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

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

Mémoire de fin d'études

pour l'obtention du diplôme de Master en Informatique

Option: *Modèle Intelligent et Décision(M.I.D)*

Thème

Approche Multi-Agent pour la reconnaissance de Diabète

Réalisé par :

- **Melle BELLIFA Imane**

Présenté le 03 Juillet 2011 devant la commission composé de MM.

Président :	-	*LAHASAINI.M*		
Encadreur :	-	*CHIKH.A*	/ -	*BENABED.A*
Examinateurs :	-	*BENAZZOUZ.M*	/ -	*ILLES.N*
	-	*BENZIANE.Y*	/ -	*SMAHI.M.I*
	-	*HADJILA.F*		

Année universitaire : 2010-2011

Remerciement

Avant tout, je rends grâce à DIEU tout puissant de m’avoir accordé la volonté et le courage pour réaliser ce mémoire

Au terme de ce travail je tiens tout d’abord à exprimer ma profonde gratitude à mon encadreur Mr CHIKH Amine.

Ma gratitude va également à Mr BENABED Amine qui a toujours su porter un regard critique sur mon travail et de m’avoir guidé tout au long de ce travail.

Mes remerciements s’adressent à tous les membres du jury

pour l’honneur qu’ils m'ont fait en acceptant

de juger ce travail.

Dédicace

À mes très chers parents
Que Dieu les garde
À toute ma famille et mes amis
À tous ceux qui sont proches de mon cœur
et dont je n’ai pas cité les noms
À toutes personnes malades
À mon pays

Je dédie ce modeste travail.

Table des matières

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ABREVIATION

ACC : Agent Communication Channel

ACL : Agent Communication Language

ADELFE : Atelier pour le DEveloppement de Logiciels à Fonctionnalité Emergente

BDI : Believe Desire and Intention

CDSMA : Classification de diabéte à base des Système Multi-Agent

DF : Directory Facilitator.

FIPA : Foundation for Intelligent Physical Agents

JADE : Java Agent Development framework

IA : Intelligence Artificielle

IAD : Intelligence Artificielle Distribuée

IAP : Intelligence Artificielle Parallèle

IHM : Interface Homme/Machine

KIF : Knowledge Interchange Format

KQML : Knowledge Query and Manipulation Language

RMA : Remote Monitoring Agent

RMI : Remonte Management Interface

SMA : Système Multi-Agent

Weka :Waikato Environment for Knowledge Analysis)

D'après un article de presse d'El Watan, il est mentionné que Le nombre des diabétiques en Algérie est passé d'un million de personnes en 1993, à plus de 2 500 000 personnes en 2007, soit 10% de la population. Ces statistiques établies par l'OMS sont confirmées par le président de la Fédération algérienne des associations des malades diabétiques(FAAD).

Le diabète est considéré comme une maladie chronique, invalidante et coûteuse, qui s'accompagne souvent de graves complications.

Notre objectif dans ce travail de mémoire de master est de proposer un modèle de classification des personnes diabétiques basé sur une approche distribuée dite SMA (système multi-agent).

Le but est de renforcer le diagnostic dans ce domaine.

Nous proposons une architecture basée sur les agents, cette architecture est entrainée et testée sur une base de données de diabète. Notre mémoire est composée de trois chapitres :

Chapitre I : présente un état de l'art de diabète

Chapitre II : présente le contexte d'étude, à savoir, les systèmes multi-agents dans lequel nous introduisons les aspects théoriques liés à ce concept.

Cela comprend une courte introduction à l'intelligence artificielle distribuée, la définition d'agent et de système multi-agent, un aperçu des théories et paradigmes utilisés dans le domaine, et finalement une présentation des méthodes de construction et d'analyse de tels systèmes.

Chapitre III : Dans ce dernier chapitre nous décrivons le développement de notre système, et cela en présentant en premier lieu les choix effectués pour sa réalisation, et nous abordons en deuxième lieu l'architecture du système ainsi que son fonctionnement En dernier lieu, nous achèverons ce chapitre par la présentation de multiples interfaces de notre système (produit final).

Finalement, nous terminons par une conclusion générale qui récapitule les principales observations concernant l'évolution du travail et nous indiquons également comment les travaux réalisés tout au long de ce mémoire pourraient être améliorés, en donnant la perspective de nouvelles approches de recherche.

Chapitre I :

Etat de l’art de Diabète

- **Introduction**
- **Définition de Diabète**
- **Les types de Diabète**
- **Les symptômes de Diabète**
- **Les complications liées aux Diabète**
- **Mesure de Diabète**
- **Conclusion.**

Introduction :

Le diabète est un problème croissant de santé dans le monde d'aujourd'hui. Saviez-vous que environ 20,8 millions d'enfants et d'adultes sont diagnostiqués avec le diabète dans les seuls Etats-Unis? Ces chiffres augmentent chaque jour. La cause exacte du diabète n'est pas clair, cependant, le manque d'exercice et l'obésité sont les principaux facteurs qui causent le développement du diabète **[Web 1]** ;

Le diabète ne date pas d'hier. En fait, c'est une maladie aussi vieille que les êtres humains. D'ailleurs, le premier document qui en parle date de 4000 ans il a pu être observé et décrit par les plus grands médecins dont Aristote, Galien, Avicenne et Paracelse. Pendant des siècles, les personnes atteintes de diabète étaient condamnées à mourir parce qu'il n'y avait aucun traitement pour les soigner.

Encore aujourd'hui, on ne peut pas guérir le diabète. C'est pourquoi on dit que c'est une maladie chronique. Toutefois, on peut le contrôler et améliorer la qualité de vie des personnes qui en sont atteintes.

En fait, la meilleure façon de bien vivre avec cette maladie, c'est de la comprendre.

I. Définition de Diabète :

Afin de bien comprendre ce qu'est le diabète, il faut savoir comment notre corps prend de l'énergie dans les aliments.

Les êtres vivants ont besoin d'énergie pour vivre. Il suffit de manger pour retrouver de l'énergie. Tous les aliments (le pain, les pommes de terre, les pâtes, etc.) contiennent des sucres, qu'on appelle des glucides. Par contre, pour nourrir les cellules de notre corps, il faut transformer ces glucides en une autre sorte de sucre : le glucose.

Cette transformation se produit pendant la digestion. Ainsi, la principale source d'énergie qui nous permet de fonctionner, c'est le glucose.

Le glucose est produit par le foie. Il passe dans le sang et le **taux de glucose** (qu'on appelle aussi **taux de sucre** ou **glycémie**) augmente.

À ce moment, le corps envoie un signal à un organe près de l'estomac qui s'appelle le pancréas. Le pancréas produit alors de l'insuline.

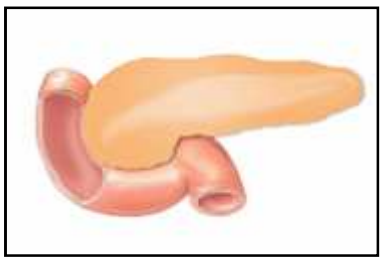

Figure I.1 : Le Pancréas

L'insuline a beaucoup d'importance. Elle fonctionne un peu comme une clé. C'est elle qui permet au glucose d'entrer dans les cellules pour les nourrir.

Lorsque le glucose entre dans les cellules, le corps reçoit l'énergie dont il a besoin.

Le diabète se développe lorsque le corps manque d'insuline ou ne réussit pas à utiliser celle qu'il produit. Alors, le glucose ne peut plus entrer dans les cellules et s'accumule dans le sang. C'est ce qu'on appelle faire de l'hyperglycémie **[Web 2].**

En termes plus « médicaux et officiels», le diabète est défini par une glycémie supérieure à 1,26 g/l (7 mmol/l) après un jeûne de 8 heures et vérifiée à deux reprises. Il est aussi défini par la présence de symptômes de diabète (polyurie, polydipsie, amaigrissement) associée à une glycémie (sur plasma veineux) supérieure ou égale à 2 g/l (11,1 mmol/L) ainsi que par une glycémie supérieure ou égale à 2 g/l (11,1 mmol/l) 2 heures après une charge orale de 75 g de glucose (critères proposés par l'Organisation Mondiale de la Santé). **[Web 1].**

II. Les types de diabète : Il y a deux types principaux de diabète : diabète de type 1 et diabète de type 2.Le diabète de type 1 apparaît souvent pendant l'enfance ou l'adolescence. On n'en connaît pas la cause. Par contre, on sait que ce sont les défenses naturelles du corps qui détruisent les cellules du pancréas. Ainsi, le corps ne peut plus produire d'insuline ou en produit très peu et le glucose reste dans le sang **[Web 2]** et des injections quotidiennes d'insuline sont nécessaires pour maintenir la vie. Ce type de diabète est généralement présent dès l'enfance et est également appelé insulino-dépendant **[Web 1].** Le diabète de type 1 touche environ une personne diabétique sur dix.

Le diabète de type 2 ou diabète non insulino dépendant (DNID) : C'est le type le plus commun de diabète pour sa part, touche plutôt les adultes de plus de 40 ans. Même si on n'en connaît pas très bien les causes, on peut identifier certains facteurs de risque : obésité, manque d'exercice physique, alimentation trop riche en gras. Certaines personnes atteintes de cette

maladie ne produisent plus assez d'insuline. D'autres en produisent, mais elle ne fait plus son travail. Le diabète de type 2 est le plus courant. La plupart des diabétiques de type 2 sont gérés par l'alimentation et les médicaments oraux. Il touche 9 personnes diabétiques sur 10. **[Web 2]**

Le diabète gestationnel : Ce type se développe pendant la grossesse et disparaît généralement après la naissance du bébé. Il touche certaines femmes qui n'avaient pas la maladie avant d'être enceintes. Il apparaît en général autour de la 24e semaine de grossesse. Les facteurs de risque sont connus : avoir plus de 35 ans, avoir un surplus de poids, avoir déjà eu un bébé de plus de 9 livres et avoir une histoire de diabète dans la famille. **[Web 2]** Si une femme a déjà connu un diabète de grossesse, elle a plus de risques d'en développer un deuxième. Il est donc important que la mère soit suivie durant toute la période de sa grossesse.

Le diabète ne touche pas seulement la mère, mais aussi le bébé. Par contre, pour le bébé, le problème se règle dans les heures qui suivent la naissance ; en ce qui concerne la mère, 9 fois sur 10, le diabète disparaît après l'accouchement.

La différence entre le type1 et le type2 de diabète :

Il y a quand même une différence entre les deux types de diabète :

Dans le cas du diabète de type 1, les gens ont toujours faim, mais ils perdent quand même du poids. C'est pourquoi on appelle parfois ce type de maladie : **diabète maigre**.

Dans le cas du diabète de type 2, les personnes ont faim et prennent du poids. C'est pourquoi on l'appelle aussi **diabète gras**.

Il faut toutefois ajouter que le diabète est une maladie sournoise : quand les symptômes apparaissent, c'est souvent parce que la maladie est très avancée.

III. Les symptômes du diabète en générale :

- Polyurie:besoin fréquent d'uriner.
- Soif
- Perte de poids
- Fatigue – souvent présents dans le diabète. Mais d'autres troubles peuvent également entraîner de la fatigue. Donc, ce n'est pas suffisant.
- Démangeaisons des organes génitaux et de la peau

- Troubles visuels (tels que la vision floue)
- Affections de la peau (par exemple, les furoncles)
- Douleurs et / ou engourdissement des extrémités **[Web 1]**

IV. Les complications liées au diabète : Le diabète s'attaque aux veines et aux artères, surtout aux plus petites. Le sang y circule plus difficilement et ne peut plus bien nourrir les cellules qui les entourent. Les muscles et les nerfs peuvent être abîmés. De plus, comme le sang circule moins bien, le corps combat moins les infections. C'est pour ces raisons que les complications liées au diabète sont nombreuses et souvent graves.

Voici les principales :

Problèmes visuels

Dans les yeux, il y a beaucoup de petits vaisseaux sanguins. Si le sang circule moins bien, certaines parties de l'œil peuvent se détériorer lentement. Parfois, il n'y a aucun symptôme avant des années et d'autres fois, la vue peut baisser ou s'embrouiller. Si on laisse aller les choses, on peut même devenir aveugle.

Pied diabétique : Le diabète peut endommager les nerfs dans les pieds. Dans certains cas, ils sont tellement abîmés qu'on ne sent plus la douleur causée par les blessures. Il arrive qu'elles s'infectent avant qu'on s'en rende compte. De plus, comme la circulation du sang est moins bonne, le corps n'arrive pas à combattre l'infection. Si le problème s'aggrave, il peut mener à l'amputation.

Crises cardiaques et ACV (accidents cérébro-vasculaires) : Même si le diabète attaque surtout les petits vaisseaux sanguins, il peut également détériorer les plus gros. Les personnes diabétiques sont plus sensibles aux problèmes qui sont associés à la circulation du sang : haute tension, athérosclérose, phlébite, thrombose, etc. Ainsi, ils ont plus de risques de faire des crises cardiaques et des ACV. **[Web 1]**

Insuffisance rénale : Les reins ont une fonction très importante dans le corps humain. Ils servent de filtres pour éliminer les déchets de l'organisme. Comme le diabète endommage les petits vaisseaux des reins, ceux-ci ne peuvent plus jouer leur rôle. Alors, tout se dérègle : au lieu d'être évacués dans l'urine, les déchets passent dans le sang. De plus, certaines substances nutritives qui devraient passer dans le sang (comme les protéines)

sont au contraire évacuées par l'urine. Avec le temps, ces problèmes peuvent causer de l'insuffisance rénale. Les personnes atteintes de cette maladie doivent utiliser un appareil pour filtrer le sang durant toute leur vie.

Infections fréquentes : Le fait d'avoir un taux élevé de glucose dans le sang empêche le corps de bien combattre les infections. Ainsi, les personnes diabétiques ont plus de risques de développer des infections de la peau, de la vessie ou des gencives **[Web 2].**

V. Mesure du diabète : (Glycémie à jeun) : Sa mesure au laboratoire est le moyen le plus simple et le plus connu pour «mesurer le diabète». Sa valeur normale à jeun, ou dans la journée avant les repas, est comprise **entre 0,6 et 1,0 g/l**. On peut aussi la mesurer 1 h 30 après le début du repas (glycémie postprandiale) et sa valeur normale est **inférieure à 1,5 g/l**.

Mais on peut également la mesurer très facilement soi-même, à domicile ou sur son lieu de travail, avec un petit appareil appelé «lecteur de glycémie» qui analyse une goutte de sang, qui est prélevée au bout du doigt avec une sorte de stylo appelé «autopiqueur». On parle de «glycémie capillaire» car le sang provient des tout petits vaisseaux appelés capillaires.

Figure I.2 : Un patient mesure le taux de glycémie **[Web 2]**

Conclusion :

Nous avons présenté dans ce chapitre les éléments nécessaires pour définir et découvrir le diabète et nous pouvons dire aussi que la recherche scientifique dans le domaine médical dans de nombreux pays, a permis au corps médical de mieux connaître la maladie et de diagnostiquer les symptômes . Grâce à un bon contrôle du taux de sucre sanguin on peut mieux prévenir les complications, en particulier vasculaires, redoutables à long terme, portant sur les yeux, les reins, le cœur, le système nerveux.

Les meilleurs résultats sont obtenus lorsque le patient lui-même devient son propre thérapeute! Pour cela, il doit comprendre sa maladie, l'accepter et vivre avec elle en faisant ce qu'il faut pour la maîtriser et la traiter.

Chapitre II :

Agent Et Système Multi-agent

- **Introduction**
- **Historique**
- **Concept d’agent**
- **Le système multi agent**
- **Conclusion**

Introduction :

Le thème des systèmes multi-agents, s'il n'est pas récent, est actuellement un champ de recherche très actif. Cette discipline est à la connexion de plusieurs domaines en particulier de l'intelligence artificielle, des systèmes informatiques distribués et du génie logiciel. C'est une discipline qui s'intéresse aux comportements collectifs produits par les interactions de plusieurs entités autonomes et flexibles appelées agents, que ces interactions tournent autour de la coopération, de la concurrence ou de la coexistence entre ces agents. Ce chapitre introduit, tout d'abord, les notions d'agents et de systèmes multi-agents, et détaille par la suite Les travaux sur les SMA qui sont poursuivis activement depuis le début des années 90. Des nouvelles tendances se sont concrétisées tel l'accent mis sur l'apprentissage collaboratif, les interfaces adaptatives et les systèmes multi-agents. Ces recherches sur les agents sont guidées par la nécessité d'avoir des environnements interactifs qui tracent à la fois le comportement du système informatique et celui des usagers (tuteur ou apprenant). En effet, lorsqu'il s'agit de concevoir des systèmes informatiques distribués qui manipulent des connaissances hétérogènes, la technologie agent se révèle bien adaptée. Les systèmes multi-agents permettent non seulement le partage ou la distribution de connaissance, mais aussi, de faire coopérer un ensemble d'agents et de coordonner leurs actions pour l'accomplissement d'un but commun.

Historique :

L'évolution des domaines d'application de l'Intelligence Artificielle (IA) pour recouvrir des domaines complexes et hétérogènes tels que l'aide à la décision, la reconnaissance et la compréhension des formes, la conduite des processus industriels, etc., a montré les limites de l'approche classique de l'IA qui s'appuie sur une centralisation de l'expertise au sein d'un système unique.

Les travaux menés au début des années 70 sur la distribution ont contribue à la naissance d'une nouvelle discipline : l'Intelligence Artificielle Distribuée (IAD)

[FER, 88]. L'IAD a pour but de remédier aux insuffisances de l'approche classique de l'IA en proposant la distribution de l'expertise sur un groupe de sous-système devant être capables de travailler et d'agir dans un environnement commun et résoudre les conflits éventuels. D'ou la naissance de notions nouvelles en IA, telles que la coopération, la coordination d'actions, la négociation et l'émergence.

L'IAD peut alors être définie comme étant la branche de l'IA qui s'intéresse à la modélisation de comportement intelligent par la coopération entre un ensemble d'entité **[HUH, 87]**.

Nous présentons trois axes fondamentaux dans la recherche en IAD :

- Les systèmes multi-agents (SMA) **[DUR, 90]** : Il s'agit de faire coopérer un ensemble d'agents dotes d'un comportement intelligent et de coordonner leurs buts et leurs plans d'actions pour la résolution d'un problème.

- La résolution distribuée des problèmes (RPD): Elle s'intéresse a la manière de diviser un problème particulier sur un ensemble d'entités distribuées et coopérantes. Elle s'intéresse aussi à la manière de partager la connaissance du problème et d'en obtenir la solution.

- L'Intelligence artificielle parallèle (IAP) **[DAV, 80] [FEH, 83]** : Elle concerne le développement de langages et d'algorithmes parallèles pour l'IAD. L'IAP vise l'amélioration des performances des systèmes d'intelligence artificielle sans, toutefois, s'intéresser à la nature du raisonnement ou au comportement intelligent d'un groupe d'agents.

I. Concept d'agent :

La notion d'agent n'est pas simple à définir. Il existe en effet plusieurs définitions ou significations données à cette notion. C'est la raison pour laquelle plusieurs auteurs essayent d'en donner une définition avant de se pencher sur l'utilisation de ce paradigme dans tel ou tel contexte.

I.1 Définition :

Si on se réfère à la définition du dictionnaire : du latin : « agens » : celui qui agit. Le terme «agir » est défini par le petit *Larousse* : Agir (latin agere, faire) faire quelque chose, s'occuper, produire un effet.
Définition : Un agent est une personne qui agit pour le compte d'autrui.
Appliquée au domaine informatique, la définition du *Media Dico* d'un agent fournit plus de précisions :
Définition : Un agent est un composant logiciel effectuant une tâche précise de manière autonome.
Le terme « agent » est un terme qui reste relativement vague, il n'y a pas de consensus sur la définition. Voici l'une des premières définitions de l'agent due à Ferber **[FER, 95]:**

« On appelle **agent**, une entité réelle ou abstraite qui est capable d'agir sur elle-même et sur son environnement, qui dispose d'une représentation partielle de cet environnement, qui, dans un univers multi-agent, peut communiquer avec d'autres agents et dont le comportement est la conséquence de ses observation, de sa connaissance et des interactions avec les autres agents».

