N° d'ordre:

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université des Sciences et de la Technologie Houari Boumediene

Faculté d'Electronique et d'Informatique
Département d'informatique

MEMOIRE

Présenté pour l'obtention du Diplôme de
Magister en Informatique
Spécialité: Intelligence Artificielle et Bases de Données Avancées

Par
KOUIDER EL OUAHED Abdellah

Sujet

Interactions dans les systèmes multi-agents

Développement d'une plateforme multi-agents basée sur le modèle d'interaction dynamique

Date de soutenance: le 13 février 2006

Devant le Jury:

Président:	Mme BOUKALA, maître de conférences à l'USTHB
Examinateurs:	Mme SOUAMI, maître de conférences à l'USTHB
	Mr ALI BENAMARA, maître de conférences à l'Université de Chlef
Directeur:	Mr H. AZZOUNE, maître de conférences à l'USTHB

Résumé

Dans le cadre de ce mémoire nous nous intéressons à l'interaction dans les systèmes multi-agents (SMA). Après une étude bibliographe réalisée sur le paradigme des systèmes multi-agents, les modèles d'interactions et les plateformes multi-agents, nous avons adopté le modèle d'interaction dynamique pour le développement d'une plateforme multi-agents.

Le Modèle d'Interaction Dynamique (MID) est un modèle où le mécanisme d'interaction joue un rôle dynamique; c'est-à-dire qu'une partie de la gestion de l'interaction est déplacée de l'agent vers le mécanisme d'interaction. Ce transfert se fait sous forme d'une délégation des tâches, concernant l'interaction, qui sont indépendantes du domaine d'application du système multi-agents. Cette répartition permet le développement des interactions indépendamment des agents et laisser les agents se concentrer sur leurs tâches.

La mise en œuvre de ce modèle d'interaction est effectuée en réalisant une plateforme multi-agents basée sur ce modèle. Cette plateforme est développée selon la norme FIPA (Foundation for Intelligent Physical Agent), offre un ensemble de services permettant la mise en œuvre d'applications SMA.

A son état de prototype, cette plateforme, en plus de la mise en oeuvre du modèle MID, elle nourrit trois objectifs : elle constitue une base logicielle expérimentale, en évolution permanente, pour les besoins de recherche de l'équipe, elle participe également à l'intégration des techniques multi-agents dans des projets scientifiques et elle servira comme support pour la mise en œuvre des applications SMA.

Enfin, certaines applications SMA considérées comme Benchmark, ont été mises en œuvre pour tester les performances de cette plateforme et valider ses capacités d'adaptations et réutilisation.

Nous finissons ce travail par une conclusion générale où nous présentons les conclusions de cette étude et les perspectives.

Mots clés: Système multi-agents, Interactions, Modèle d'interaction, Plateforme multi-agents.

Remerciements

Je tiens à remercier tout particulièrement les personnes ayant permis l'aboutissement de ce travail :

Je remercie donc vivement Hamid AZZOUNE, Maître de conférences à l'USTHB, de m'avoir proposé ce sujet et de m'avoir encadré. Je remercie, également, Ahmed HARBOUCHE, Maître assistant chargé de cours à l'université de Chlef, pour ses nombreux conseils. Je les remercie de leur disponibilité, de leur réactivité à la lecture et à la correction de mes documents, et de leur soutien.

Mes remerciements iront naturellement vers tous ceux qui ont accepté avec bienveillance de participer au jury:

Je remercie Mme BOUKALA, Maître de conférences à l'USTHB, d'avoir accepté de présider le jury. Je remercie, Melle SOUAMI, Maître de conférences à l'USTHB, d'avoir accepter de participer au jury. Je salue également, A. ALI BENAMARA, Maître de conférences à l'université de Chlef, de sa participation au jury et d'avoir manifesté un grand intérêt à mes travaux et pour son soutien.

J'adresse mes remerciements à toutes les personnes de l'équipe IMAS de l'université de Chlef, qui ont contribué de près ou de loin à ce travail. Je tiens à remercier en particulier mes étudiants PFE cinquième année, d'avoir contribuer à ce travail.

Enfin un grand merci à tous mes amis et mes collègues au département d'informatique, université de Chlef, qui m'ont encouragé de près ou de loin pendant la fin de ce travail, je pense particulièrement à tous les membres de ma famille.

Sommaire

Table des figures

Partie I:

Etat de l'art sur les systèmes Multi-agents

Chapitre 1

Introduction générale

Dans ce chapitre, on introduit le contexte dans lequel se situe ce mémoire, à savoir les systèmes multi-agents et les environnements de développements multi-agents. Nous présentons ensuite l'objectif de cette étude. Enfin, nous présentons la démarche suivie et l'organisation du manuscrit.

L'intelligence artificielle classique (IA) consiste à modéliser le comportement intelligent d'un seul agent. L'intelligence artificielle distribuée (IAD) s'intéresse à des comportements intelligents qui sont le produit de l'activité coopérative de plusieurs entités autonomes.

La résolution d'un problème en IAD se fait grâce à ces entités qui se coordonnent et interagissent afin de tirer le meilleur de leurs caractéristiques et de leurs savoir-faire individuels pour une résolution collective du problème.

La résolution ne consiste pas seulement à l'exécution d'une entité informatique, mais d'un système composé de plusieurs entités informatique. De plus, cette résolution ne se contente pas d'être la somme des résolutions partielles de chaque entité composant le système mais de combiner ces résolutions par des interactions entre les entités.

Les principes de base de l'IAD consistent donc en la distribution des connaissances et des informations nécessaires à la résolution d'un problème parmi les différentes composantes en interactions d'un système. Les composantes de ces systèmes sont des entités plus au moins intelligentes désignées par le terme « agent ».

L'IAD regroupe deux grands domaines, à savoir la résolution distribuée de problème (RDP) et les systèmes multi-agents (SMA). La RDP s'intéresse à la manière de diviser un problème particulier sur un ensemble d'agents distribués et coopérants et la manière du partage de la connaissance du problème et d'en obtenir la solution. La RDP est une approche descendante (" Top-down"), Le problème à résoudre est décomposé et des agents sont conçus pour résoudre les parties. Les SMA s'occupent du comportement d'un ensemble d'agents autonomes qui essaient de résoudre un problème, ce qui pose des questions d'organisation et d'interaction entre les agents. Dans les SMA, l'approche est ascendante ("Bottom-up"), les agents sont d'abord créés avec leurs comportements, leurs buts et leurs capacités, et sont ensuite mis ensemble pour résoudre un problème donné.

Les agents sont considérés comme un nouveau modèle informatique des systèmes de calcul distribués, complexes, ouverts et hétérogènes. De nos jours, un nombre de plus en plus grand de systèmes logiciels sont conçus en termes d'agents et de systèmes multi-agents dans lesquels les composantes logicielles agissent plutôt comme des entités individuelles indépendantes, notamment agents, au lieu d'être uniquement des composantes du système. Les systèmes multi-agents comprennent des idées et techniques venant de plusieurs disciplines : calcul distribué, intelligence artificielle, sociologie, sciences de la gestion et des organisations, mais aussi biologie, psychologie et philosophie.

Notre étude se concentre sur les SMA, plus précisément sur les SMA où les agents peuvent entrer et sortir dynamiquement. Ces systèmes sont nommés SMA ouverts (SMAO*).*

Selon l'approche VOYELLE, développée par l'équipe MAGMA, un SMA est décomposé en quatre parties (ou briques) [Dem01] :

SMA = Agents + Environnement + Interactions + Organisations

– Agents, qui concernent les modèles (ou les architectures) utilisés pour la partie active de l'agent, depuis un simple automate à un système complexe 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. Elles comprennent toutes les interactions réalisées par les agents, soit avec l'environnement, soit avec d'autres agents.

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

Les systèmes multi-agents sont des systèmes complexes, distribués et les concepts qu'ils manipulent sont de haut niveau d'abstraction (coopération, autonomie,…) ce qui rend la programmation lourde et difficile avec les outils de programmation classiques. La mise au point d'environnement de développement ou plateforme multi-agents est nécessaire pour renforcer le succès de la technologie multi-agents. Les plateformes multi-agents permettent aux développeurs des SMA de concevoir et réaliser leurs applications sans perdre de temps à réaliser des fonctions de base pour la création et l'interaction entre agents et éliminent, dans la plupart des cas, la nécessité d'être familier avec les différents concepts théoriques des systèmes multi-agents.

