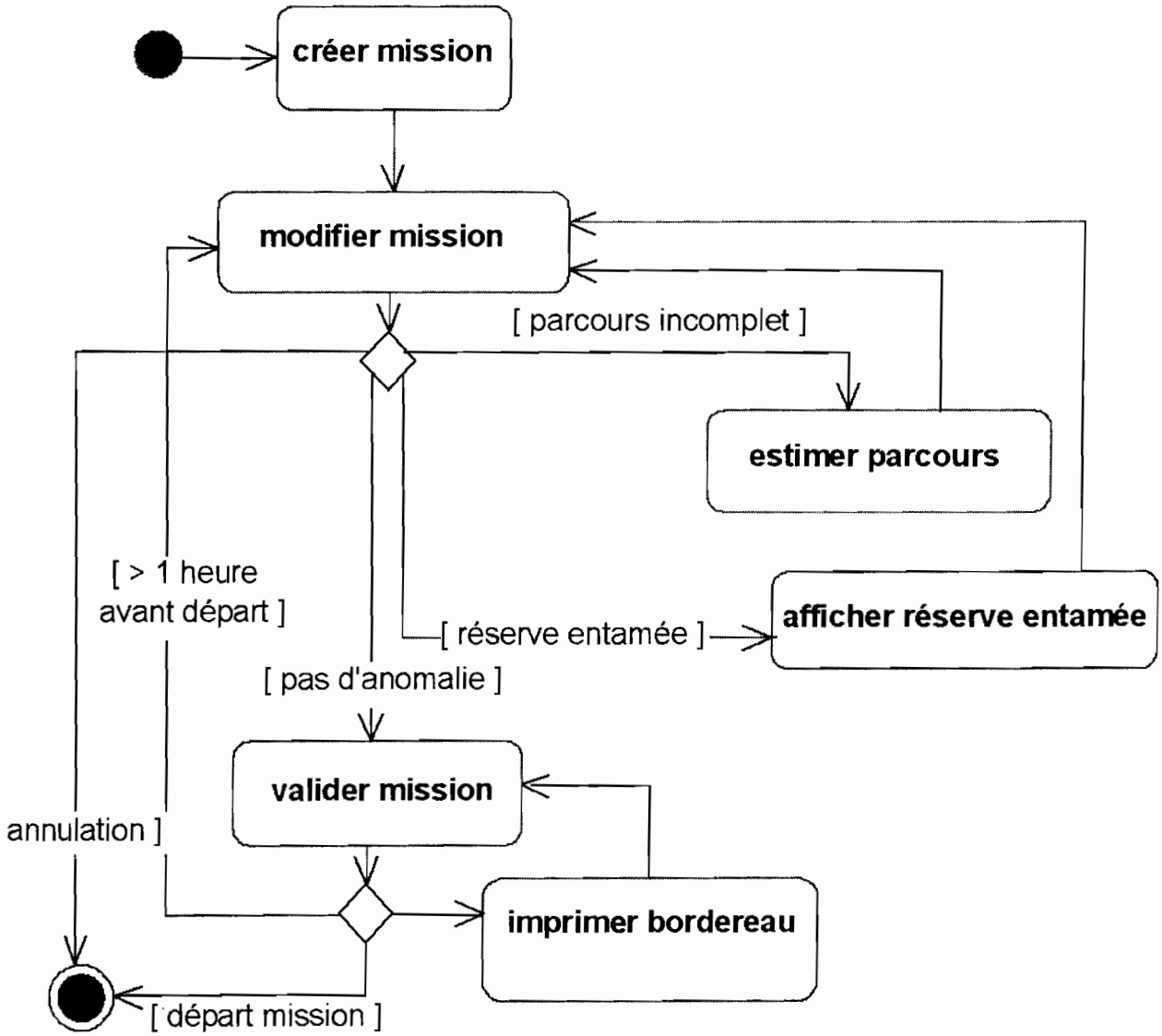


Figure A 21 : exemple de diagramme d'activité

ANNEXE 2 : Méthode de calcul du cout de développement

Le modèle le mieux documenté dont les paramètres sont adaptables à l'environnement est le **modèle COCOMO** (acronyme pour COnstructive COst MOdel) ou encore COCOMO simple, premier modèle datant de 1981, et développé par Dr. Barry Boehm pour estimer le coût, en nombre d'homme-mois, et le temps de développement d'un produit logiciel. A l'origine il a été construit sur une étude de 63 projets logiciels de 2000 à 100.000 lignes de code dans l'entreprise TRW Inc. (Société Américaine spécialisée dans l'Automobile et le Transport). Ce modèle existe en trois versions : simple, intermédiaire et détaillé.

Nous allons présenter les grandes lignes du modèle simple car c'est lui qui est utilisé pour notre cas précis, afin d'introduire la modélisation comme outil d'estimation des coûts et d'illustrer ses avantages en matière de gestion de projet.

Le modèle COCOMO simple est destiné à donner des estimations approximatives de coûts. Il s'appuie uniquement sur la taille estimée du logiciel et sur le type de logiciel à développer. Des familles différentes sont proposées pour trois types de projets :

- **projets de mode organique :** ces projets sont réalisés par une équipe de taille relativement petite travaillant dans un environnement familier et dans un domaine d'application connu de l'équipe. En conséquence, le surcoût dû à la communication est faible, les membres de l'équipe savent ce qu'ils ont à faire et le font rapidement.
- **projets de mode semi détaché :** ce mode représente un intermédiaire entre le mode organique et le mode embarqué décrit ci-dessous. Pour des projets de mode semi détaché, l'équipe de projet peut être composée de programmeurs de divers niveaux d'expérience. Les membres de l'équipe ont une expérience limitée dans ce type de système. Ils peuvent être totalement inexpérimentés en ce qui concerne quelques-uns des aspects du système à développer, mais pas tous.