Une autre définition, proposée par N. Jennings et M.Wooldridge **[JW, 98]** :

« Un **agent** est un système informatique, situé dans son environnement, et qui agit de façon autonome et flexible pour atteindre les objectifs pour lesquels il a été conçu».

Pattie Maes **[MAE, 1995]** du groupe de recherche SMA au *MIT* propose une bonne définition générique du terme agent: « *An agent is a computational system that inhabits a complex, dynamic environment. The agent can sense, and act on, its environment, and has a set of goals or motivations that it tries to achieve through these actions.* »

[JEN et al. 98] définissent un agent comme un système informatique situé dans un certain environnement, capable d'exercer de façon autonome des actions sur cet environnement en vue d'atteindre ses objectifs. D'après cette définition un agent intelligent se caractérise par les propriétés suivantes :

I.2 Principales caractéristiques d'un agent : Un agent est une entité qui doit posséder les caractéristiques suivantes **:**

Autonomie

- Un agent a un certain degré d'autonomie.
- Il possède certains états (non-accessibles aux autres agents et composants du système).
- Il peut prendre certaines décisions par rapport à ses états (sans intervention externe directe).

Situé

- Un agent est situé dans son environnement (physique ou virtuel).
- Il a une représentation de son environnement.

Réactif

- Un agent peut percevoir son environnement via des senseurs.
- Il peut agir sur son environnement via des effecteurs.

Social

- Un agent est capable d'interagir et de communiquer avec les autres agents (par des langages de communication).

- Il est capable de coopérer pour résoudre des problèmes ou effectuer des tâches.

Proactif

- Un agent est capable de « prendre de l'initiative » pour atteindre son but ou effectuer des tâches (et d'adopter les comportements appropriés).

Capacité à agir : un agent est mû par un certain nombre d'objectifs qui guident ses actions, il ne répond pas simplement aux sollicitations de son environnement;

Actif

- Un agent est toujours actif. Il s'exécute donc nécessairement dans un *thread* ou un *process* indépendants.

Apprentissage

- Un agent est capable d'apprendre et d'évoluer en fonction de cet apprentissage.
- Il est capable de changer de comportement (en fonction des expériences passées).

Il sort de ses définitions trois notions fondamentales permettant de caractériser un agent : percevoir, décider et agir :

- Percevoir : Le fait de percevoir implique le fait que l'agent soit plongé dans un univers et qu'il communique dans cet univers. Il peut ou non percevoir les actions des autres agents, percevoir des mouvements d'objets ou tout type de transformation de son environnement.

- Décider : cela induit le fait qu'un agent ait des connaissances qui lui sont propres et un comportement autonome lui permettant d'exploiter ces connaissances et de « raisonner »… Il décide en fonction de ses perceptions.

- Agir : l'agent peut agir sur son environnement. Il peut manipuler des objets, les déformer, communiquer avec d'autres agents, engager une coopération ou un conflit. Toute action en tant que telle modifie d'une certaine façon une partie de l'environnement.

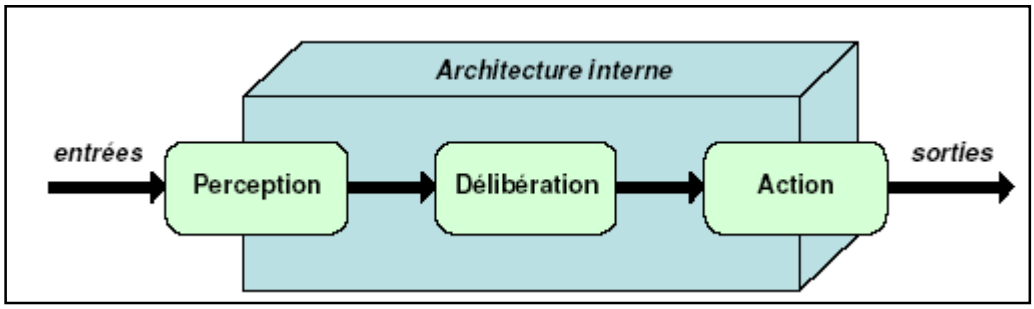

Figure II.1 : Architecture d'agent

I.3 Environnement

Un agent ne peut exister sans environnement. L'environnement est une structure dans laquelle l'agent évolue. Un agent va agir sur son environnement et l'environnement va agir sur l'agent.

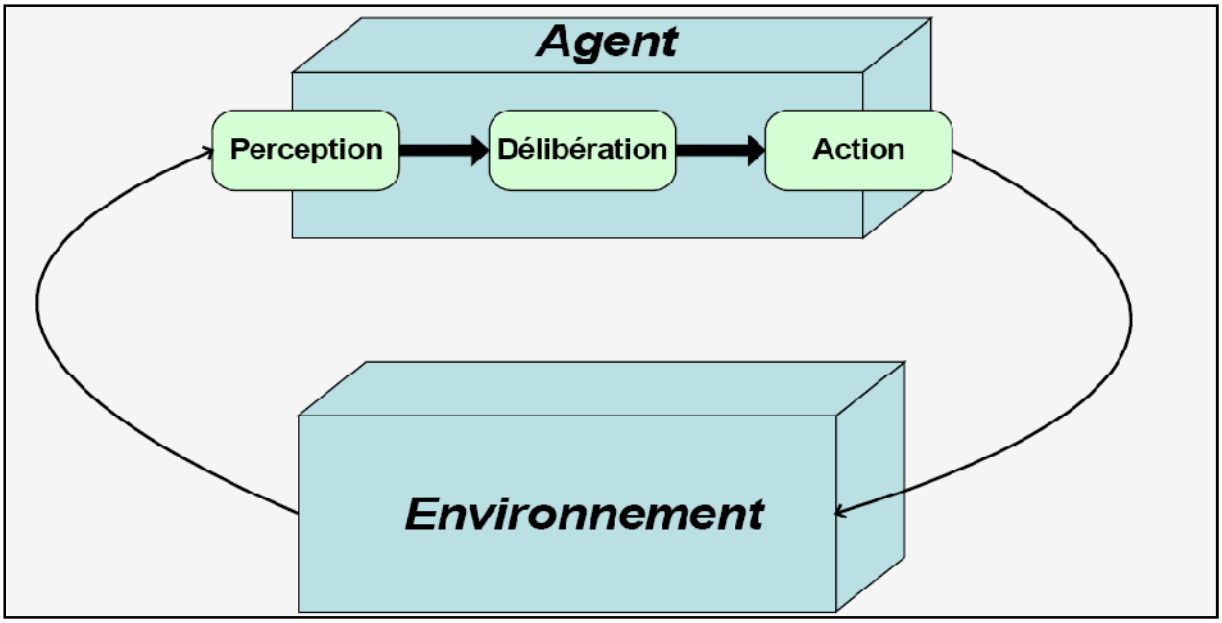

Figure II.2 : Architecture d'un environnement

D'après **[CHD, 02]** la notion Agent-Environnement est définit comme suit :

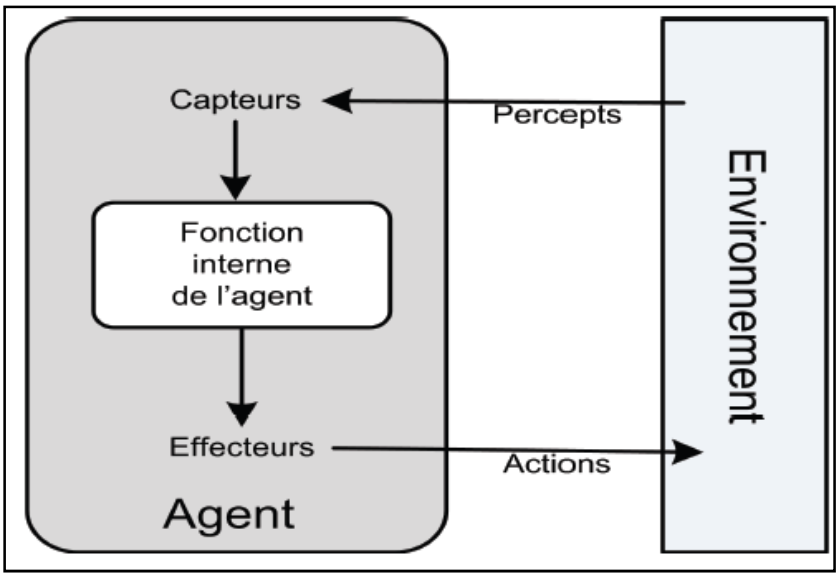

Figure II.3 : Agent-Environnement

L'agent perçoit son environnement à l'aide des capteurs, choisit la ou les actions à faire selon sa fonction interne et exécute ces actions dans l'environnement à l'aide de ses effecteurs.

La fonction interne propre à l'agent détermine les actions qu'il doit effectuer et elle varie selon le type d'agent et peut tenir compte du percept courant (Agent a reflexe simple) ou encore d'un historique de percepts.

Ainsi, un agent perçoit son environnement à l'aide de capteurs, pose des actions à l'aide de ses effecteurs et l'on évalue son efficacité à l'aide d'une mesure de performance En fait une **mesure de performance** doit refléter le succès du comportement d'un agent

Type d'agent	Mesure de performance	Environnement	Effecteurs	Capteurs
Conducteur automobile	Sécurité, rapidité, confort	Réseaux routiers, autres véhicules, piétons	Volant, accélérateur, frein	Caméras, détecteurs lasers, détecteurs infrarouges
Robot aspirateur	Propreté, rapidité	Planchers, meubles, obstacles	Roues motorisées, aspirateur	Détecteurs de poussières, détecteurs d'obstacles
Joueur échec	Victoires, nombre de pièces	Échiquier	Écran pour afficher le coup à jouer	Caméra pour observer le jeu

Table II.1 : Exemple d'environnements de tâche pour différents agents.

L'ensemble des éléments [Performance, Environnement, Effecteurs et Capteurs] forme l'environnement de la tâche.

Un environnement de tâche a de plus certaines propriétés ; ainsi il pourrait être :

Entièrement ou partiellement observable: Si l'état complet de l'environnement est accessible via les capteurs d'un agent, on dira de cet environnement qu'il est entièrement observable. Dans ce cas, tous les aspects importants au processus de décision d'un agent sont fournis par l'environnement. Dans le cas contraire, on dira de l'environnement qu'il est partiellement observable. Ce cas pourrait se produire en raison du bruit qui affecte les données transmises par les capteurs ou simplement par l'imprécision qui caractérise ces capteurs.

Déterministe ou stochastique : S'il est possible de déterminer totalement le prochain état de l'environnement en tenant compte de l'état courant et des effets des actions, on dira que l'environnement est déterministe. De cette manière, il n'y a pas de place à l'incertain concernant l'effet d'une action sur l'environnement de l'agent. Dans le cas contraire, l'environnement est dit stochastique. C'est le cas par exemple d'un environnement complexe comme par exemple la conduite automobile où il très difficile de suivre les aspects non observés de la circulation.

Épisodique ou séquentiel : Dans une tâche épisodique, l'agent perçoit puis exécute une action unique ; et le choix de cette action dépend uniquement de l'épisode lui même. La décision dans ce cas, n'influe pas sur les épisodes qui suivent. Dans les environnements séquentiels au contraire, la décision courante est susceptible d'affecter toutes les décisions futures. Ainsi, par exemple, un agent analyseur d'images a un environnement épisodique ; alors qu'un agent joueur d'échecs à un environnement séquentiel.

Statique ou dynamique : Un environnement statique ne change pas d'état pendant que l'agent délibère. Si c'est le cas, l'environnement est dit dynamique. Dans le premier cas, la prise de décision est plus ou moins facile dans la mesure où l'agent n'a pas à se soucier des changements qui risquent de se produire. Dans le deuxième cas en revanche, l'agent se doit de toujours déterminer « quoi faire » en fonction des changements intervenus dans l'environnement. L'agent conducteur d'automobile est l'exemple type d'agent à environnement dynamique.

Discret ou continu : Un environnement est dit discret lorsque l'état d'un tel environnement, son temps de gestion ainsi que les percepts et les actions qui le caractérise sont discrets. Ainsi un robot de surveillance par exemple, pourrait être vu comme un agent où le temps et les actions seraient continues Mono agent ou Multi agent. La question est de savoir ici si la mesure de performance attachée à un agent est indépendante ou dépendante d'autres agents. Ainsi, jouer aux échecs, nettoyer une salle à plusieurs constituent des environnements multi-agents ; tandis que nettoyer une salle tout seul est un environnement mono-agent.

I.4 Typologie des agents :

Il existe deux grandes écoles de pensée dans la communauté des agents : l'école cognitive qui conçoit les agents comme des entités intelligentes et l'école réactive qui conçoit les agents comme des entités très simples réagissant directement aux modifications de l'environnement**.**

I.4.1 Les agents cognitifs :

Un système cognitif comprend un petit nombre d'agents **[Dieng90]** dont chacun est assimilable, suivant le niveau de ses capacités, à un système expert plus ou moins sophistiqué, on parle d'agent de forte granularité.

Ils regroupent plusieurs sous-types d'agents définis de la façon suivante :

Les **« agents intelligents » [TrEsp99]** combinent les trois caractéristiques (autonomie, coopération, adaptativité) à leur plus haut niveau. Ils sont capables de négocier avec d'autres agents et d'acquérir ou de modifier leurs connaissances. Ainsi, ils planifient leurs actions. Le terme d'agent intelligent est souvent utilisé pour caractériser des agents dotés de la capacité d'apprentissage.

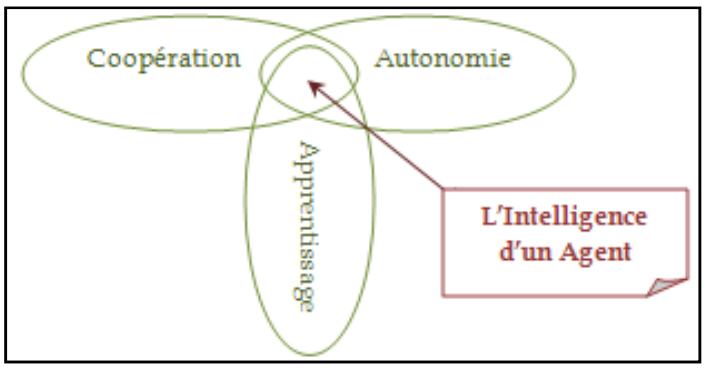

Figure II.4 : Agent Intelligent

Les **« agents collaborant » [Nwa96]** : ce sont des agents cognitifs non apprenants. Ils sont donc à la fois fortement autonomes et coopérants, mais leur adaptabilité ne va pas jusqu'à l'adaptation et à l'acquisition des connaissances. Les agents collaborant sont surtout utilisés dans les domaines qui nécessitent une décentralisation comme par exemple la maintenance de réseau, ou encore pour simuler le comportement d'organisations humaines ou animales **[Bourr]**.

Les agents interface [MAE, 95]

Les agents interface sont des agents relativement autonomes qui utilisent différentes techniques d'apprentissage pour effectuer des tâches pour leur utilisateur. On peut voir un agent interface comme un assistant personnel qui effectue diverses tâches. Les principales tâches de ce type d'agent est de fournir du support et de l'assistance à l'utilisateur. Entre autre, ces agents peuvent filtrer des courriels, acheter ou vendre des articles sur le Web ou être votre compagnon dans *MS Word*. Un des principaux mandats de ces agents est de faciliter et accélérer quelques tâches spécifiques de l'utilisateur. La plupart du temps, les agents interface observent et enregistrent les actions de l'utilisateur. Par la suite, ils peuvent lui suggérer une meilleure façon d'effectuer ce travail. Les agents apprennent de différentes façons. Parmi les plus courantes, on retrouve l'apprentissage par observation, par imitation de l'utilisateur, l'apprentissage par réception de *feedback6* (négatif ou positif) de l'utilisateur et réception d'instructions explicites de l'utilisateur (Figure 5).

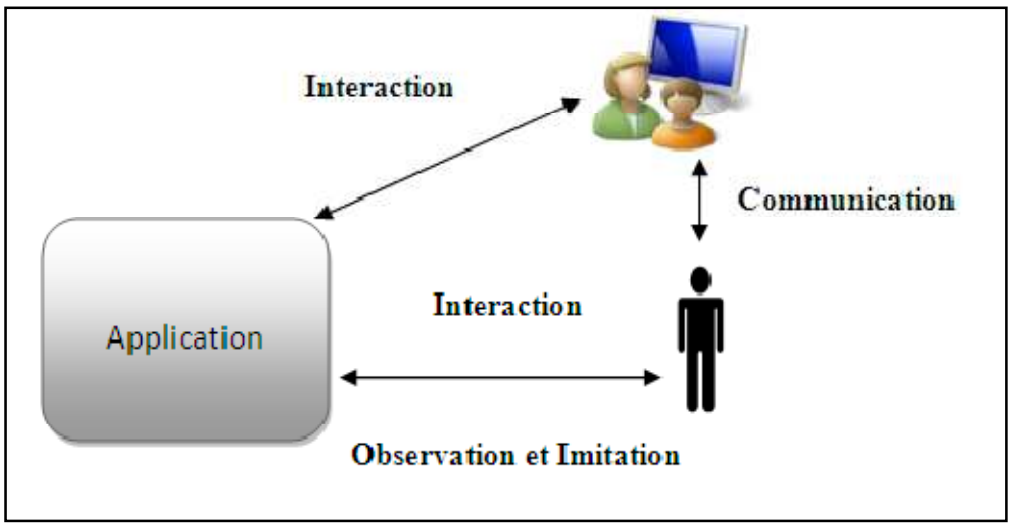

Figure II.5 : Architecture d'un agent interface simple

Les **« agents informations » [Rho00, RGK97]** sont dédiés à la recherche d'information, principalement sur l'Internet. Ces agents (tels que softbot **[Etz94])** possèdent une grande autonomie; ils agissent seuls, soit en fonction d'un calendrier, soit en fonction d'un manque d'information, soit en fonction d'une nouvelle disponibilité d'information. Ils sont capables d'adapter leurs fonctionnements en fonction du besoin de l'utilisateur ou de la quantité ou pertinence de l'information (par exemple, si un nouveau site propose des informations plus pertinentes, ce site sera alors privilégié pour les recherches futures).

Toutefois, ces agents agissent le plus souvent de manière isolée, ce qui peut entraîner un

Synthèse

La figure II.6 résume ces différents types d'agents cognitifs ainsi que leurs degrés d'autonomie, d'apprentissage et de coopération. Les agents «collaborant » sont autonomes et coopérants, leurs capacités de négociation impliquent une faculté d'adaptation, de modification de leur comportement. Les agents « interfaces » réagissent aux sollicitations de l'utilisateur, ils sont donc peu autonomes. Ils s'adaptent au comportement de l'utilisateur (par exemple, ils peuvent changer leur représentation) et communiquent leurs déductions sur son état (débutant, occasionnel, expert) à d'autres agents interfaces. Les agents « informations » peuvent agir de façon très autonome pour la recherche d'une information au travers de différents sites, mais souvent indépendamment des autres agents. Ils sont capables d'adapter leur stratégie de recherche en fonction des informations trouvées ou non trouvées (par exemple, pour modifier la requête ou chercher de nouvelles sources d'information).