Une plateforme multi-agents est un outil permettant de faciliter la construction et l'exploitation d'un système multi-agents, plus le développeur est assisté dans ses différentes tâches plus la plateforme est performante.

Dans le cadre de l'approche VOYELLES, notre travail se concentre sur la composante **Interaction**. Notre objectif est d'effectuer une synthèse sur les travaux réaliser sur les interactions puis la mise en œuvre du modèle d'interaction dynamique en développant une plateforme expérimentale (Multi-Agents Plateform "MAP") de construction de systèmes multi-agents fondée sur ce modèle. Cette plateforme sera utilisée pour de futurs travaux au sein de l'équipe de recherche[1]. Le développement de cette plateforme est également l'occasion d'explorer et de découvrir le domaine des systèmes multi-agents et de la construction d'applications complexes et réparties.

Par ailleurs cette plateforme va permettre à des utilisateurs non spécialisés de développer de façon simple leur propre système multi-agents réparti sur différentes machines. L'utilisateur va pouvoir définir des agents sans avoir à se préoccuper des problèmes de gestion des communications et d'interactions entre agents. Par ailleurs le système est ouvert en permettant de rajouter ou de supprimer des agents dynamiquement.

Nous proposons d'adopter le modèle d'interaction dynamique MID (présenté au chapitre 3), permettant de déléguer une partie de l'interaction au mécanisme d'interaction. Cette délégation implique le déplacement de plusieurs responsabilités vers le mécanisme d'interaction qui devient de cette façon responsable pour la partie de l'interaction qui est indépendante du domaine d'application. La distribution des tâches concernant l'interaction entre l'agent et le mécanisme d'interaction permet à l'agent de se concentrer sur le sujet de l'interaction et de laisser de côté les questions qui portent sur *comment* et *avec qui* interagir.

Cette plateforme est conçue selon la norme FIPA (Foundation for Intelligent Physical Agent) tout en s'inspirant des autres plateformes existantes telle que JADE [Jade03]. Cette plateforme est entièrement réalisée en Java, qui un langage orienté objet de haut niveau et multi-plateformes, offrant tous les mécanismes nécessaires à la programmation de tâches concurrentes, et permettant l'ouverture des applications sur un réseau comme Internet. Ceci nous a permis de mettre en relation des architectures d'agents hétérogènes, implémentées avec n'importe quel langage supportant les communications réseau. Cette plate-forme offre la possibilité de distribuer physiquement les agents et la distribution physique de l'environnement sur plusieurs sites (machines), autorisant la mobilité des agents de sites en sites.

Pour atteindre ce but, nous avons suivi la démarche suivante: d'abord, nous avons réalisé une étude bibliographique sur les SMA et précisément sur l'impact des interactions dans les SMA et des modèles d'interaction utilisés dans la construction des SMA. Ensuite, nous avons étudié quelques plateformes multi-agents existantes. Nous avons ensuite opté pour la mise en œuvre du modèle d'interaction dynamique en développant la plateforme Multi-agents (MAP) selon les spécifications FIPA (Foundation for Intelligent Physical agent). Enfin le fonctionnement de cette plateforme est testé par la mise en œuvre de quelques applications considérées comme

[1] L'équipe de recherche IMAS – département d'informatique- université de Chlef. Son principal objectif, porte sur l'étude des interactions dans les SMA et le développement d'une plateforme Multi-agents générique dans une perspective de Programmation Orientée Multi-Agents.

Benchmarks (Ventes aux enchères, le jeu Pierre-Papier-Ciseau) et l'exécution de ces applications sur la plateforme.

Ce manuscrit est organisé en deux parties. La première partie couvre un état de l'art sur les systèmes multi-agents et les interactions dans les SMA. D'abord, au deuxième chapitre nous présentons le paradigme des systèmes multi-agents où nous introduisons les concepts de base d'un SMA à savoir l'Agent, l'Interaction et l'Organisation dans un SMA. Et comme l'interaction fait l'objet de notre étude, le troisième chapitre est consacré pour les techniques et modèles d'interactions, où nous présentons les techniques utilisées pour réaliser les interactions ainsi que les modèles d'interaction où nous présentons le modèle d'interaction dynamique MID. Le quatrième chapitre présente la norme des plateformes des systèmes multi-agents et quelques plateformes multi-agents existantes (MADKIT, MASK et JADE) que nous jugeons utile pour cette étude.

La deuxième partie de ce manuscrit est composée de quatre chapitres. Dans le chapitre cinq, nous présentons l'architecture et les modèles choisis pour le développement de la plateforme multi-agents MAP. L'implémentation de cette plateforme est présentée dans le chapitre suivant. Le chapitre sept est consacré pour la mise en œuvre des applications tests de la plateforme MAP et les résultats obtenus. Nous concluons ce travail par une conclusion présentée dans le chapitre huit où nous donnons les perspectives.

Chapitre 2

Le paradigme Multi-agents

Le fondement du paradigme multi-agents est évidemment le concept d'agent qui définit les caractéristiques des composants de base d'un système. Afin de permettre une meilleure approche de la conception des SMA, il est nécessaire de prendre en compte également le concept d'interaction et le concept d'organisation de ces interactions [Fois98].

Le terme interaction désigne l'enchaînement d'échanges d'informations ou d'influences ayant lieu entre les agents. Ce concept est l'un des aspects les plus importants de l'IAD [Deck87] dans la mesure où il permet de relier les agents pour qu'ils constituent un système.

Mais ce concept d'interaction permet de mettre en relation uniquement un sous-ensemble d'agents composant le système. Il est nécessaire alors d'utiliser un autre concept plus général pour exprimer les relations liant tous les membres de la société : l'organisation. Ce concept a pour objectif principal de contrôler et de coordonner l'ensemble des interactions, afin de structurer les activités des agents et permettre au système de se comporter comme un tout efficace et cohérent.

Le but de ce chapitre est de présenter les caractéristiques respectives de ces trois concepts de base d'un SMA à savoir l'agent, l'interaction et l'organisation.

1. Concept d'agent:

Le concept de base du paradigme multi-agents est l'agent. Il correspond à une entité active composant un système multi-agents. Il intervient dans la résolution de problème grâce à ses connaissances, ses compétences (services) et son comportement en participant à la résolution collective par le biais d'interaction avec les autres agents du système.

1.1. Définition:

Plusieurs définitions ont été données pour le terme d'agent, nous retenons deux définitions celle de Ferber [Ferb95] et de Wooldridge [Wool95]

D'après Ferber [Ferb95], un agent est une entité physique ou virtuelle :

a. qui est capable d'agir dans un environnement,
b. qui peut communiquer directement avec d'autres agents,
c. qui est mue par un ensemble de tendances (sous la forme d'objectifs individuels ou d'une fonction de satisfaction, voire de survie, qu'elle cherche à optimiser),
d. qui possède des ressources propres
e. qui est capable de percevoir (mais de manière limitée) son environnement,
f. qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune),
g. qui possède des compétences et offre des services,
h. qui peut éventuellement se reproduire
i. dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et fonction de sa perception, de ses représentations et des communications qu'elle reçoit.

Pour Wooldrige [Wool95], Un agent est un système informatique, situé dans un environnement, et qui agit d'une façon autonome pour atteindre les objectifs (buts) pour lesquels il a été conçu.

Trois fonctions principales sont distinguées dans la structures d'un agent: *Percevoir, Décider* et *Agir*. Ceci introduit qu'un agent a des connaissances qui lui sont propres et un comportement autonome lui permettant d'exploiter ses connaissances, de raisonner et de décider comment et quand agir en fonction de ses interactions avec les autres agents, en fonction de ses perceptions de son environnement et selon les objectifs qu'il cherche à satisfaire.