- **projets de mode embarqué :** la caractéristique principale d'un projet de mode embarqué est que le système doit fonctionner sous des contraintes particulièrement fortes. Le système à développer est une partie d'un système complexe et fortement connecté de matériels et de logiciels, de

normes et de procédures opérationnelles. En conséquence, les modifications de spécifications destinées à contourner des problèmes logiciels sont en général impossibles et les coûts de validation extrêmement élevées. Du fait de la nature même de ces projets, il est inhabituel de disposer d'ingénieurs logiciels expérimentés dans le domaine d'application.

Les formules permettant de calculer le coût, ou plus exactement l'effort requis, exprimé en homme-mois, pour le développement du logiciel sont les suivantes :

Mode organique : effort = 2.4 (KLSL) $^{1.05}$

Mode semi détaché : effort = 3 (KLSL) $^{1.12}$

Mode embarqué : effort = 3.6 (KLSL) $^{1.20}$

Où KLSL est le nombre de Kilo-Lignes-Sources Livrées.

Le modèle COCOMO simple permet également d'estimer le temps de développement (TDEV) nécessaire au projet. Le temps de développement est le temps requis pour terminer le projet, en supposant que les ressources de personnel requises soient disponibles. Les équations pour les différents modes de projets sont les suivantes :

Mode organique *TDEV* = 2.5 (*effort*) $^{0.38}$

Mode semi détaché *TDEV* = 2.5 (effort) $^{0.35}$

Mode embarqué *TDEV* = 2.5 (*effort*) $^{0.32}$

Le nombre de personnes requis pour réaliser le projet dans cet intervalle de temps est donc : *N* = *effort*/*TDEV*.