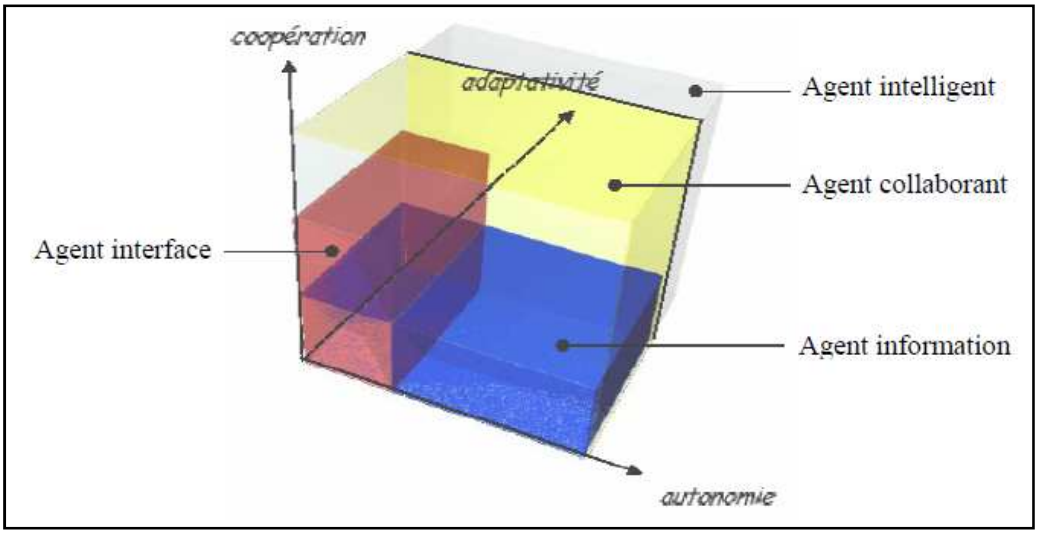

Figure II.6 : Degrés d'autonomie, de coopération et d'adaptativité des principaux agents cognitifs

I.4.2 Les Agents réactifs :

Les défenseurs de cette approche partent du principe suivant : dans un système multi agents, il n'est pas nécessaire que chaque agent soit individuellement « intelligent » pour parvenir à un comportement global intelligent. En effet, des mécanismes simples de réactions aux évènements peuvent faire émerger des comportements correspondant aux objectifs poursuivis. Cette approche propose la coopération d'agents de faible granularité (fine grain) mais beaucoup plus nombreux.

Les agents réactifs sont de plus bas niveau, ils ne disposent que d'un protocole et d'un langage de communication réduite, leurs capacités répondent uniquement à la loi stimulus/action.

Les premiers travaux relatifs à cette approche ont été réalisés en 1986 par R. Brooks **[Bro89].** D'après lui, plusieurs milliers de micro-robots identiques, d'une taille aussi petite que possible, travaillant ensemble sur une tâche donnée pourront être plus efficaces qu'un seul gros robot spécialisé.

Le tableau 2 résume les différentes propriétés des deux approches :

Systèmes d'agents cognitifs	**Systèmes d'agents réactifs**
Représentation explicite de l'environnement Petit nombre d'agents	Pas de représentation explicite
L'agent peut tenir compte de son passé	Pas de mémoire de son historique
Agents complexes	Fonctionnement stimulus/Réponse
Petit nombre d'agent	Grand nombre d'agent

Table II.2 : Les agents cognitifs et réactifs

I.5 Architecture d'agents : [CHD, 02]

Il existe plusieurs manières de concevoir des agents, mais peu importe l'architecture adoptée, un agent peut toujours être vu comme une fonction liant ses perceptions à ses actions (voir figure 2.1). Plus précisément, un agent perçoit l'environnement à l'aide de ses capteurs et il agit sur son environnement à l'aide de ses effecteurs. Ce qui différencie les différentes architectures d'agents, c'est la manière dont les perceptions sont liées aux actions.

Les architectures d'agents peuvent être regroupées en agents réactifs et agents délibératifs comme suit :

– les agents réactifs.

1. les agents à réflexes simples,
2. les agents conservant une trace du monde.

– les agents délibératifs.

1. les agents ayant des buts,
2. les agents utilisant une fonction d'utilité,
3. les agents du type BDI (Beliefs-Desires-Intentions)

Les sections suivantes détaillent ces types d'architectures.

I.5.1 Agents à réflexes simples : Ce type d'agent agit uniquement en se basant sur ses perceptions courantes.

Il utilise un ensemble de règles prédéfinies, du type Si *condition* alors *action*, pour choisir ses actions. Par exemple, pour un agent en charge du contrôle de la défense d'une frégate, on pourrait avoir la règle suivante :

Si missile-en-direction-de-la-frégate **alors** lancer-missiled'interception

Comme on peut le constater, ces règles permettent d'avoir un lien direct entre les perceptions de l'agent et ses actions. Le comportement de l'agent est donc très rapide, mais peu réfléchi. À chaque fois, l'agent ne fait qu'exécuter l'action correspondant à la règle activée par ses perceptions. Il présente l'avantage d'être fort simple mais en pratique il est très limité. En effet, un tel agent ne peut fonctionner correctement que s'il peut prendre la bonne décision sur la base du seul percept courant détecté, c'est à dire seulement si l'environnement est entièrement observable.

I.5.2 Agents conservant une trace du monde :

Le type d'agent qui a été décrit précédemment, ne peut fonctionner que si un tel agent peut choisir ses actions en se basant uniquement sur sa perception actuelle. Par exemple, si le radar de la frégate détecte un missile et que l'instant d'après, il le perd de vue, dû à un obstacle, cela ne signifie nullement qu'il n'y a plus de missile. Dès lors, l'agent en charge du contrôle de la

frégate doit tenir compte de ce missile dans le choix de ses actions et ce, même si le radar ne détecte plus le missile. Un tel problème survient parce que les capteurs de l'agent ne fournissent pas une vue *complète* du monde. Pour régler ce problème, l'agent doit maintenir des informations internes sur l'état du monde dans le but d'être capable de distinguer deux situations qui ont des perceptions identiques, mais qui, en réalité, sont différentes. L'agent doit pouvoir faire la différence entre un état où il n'y a pas de missile et un état où le missile est caché, même si ses capteurs lui fournissent exactement les mêmes informations dans les deux cas. Pour que l'agent puisse faire évoluer ses informations internes sur l'état du monde, il a besoin de deux types d'information. Tout d'abord, il doit avoir des informations sur la manière dont le monde évolue, indépendamment de l'agent. Par exemple, il doit savoir que si un missile avance à une vitesse de 300 m/s, alors 5 secondes plus tard, il aura parcouru 1500 mètres.

L'agent doit avoir ensuite des informations sur la manière dont ses propres actions affectent le monde autour de lui. Si la frégate tourne, l'agent doit savoir que tout ce qui l'environne tourne aussi. Il doit donc mettre à jour la position relative de tous les objets autour de la frégate.

Dès lors, la structure d'un tel agent pourrait être vue comme une amélioration de l'agent à réflexes simples précédent et elle viserait à améliorer le processus d'évaluation de l'environnement à l'instant t par :

1. une trace du monde reflétant l'instant (t – 1) ;
2. une dynamique qui indiquerait comment le monde évolue ;
3. une fonction d'évaluation de l'impact des actions de l'agent.

I.5.3 Agents ayant des buts :

Dans la section précédente, les agents utilisaient leurs connaissances sur l'état actuel de l'environnement pour choisir leurs actions mais l'agent a besoin, en plus de la description de l'état actuel de son environnement, de certaines informations décrivant ses buts. Lesquels buts peuvent être vus comme des situations désirables pour l'agent.

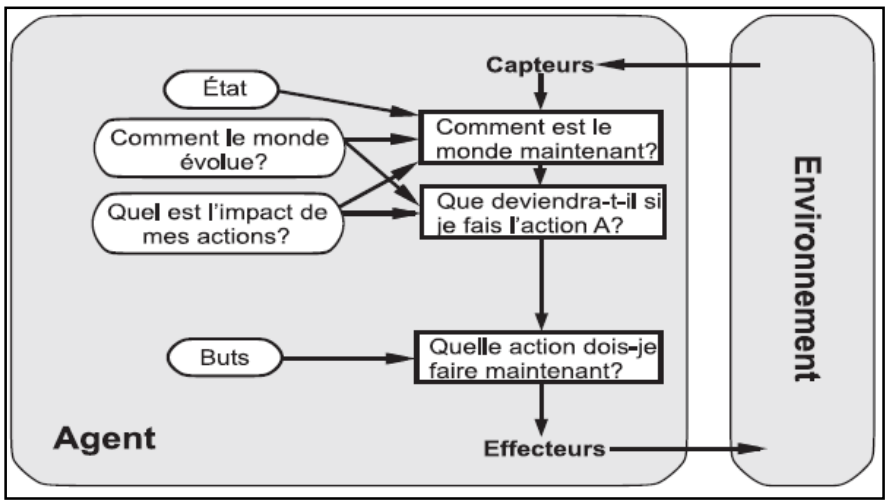

Figure II.7 : Schéma d'un agent ayant des buts

I.5.4 Agents utilisant une fonction d'utilité :

Dans plusieurs situations, les buts ne sont pas suffisants pour générer un comportement de haute qualité. Par exemple, s'il y a plusieurs chemins possibles pour atteindre le port, certains seront plus rapides et d'autres plus dangereux. Dans cette situation, l'agent raisonnant uniquement sur ses buts n'a pas de moyens pour choisir le meilleur chemin, son seul but étant de se rendre à destination. En effet, les buts ne procurent qu'une simple distinction entre les états où l'agent est satisfait et les états où il ne l'est pas. En fait, l'agent doit plutôt s'appuyer sur une manière plus fine d'évaluer les états pour être en mesure de reconnaître pour chacun des états son degré de satisfaction. Pour cela, on dit que l'agent va *préférer* un état à un autre si son utilité est plus grande dans le premier état que dans le deuxième.

Généralement, l'utilité est une fonction qui attribue une valeur numérique à chacun des états. Plus l'état a une grande valeur, plus il est désirable pour l'agent.

I.5.5 Agents BDI :

L'architecture BDI est une autre approche utilisée dans la conception des agents délibératifs. BDI est un acronyme qui signifie, en anglais, *Belief, Desire, Intentions*. Ce qui se traduit en français par croyances, désirs et intentions. Les agents se basent donc sur ces trois aspects pour choisir leurs actions.

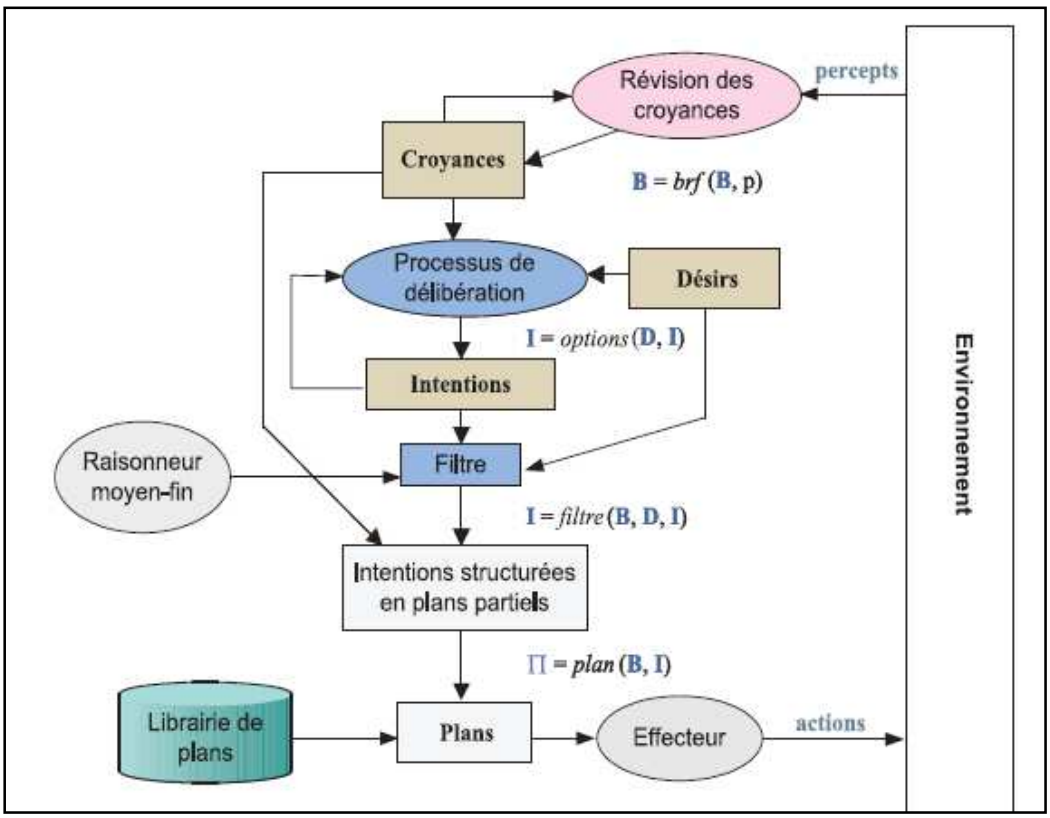

Figure II.8 : Architecture BDI

Une architecture ayant sept composantes, telles que présentées sur la figure 10 :

– Un ensemble de *croyances* courantes, représentant les informations que l'agent possède à propos de son environnement courant ;

– Une *fonction de révision des croyances (fonction brf())*, qui prend les entrées des capteurs et les croyances actuelles de l'agent et qui détermine un nouvel ensemble de croyances ;

– Une *fonction de génération des options (fonction options())*, qui détermine les options disponibles pour l'agent (i.e. ses désirs), en se basant sur les croyances courantes de l'agent à propos de son environnement et sur ses intentions courantes ;

– Un ensemble de *désirs*, représentant les options disponibles à l'agent ;

– Une *fonction de filtre (fonction filtre())*, qui représente le processus de délibération de l'agent et qui détermine les intentions de l'agent en se basant sur ses croyances, ses désirs et ses intentions courantes ;

– Un ensemble d'*intentions structurées (fonction plan())* courantes, représentant le centre d'attention actuel de l'agent, c'est-à-dire les buts envers lesquels il s'est engagé et envers lesquels il a engagé des ressources ;

– Une *fonction de sélection des actions*, qui détermine l'action à effectuer en se basant sur les intentions courantes de l'agent.

I.5.6 Les agents hybrides :

L'architecture hybride est une architecture composée d'un ensemble de modules organisés dans une hiérarchie, chaque module étant soit une composante cognitive, soit une composante réactive. De cette manière, le comportement proactif de l'agent, dirigé par les buts, est combiné avec un comportement réactif aux changements de l'environnement, afin d'obtenir simultanément les avantages des architectures cognitives et réactives, tout en éliminant leurs limitations **[SEC, 03]**.

La plupart des architectures hybrides considèrent que trois couches suffisent amplement. Ainsi, au plus bas niveau de l'architecture, on retrouve habituellement une couche purement réactive, qui prend ses décisions en se basant sur des données brutes en provenance des senseurs. La couche intermédiaire fait abstraction des données brutes et travaille plutôt avec une vision qui se situe au niveau des connaissances de l'environnement. Finalement, la couche supérieure se charge des aspects sociaux de l'environnement, c'est à dire du raisonnement tenant compte des autres agents **[CHD, 02]**.

I.6 Fonctionnement d'agents :

Les agents sont immergés dans un environnement dans lequel et avec lequel ils interagissent. D'où leur structure autour de trois fonctions principales : percevoir, décider et agir. Parmi les sous-fonctions importantes d'un agent on peut citer : le raisonnement et l'apprentissage.

I.6.1 Le raisonnement des agents :

Un agent a la possibilité d'acquérir des connaissances sur l'environnement externe. Il a aussi des capacités d'interaction avec les autres agents. En fonction des connaissances et croyances dont il dispose et des buts qu'il se fixe suite à une perception ou à une interaction avec le monde extérieur, l'agent doit élaborer un plan d'action. Pour cela, il doit décider quel serait le but à retenir et à satisfaire en premier, ensuite planifier en fonction de ce but et passer à l'exécution.

On dit d'un agent qu'il est un agent de raisonnement s'il intègre les éléments suivants.

- **la base de faits** qui constitue l'état du programme; elle peut être vue comme la perception en temps réel, par l'agent, du monde à un instant T;
- **la base de règles** qui est généralement la connaissance du travail fourni à l'agent par son concepteur ou par l'utilisateur;
- **La structure de contrôle**, qui est une structure procédurale du moteur raisonnement, et qui détermine, entre autres choses, quand réévaluer une règle donnée;

- **les interfaces événement/action** qui sont des interfaces de programmation par lesquelles l'agent reçoit des événements et effectue des actions. Les événements sont délivrés à l'agent par d'autres applications ou services appelés aussi capteurs.

Les actions décidées par le système sont exécutées par des acteurs. **[CAG, 98]**

I.6.2 L'apprentissage pour les agents :

L'apprentissage est la modification du comportement comme résultat de l'expérience. Pour les agents fondés sur les règles, l'apprentissage implique que l'agent ait la possibilité de modifier automatiquement la base de règles et les faits à long terme d'une ou de plusieurs manières.

- **ajouter de nouvelles règles ou modifier des règles anciennes**. Si l'agent est capable de reconnaître un nouveau comportement intéressant, il doit être capable de proposer une nouvelle règle ou du moins d'attirer l'attention u développeur sur sa croyance en un nouveau comportement souhaité et de l'inviter à créer une nouvelle règle ;
- **ajouter de nouveaux faits ou modifier d'anciens faits**. Dans ce cas l'agent doit être capable de connaître de nouveaux faits par exemple reconnaître un nouvel utilisateur important à prendre en compte. **[CAG, 98].**

II. Les systèmes multi agents SMA :

Les systèmes multi-agents s'appuient sur le principe suivant : au lieu d'avoir un seul agent en charge de l'intégralité d'un problème, on considère plusieurs agents qui n'ont chacun en charge qu'une partie de ce problème. La solution au problème initial est alors obtenue au travers de l'ensemble des comportements individuels et des interactions, c'est à dire par une résolution collective.

II.1 Définition :

Ferber **[Ferb95],** définit un système multi-agents de la façon suivante (figure II.9) :

On appelle système multi-agents (ou SMA), un système composé des éléments suivants:

1. Un environnement **E**, c'est-à-dire un espace disposant généralement d'une métrique.
2. Un ensemble d'objets **O** situé dans **E.** Ces objets sont passifs, c'est-à-dire qu'ils peuvent être perçus, créés, détruits et modifiés par les agents.
3. Un ensemble **A** d'agents, qui représentent les entités actives du système.
4. Un ensemble de relations **R** qui unissent des objets (et donc des agents) entre eux.
5. Un ensemble d'opérations **Op** permettant aux agents de **A** de percevoir, produire, consommer, transformer et manipuler des objets de **O**.
6. Des opérateurs chargés de représenter l'application de ces opérations et la réaction de l'environnement envers les tentatives de modification.