1.2. Typologie des agents:

Les agents peuvent être classer en deux catégories obtenues en fonctions de leurs capacité de raisonnement: les agents réactifs et les agents cognitifs. Plusieurs définitions sont données pour ces types d'agents, voici une synthèse :

a. Agents réactifs :

Ce sont des agents qui réagissent uniquement à leur perception et qui agissent en fonction de cette perception. C'est le niveau de complexité le plus bas des agents. Ce type d'agents ne peut pas vraiment être qualifié d'intelligent, étant donné que leur fonctionnement est principalement basé sur le principe du stimilus/action. Ce principe permet aux agents d'agir grâce à des réflexes conditionnés. Néanmoins, ils présentent des caractéristiques intéressantes : simplicité de la description du comportement local et émergence de comportements globaux. Un SMA constitué d'agents réactifs possède généralement un grand nombre d'agents et présente un comportement global intelligent.

b. Agents cognitifs :

C'est le principe le plus complexe. De tels agents sont non seulement capables de percevoir et d'agir sur leur environnement, mais en plus ils ont des capacités de cognition leur permettant de raisonner sur les autres ou sur l'avancement de la résolution. Ils font souvent appel à des modes de communication plus complexes qu'une simple perception. Ils communiquent généralement grâce à des structures de données partagées ou par des communications directes.

Les agents cognitifs possèdent une représentation partielle de l'environnement, des buts explicites, ils sont capables de planifier leur comportement, mémoriser leurs actions passées, communiquer par envoi de messages, négocier, etc. Un SMA constitué d'agents cognitifs possède communément peu d'agents.

Les notions mentales, couramment utilisées dans les SMA, sont données par les exemples ci-dessous:

- *connaissances* – Omar connaît le fait que les humains sont mortels
- *croyances* – Omar a pris son parapluie parce qu'il croit qu'il va pleuvoir
- *désirs, buts* – Omar désire avoir son diplôme
- *intentions* – Omar a l'intention de travailler dur pour avoir sa thèse
- *choix, décisions* – Omar a décidé de faire une thèse
- *engagements* – Omar ne va pas s'arrêter de travailler avant d'avoir fini sa thèse
- *conventions* – si, par hasard, Omar décide d'abandonner sa thèse, il va le dire à son professeur
- *obligations* – Omar doit travailler pour entretenir sa famille

Selon le type d'agents utilisés on parle de système réactif ou de système cognitif. Le tableau suivant présente une comparaison entre ces deux systèmes :

Systèmes d'agents cognitifs	*Systèmes d'agents réactifs*
Représentation explicite de l'environnement	Pas de représentation explicite
Peut tenir compte de son passé	Pas de mémoire de son historique
Agents complexes	Fonctionnement slimilus/Action
Petit nombre d'agents	Grand nombre d'agents

Il est possible de concevoir des systèmes hétérogènes comportant les deux types d'agents. Il est ainsi possible de doter les agent cognitifs de capacité de réaction aux événements : on parle alors d'agents hybrides.

1.3. Modèles d'agents:

Les systèmes multi-agents ont de nombreux domaines d'application, ce qui entraîne des attentes parfois très différentes sur les fonctions nécessaires d'un agents. Les critères suivants apparaissent souvent dans la littérature: autonomie, sociabilité, négociation, coopération, conversation à haut niveau,… . Il faut noter qu'il n'existe pas de modèle d'agent répondant à l'ensemble de ces caractéristiques. C'est en fonction de l'application visée, seul un sous ensemble de ces caractéristiques est nécessaire, il faut alors s'orienter vers le modèle d'agent qui semble le plus adapté.[Wool00]

Dans la suite de cette section, nous présentons un modèle général d'agent, puis deux modèles, qui sont généralement utilisés pour les agents intelligents, à savoir le modèle d'agent qui reposent sur une modélisation des croyances, des désirs et des intentions et les agents à couches.

a. Structure conceptuelle des agents:

Un agent est situé dans un environnement. Pour modéliser la structure de l'agent il faut avoir un modèle de l'environnement. L'environnement peut être vu comme étant dans un état ***e*** parmi un ensemble d'états $E=\{e_1, .., e, ..\}$ L'environnement peut changer son état soit d'une manière spontanée soit comme résultat des actions de l'agent.

L'évolution de l'environnement se modélise différemment selon les caractéristiques que l'on prend en compte, et les simplifications que l'on s'autorise. Les principales distinctions à faire sur les types d'environnements sont :

- **Environnement accessible ou inaccessible**. Un environnement est accessible à l'agent si l'agent peut percevoir entièrement l'état de l'environnement ou, au moins, tous les traits de l'environnement qui sont significatifs du point de vue des actions de l'agent. Sinon, l'environnement est inaccessible.
- **Environnement déterministe ou non déterministe**. Si l'état suivant de l'environnement est déterminé d'une manière unique par l'état courant et l'action de l'agent, alors l'environnement est déterministe. Si le résultat est incertain, notamment si, par suite d'une action de l'agent, l'environnement peut évoluer de différentes manières, alors on est dans le cas non déterministe.

- **Environnement statique ou dynamique**. Si l'environnement ne peut pas changer d'état sans l'intervention de l'agent, on est dans le cas statique. L'environnement est dynamique si son état peut se modifier sans l'action de l'agent dans l'intervalle de temps entre deux perceptions de l'agent.
- **Environnement discret ou continu**. Si tout passage d'un état de l'environnement à un autre nécessite le passage par une séquence d'états intermédiaires, alors on a un environnement continu ; sinon, l'environnement est discret.

Les caractéristiques de l'environnement influencent la façon dont on conçoit un agent car il faut tenir compte de l'évolution de l'environnement, de la capacité de l'agent de saisir cette évolution et de sa capacité à décider en conséquence. Par exemple, si on a plusieurs agents qui agissent dans un même environnement, chaque agent va percevoir l'environnement comme dynamique et non déterministe, car l'état de l'environnement changera en raison des actions des autres agents, et une même action exécutée dans un certain état aura des résultats différents en fonction des actions de ces autres agents.

Pour modéliser la structure d'un agent, plusieurs fonctions sont considérées:

- ***voir* : E -> P** est la fonction qui décrive la capacité d'observation de l'environnement, où E est l'ensemble d'états de l'environnement et P est l'ensemble des perceptions de l'agent ;
- ***action* : P -> A** est la fonction qui représente le processus de décision de l'agent : elle détermine quelle action $a \in A$ (A étant l'ensemble des actions disponibles pour l'agent) l'agent choisit en fonction de la perception $p \in P$ qu'il a sur son environnement ;
- ***env* : E x A_1 x … A_n ->P(E)**, A_j, j =1..n étant l'ensemble des actions disponibles à l'agent j.
- ***inter* : P -> I** est la fonction qui représente la décision de l'agent concernant son interaction avec un autre agent, où I est l'ensemble des interactions disponibles pour l'agent et $i \in I$ est l'interaction que l'agent choisit (par exemple un message) en fonction d'une perception $p \in P$ qu'il a de l'environnement ;
- ***rinter* : I -> A** est la fonction qui décrit la réaction de l'agent, notamment l'action décidée par l'agent, à une interaction transmise par un autre agent.

En général, les agents réactifs interagissent uniquement avec l'environnement. Tandis que les agents intelligents interagissent avec l'environnement et les autres agents du système, cette interaction définit la dimension sociale des agents. Mêmes les agents réactifs peuvent parfois interagir, d'une manière directe ou indirecte, avec les autres agents du système.