ANNEXE 3 : ORGANIGRAMME SETO

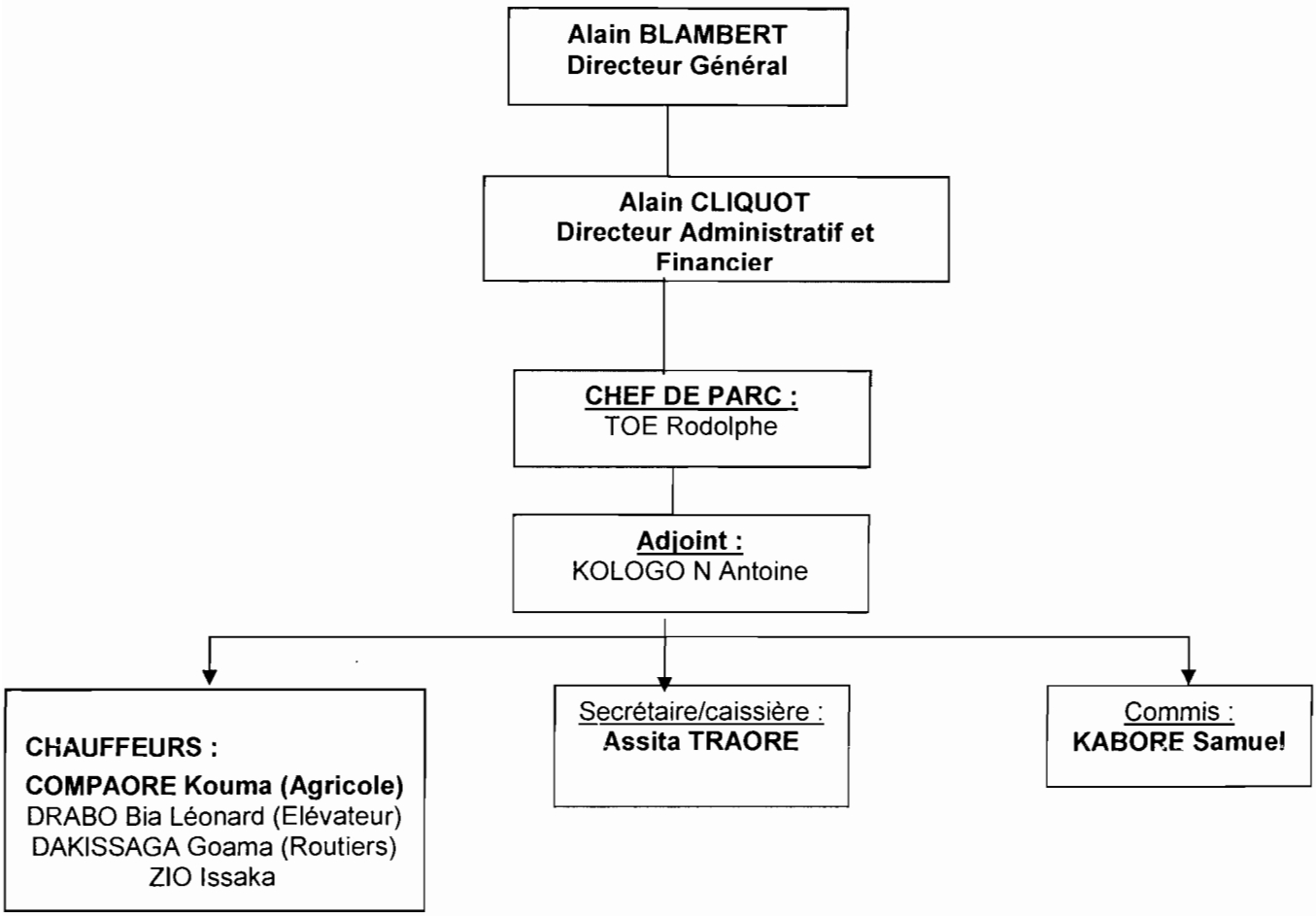


Figure A22 : organigramme SETO

ANNEXE 4 : ORGANIGRAMME SDV-BURKINA

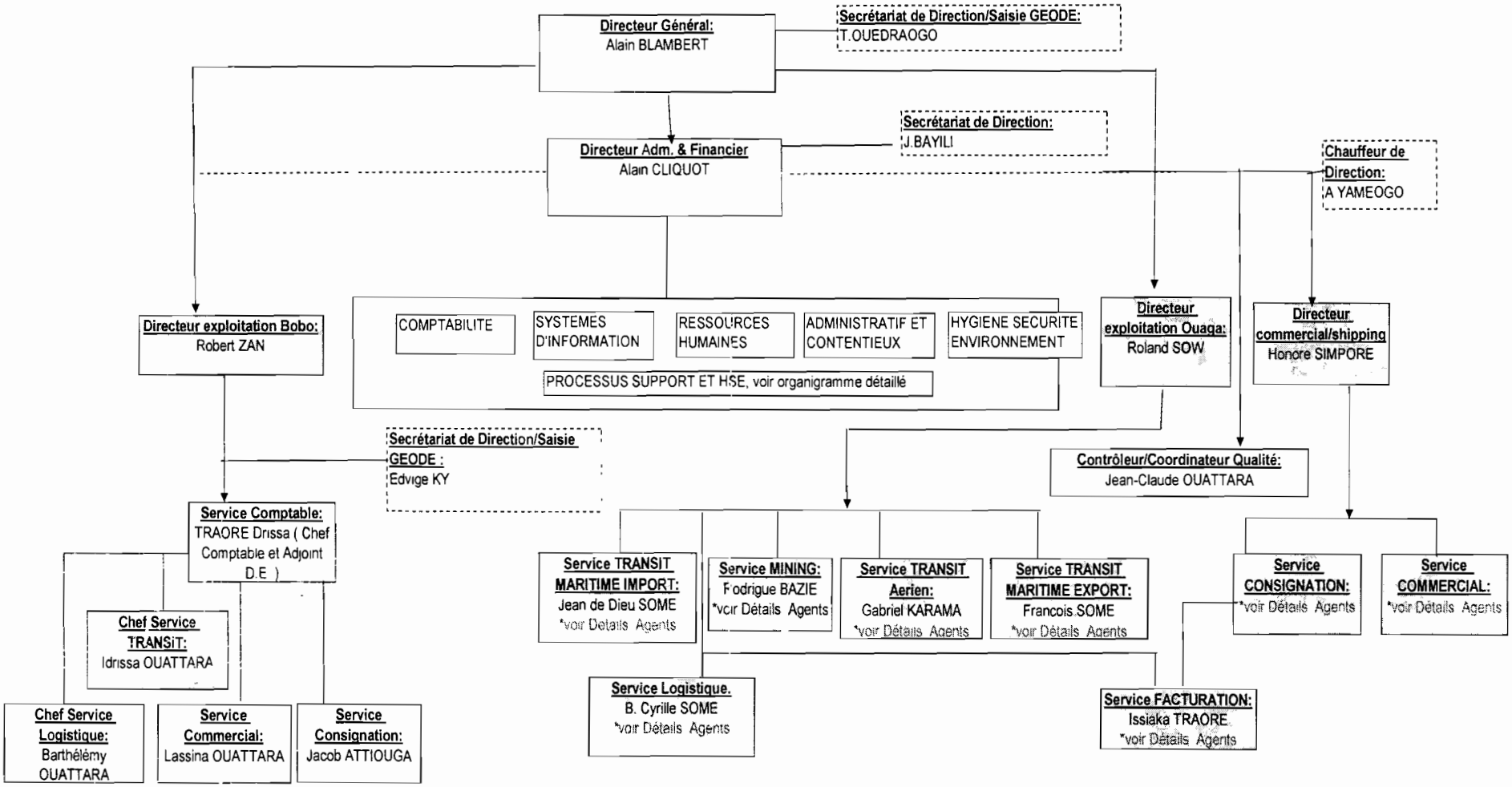


Figure A23 : Organigramme SDV-Burkina