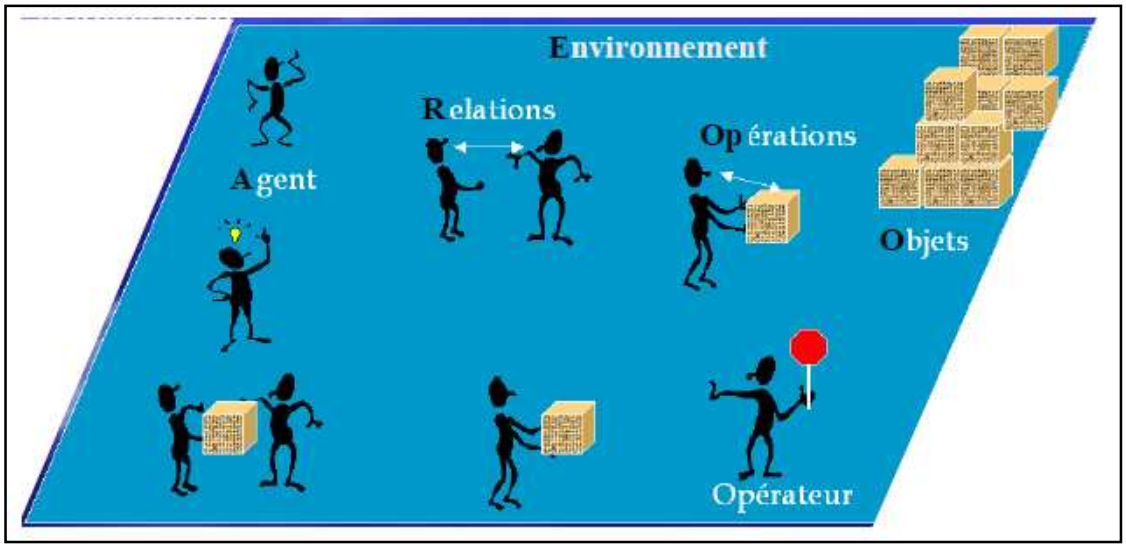

Figure II.9 : Architecture d'un SMA

II.2 Caractéristique d'un SMA :

Généralement, un SMA possède les caractéristiques suivantes :

- Les agents agissent et travaillent indépendamment les uns des autres.
- Chaque agent est une partie du système.
- Chaque agent travaille dans le but d'accomplir ses tâches.
- Chaque agent est capable de communiquer et d'interagir avec d'autres agents.
- Un agent coopère avec les autres agents lorsque nécessaire.
- Un agent est capable de coordonner ses activités avec les autres agents pour accéder à des ressources et à des services partagés dont il a besoin (pour réaliser ses buts).
- Les agents ont un but commun.
- Chaque agent a une vue partielle du SMA.

Et nous pouvons dire qu'un SMA peut prendre une des propriétés suivantes :

Ouvert : les agents y entrent et en sortent librement (ex: une application de commerce électronique, etc.).

Fermé : l'ensemble d'agents reste le même.

Homogène : tous les agents sont construits sur le même modèle.

Hétérogène : des agents de modèles différents, de granularités différentes.

Mixte (ou non) : les agents « humains » sont partie intégrante du système, comme le serait un

II.3 Les interactions dans les SMA :

III.3.1 Les types d'interaction :

Une interaction « est une mise en relation dynamique de deux ou plusieurs agents par le biais d'un ensemble d'actions réciproques.les interactions s'expriment ainsi à partir d'une série d'actions dans les conséquences exercent en retour une influence sur le comportement futur des agents » **[FER, 97].**

Nous allons donner par la suite quelques définitions de la communication, la coopération, la coordination et la collaboration, ces types d'interactions étant parfois difficiles à différencier.

A) La communication :

La communication est la base de la résolution coopérative des problèmes. Elle permet de synchroniser les actions des agents et résoudre les conflits de ressources et de buts par la négociation. Dans les systèmes multi-agents, les agents ne disposent d'aucune mémoire commune. La communication entre les agents repose explicitement sur des mécanismes d'envoi de message, de réception et de synchronisation.

a) Protocoles de communication

Dans un système distribué, il faut que les entités disposent d'un langage commun pour pouvoir échanger des informations et coopérer pour la résolution d'un problème. Ce langage appelé mécanisme de communication interprocessus est un ensemble de primitives connues par chaque entité et dont l'ensemble constitue un protocole de communication.

Les primitives de base d'un protocole de communication sont les suivantes :

- établissement d'une connexion entre deux entités ;
- identification du nœud destinataire dans un réseau de communication ;
- envoi de données ;

b) Architecture de communication

La communication entre les agents dans les SMA peut être organisée suivant trois schémas différents :

- **Réseaux en anneau :** Les interactions dans ce genre d'organisations ont trop lentes, un message destiné à un agent doit transiter par n-1 agents.

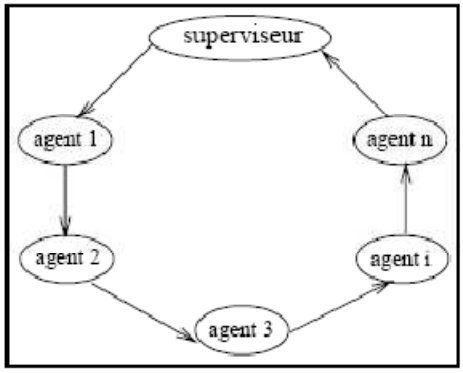

Figure II.10 : Réseau en anneau

- **Réseaux en étoiles :** Ils présentent l'avantage de l'accès rapide entre le superviseur et les autres agents. La nature bidirectionnelle des connexions rend complexe la gestion des interactions inter-agents.

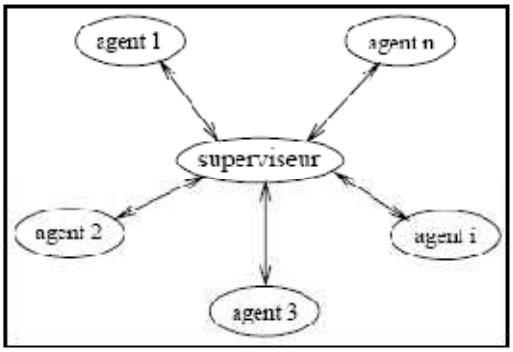

Figure II.11 : Réseau en étoile

- **Réseaux hybrides :** Ce type d'organisation est une solution hybride reliant deux types d'organisation : un réseau en bus et un réseau en étoile.

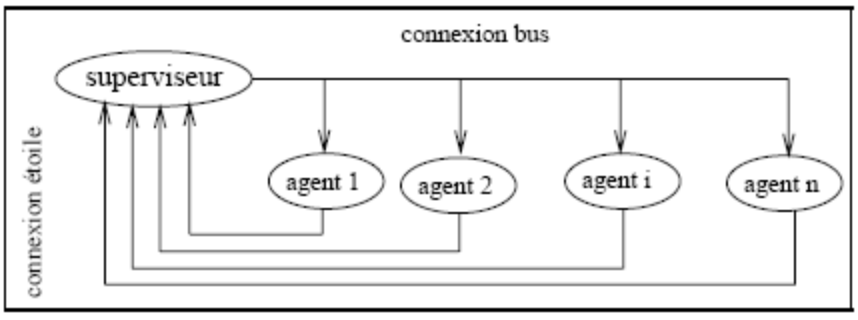

Figure II.12 : Réseau hybride

c) Mode de communication :

Il existe deux principaux modes de communication : la communication par envoi de messages et la communication par partage d'informations.

- **Communication par envoi de messages :** Les agents sont en liaison directe et envoient leurs messages directement et explicitement au destinataire. La seule contrainte est la connaissance de l'agent destinataire. Les systèmes fondés sur la communication par envoi de messages relèvent d'une distribution totale à la fois de la connaissance, des résultats et des méthodes utilisées pour la résolution du problème.

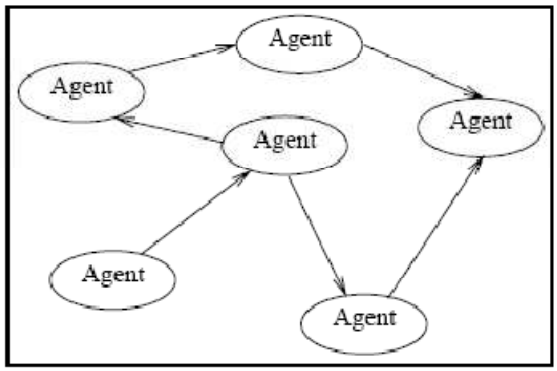

Figure II.13 : Communication par envoi des messages

-Communication par partage d'informations : Les composants ne sont pas en liaison directe mais communiquent via une structure de données partagée, ou on trouve les connaissances relatives à la résolution (état courant du problème) qui évolue durant le processus d'exécution. Cette manière de communiquer est l'une des plus utilisées dans la conception des systèmes multi-experts. L'exemple parfait d'utilisation de ce mode de communication est l'architecture de blackboard, on parle plutôt de sources de connaissances que d'agents. Ce mode de communication n'existe pas dans les systèmes multi-agent où l'on dispose que d'une vision partielle du système alors que la communication par partage d'informations suppose l'existence d'une base partagée sur laquelle les composants viennent lire et écrire.

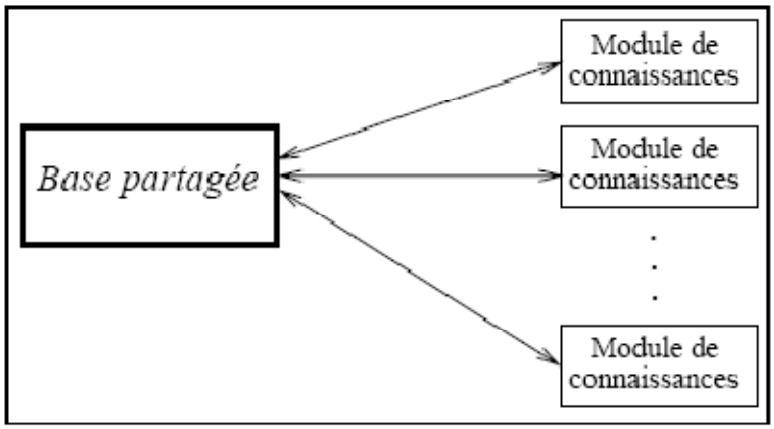

Figure II.14 : Communication par partage d'information

d) Langages de communication :

Lorsque la notion de groupe d'agents est apparue, les recherches sur des mécanismes permettant aux agents de communiquer entre eux se sont amorcées. Celles-ci ont mené à trois langages de communication et de représentation de l'information qui sont devenus des « standards » en SMA : KQML, FIPA ACL et KIF.

KQML (Knowledge Query ManipulationLanguage): Ce langage de communication se base sur la théorie des actes de langages **[Finin, 93], [Finin, 94]**, **[Finin, 94b]**, **[Finin, 94c]**. KQML détermine un format pour les messages et un protocole pour la réception et l'envoi de ces derniers. KQML permet aux agents de partager des informations avec les autres agents du système afin de coopérer pour résoudre un problème.

Le langage KQML spécifie le format des messages échangés par les agents, sans être concerné par le format de l'information transportée. Un message KQML peut être vu comme un objet, défini par l'information clé-la 'performative'- et un nombre variable d'attributs.

FIPA ACL : FIPA (Foundation For Intelligent Physical Agent) : est une organisation qui tente d'instaurer des standards dans le domaine des SMA **[FIPA, 97]**. Le langage (FIPA ACL) est un langage semblable à KQML auquel des protocoles d'interaction comme le *contractnet* et autre protocole populaire ont été ajoutés.

FIPA-ACL possède 21 actes communicatifs qui peuvent être groupés selon leurs fonctionnalités de la façon suivante :

- **Passage d'information**: inform*, inform-if (macro act), inform-ref (macro act), confirm*, disconfirm*;
- **Réquisition d'information** : query-if, query-ref, subscribe;
- **Négociation** : accept-proposal, cfp, propose, reject-proposal;

▪ **Distribution de tâches** (ou exécution d'une action) : request*, request-when, request whenever, agree, cancel, refuse;

▪ **Manipulation des erreurs** : failure, not-understood.

Les actes communicatifs peuvent être :

▪ **Les actes communicatifs primitifs,** définis de façon élémentaire (mentionnés ci dessus par une étoile '*') ;

▪ **Les actes communicatifs composés**, définis à partir d'autres actes par l'une des opérations suivantes :

• un acte fait partie du contenu d'un autre acte; à travers l'opérateur de composition " ; " pour indiquer une séquence d'actions;

• à travers l'opérateur de composition " | " pour indiquer un choix non déterministe de l'action.

B) La coopération :

La coopération peut être considérée comme une attitude adoptée par les agents qui décident de travailler ensemble. Durfee et ses collègues **[DUR 89]** ont proposé quatre buts génériques pour établir la coopération dans ce groupe d'agents :

-Augmenter le taux de finalisation des tâches grâce au parallélisme,

-Augmenter le nombre de tâches réalisables grâce au partage de ressources (information, expertise, dispositifs physiques, etc.),

-Augmenter les chances de finaliser des tâches en les dupliquant et en utilisant éventuellement des modes de réalisation différents,

-Diminuer les interférences entre tâches en évitant les interactions négatives.

C) La négociation :

La technique de la négociation pour la résolution de confits ne fixe aucun plan préalable, mais elle résout le problème du conflit au moment ou il se pose. Elle considère qu'il est souvent plus facile de mettre en œuvre des mécanismes de résolution de conflits fondés sur des agents que de planifier l'ensemble des actions et leurs interactions.

Composantes de la négociation dans les SMA :

Pour modéliser la négociation dans un logiciel multi-agents, il faut prendre en compte les aspects suivants.

• Le langage de négociation est utilisé par les agents pour échanger des informations pendant la négociation; il est composé d'un ensemble de primitives de communication.

• Le protocole de négociation est l'ensemble des règles concernant : les participants possibles dans la négociation, les propositions légales que les participants peuvent faire, les états de la négociation (par exemple l'état initial où commence la négociation) et une règle pour

déterminer quand on est arrivé à un accord ou quand il faut s'arrêter parce qu'aucun accord n'a pu être trouvé.

• L'objet de négociation représente le sujet autour duquel tourne la négociation.

• Le processus de décision, c'est à dire le modèle que l'agent utilise pour prendre des décisions pendant la négociation. La partie la plus importante de la prise des décisions dans ce cas est la stratégie de négociation qui permet de déterminer quelle primitive de négociation l'agent doit choisir à un certain moment. Pour prendre une décision adéquate, un agent doit être capable de faire un raisonnement stratégique, notamment raisonner en tenant compte de ce que font/décident les autres agents et, s'il parvient à le savoir ou à le supposer, quel est le modèle de décision des autres agents.

• Les participants sont les agents qui prennent partie dans la négociation. Selon le nombre des agents et des interactions, on distingue les cas de figure suivants :

- **Négociation un à un :** un agent négocie avec un autre, par exemple la négociation du prix d'achat d'une maison avec le représentant d'une agence immobilière;

- **Négociation un à plusieurs :** un seul agent négocie avec plusieurs autres agents, par exemple les enchères où un agent veut vendre un objet;

- **Négociation plusieurs à plusieurs :** plusieurs agents négocient avec plusieurs autres agents en même temps, par exemple les participants à des enchères électroniques.

II.3.2 Modélisation des interactions :

Les interactions entre agents sont souvent modélisées avec des approches logiques, mais aussi avec des outils comme des graphes de transition d'automates à états finis [Win86]. Les figures 17 et 18 représentent respectivement un exemple d'interaction modélisé par un automate et à l'aide d'un réseau de Pétri. Dans cet exemple, un état de l'automate représente l'état d'un système à deux agents A et B et une transition représente une interaction. Avec les réseaux de Pétri [Segh99], les places du réseau représentent les états internes de l'agent et les messages en cours d'acheminement, alors que les transitions représentent des synchronisations sur les messages et des conditions d'application d'actions.

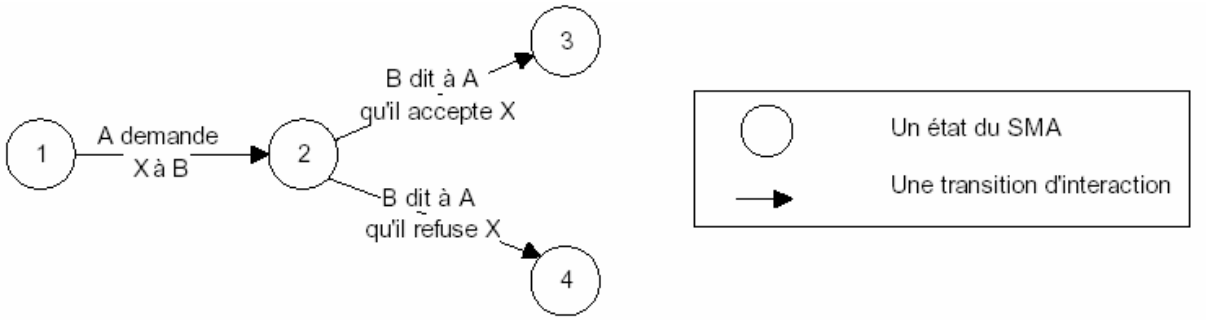

Figure II.15: Un exemple d'automate à états modélisant une interaction

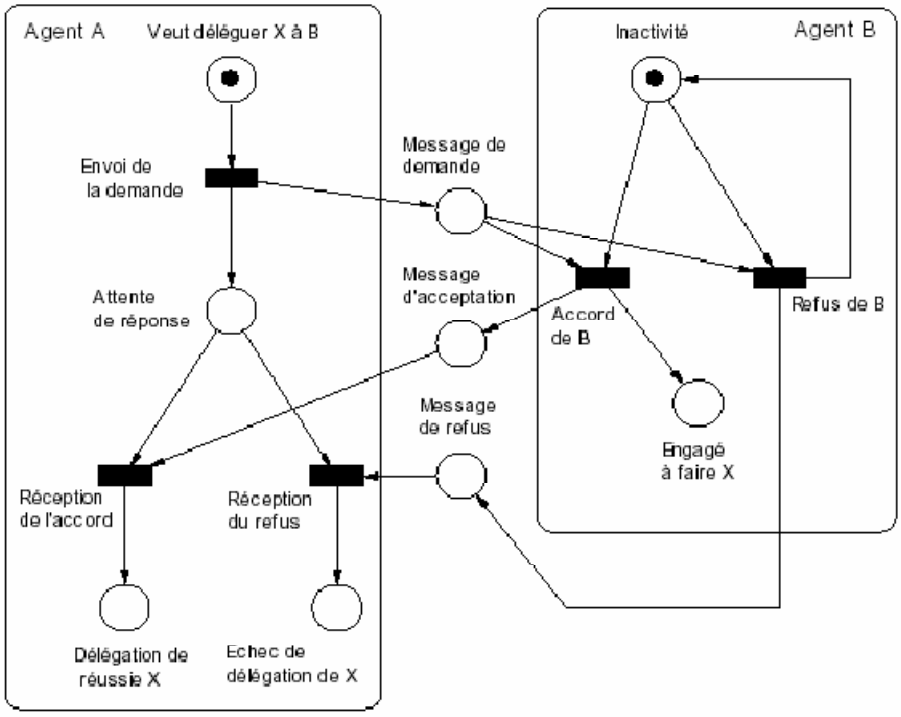

Figure II.16 : Un exemple de réseau de Pétri modélisant la même interaction que la figure 17

Modéliser les interactions permet d'une part à un observateur de comprendre la façon dont elles se déroulent et d'autre part de concevoir le comportement d'un agent qui respecte, par exemple, un protocole de communication.

II.3.3 Les situations d'interaction :

La façon dont se réalisent les interactions permet d'obtenir diverses situations au niveau du système multi-agents. Le Tableau 3 montre la classification des situations d'interaction donnée par Ferber [Ferb95] en fonction de la compatibilité des buts des intervenants, de la disponibilité des ressources et de la capacité des agents à atteindre leur but en termes de compétences individuelles.