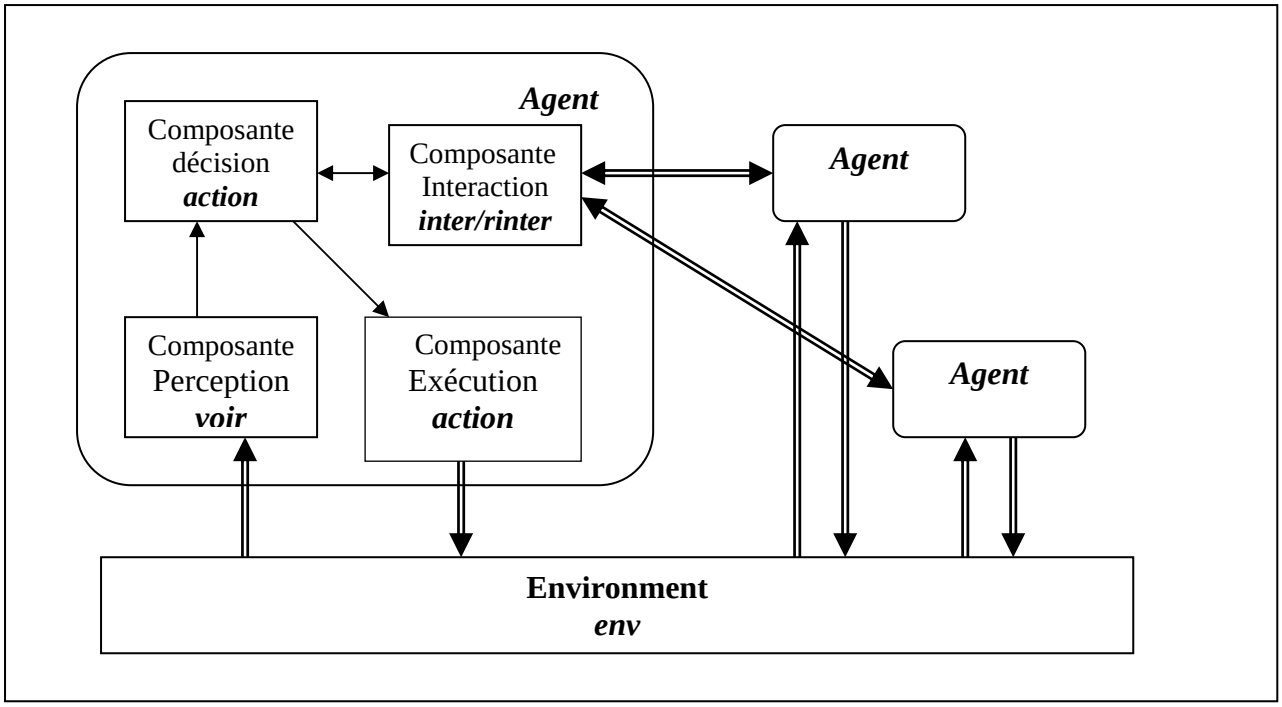


FIG. 2.1. Structure générale d'un agent

Il faut noter que la structure présentée dans la figure 2.1 ci-dessus caractérise les agents intelligents en général, et que la composante interaction avec d'autres agents n'est pas souvent présente dans le cas des agents réactifs.

b. Les Agents BDI (Beliefs, Desirs and Intentions):

Les agents BDI pour Beliefs, Desirs and Intentions, sont inspirés des travaux sur le raisonnement pragmatique, c'est-à-dire le processus de décision permettant de sélectionner les actions à effectuer pour atteindre ses objectifs [Wool95]. Dans ce modèle, le processus se décompose en deux phase: se fixer certains buts et définir la manière de les atteindre. La première phase est appelée phase de délibération, tandis que la seconde correspond à une phase de planification. La figure 2.2 illustre le fonctionnement d'un agent BDI. L'agent est caractérisé par:

- *Ses croyances*, les connaissances qu'il a du monde,
- *Ses désirs*, les objectifs accessibles en fonction de ses croyances et ses intentions,
- *Ses intentions*, qui caractérisent les objectifs courants ou en cours de réalisation de l'agent.

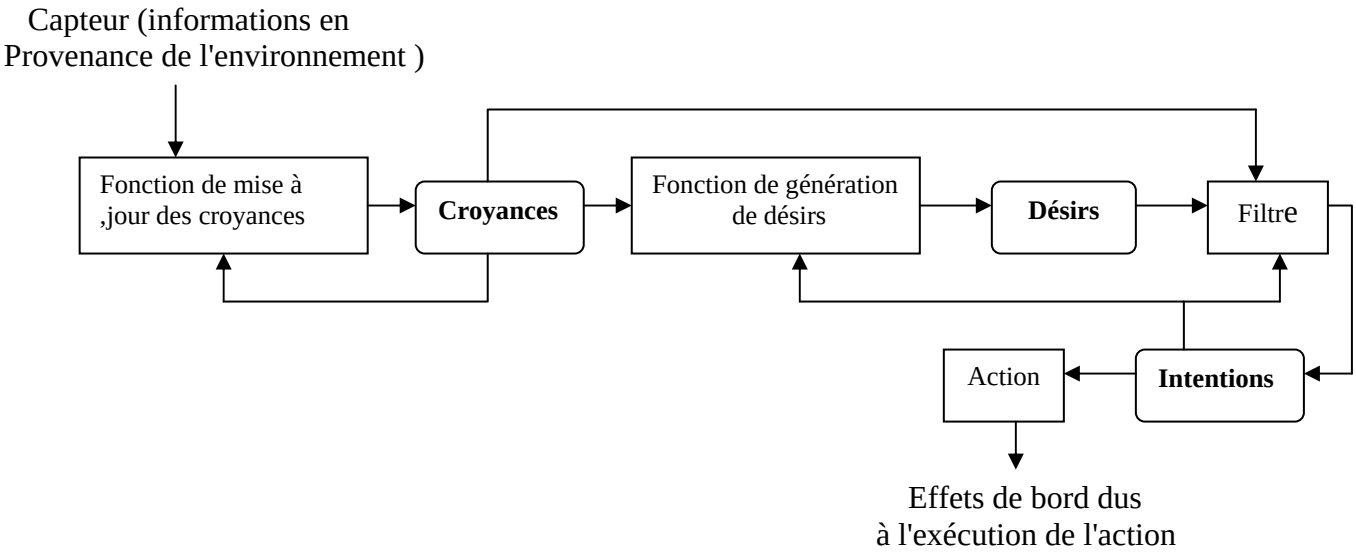


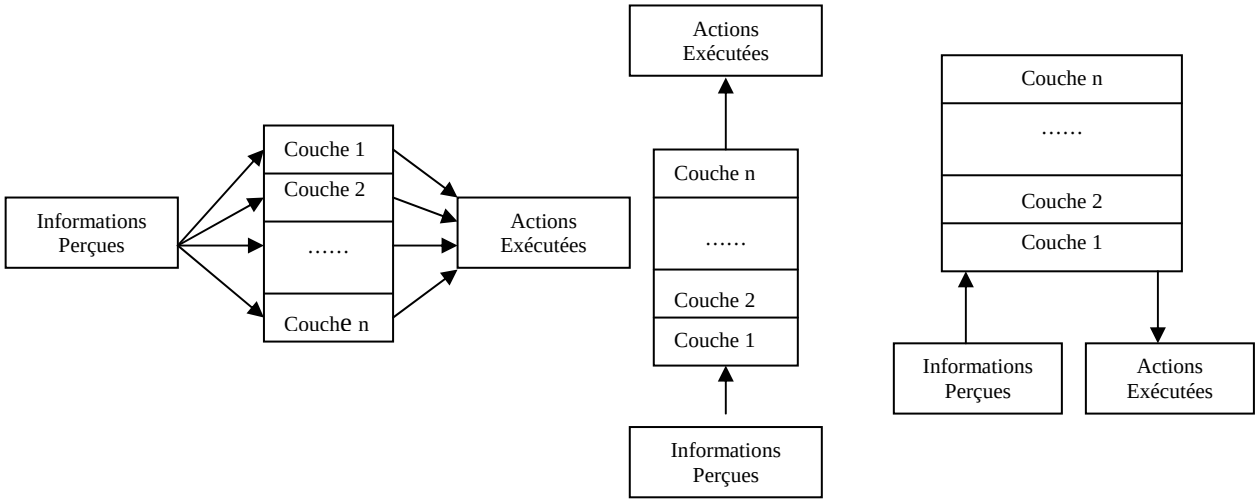
FIG 2.2. Une architecture schématique d'agent BDI

Dans ce modèle, les intentions jouent un rôle central: elles orientent la planification, elles limitent les délibérations futures, elles sont persistantes et elles influencent les croyances futures sur lesquelles se base le raisonnement. L'équilibre entre ces différents aspects représente l'une des clés de la conception d'agents BDI. Particulièrement, le compromis entre la poursuite d'une intention ou son abandon est une fonction particulièrement difficile à concevoir.