ANNEXE 5 : ORGANIGRAMME SNTB

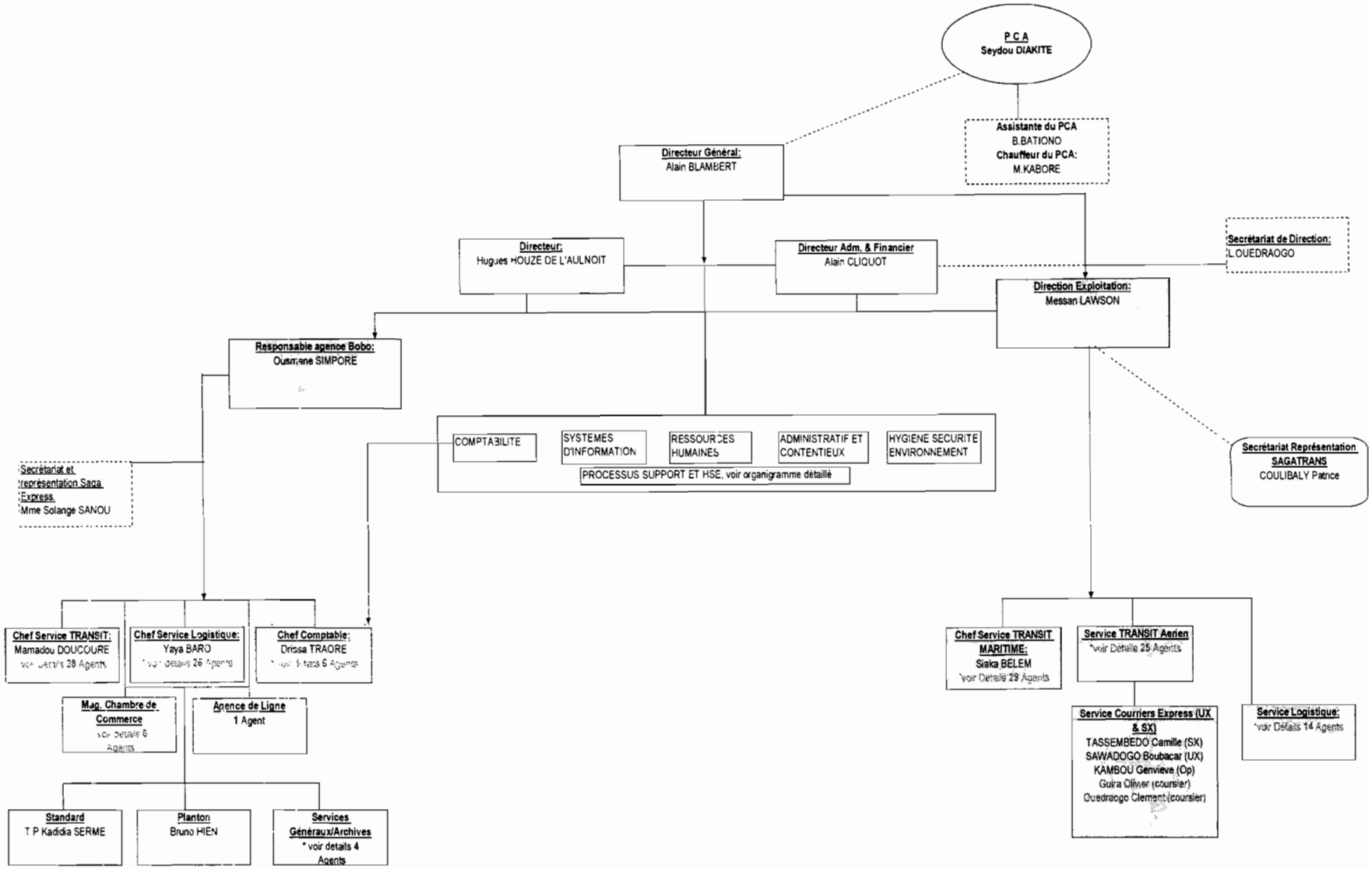


Figure A24 : Organigramme SNTB

ANNEXE 6 : FICHE D'ENREGISTREMENT DES INCIDENTS

FICHE D'ENREGISTREMENT DES INCIDENTS INFORMATIQUES								
NO ORD	DATE jj/mm/aaaa	HEURE hh:mn	SOCIETE	SITE	ID.AUTEUR APPEL	TYPE CONTACT	DESCRIPTION	INITIAL EQUIPE

Figure A25 : exemplaire de la fiche d'enregistrement des incidents

LISTE DES FIGURES

LISTE DES TABLEAUX