Buts	Ressources	Compétences	Types de situation	Catégorie
Compatibles	Suffisantes	Suffisantes	Indépendance	Indifférence
Compatibles	Suffisantes	Insuffisantes	Collaboration simple	Coopération
Compatibles	Insuffisantes	Suffisantes	Encombrement	
Compatibles	Insuffisantes	Insuffisantes	Collaboration coordonnée	
Incompatibles	Suffisantes	Suffisantes	Compétition individuelle pure	Antagonisme
Incompatibles	Suffisantes	Insuffisantes	Compétition collective pure	
Incompatibles	Insuffisantes	Suffisantes	Conflits individuels pour des ressources	
Incompatibles	Insuffisantes	Insuffisantes	Conflits collectifs pour des ressources	

Table II.3 : Classification des situations d'interaction

II.4 L'organisation dans les SMA :

L'organisation sociale d'un système Multi-Agent est la manière dont le groupe est constitué, à un moment donné, pour pouvoir fonctionner. Elle décrit l'ensemble des composants fonctionnels du système, leur nature, leur responsabilité et leur besoin en ressources ainsi que les liens de communications entre agents.

Il existe plusieurs modèles d'organisation, nous citons parmi eux :

• **l'organisation à membre unique**, où le seul agent présent dans l'organisation accomplit toutes les tâches;

• **l'organisation en groupe simple** : dès qu'un groupe existe, on peut avoir une coordination ou une coopération entre les agents afin d'atteindre un but commun;

• **l'organisation hiérarchique simple** : lorsqu'un agent ne peut pas réaliser une tâche complexe, cette dernière est divisée en sous tâches qui sont ensuite distribuées à un ensemble d'agents.

II.5 Les méthodologies de conception des SMA :

Les méthodologies orientées agents sont devenues une priorité pour le développement de systèmes complexes. Il s'agit avant tout de tirer parti des modèles, méthodes et outils déjà proposés pour faciliter la construction de systèmes multi-agents. La plupart des méthodologies pour les systèmes multi-agents sont des extensions des méthodologies orientées-objets et restent ainsi incomplètes. De plus, peu d'efforts ont été faits pour la normalisation des méthodologies orientées agents et les plates formes d'exécution.

Dans ce qui va suivre, nous allons détailler quelques méthodologies les plus utilisées.

1) La méthode des VOYELLES : L'approche Voyelles, créée par Yves Demazeau du laboratoire IMAG, analyse les systèmes multi-agents selon quatre points de vue : Agents, Environnements, Interactions, et Organisations (les voyelles **A**, **E**, **I**, **O**), d'égale importance paradigmatique **[Demazeau, 1995**].

A noter que dans des systèmes de simulation comme la réalité virtuelle, on trouve parfois la voyelle **U** (oubliée dans AEIO), **U** comme Utilisateur, pour prendre en compte la participation active de l'opérateur humain à la simulation (l'homme est dans la boucle).

-**Agents**, qui concernent les modèles (ou les architectures) utilisés pour la partie active de l'agent, depuis un simple automate à un complexe système à base de connaissances.

-**Environnements**, qui sont les milieux dans lesquels sont plongés les agents. Ils sont généralement spatiaux dans la plupart des applications multi-agents.

-**Interactions**, qui concernent les infrastructures, les langages et les protocoles d'interactions entre agents, depuis de simples interactions physiques à des interactions langagières par actes de langage.

-**Organisations** qui structurent les agents en groupes, hiérarchies, relations, etc.

Outre cette décomposition en quatre briques, l'approche Voyelles est guidée par trois principes :

- Le premier est le principe déclaratif. Comme nous venons de le voir, et d'un point de vue déclaratif, un système multi-agents est composé d'agents, d'environnements, d'interactions, et d'organisations.

  SMA = Agents + Environnement + Interactions + Organisations
- D'un point de vue computationnel, les fonctionnalités d'un système multi-agents incluent les fonctionnalités individuelles des agents enrichies des fonctionnalités qui résultent de la valeur ajoutée par le système multi-agents lui-même, parfois appelée intelligence collective. Ceci constitue le principe fonctionnel.

Fonction(SMA) = Fonctions(Agents) + Fonction collective

- Enfin, pour capturer l'esprit de ce que doit être la programmation orientée multi agents, les systèmes multi-agents peuvent être considérés à un niveau d'abstraction supérieur comme des entités multi-agents. Ce dernier principe est appelé le principe de récursion.

SMA* = SMA

- D'après le paradigme Voyelles, pour un problème à résoudre (ou un système à simuler) dans un domaine donné, l'utilisateur choisit le modèle d'agent, le modèle d'environnement, le modèle d'interaction et le modèle d'organisation qu'il instancie ensuite, ainsi que leurs dynamiques selon les trois principes Voyelles, de manière à engendrer le système multi-agents computationnel et déployable qui résoudra le problème dans le domaine considéré.

L'analyse d'un SMA sous tous ces aspects nous parait digne d'intérêt mais la prise en charge dynamique des rôles et des tâches ne nous a pas paru facile à représenter.

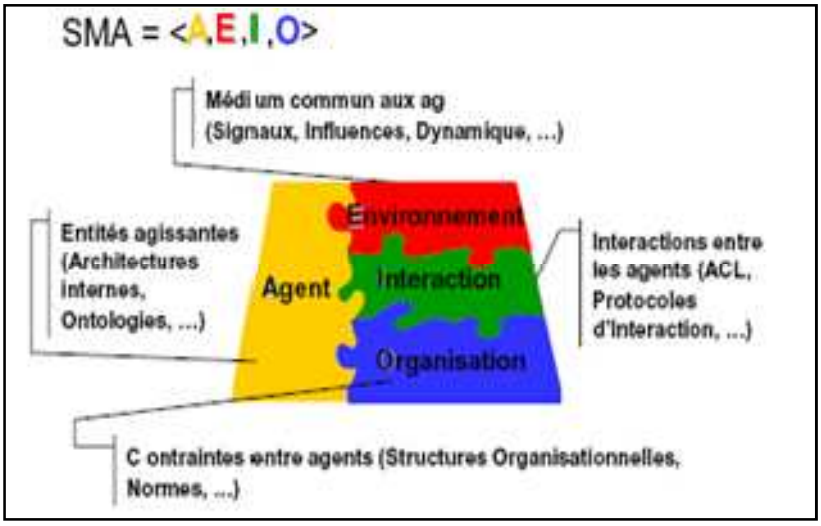

Figure II.17 : Méthode des VOYELLES

2) <u>La méthode Aalaadin</u> : Ce modèle est proposé par Ferber, il repose sur les notions de **rôle**, de **groupe** et d'**agent**.

- Un agent est une entité communicante qui joue un ou plusieurs rôles dans des groupes (un ensemble atomique d'agents). Chaque agent peut appartenir à différents groupes et les groupes peuvent se chevaucher.

- Un rôle dans Aalaadin est une représentation abstraite de la fonction, du service ou tout simplement l'identificateur d'un agent au sein d'un groupe. Chaque agent peut tenir plusieurs rôles, mais chaque rôle est local à un groupe. La communication entre les agents est possible

grâce aux rôles qu'ils assument et le contrôle sur les communications est effectué par le groupe.

Dans Aalaadin, la notion d'organisation correspond à une relation structurelle entre les rôles définis au sein des groupes et la notion d'interaction n'est pas explicitement représentée ni manipulable dans le système. Les principales étapes de cette méthode sont :

- l'analyse, qui permet d'identifier les fonctions du système et les dépendances au sein de communautés identifiées. Il convient de définir quels sont les mécanismes de coordination et d'interaction entre les entités d'analyse.
- la conception, qui contient l'identification des groupes et des rôles dans des diagrammes de structures organisationnelles.
- la réalisation, qui commence par le choix de l'architecture d'agent. Puis la gestion des entités du domaine permet d'implanter le système à partir d'organisations concrètes. Le mieux étant d'utiliser MadKit comme plate-forme de prototype et de simulation.

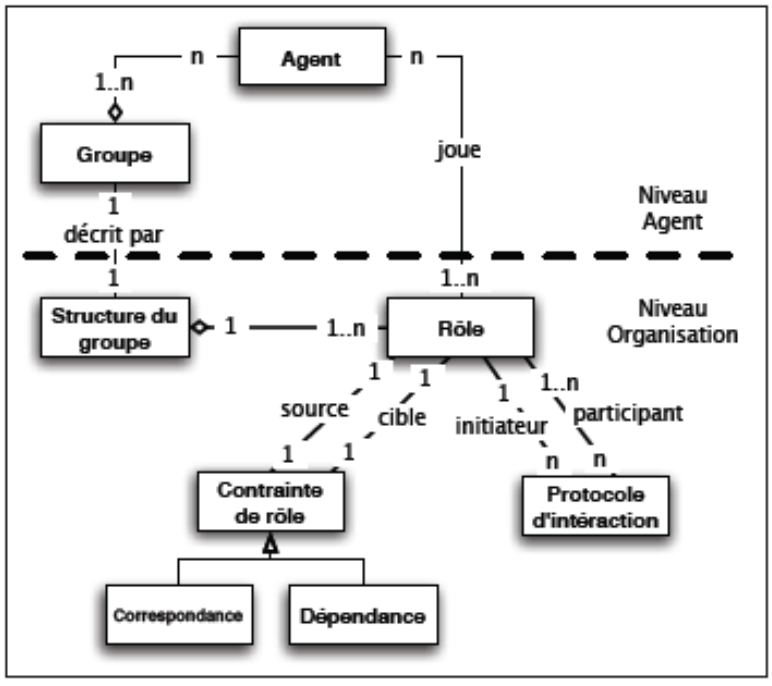

Figure II.18: Méthode Aalaadin

3) AMAS/Adelfe :

De façon similaire à MESSAGE/UML, la méthode ADELFE (Atelier pour le DÉveloppement de Logiciels à Fonctionnalité Émergente) est une extension d'UML qui tente de prendre en compte les notions associées au paradigme multi-agent. La méthode ADELFE a en fait été

développée afin de fournir un outil applicatif de la théorie des AMAS (« Adaptive Multi-Agent Systems »).

Selon la théorie AMAS, pour tout système fonctionnellement adéquat réalisant la fonction souhaitée, il existe au moins un système dont les agents sont en interactions coopératives. Cela signifie que pour concevoir un système réalisant la fonction attendue, il suffit que les agents le composant aient pour attitude sociale la coopération.

Le processus d'ADELFE est basé sur le Rational Unified Process et y ajoute des activités spécifiques à l'ingénierie orientée agent. Les notations sont une extension des notations

UML et A-UML. La méthode ADELFE propose une plateforme comprenant un outil de modélisation graphique ; une bibliothèque de composants permettant des simulations et un prototypage rapide et une technique d'auto-assemblage des composants logiciels que sont les agents adaptatifs.

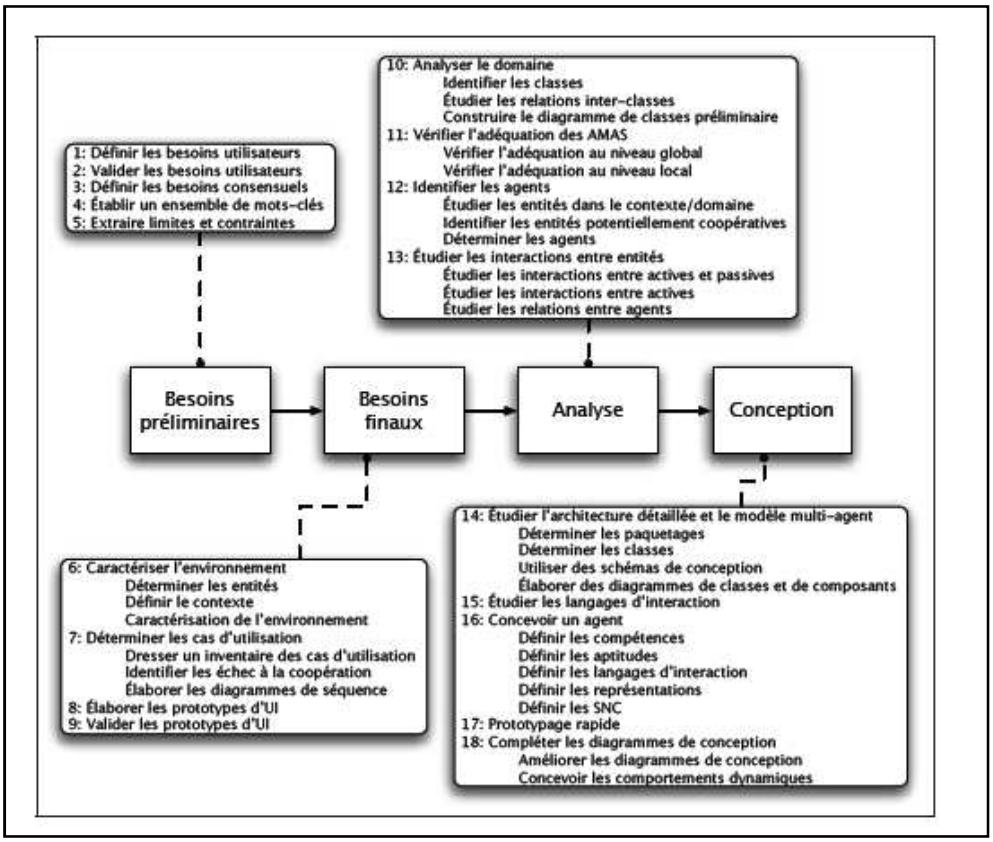

Figure II.19 : Méthode ADELFE

La méthode ADELFE comprend les phases suivantes :

Une phase d'analyse : Dans cette phase on vérifie la pertinence d'une approche multi-agent. Cela entend l'étude de critères tels que :

- complexité grande,

- absence d'autres méthodes,
- frontières mal définies,
- distribution physique ou fonctionnelle,
- environnement évolutif,
- besoin d'interactions-coopération entre entités.
- Puis, la suite de l'analyse est similaire à celle de UML;

Une phase de conception : Pour la conception d'un agent, ADELFE propose une description d'agent générique composée de sept modules :

- communication avec les autres agents,
- communication avec l'environnement,
- croyances sur lui-même,
- croyances sur les autres agents,
- croyances sur son environnement,
- compétences,
- attitude sociale coopérative.

II. 6. Les plateformes de développement des SMA : La notion de plate-forme est liée à l'implémentation des systèmes multi-agents : elle correspond à l'environnement qui permet de gérer le cycle de vie des agents et dans lequel les agents ont accès à certains services. Nous en présentons dans ce qui suit trois dont le choix a dépendu des méthodologies de conception précédemment citées.

- **Jack :** est constitué d'un éditeur gestionnaire de projets, d'un langage de programmation JAL (Jack Agent Language) et d'un compilateur. Le gestionnaire de projets est une interface qui possède un éditeur de textes où se fait l'implémentation du système. La compilation (passage de JAL à Java) et l'exécution du système se font aussi à l'intérieur de cette interface. Le langage JAL est une extension à Java. Aucune méthodologie n'est proposée. Les agents sont basés sur un modèle BDI. Aucun éditeur n'est disponible pour le développement ou le déploiement des systèmes. Jack est très long à maîtriser, il faut apprendre le langage JAL et connaître le modèle BDI. De plus, le manque de support graphique complique l'implémentation et le déploiement des systèmes. **[HEDJ, 03]**

- **Jade :** est un outil qui répond aux normes FIPA97. Jade fournit des classes qui implémentent « JESS » pour la définition du comportement des agents.

L'outil possède trois modules principaux (nécessaires aux normes FIPA). Le DF « Director Facilitor » fourni un service de pages jaunes à la plate-forme. L'ACC « Agent Communication Chanel » gère la communication entre les agents. L'AMS « Agent

Management System » supervise l'enregistrement des agents, leur authentification, leur accès et utilisation du système.

Les agents communiquent par le langage FIPA ACL. Un éditeur est disponible pour l'enregistrement et la gestion des agents. Aucune autre interface n'est disponible pour le développement ou l'implémentation. À cause de cette lacune, l'implémentation demande beaucoup d'efforts. Elle nécessite une bonne connaissance des classes et des différents services offerts. **[HEDJ, 03]**

- **MadKit :** est un environnement basé sur la méthodologie Aalaadin ou AGR (agent / groupe / rôle). L'outil fournit un éditeur permettant le déploiement et la gestion des SMA (G-box). La gestion faite via cet éditeur offre plusieurs possibilités intéressantes. L'outil offre aussi un utilitaire pour effectuer des simulations. **[HEDJ, 03].**

Conclusion :

Dans cette partie, nous avons essayé de présenter un état de l'art sur les notions théoriques en rapport direct avec le thème de notre mémoire. Cet état de l'art présente de manière plus ou moins détaillée des définitions et explications sur les concepts d'agents et de Système Multi Agent.

Et puisque les SMA servent aujourd'hui à toutes sortes d'applications dans divers domaines, nous citons à titre d'exemple la classification, la segmentation d'image, la reconnaissance de forme…

Et comme nous nous intéressant à la classification dans le secteur médical comme le titre l'indique le chapitre suivant présente en détail l'utilité des SMA dans la classification.

Chapitre III :

REALISATION

- **Introduction**
- **Environnement de développement**
- **Caractéristiques du prototype**
- **Réalisation du système multi agent**
- **Interfaces**
- **Conclusion**

Introduction :

Un projet informatique requiert la mise en œuvre de différentes technologies pour sa phase de développement. Cette partie présente en premier lieu les choix effectués pour la réalisation de notre système en termes de langages, d'un système de communication à travers une plateforme multi-agent et de bien d'autres composants indispensables. Ensuite, elle décrit son fonctionnement ainsi que la réalisation du Système Multi-Agent, et en dernier lieu les interfaces de l'application CDSMA.

I. Environnement de développement :

I.1 Choix de la plate-forme de développement des agents :

JADE (Java Agent Development Framework - Bellifemine, Poggi,Rimassa, 1999) est une plate-forme de développement d'agents gratuite et Open Source développée par CSELT (Groupe de recherche de Gruppo Telecom, Italie) et qui résulte principalement des activités de recherche.

Ces principales caractéristiques sont :

JADE simplifie l'implémentation d'un SMA à travers un MiddleWare répondant aux spécifications de la FIPA une librairie de classes que les utilisateurs peuvent utiliser et étendre ainsi un ensemble d'outils graphiques qui permettent le débuggage et l'administration du SMA à concevoir;

JADE assure une communication transparente par l'échange de messages dans le langage normalisé FIPA-ACL;

JADE diminue l'effort de programmation car elle implémente deux agents (DF et AMS) dont les fonctionnalités sont utiles à notre application;

JADE a comme but la construction des systèmes multi-agents et la réalisation d'applications conformes à la norme FIPA (FIPA, 1997). JADE comprend deux composantes de base : une

plate-forme agents compatible FIPA et un paquet logiciel pour le développement des agents Java

JADE offre la possibilité de créer des agents intelligents en délégant le raisonnement à JESS, où JESS est le moteur qui exécute tout le raisonnement nécessaire.

La norme FIPA pour les systèmes multi-agents :

Les premiers documents de spécification de la norme FIPA (FIPA 1997), appelés spécifications FIPA97, établissent les règles normatives qui permettent à une société d'agents d'inter-opérer. Tout d'abord, les documents FIPA décrivent le modèle de référence d'une plate-forme multi-agents (figure) où ils identifient les rôles de quelques agents clés nécessaires pour la gestion de la plate-forme, et spécifient le contenu du langage de gestion des agents et l'ontologie du langage.

Figure III.1 Le modèle de référence pour une plate-forme multi-agents FIPA

Dans la figure **III.1** nous voyons trois rôles principaux dans une plate-forme multi agents FIPA

- Le **Système de Gestion d'Agents** (Agent Management System - AMS) est l'agent qui exerce le contrôle de supervision sur l'accès à et l'usage de la plate-forme ; il est responsable de l'authentification des agents résidents et du contrôle d'enregistrements.

- Le **Canal de Communication entre Agents** (Agent Communication Channel - ACC) est l'agent qui fournit la route pour les interactions de base entre les agents dans et hors de la plate-forme ; c'est la méthode de communication implicite qui offre un service fiable et précis pour le routage des messages ; il doit aussi être compatible avec le protocole IIOP pour l'interopérabilité entre les différentes plates-formes multi-agents.
- Le **Facilitateur d'Annuaire** (Directory Facilitator - DF) est l'agent qui fournit un service de pages jaunes à la plate-forme multi-agents.