Le modèle BDI est intéressant non seulement car il est intuitif, mais aussi car il fournit une bonne décomposition fonctionnelle, qui identifie clairement les sous-systèmes nécessaires à la conception d'un agent. Néanmoins, la principale difficulté reste toujours l'implémentation efficace de ces sous-systèmes.

c. Les agents à couches:

Les architectures à couches consistent à décomposer le comportement d'un agent en deux partie: la partie réactive et la partie proactive [Wool95]. Il y a donc au minimum deux couches, mais il est possible d'avoir une hiérarchie complexe de couches qui interagissent entre elles. La figure 2.3 illustre les deux grandes classes d'architecture à couches et la gestion du flux de contrôle entre les couches. La première classe est caractérisée par des couches horizontales, qui reçoivent toutes la même information et qui proposent des actions en réaction à ces informations. Dans la deuxième classe, les couches sont organisées verticalement et une couche au plus reçoit les informations et/ou produit les actions.



(a) Couches horizontales ***(b) Couches Verticales (Contrôle en une passe)*** ***(c) Couches verticales (Contrôle en deux passes)***

FIG.2.3. Architectures et flux de contrôle dans les agents à couches.

L'avantage principal des couches organisées horizontalement est qu'il suffit d'ajouter des couches à l'agent pour qu'il exhibe un nouveau comportement. Mais comme toutes les couches proposent des actions, il est indispensable d'introduire un médiateur qui définira à chaque moment quelle est la couche prioritaire. C'est ce point de centralisation qui est particulièrement gênant dans cette architecture.

Ce problème est justement partiellement résolu avec les architectures à couches verticales. Dans ces modèles, l'information traverse les couches séquentiellement, réduisant ainsi les interactions entre les différentes couches. En fonction du mécanisme de contrôle, l'information ne traverse qu'une fois chacune des couches (figure-b) ou deux fois, lors de la remontée des informations et de la descente des actions (figure-c)

Les architectures à couches sont actuellement les plus utilisées. Les couches représentent naturellement les différentes fonctionnalités de l'agent: réactivité, proactivité, et comportement social. Le principal avantage de cette approche est aussi son principal inconvénient: c'est une approche pragmatique, c'est-à-dire facilement implémentable et praticable, qui manque de clarté conceptuelle et sémantique. D'autre part, la gestion des interactions entre les différentes couches, bien que simplifiée dans le cas des architectures à couches verticales, reste difficile à maîtriser.

2. Concept d'interaction:

La notion d'interaction est une notion qui prend place à l'intérieur des agents, via la faculté de perception, d'action et de communication, mais qui prend corps entre les agents. Elle peut être défini comme un processus, qui consiste en l'enchaînement d'actions réciproques de la part des agents. Ce concept est l'un des aspects les plus

important de l'IAD, dans la mesure où il permet de relier les agents qui peuvent échanger des informations et s'influer afin de constituer un système

Pour un agent, interagir constitue à la fois la source de sa puissance et l'origine de ses problèmes [Ferb95]. En effet, c'est parce que les agents interagissent que les agents peuvent accomplir leurs actions locales, mais aussi à cause de leurs interactions avec les autres agents qu'ils doivent coordonner leurs actions et résoudre des conflits.

Les interactions sont à la base du fonctionnement des systèmes informatiques constitués de plusieurs entités et notamment les systèmes multi-agents et les systèmes répartis. La notion d'interaction est néanmoins difficile à cerner, car elle est définie par les entités qui interagissent (leurs objectifs, leurs intentions,…), mais elle est aussi liée à l'organisation existante entre les différents composants du système, à l'environnement qui supporte l'interaction et aux moyens de communications pour la mise en œuvre des interactions.

Selon [Bras96], le terme d'interaction peut être décliné de deux façons différentes, si elle est faible ou forte:

- Elle est faible, si elle n'est composée que d'un ensemble d' « action-influence unidirectionnelle », c'est-à-dire que les agents agissent « en direction » des autres.

- Elle est forte, si elle est composée d'un ensemble de « co-action-influence mutuelle ».

2.1. Interaction et SMA réactifs:

Malgré l'intelligence faible des agents réactifs, ceux-ci utilisent généralement des interactions fortes où l'enchaînement des actions de communication émerge dans l'environnement [Ferb95]. Le contrôle de cet enchaînement est basé sur des comportements réflexes de type stimuli/action stipulant le comportement de l'agent. Les agents produisent alors localement des actions dans l'environnement en fonction de leur perception selon des principes réflexes. Dans ce cadre, la problématique de l'interaction consiste à spécifier ces réflexes pour chaque agent pour obtenir un enchaînement viable de ces actions qui émerge de la multitude des agents et des réflexes associés. La description de cet environnement est donc réalisée de manière implicite.

2.2. Interactions et SMA Cognitifs:

Dans le cadre de ce mémoire, nous nous intéressons aux interactions dans les systèmes multi-agents composés d'agents cognitifs (agents communicants). Les agents de ces systèmes se caractérisent par une représentation explicite de leurs connaissances et des autres agents leur permettant d'interagir par l'échange direct de messages.

Pour les agents cognitifs, une perception ne consiste plus uniquement en un signal dans un environnement, mais à une combinaison de signaux formant un

message. La sémantique des communications et alors beaucoup plus riche et complexe. La notion d'interaction reliant les agents s'apparente alors plus à la notion de conversation qu'à une notion d'action.

Les approches du paradigme multi-agents considèrent que la notion de langage d'interaction est nécessaire à un niveau global [Dema95], pour que les agents se comprennent entre eux. Cette notion est mise en place par un langage commun, à l'ensemble des agents, définissant pour le système entier une syntaxe et une sémantique d'échange.

2.2.1. Actes de langage:

La théorie des actes de langage (speech acts) est une théorie de la communication fondée dans les années 60 par le philosophe J.L. Austin. Le postulat à la base de la théorie affirme que la production d'un énoncé s'assimile à l'accomplissement d'actions: si agir c'est transformer l'état des choses, parler c'est, également, transformer l'état mental des interlocuteurs. Une conséquence de ce postulat est qu'il doit être possible d'identifier les constituants élémentaires de la communication. Ceux-ci sont appelés *actes de langage.*

J.L. Austin s'est intéressé aux différents types d'actes de langage que constituent les actions intentionnelles effectuées au cours d'une communication. Il distingue trois types d'actes de langage : actes locutoires, actes illocutoires et actes perlocutoires.

- Les *actes locutoires* se rapportent à la formulation d'un énoncé. Ces actes sont satisfaits lorsque l'énoncé est correctement formulé.
- Les *actes illocutoires* désignent l'action effectuée sur l'auditeur lors de la formulation de l'énoncé (ex., donner un ordre, poser une question). Ils sont accomplis avec succès lorsque l'effet attendu sur auditeur est obtenu (ex.: affirmer, questionner, demander, promettre, ordonner, prévenir, répondre, etc). La force illocutoire est souvent exprimée grâce à un verbe qualifié de performatif, qui manque le type d'acte. Ce sont les **performatifs** qui indiquent dans les systèmes les différents types d'actes de langages qui peuvent émettre et interpréter les agents.
- Les *actes perlocutoires* rapportent aux conséquences indirectes visées par les actes locutoires et illocutoires. Ils ne sont pas codés dans les actes de communication par des performatifs et dépendent du contexte dans lequel s'effectue la communication. (Conséquences des actes illocutoires) Ils sont satisfaits lorsqu'ils sont reconnus par l'auditeur (ex.: convaincre, faire croire, etc.). Exemple : « Il pleut » est un acte illocutoire d'affirmation, qui peut être pris par le destinataire comme une tentative de le persuader de se munir de son parapluie.

Un énoncé peut réussir ou échouer. L'acte de langage échoue dans l'un des cas suivants :

- La transmission de l'acte est mal faite ou que le destinataire ne comprend pas le langage utiliser par l'émetteur.