Chaque instance du JADE est appelée conteneur " container ", et peut contenir plusieurs agents. Un ensemble de conteneurs constituent une plateforme. Chaque plateforme doit contenir un conteneur spécial appelé main-container et tous les autres conteneurs s'enregistrent auprès de celui-là dés leur lancement

I.2 Choix du langage de programmation :

Afin de réaliser l'interface permettant aux utilisateurs (médecins) de manipuler le prototype, nous avons choisi le langage Java. Ce choix a été motivé par les raisons suivantes :

-Les agents développés sous la plate-forme JADE, sont entièrement écrits en Java.

Ce langage s'est donc imposé comme étant une conséquence de nos précédents choix en terme de plate-forme de développement du SMA (JADE);

-Java assure une totale indépendance des applications vis-à-vis de l'environnement d'exécution : c'est à dire que toute machine supportant Java est en mesure d'exécuter un programme sans aucune adaptation (ni recompilation, ni paramétrage de variables d'environnement);

-Une programmation orientée objet et une bibliothèque immense d'objets: dès sa naissance, les programmeurs de Sun ont doté leur langage d'une des plus grandes bibliothèques d'objets prêts à l'emploi;

- Un accès simplifié aux bases de données ;

En choisissant le langage Java, nous avions deux possibilités pour développer les interfaces du Systèmes Multi Agent. Java permet, en effet, de nous faciliter la création des agents et la distribution des différents classificateurs à ces agents.

En ce qui concerne l'environnement de développement, nous avons choisi NetBeans 6.9.1 qui est un environnement de développement intégré (EDI) pour le développement d'applications orientées objet.

I.3 Le choix de la base de données :

Notre prototype doit contenir une base de données a priori pour que les différents agents concernés par la classification passent par une phase d'apprentissage.

Notre BDD s'appelle PIMA (Maladie de Diabète).

C'est une base de données qui contient :

-Nombre d'instances : 768 cas.

-Nombre d'attributs : 8 descripteurs de diabète plus la classe

Chaque attribut représente un symptôme de cette maladie :

1. Nombre de grossesse
2. Le taux de glucose
3. La pression artérielle (mm Hg)
4. Epaisseur de pli de peaux (mm)
5. Le taux d'insuline (mu U/ml)
6. Index de masse corporelle
7. Facteur génétique
8. Age (an)
9. La classe (1 ou 0)

   Si classe =1 implique Présence de la maladie

   Si classe =0 implique Absence de la maladie

Dans cette base nous avons :

500 cas de classe 0

Et 268 cas de classe 1

I.4 Choix des classifieurs :

Parmi les problèmes de la classification il existe une autre grande catégorie de problèmes industriels qui consiste à attribuer de façon automatique un objet à une classe, parmi d'autres classes possibles. La résolution de ce type de problème demande de représenter les objets à l'aide de descripteurs, ou entrées. La représentation choisie doit être discriminante du point de vue de la classification désirée. Les valeurs des entrées peuvent être symboliques ou réelles.

Afin de résoudre ce problème et pour obtenir une bonne classification, nous avons choisi d'intégrer les classifieurs de WEKA dans notre prototype ; Ce choix a été motivé par les raisons suivantes :

Weka (Waikato Environment for Knowledge Analysis) est un ensemble d'outils permettant de manipuler et d'analyser des fichiers de données, implémentant la plupart des algorithmes d'intelligence artificielle, entre autres, les arbres de décision et les réseaux de neurones.

Il est écrit en java, disponible sur le web

Il se compose principalement :

– De classes Java permettant de charger et de manipuler les données.

– De classes pour les principaux algorithmes de classification supervisée ou non supervisée.

– D'outils de sélection d'attributs, de statistiques sur ces attributs.

– De classes permettant de visualiser les résultats.

On peut l'utiliser `a trois niveaux :

– Via l'interface graphique, pour charger un fichier de données, lui appliquer un algorithme, vérifier son efficacité.

– Invoquer un algorithme sur la ligne de commande.

– Utiliser les classes définies dans ses propres programmes pour créer d'autres méthodes, implémenter d'autres algorithmes, comparer ou combiner plusieurs méthodes.

C'est cette troisième possibilité qui sera utilisée dans notre prototype.

Figure III.2: Interface Graphique de Weka

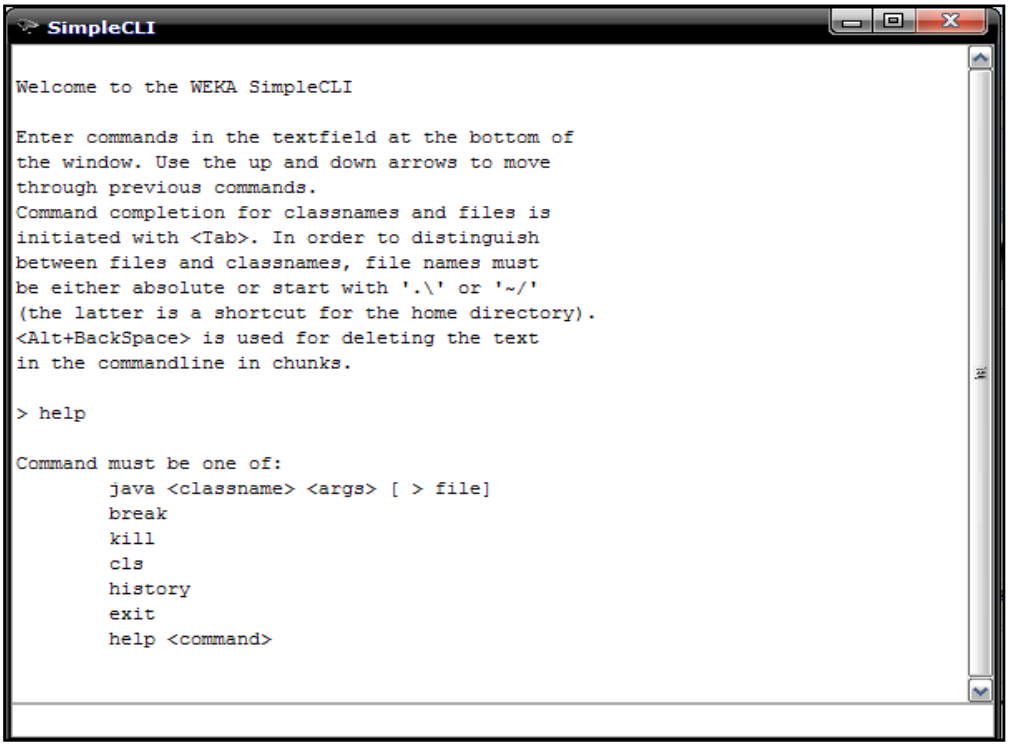

Figure III.3 : Ligne de commande de Weka

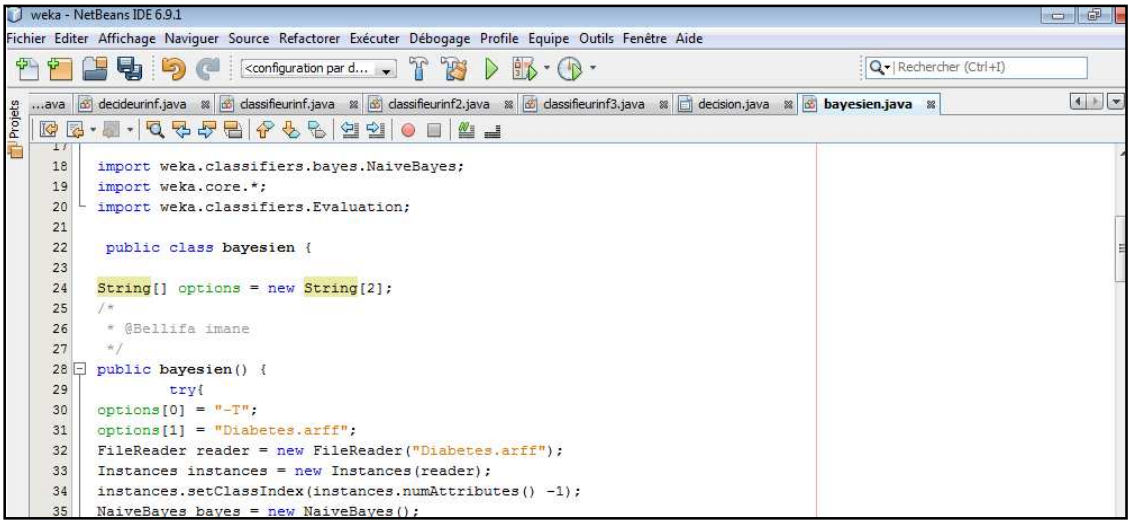

Figure III.4 : Weka intégré dans Java

Nous avons choisi le troisième mode d'utilisation (Figure III.4) pour les causes suivantes :

–Pouvoir mettre en œuvre les algorithmes en grandeur nature, sans devoir réécrire tout le code correspondant (quoiqu'une telle réécriture puisse s'avérer très fructueuse !).

–Comprendre et utiliser intelligemment les différentes sorties de ces algorithmes.

–Pouvoir programmer des agents *intelligents* en un temps raisonnable, pour des taches non triviales.

–Evaluer les performances d'un algorithme.

–Comparer les performances de deux ou plus d'algorithmes de classification.

Les 3 classifieurs présentés dans notre projet sont :

- Naive Bayesien : un modèle probabiliste
- C4.5 : Arbre de décision (J48)
- RBF : Réseau de neurone

Naive Bayesien : La classification Naive Bayesien est un type de classification Bayésienne probabiliste simple basée sur le théorème de Bayes avec une forte indépendance (dite naïve) des hypothèses. Elle met en œuvre un classifieur Bayesien naïf appartenant à la famille des classifieurs linéaires.

En terme plus simple, un classifieur Bayesien naïf suppose que l'existence d'une caractéristique pour une classe, est indépendante de l'existence d'autres caractéristiques.

Selon la nature de chaque probabiliste, les classifieurs Bayesiens naïfs peuvent être entrainés efficacement dans un contexte d'apprentissage supervisé.

C4.5 : L'algorithme C4.5 est un algorithme de classification supervisé.

Il est basé sur l'algorithme ID3 au quel il apporte plusieurs améliorations.

A partir d'un échantillon d'apprentissage composé d'une variable objectif Y et d'au moins une variable d'apprentissage X={x1, x2 …xn} C4.5 produit un modèle de type arbre de décision. Cet algorithme se base sur une mesure de l'entropie pour produire le modèle.

RBF : le réseau RBF (Radial Basis Functions) fait partie des réseaux de neurones supervisés introduit par Powell et Broomhead . Il est constitué de trois couches : une couche d'entrée qui retransmet les entrées sans distorsion, une seule couche cachée qui contient les neurones RBF qui sont généralement des Gaussiennes et une couche de sortie dont les neurones sont généralement animés par une fonction d'activation linéaire.

Chaque couche est complètement connecter à la suivante et il n'y a pas de connexions à l'intérieur d'une même couche.

II. Les caractéristiques du prototype

Notre prototype possède les caractéristiques suivantes :

- **transparence :** un utilisateur du prototype doit pouvoir accéder à tout ce qui lui est légalement possible, en termes d'information. Même lorsque le système réalise automatiquement des opérations, il informe l'utilisateur et il lui conserve une trace;
- **adaptation :** notre système est évolutif et adaptatif, cela signifie en particulier qu'au fur et à mesure de son utilisation il se conforme aux comportements globaux des usagers supposant des capacités d'apprentissage.
- **La portabilité :** du fait qu'il soit implémenté en Java, CDSMA est avant tout un système multi-plateforme, c'est-à-dire qu'il peut être exécuté sur plusieurs systèmes d'exploitation et ne nécessite aucune recompilation lors du transfert d'un environnement vers un autre.

III. Réalisation du système multi Agent :

III.1 Les Agents de CDSMA :

Comme mentionné précédemment, la plate-forme Jade nous a permis de développer nos agents, ces derniers sont réalisés grâce à l'importation de la classe « jade.core.Agent».

Sous forme de classe Java, nos agents contiennent plusieurs méthodes les caractérisant, entre autres la méthode « **Setup ()** » obligatoire pour l'initialisation de l'agent, et « **TakeDouwn()** » appelée à la fin de l'exécution d'un agent pour se désinscrire.

Afin d'implémenter les comportements des agents, nous avons défini des objets de la classe Behaviour « import jade.core.behaviours.Behaviour », chaque objet de ce type dispose d'une méthode « **action ()** » qui constitue le traitement à effectuer par celui-ci, ainsi que d'une méthode « **done ()** » qui vérifie si le traitement est terminé.

De plus, ces comportements sont classés en trois catégories :

OneShotBehaviours, CyclicBehaviour et SimpleBehaviours

- Les SimpleBehaviours: exécutent différentes opérations dépendamment d'un état ; elles se terminent quand une certaine condition est satisfaite ;
- Les CyclicBehaviour : ne sont jamais terminés et exécutent la même opération à chaque fois qu'ils sont appelés ;
- Les OneShotBehaviours : sont exécutés une seule fois et se terminent immédiatement après.

Afin de mieux comprendre ces notions, nous vous proposons le fragment de code qui suit, illustrant l'AgentMédecin(Decedeur) implémenté en classe Java étendue Agent :

```
public class decideur extends Agent {
/*création de la classe étendu Agent*/
public Params1;
private Params2;
/*quelques attributs de la classe decideur*/
public void setup() {
/*Initialisation de l'Agent*/
addBehaviour (new RecoitInfo());
/*Implémentation d'un CyclicBehaviour qui prend en charge la réception
des Messages contenants les informations concernant les résultats den
classification*/
addBehaviour (new Decision());
/*Ajout d'un SimpleBehaviour qui décider le meilleur résultat sur les
autres résultats de classification,et qui s'achève suivant le done()* /
}
```

Voici le Bihaviour (Comportement) qui reçoit les informations provenant de chaque classifieurs (NB, RBF, C4.5)

```
public class RecoitInfo extends CyclicBehaviour {
private MessageTemplate msg;
public void action() {
/ *ouverture de la méthode action contenant les tâches à effectuer*/
/ *réception d'un message qui contient les paramètres saisie par le médecin*/
Msg.setContent(parametre); //Envoi ces données à tous les classifieurs pour
faire une classification pour ce patient
myAgent.send(msg); //Envoi ces données à tous les classifieurs pour faire une
classification pour ce patient
MessageTemplate msg =MessageTemplate.MatchPerformative(ACLMessage.REQUEST);
//Attente les trois réponse des autres Agents
} // Fin action ()
} // Fin CyclicBehaviour RecoitInfo
```

Lorsque L'agent Décideur il reçoit toutes les résultats nécessaires pour prendre une décision alors le <<Behaviour Decision>> se déclenchera de la même façon décrit en dessous

III. 2 La communication entre Agents :

Les performatifs ACL utilisés

Notre système exige des moyens de communication de haut niveau, dans le sens où les messages échangés sont porteurs d'informations que les agents doivent être aptes à interpréter afin d'effectuer les actions requises.

Le langage FIPA ACL permet la mise en œuvre de conversations où l'émetteur peut exprimer son intention à travers l'envoi d'un message. Ceci est rendu possible par l'utilisation des performatifs adéquats.

Dans le cadre du fonctionnement de notre système, tous les actes de communications entre agents visent à permettre à l'émetteur de prendre une décision finale et la rendre visuel à l'utilisateur qui est dans notre cas le médecin.

Parmi la large librairie de performatifs proposée par ACL, nous n'avons eu à utiliser que les deux performatifs suivants :

- REQUEST: permet à l'agent émetteur d'exprimer qu'il attend de l'agent destinataire une réponse suite à une requête contenue dans le message.
- INFORM : permet à l'émetteur d'exprimer que le contenu du message est un fait dont il souhaite informer le destinataire. Ce performatif est utilisé en réponse à un message contenant le performatif « REQUEST ».

IV. Le fonctionnement de CDSMA

Dans ce qui suit, une description du fonctionnement des agents sera effectuée à travers les principaux comportements impliqués dans le cadre de classification d'une nouvelle instance et le lancement d'un résultat final.

• L'Agent Decideur reçoit un message de l'utilisateur (Médecin) contenant les 8 descripteurs de Diabète en exécutant le CyclicBehaviour : « ReçoitInfo » et il l'envoit à chaque classifieur.

• L'Agent Classifieur1 fait une classification de cette nouvelle information (selon le type de comportement de l'agent qui est dans ce cas Naive Bayes) qui sont les différents symptômes de diabètes d'une nouvelle instance en déclenchant un CyclicBehaviour ClassifieurNaiveBayes

- L'Agent Classifieur2 fait une classification de cette nouvelle information (selon le type de comportement de l'agent qui est dans ce cas C4.5) qui sont les différents symptômes de diabètes d'une nouvelle instance en déclenchant un CyclicBehaviour ClassifieurC4.5
- L'Agent Classifieur3 fait une classification de cette nouvelle information (selon le type de comportement de l'agent qui est dans ce cas RBF) qui sont les différents symptômes de diabètes d'une nouvelle instance en déclenchant un CyclicBehaviour ClassifieurRBFNetwork

• ChaqueAgent Classifieur (1et 2 et 3) envoi le résultat de classification après transformation en langage ACL grâce au Behaviour « EnvoiInfoR » qui informe l'AgentDecideur des résultats de classification.

• L'AgentDecideur reçoit les résultats de la classification au CyclicBehaviour : « ReçoitResult »

Cela explique comment un agent réagi lors la réception d'une nouvelle information et la transformer à l'AgentDecideur qui va donner une décision finale après la réception de tous les résultats obtenus par les différents Agents Classifieurs (Naive Bayes, RBF, C4.5) et la rendre visuelle à l'utilisateur via une interface graphique en affichant patient : Malade ou Non malade.

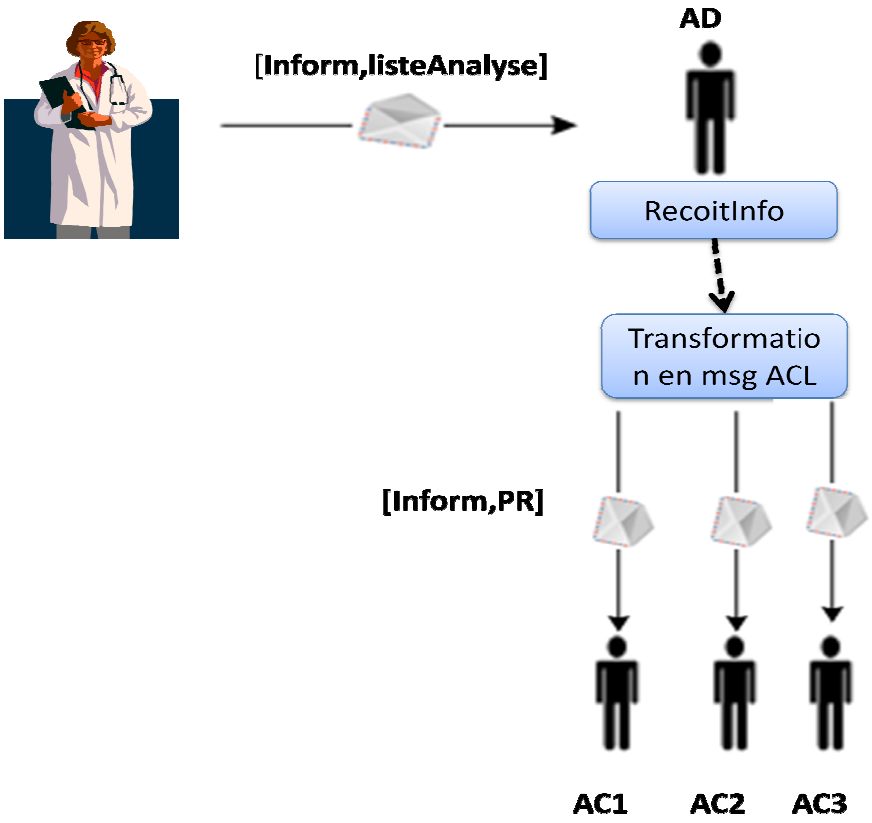

Figure III.5 : Behaviours impliqués lors l'introduction d'une ressource.