- Le destinataire interprète mal la force illocutoire de l'émetteur.
- Echec dans la réalisation effective de l'acte énoncé. Cet échec peut être dû au manque de compétence du destinataire pour réaliser cet acte. Ou un refus d'exécuter des directifs. etc…

La force illocutoire peut être identifiée selon cinq catégories de verbes:

1. Les assertifs : servent à donner une information sur le mode en affirmant quelque chose. (dire, penser, informer, affirmer, etc…) Exemple : l'année a quatre saisons.

2. Les directifs : sont utilisés pour donner des directives au destinataire.

(demander,exiger,ordonner, supplier, etc…) Exemple : Ferme la porte.

3. Les promissifs (commissifs) : Engagent le locuteur à accomplir certains actes dans l'avenir. (promettre, garantir, etc…) Exemple : j'assisterai au cours d'IA demain.

4. Les déclaratifs : Accomplissent un acte par le fait même de prononcer l'énoncé. (déclarer, stipuler,etc…) Exemple : Je sors de la salle.

5. Les expressifs : Servent à donner au destinataire des indications concernant l'état mental du locuteur. (féliciter, approuver, déplorer,etc…)Exemple : Je suis content de votre travail.

La théorie des actes de langage fait la constatation que la production d'énoncé linguistique est une action à part entière "dire c'est faire". Cette constatation est particulièrement importante dans le contexte des systèmes multi-agents, puisque les agents logiciels ont souvent comme seule forme d'action l'échange de messages. [Fois98]

La théorie des actes de langage est à la base du développement de plusieurs langages d'interaction inter-agents, parmi les quels qui sont devenus des standards (KQML et FIPA-ACL). Les deux langages seront présentés dans le chapitre suivant.

2.2.2. Conversation:

La théorie des actes de langage a montré qu'elle est un support très utile pour les langages de communication entre agents, néanmoins elle ne prend pas en compte les séquences d'interactions, elle se limite à un acte isolé.[Mazo01]

Une conversation entre agents est un enchaînement de cycle d'échange de messages. Un cycle peut être seulement un acte de communication ou un ensemble d'actes exécutés dans un ordre précis par l'ensemble d'agents. Le problème avec les conversations inter-agents est de garantir la cohérence et la continuité des échanges.

Dans la prise en compte des séquences d'interaction (conversation) deux approches se distinguent: la première plus proche des approches linguistique [Bras97] travaille sur le dialogisme, la seconde approche de l'informatique traditionnelle et des systèmes distribués utilise des protocoles d'interaction [Dema94] qui constituent une

sorte de grammaire de dialogue dictant comment former des conversations à base d'actes de langage et dans quel ordre ils doivent être considérés.

Les modèles d'interaction basés sur le dialogisme ne sont pas viables en pratique [Mazo01]. En effet, pour pouvoir réaliser les interactions sociales des conversations, il faut incarner au sein des agents une quantité significative de connaissances et une rationalité très sophistiquée, ce qui implique que les agents devraient produire leurs interactions par un processus en ligne du raisonnement très complexe à mettre en œuvre. En contre partie, les protocoles eux ne souffrent pas de cette difficulté: ils peuvent être mis en application efficacement et n'exige pas d'un agent d'avoir une architecture complexe et délibérative.

3. Le concept d'organisation:

La notion d'organisation joue un rôle fondamental au sein des systèmes multi-agents : c'est un moyen permettant d'organiser logiquement les agents, c'est un réseau de communication par défaut et c'est un média de recherche d'agent, de rôle ou de compétence. De plus, sa réification fournit un point d'entrée dans le système permettant de visualiser et d'améliorer les interactions entre agents grâce à son évolution dynamique. Cette dynamicité de l'organisation facilite la tâche du concepteur, et améliore la fiabilité du système.

Les organisations tout comme les interactions sont des objets de premier ordre dans les systèmes multi-agents. La notion d'organisation est nécessaire pour structurer les interactions qui interviennent entre les différentes entités du système. Chaque organisation est constituée d'un ensemble de rôles (les rôles abstraits joués par les agents) et dispose d'un rôle particulier qui sert de point d'entrée dans l'organisation. C'est ce rôle qui encapsule les fonctionnalités d'authentification et d'autorisation. Ce rôle a été identifié dans le modèle Aalaadin [Gutk98]. Dans Aalaadin, les agents ne peuvent communiquer qu'au travers des groupes auxquels ils participent.

Cette approche renforce le contrôle sur les communications entre les agents et formalise les responsabilités des agents au sein de l'organisation. De ce point de vue, l'organisation doit être considérée comme une entité en elle-même, indépendante des agents la constituant : les agents peuvent changer au sein de l'organisation, mais pas les rôles qui la définissent.

L'organisation, puisqu'elle gère les rôles, peut aussi constituer un mécanisme intéressant de recherche d'accointances. Utiliser l'organisation de cette manière, permet de disposer d'un système de pages jaunes applicatif distribué. Au sein de chaque système, il peut y avoir plusieurs organisations de différentes natures, que cela soit au niveau de la topologie ou des règles de gestion (admission / démission, algorithme de recherche de rôle, etc.).

3.1. Types d'organisation:

Dans les SMA, les organisations sont de deux Sortes : l'organisation statique et l'organisation dynamique :

3.1.1. Organisation statique:

De nombreux modèles organisationnels se sont inspirés de la théorie des organisations économique. Généralement, les systèmes de ce type laissent à leurs concepteurs le soin de définir eux même à priori l'organisation du système multi-agents. Les architectures multi-agents hiérarchiques sont des exemples d'organisation statique de systèmes multi-agents. Dans cette approche, les liens relationnels entre agents sont tout à fait préétablis. Ils n'émergent pas par l'évolution du système, le contrôle du système est global et fixe.

3.1.2. *Organisation dynamique:*

Dans ce modèle, la structure organisationnelle du SMA est créée dynamiquement par les agents eux-mêmes. La réalisation des buts d'un agent stimule le phénomène de réorganisation de la société multi-agents. Le phénomène est géré par un processus collectif ou individuel mettant en jeu des mécanismes d'évolution. Il existe différentes approches qui traitent du problème d'adaptation.

Les agents d'un système adoptent une structuration en fonction de leurs connaissances individuelles des interactions qui caractérisent cette structuration. L'organisation d'un système consiste alors à décrire ces connaissances d'une manière globale et à doter les agents de ces connaissances.

3.2. Les modèles organisationnels:

Dans cette section, nous présentons quelques travaux effectués dans le domaine des systèmes multi-agents autour de la notion d'organisation. Un premier critère caractérisant une organisation est sa structure topologique, c'est à dire la nature des liaisons entre les entités. La figure 2.4 ci-dessous illustre cet aspect. Parmi les modèles les plus répandus, il y a le modèle hiérarchique, le modèle orienté groupes (Aalaadin), le modèle peer-to-peer (à la base de la majorité des systèmes de partage de fichiers) et le modèle de diffusion (Broadcast).

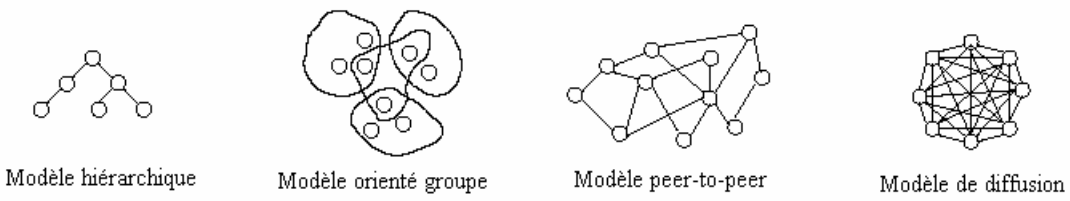


FIG.2.4. Les modèles organisationnels

Dans la section suivante, nous présenterons le modèle organisationnel Aalaadin [Gutk98].

3.2.1. Le modèle Aalaadin:

Ce modèle décompose l'analyse des structures collectives en deux niveaux:

- **Le niveau descriptif** correspond aux concepts centraux d'agent, de groupe et de rôle. C'est à ce niveau que se décrit une organisation réelle.

- **Le niveau méthodologique** définit l'ensemble des rôles possibles, spécifie les interactions et décrit les structures abstraites de groupe et d'organisation.