Lancement de vote pour prendre une décision finale :

• L'Agent Classifieur1 2 et 3 lance une classification après recevoir les informations nécessaires et obtenir des résultats ensuite chacun d'eux envoi son résultat par un message à l'AgentDecideur.

• L'AgentDecideur reçoit le message en exécutant le CyclicBehaviour : « ReçoitResultat », analyse les résultats de chaque classifieurs et faire un vote pour faire un choix final et envoi la réponse (malade ou non malade) qui sera en suite envoyer par cet Agent à l'utilisateur (médecin).

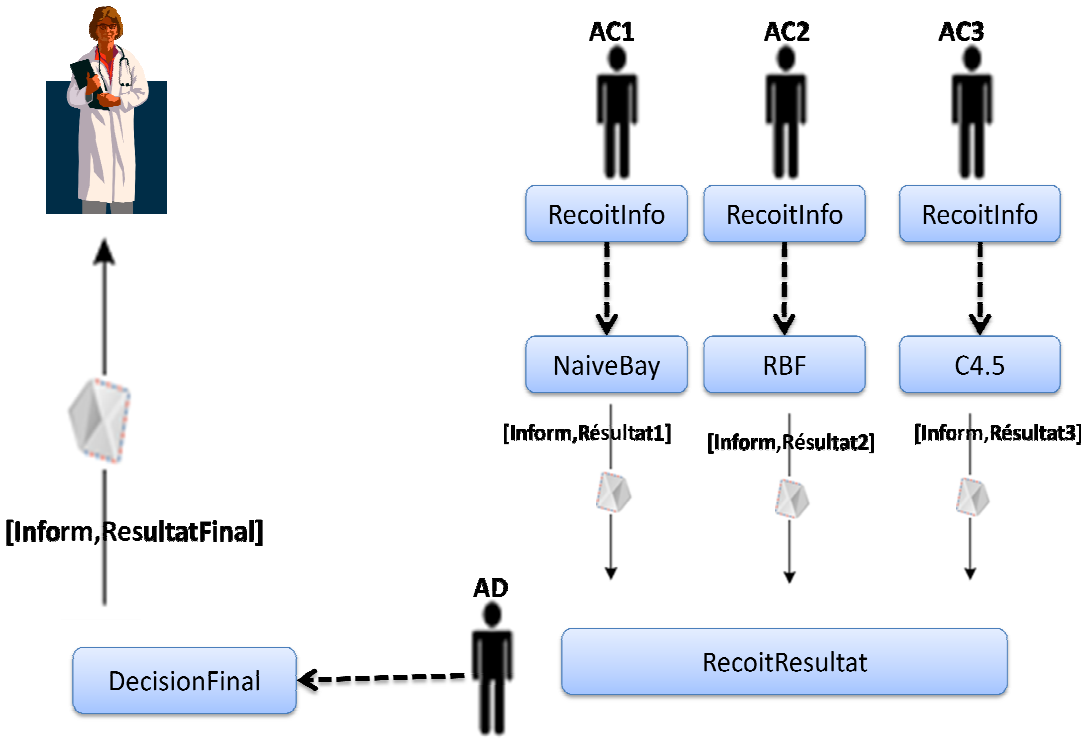

Figure III.6 : Behaviours impliqués lors lancement d'un vote pour faire un meilleurs choix.

V. Les interfaces

Dans ce qui suit, nous présentons le fonctionnement de notre système en présentant ses différentes interfaces.

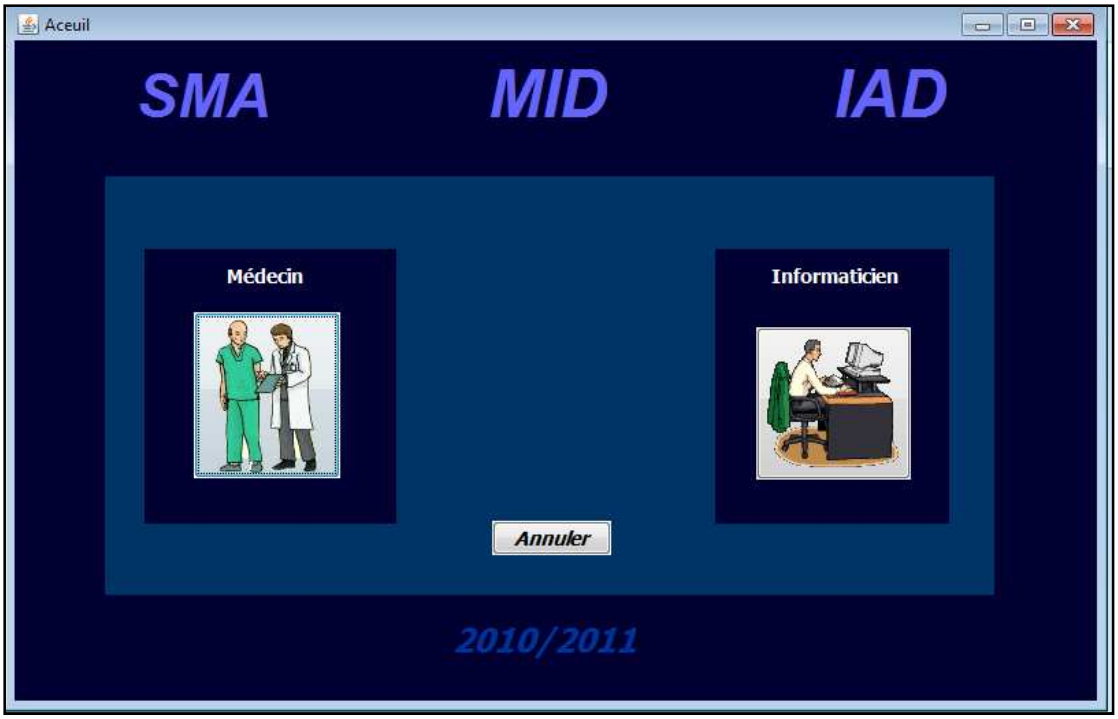

Figure III.7 : Interface Accueil de CDSMA

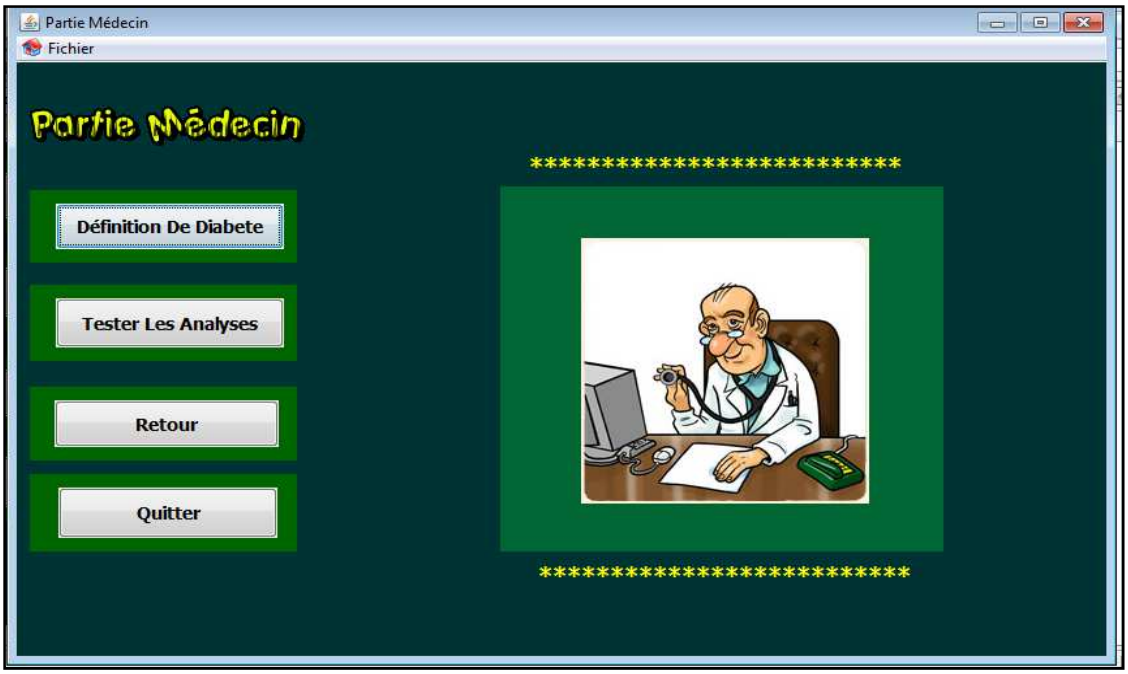

Figure III.8 : Interface Partie Médecin

Figure III.9 : Interface consultation

Figure III.10 : Interface affichage résultat de consultation

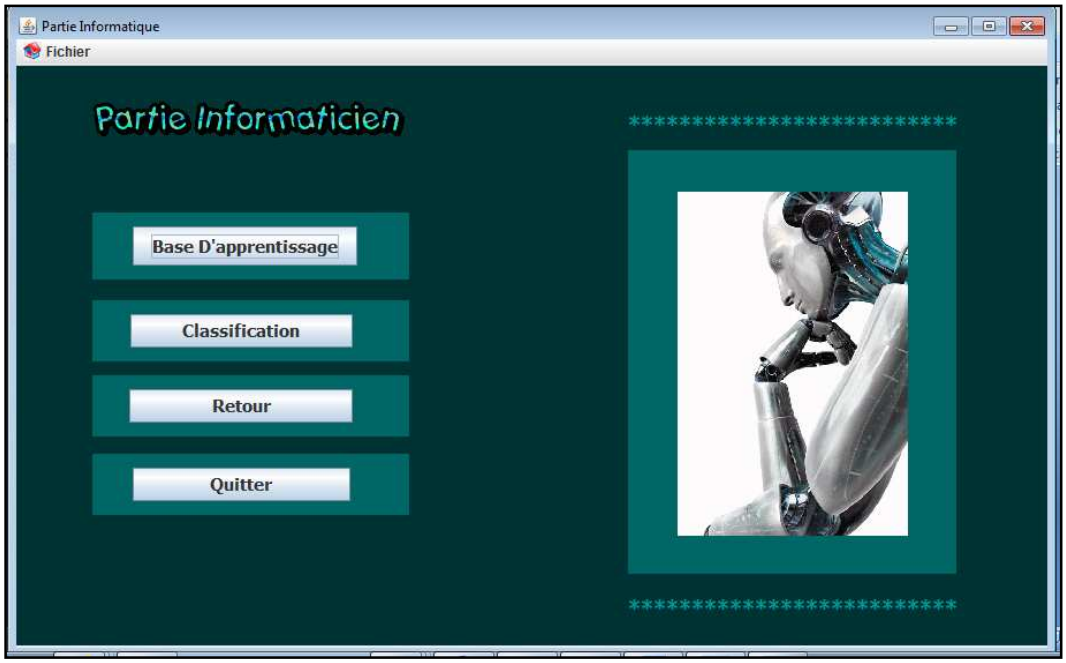

Figure III.11 : partie informatique

Figure III.12 Interface Classification Intelligente

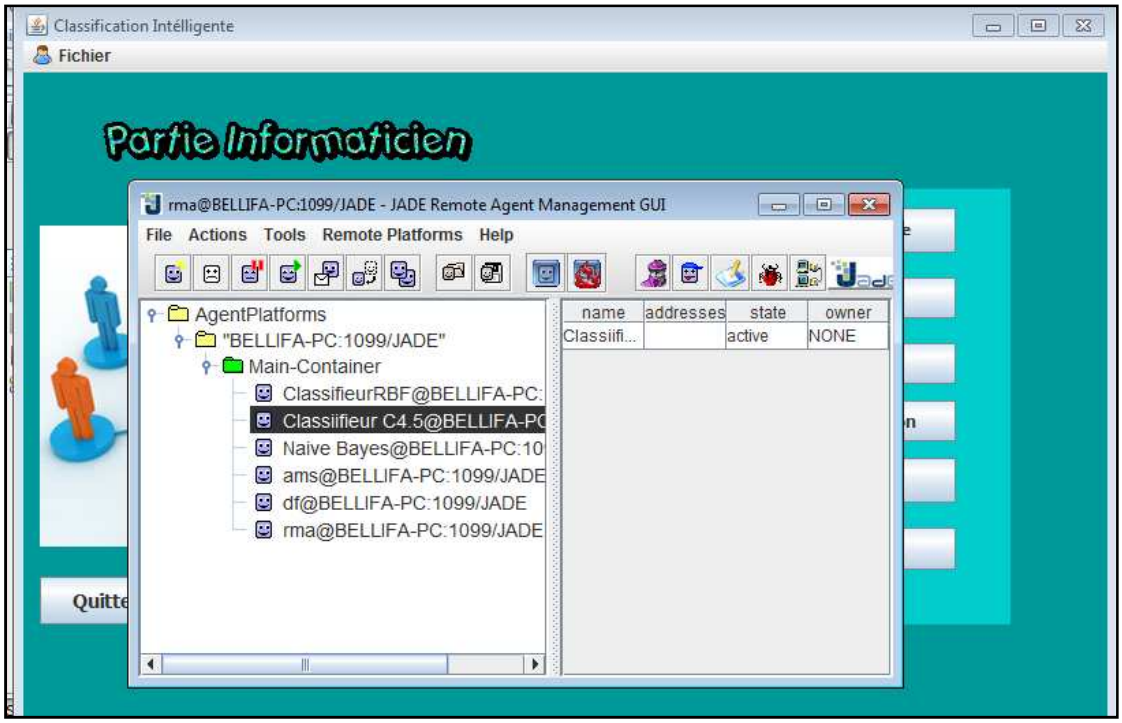

Figure III.13 : Interface Jade pour visualiser les agents

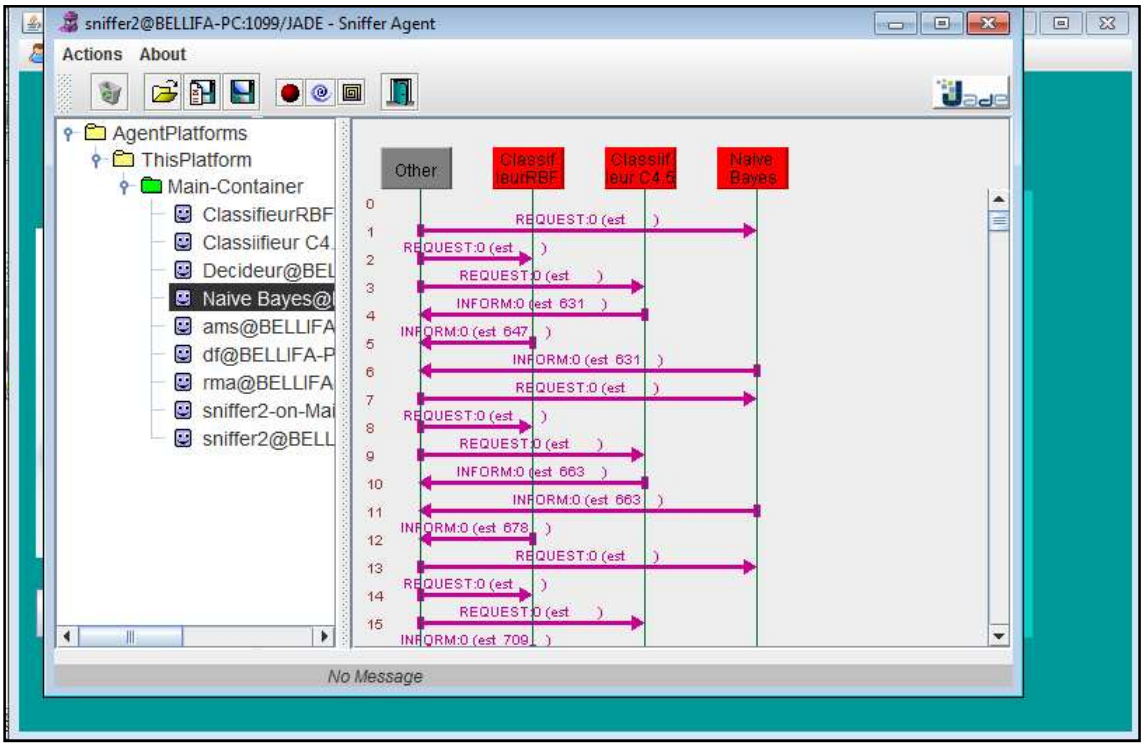

Figure III.14 Interface Sniffer pour visualiser les messages

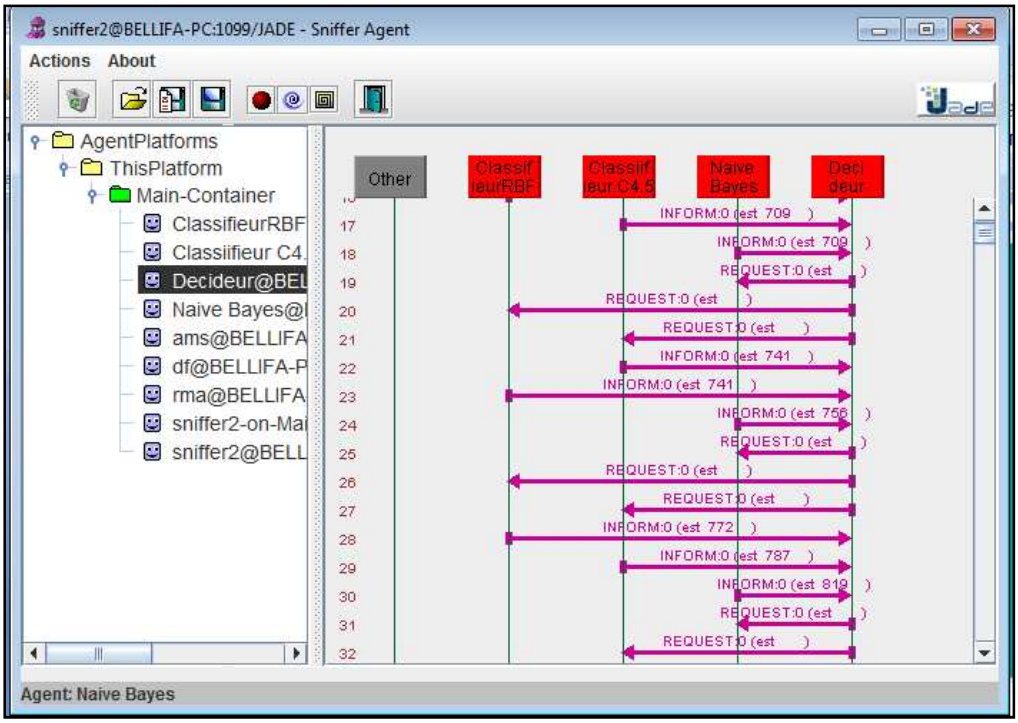

Figure III.15 : Interface Sniffer pour visualiser les messages entre agents classifieurs et l'agent Décideur