3.2.1.1. Le niveau descriptif:

Le modèle Aalaadin est basé sur trois concepts: l'*agent*, le *groupe* et le *rôle*. La figure suivante présente un diagramme de ce modèle.

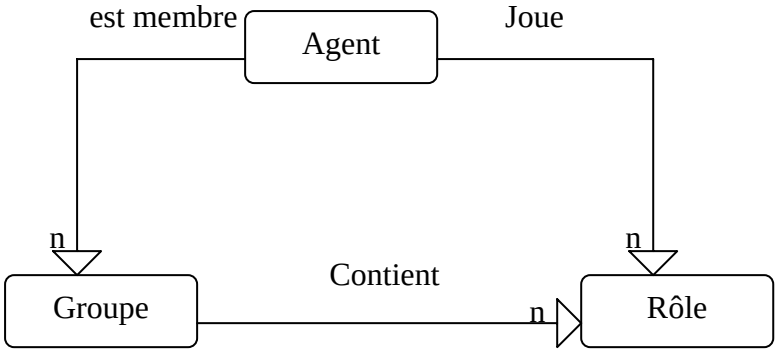


FIG.2.5.Les trois concepts centraux

Dans ce modèle, aucune contrainte ou un pré-requis n'est posé sur l'architecture interne de l'agent. Un agent est simplement défini comme une entité autonome communicante qui joue des rôles au sein de différents groupes.

Le groupe est défini comme étant la notion primitive de regroupement d'agents. Chaque agent peut être membre d'un ou plusieurs groupes. Dans sa forme la plus simple, un groupe peut être vu comme un moyen d'identifier par regroupement un ensemble d'agents, d'une manière plus évoluée, le groupe peut être vu comme un SMA usuel. Un point majeur de cette définition est que les différents groupes peuvent se recouper librement. Un groupe peut être fondé par n'importe quel agent.

Le rôle est une représentation abstraite d'une fonction, d'un service ou d'une identification d'un agent au sein d'un groupe particulier. Chaque agent peut avoir plusieurs rôles, un même rôle peut être tenu par plusieurs agents, et les rôles sont locaux aux groupes. La tenue d'un rôle dans un groupe préexistant doit être demandée par l'agent et n'est pas forcément accordée. La maîtrise de l'hétérogénéité des situations d'interaction est rendue possible par le fait qu'un agent peut avoir plusieurs rôles distincts au sein de plusieurs groupes, et que les interactions sont toujours locales à un groupe.

Pour résumer, un agent *a* peut entrer dans un groupe *g* pour y jouer un rôle *r* si une fonction d'acceptation booléenne *fg r (a)* est évaluée à vrai.

Un rôle important et particulier dans un groupe est le rôle de gestionnaire de groupe, confié initialement au fondateur. Ce rôle correspond à la responsabilité de gestion des demandes d'admission dans le groupe et de tenue de rôle. Il peut être vu

comme l'agent responsable de l'évaluation des fonctions d'acceptations définies, et donc d'une certaine manière comme " l'œil du concepteur " au sein du groupe.

3.2.1.2. Le niveau méthodologique:

Cette section décrit les concepts de structure de groupe et structure organisationnelle, ainsi que leur relation avec le modèle descriptif. La figure suivante représente le modèle Aalaadin complet.

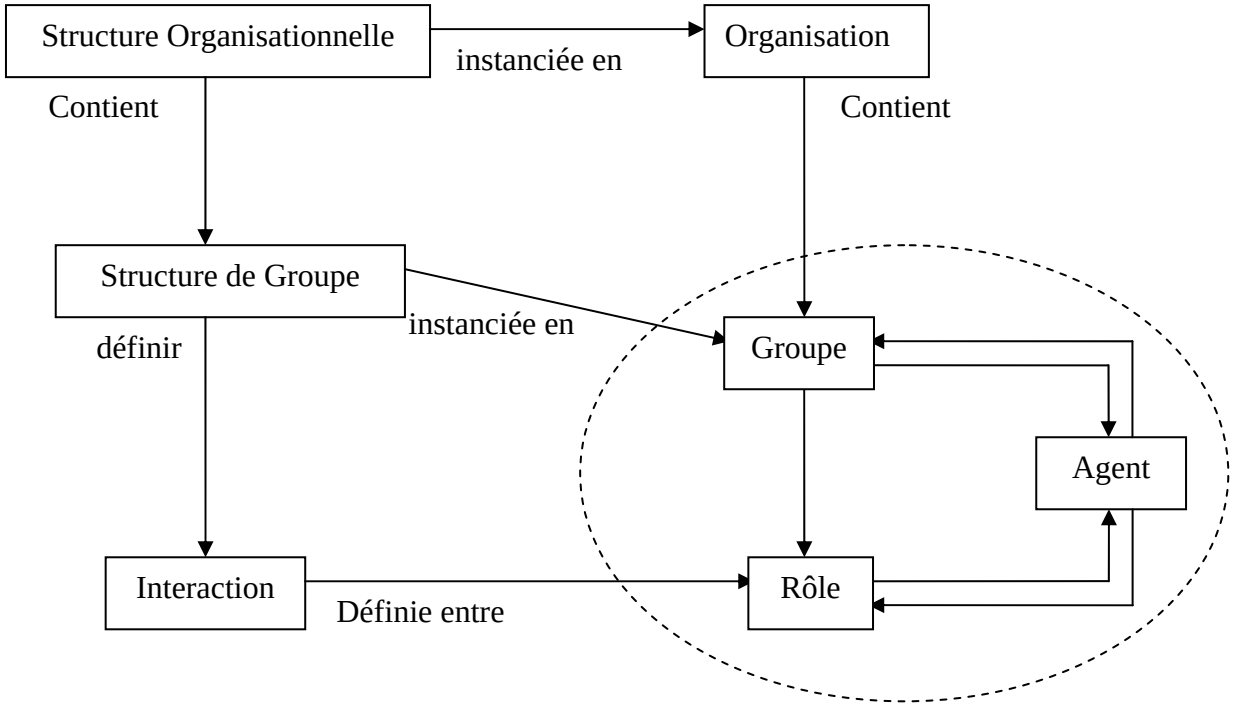


Fig. 2.6. Le Modèle Aalaadin complet

a. Structure de groupe:

La structure de groupe est une description abstraite d'un groupe. Elle identifie la totalité des rôles et des interactions qui peuvent survenir au sein d'un groupe. La structure de groupe se définit comme un tuple: $S = < R , G , L >$

- *R* est un ensemble fini d'identifiants de rôles. C'est l'énumération de tous les rôles qui peuvent être joués par des agents dans le groupe.
- *G* est le graphe des interactions. C'est un graphe orienté étiqueté spécifiant l'ensemble des interactions entre rôles deux à deux. $G: R \times R \longrightarrow L$, l'orientation correspond au rôle initiant l'interaction. Chaque arc représente une interaction nommée par l'étiquette de l'arc et initié par un agent ayant un rôle *ri* avec un agent ayant un rôle *rj*
- *L* est le langage d'interaction. C'est le formalisme choisi pour décrire chaque interaction. Chaque relation dans le graphe est définie par un protocole d'interaction *p* décrit dans *L*.

Notons que la structure de groupe peut être instanciée d'une manière partielle dans le SMA final: il n'est pas requis que l'ensemble des rôles définis dans la structure de groupe se retrouvent simultanément dans le groupe réel, ceci dépendant de la dynamique interne du groupe.

b. Structure organisationnelle:

La structure organisationnelle est définie comme un ensemble de structures de groupe définissant un modèle d'organisation multi-agents. Cette structure organisationnelle peut être vue comme la spécification globale du problème initial (par exemple avoir à définir une structure de marché). Les différentes structures de groupe mises en œuvre - spécifiables indépendamment - permettent une gestion viable de l'hétérogénéité des langages de communications et des modèles de domaine et d'agents, le tout au sein d'un même système.

La structure organisationnelle est définie comme un couple $O = <S, Rep>$ où:
- *S* est un ensemble de structure de groupes.
- *Rep* est le graphe des représentants. C'est un graphe étiqueté où chaque arc *Sa Sb* réunit deux rôles *r1 r2* , où $S_a \in S$ *et* $S_b \in$ S, avec *r1* et *r2* étant des rôles définis dans les structures de groupes *Sa* et *Sb* respectivement.