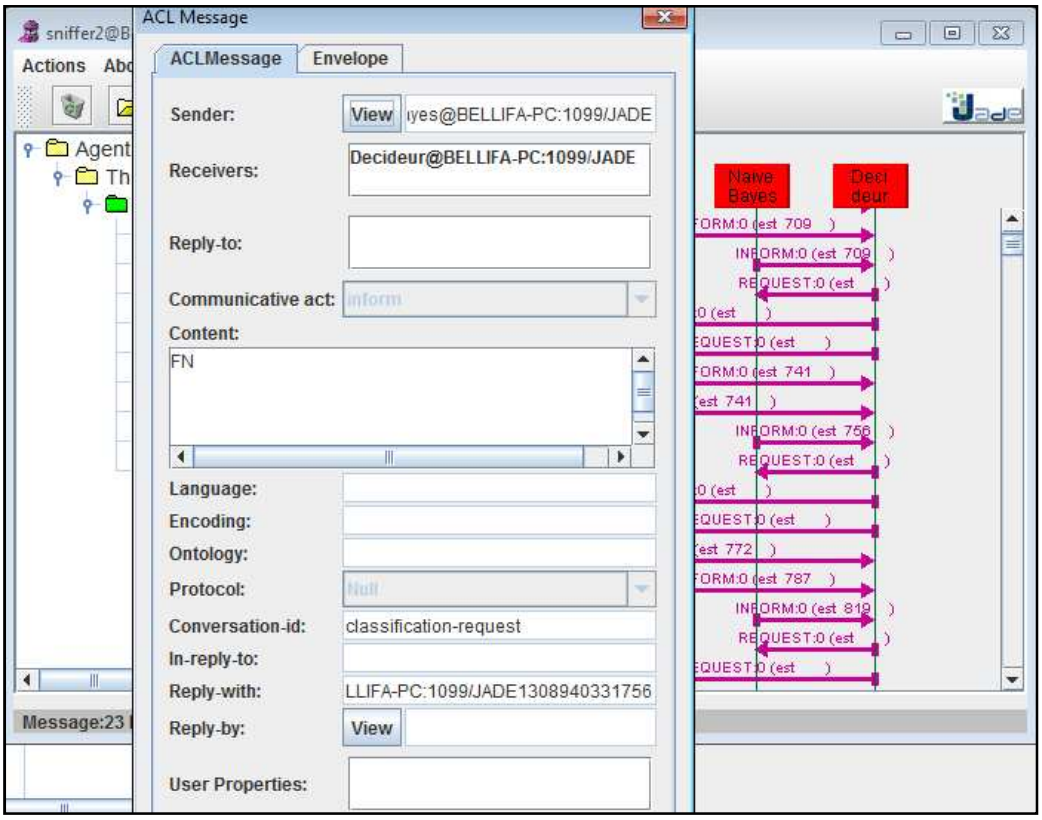

Figure III.16 : Visualisation d'un message ACL

Phase d'évaluation des résultats :

Pour évaluer notre système nous sommes basées sur les notions suivantes :

La sensibilité, la spécificité et le taux de classification correcte ;

SE=VP/VP+FN ;

SP=VN/VN+FP ;

CC=VN+VP/ (VN+VP+F N+FP) avec :

VP : Nombres positifs classés positivement

VN : Nombres négatives classés négativement

FP : Nombres négatives classés positivement

FN : Nombres positifs classés négativement

Par exemple dans notre cas

Si l'attribut a comme classe 1 et le classifieur reconnait la bonne classe 1 alors c'est un VP

Si l'attribut a comme classe 0 et le classifieur reconnait la bonne classe 0 alors c'est un VN

Si l'attribut a comme classe 0 et le classifieur affecte la classe 1 au lieu de 0 alors c'est un FP

Si l'attribut a comme classe 1 et le classifieur affecte la classe 0 au lieu de 1 alors c'est un FN

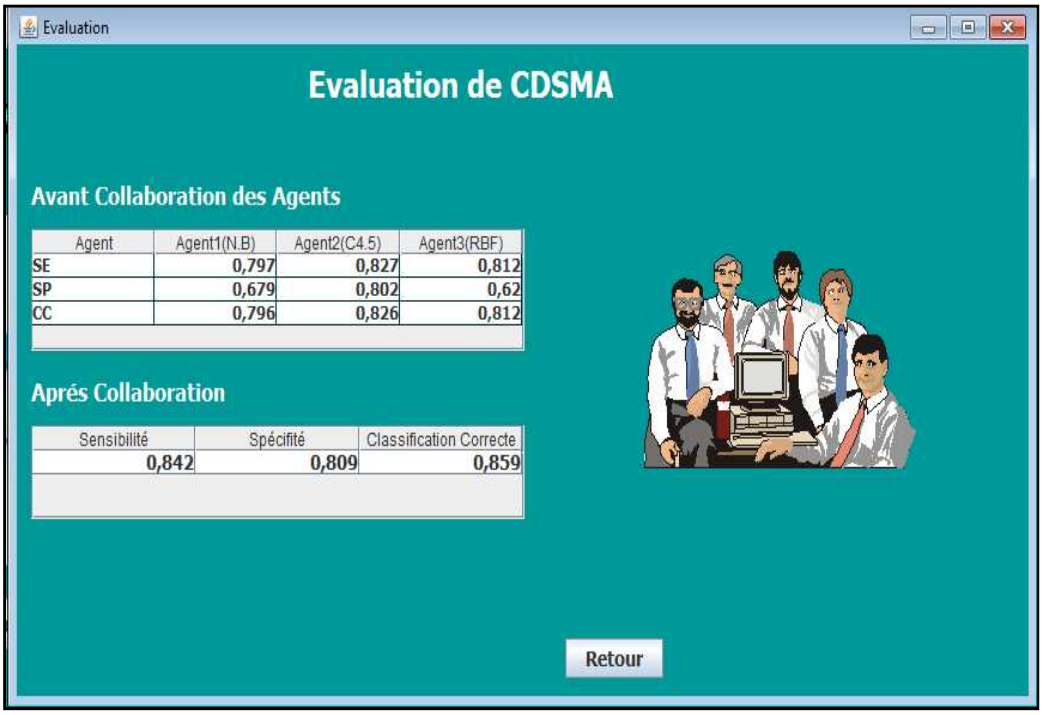

Figure III.17 : Interface évaluation de CDSMA

Le premier tableau concerne la classification des attributs par chaque Agent Classifieur avant la communication et la collaboration entre eux ; d'après les résultats obtenus nous avons remarqué que le meilleur classifieur dans notre système est l'Agent 2(arbre de décision C4.5) avec un taux de reconnaissance égale à 82% ; en deuxième position l'Agent 3(RBF) avec un CC=81% et en dernière position l'Agent 1 (Naïve Bayes) avec un CC=79%

Pour faire une comparaison entre un travail réaliseé dans un système mono Agent et un système Multi Agent nous avons crée un Agent DecideurINF qui va demander aux différents Agents les VP, VN, FP et FN et lui va calculer le CC, SE, SP et tous ça se réalise via une communication entre l'Agent DecideurINF et les Agents Classifieurs

Pour mieux comprendre nous allons voir tous ça dans la plate-forme Jade à l'aide de Sniffer qui nous permet de visualiser les messages envoyer entre ces Agents lors de leur communication

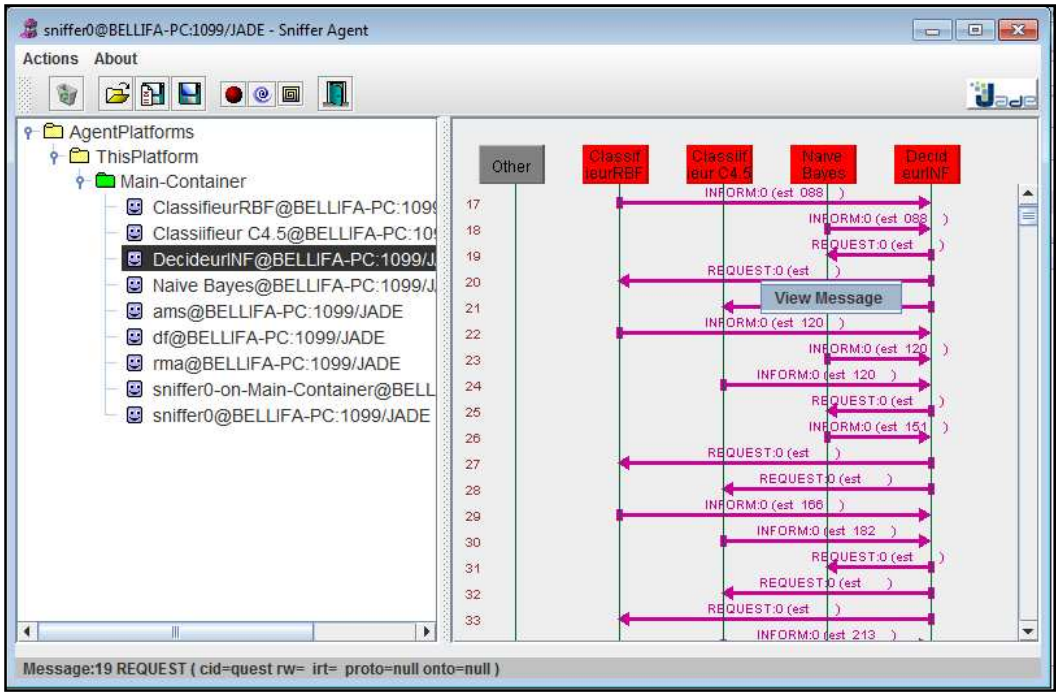

Figure III.18 : Interface Sniffer pour visualiser les messages entre agents classifieurs et l'agent DécideurInf

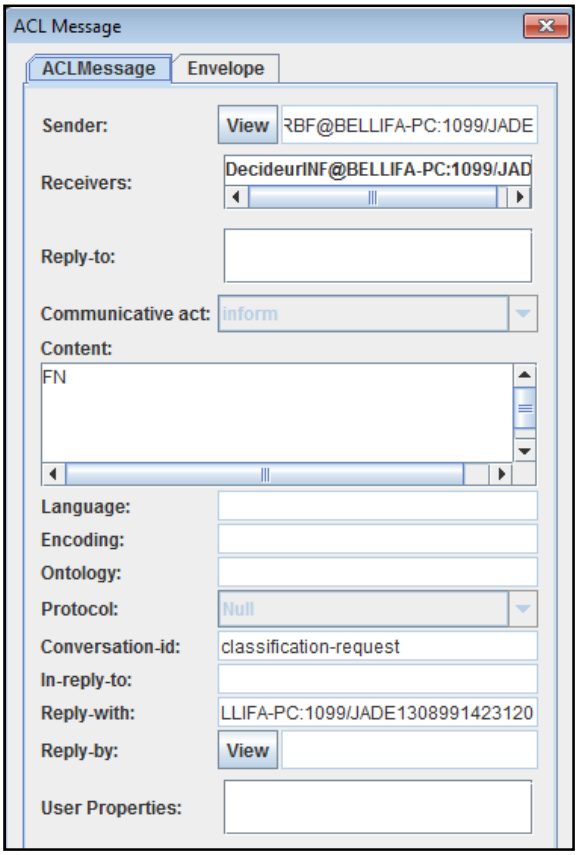

Figure III.19 : Interface Sniffer pour visualiser les messages agent RBF et l'agent DécideurInf

Comme Exemple voici un message ACL envoyé d'Agent Classifieur (RBF) a l'Agent DecideurINF dont le contenu est un FN pour l'instance 20

Apres la classification de tous les Agents l'agent DecideurINF évaluera le système comme suit :

Le taux de classification correcte CC=85.9%

La sensibilité SE=84.2%

La spécificité SP=80%

Conclusion

Dans ce travail, nous avons présenté une application baptisée « CDSMA » exploitant la technologie multi-agent, pour une classification d'une base de données dans le domaine médical.

Initialement la problématique de classification était de classifier une nouvelle instance
Dans notre solution, nous avons évolué vers une distribution de paramètres (Symptôme de diabète) en se basant sur des entités réelles appelées agent qui peuvent communiquer entre eux, cela a donné la naissance à ce que nous avons appelé « la communication Collaborative assistée par un «Système Multi-Agent ».

Vu le nombre des personnes diabétiques qui ne cesse d'augmenter dans le monde, et sachant les complications résultantes de cette maladie, une prise en charge rigoureuse de cette maladie est devenue une priorité absolue. Afin d'apporter une solution efficace que peut renforcer le diagnostic, nous proposons dans ce travail de master une approche intelligente et distribuée basée sur les systèmes multi-agents.

Les Systèmes multi-agent sont utilisés aujourd'hui dans plusieurs applications, nous citons à titre d'exemple la classification des données, la reconnaissance de formes, la segmentation d'image…etc.

Nous avons évalué notre modèle sur une base de données diabétique (PIMA INDIEN), les résultats obtenu ont été très prometteurs et montrent l'intérêt des modèles dits intelligents dans le domaine médical. Comme un travail futur nous proposons l'hybridation des techniques utilisées avec la logique floue afin d'augmenter l'interprétabilité du modèle classifieur, ainsi nous proposons la possibilité d'enrichir la base de données d'apprentissage afin d'augmenter la précision des résultats.

BIBLIOGRAPHIE

[Bourr] :T. Bourron. Application des systèmes multi-agents dans les télécommunications : États de l'art, enjeux et perspectives.

[Bro89] R. A. Brooks. A Robot that Walks: Emergent Behaviors From a Carefully Evolved Network. A.I. Memo 1091, Massachusetts Institute of Technology, 1989.

[CAG, 98]: A. CAGLAYAN, C. HARRISON « *Les Agents, Applications bureautiques, Internet, Intranet* ». InterEditions, 1998.

[CHD, 02]: B. CHAIB-DRAA, I. JARRAS, « *Aperçu sur les systèmes multi agents* », Série Scientifique, Montréal, 2002.

[DAV, 80]: Davis. Report on the Workshop on Distributed AI. SIGART Newsletter 73, pages 42-52, 1980

[DEM, 95]: Y. DEMAZEAU, « *From Interactions to Collective Behaviour in Agent-Based Systems* », Proceeding of the First European Conference on
Cognitive Science, Saint-Malo, p. 117-132, 1995.

[Dieng90]: R. Dieng. Relations Linking Cooperating Agents. In Proceedings of the 2nd European Workshop MAAMAW'90, pages 185-202, Saint-Quentin en Yvelines-France, August 1990.

[DUR, 90]: E. H. Durfee and T. A. Montgomery. A Hierarchical Protocol for Coordinating Multiagent Behaviors. In Proceedings of the 8th National Conference on Artificial Intelligence, pages 86-93, Cambridge, July- August 1990. Volume One

[Etz94] Etzioni O., Weld D., An Softbot-based interface to the Internet. *Communication of the ACM*, 1994, 37, 7, pp. 72-76.

[FEH, 83]: M. Fehling and L. Erman. Report on the Third Annual Workshop on DAI. SIGART Newsletter 84, pages 3-12, April 1983.

[FER, 88]: J. Ferber and M. Ghallab. Problématiques des univers Multi-Agent intelligents.

[Ferb95] : J.Ferber, Les Systèmes Multi-Agents. Vers une intelligence collective. InterEditions, 1995

[FER, 97]: J. FERBER, « *Les Systèmes Multi‑Agents : vers une Intelligence Collective* » Inter-éditions, 1997.

[FIN, 93]: T. FININ, G. WIEDERHOLD, "A*n Overview of KQML: A nowledge Query and Manipulation Language*", Department of Computer Science, Stanford University, 1993.

BIBLIOGRAPHIE

[FIN, 94]: T. FININ, R. FRITZON, D. MCKAY, R. MCENTIRE, "*KQML as an agent communication language*", In Proceedings of the Third International
Conference on Information and Knowledge Management (CIKM'94), ACM Press, 1994.

[FIN, 94b]: T. FININ, R. FRITZON, D. MCKAY, R. MCENTIRE. "*KQML - A Language and Protocol for Knowledge and Information Exchange*", (Technical Report No. CS-94-02). University of Maryland, Department
of Computer Science, 1994.

[FIN, 94c]: T. FININ, J. WEBER, G. WIEDERHOLD, M. GENESERETH
R. FRITZON, J. McGuire, S. Shapiro, D. McKay, R. Pelavin, C. Beck, "*Specification of the KQML Agent - Communication Language plus example agent policies and architectures (DRAFT Report)*", DARPA Knowledge Sharing Initiative External Interface Working Group, 1994.

[HUH, 87]: M. N. Huhns. Distributed Artificial Intelligence. Pitman Publishing-
Morgan Kaufman, 1987.

[Jenn98] N. R. Jennings, M. Wooldridge, and K. Sycara. A roadmap of agent research and development. Int Journal of Autonomous Agents and Multi-
Agent Systems, 1(1):7- 38, 1998.

[JW, 98]: Nicholas R. Jennings and Michael J. Wooldridge, Applications of intelligent agents, Agent Technology Foundations, Applications, and Markets , Springer-Verlag, 1998.

[MAE, 95]: Maes P., Intelligent Software, Scientific American, volume 273, numéro3, septembre 1995, pp. 84-86.

[Nwa96]: Nwana H. S., Software Agents: An Overview. In Knowledge Engineering Review, Vol. 11(3), pp.205-244, 1996.

[RGK97] Rus D., Gray R., Kotz D., Transportable Information Agent. *Journal f Intelligent Information systems.* 9, 3, pp. 215-238, 1997

[Rho00] Rhodes J., Just-In-Time Information Retrieval. Ph.D. Thesis, MIT Media Lab, May 2000.

[SEC, 03]: Y. SECQ, « *RIO : Rôles, Interactions et Organisations une méthodologie pour les systèmes multi - agents ouverts* », Thèse Doctorat, Université des
Sciences et Technologies de Lille, France, 2003

BIBLIOGRAPHIE

[TrEsp99] : Tranvouez E., Espinasse B., Protocoles de coopération pour le réordonnancement d'atelier. Ingénierie des Systèmes Multi-Agent (Actes des Journées Francophones de l'Intelligence Artificielle Distribuée et des Systèmes Multi-Agents), (Novembre 1999 ; Saint Gilles), JFIADSMA'99, Ed. par Marie ierre Gleizes, Paris : Hermes, 1999.

[Web 1]: http://www.gsk.fr/gsk/votresante/diabete/definition.html

[Web 2]: http://obnet.chez-alice.fr/p0461.htm

Résumé :

Notre travail consiste à implémenter un modèle intelligent basé sur les Système Multi Agents, chaque agent est basé sur une technique dite intelligente (réseaux bayésiens, arbre de décision et les réseaux de neurones à base radiale). Nous avons évalué notre approche sur une base de données médicale, les résultats obtenus constituent un début vers la réalisation d'un système d'aide au diagnostic médical intelligent et robuste, ce dernier peut aider les experts du domaine médicale à prendre la meilleure décision.

Mots clés : Système Multi Agents, modèle intelligent, arbre de décision, réseaux bayesiens, réseaux de neurones

Abstract:

Our job is to implement an intelligent model based on MAS, each agent is based on an intelligent technique (Bayesian networks, decision trees and neural networks radial basis). We evaluated our approach on a medical database, the results are a start towards achieving a system of medical diagnosis using intelligent and robust, and it can help medical experts to make the best decision.

Keywords: multi-agent system, intelligent design, decision trees, Bayesian networks, neural networks

ملخص:

مهمتنا هي لتطبيق نموذج يستند إلى عامل الذكاء المتعدد النظام ، ويقوم كل وكيل على تقنية ذكية (شبكات بايزي والأشجار القرار والشبكات العصبية)
قمنا بتقييم نهجنا على قاعدة بيانات طبية، فإن النتائج هي نقطة البداية نحو تحقيق نظام التشخيص الطبي باستخدام نماذج ذكية وقوية، وأنها يمكن أن تساعد الخبراء الطبيين لتقديم أفضل قرار.

الكلمات الرئيسية: نظام متعدد وكيل. وتصميم ذكي والأشجار المقرر ، بايزي شبكات، والشبكات العصبية