Un représentant entre deux structures de groupes *A* et *B* est un agent qui aura simultanément le rôle *ra,i* dans le groupe *a* et le rôle *rb ,j* dans le groupe *b*.

Tout comme les relations entre structures de groupes et groupes réels, l'organisation réelle finale n'est qu'une manifestation possible de la structure organisationnelle. Elle peut ne pas inclure tous les groupes définis dans la structure organisationnelle abstraite.

Exemple de communauté d'agents représentant une structure de marché:

Dans la figure 2.7, ci-dessous, trois groupes sont définis. Le lien entre les deux modèles d'interaction est réalisé par le rôle de *Broker*, qui est un représentant du groupe des *fournisseurs* via un rôle de *Broker* au sein de ce groupe. Les clients interagissent avec l'intermédiaire pour chercher un fournisseur adéquat. La structure de groupe *contrat* est définie pour réunir les deux agents impliqués dans la phase finale d'une négociation. Une organisation réelle possible découlant de cette structure organisationnelle est montrée dans la figure 2.8.

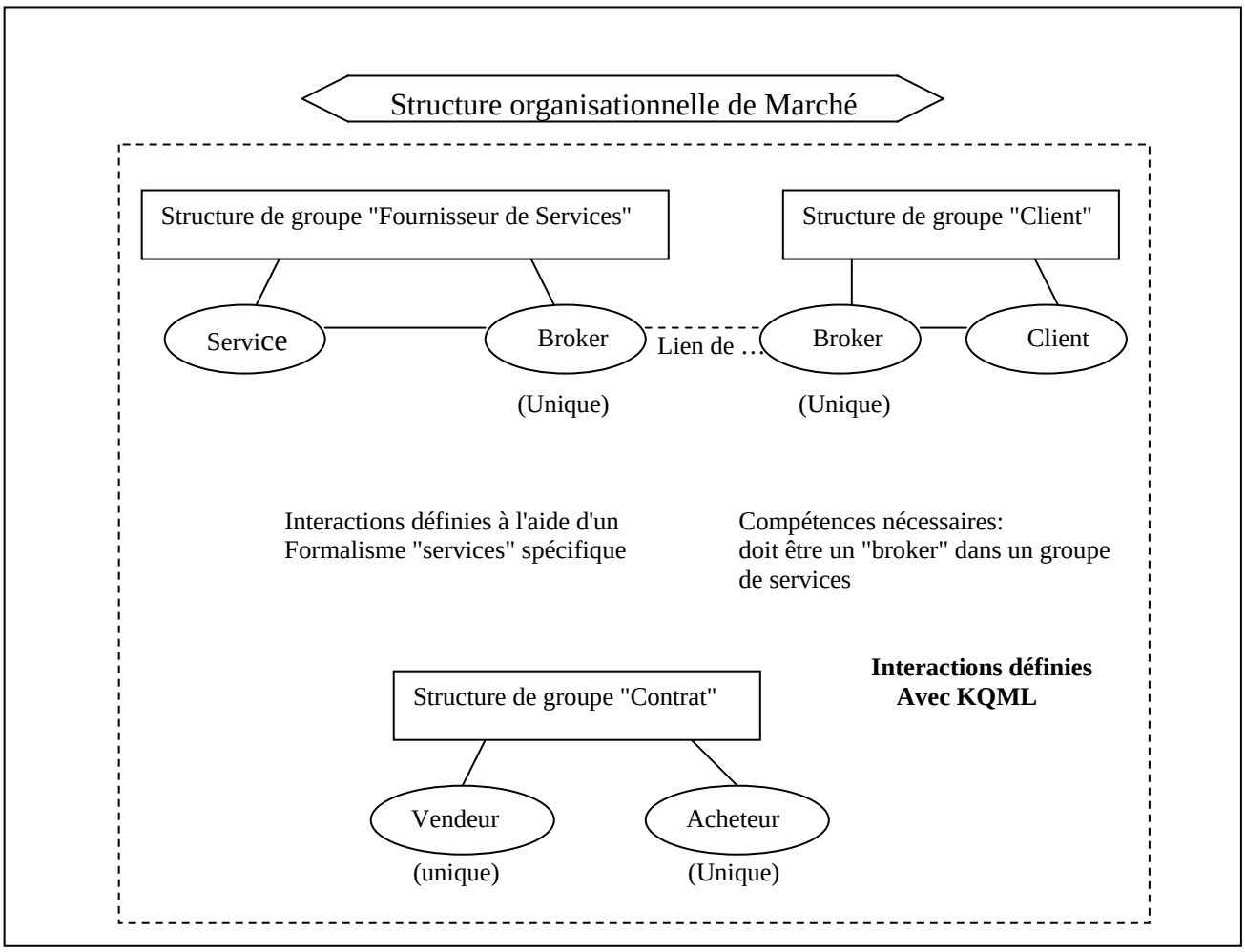


Fig.2.7.Structure organisationnelle de "marché"

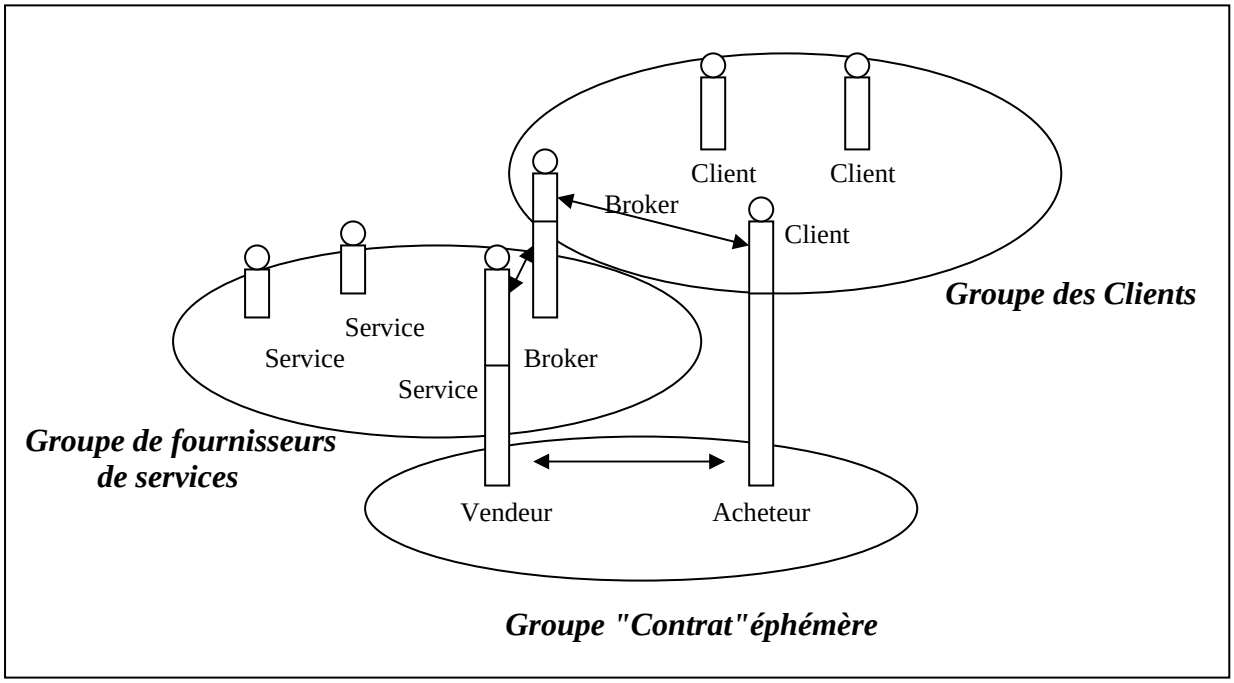


Fig 2.8. Une instance d'organisation de "marché"

4. Conclusion:

Dans ce chapitre nous venons de présenter les différents concepts de base du paradigme multi-agents. L'agent, quelque soit son type, est l'entité de base de toute approche multi-agents. Il est caractérisé par son autonomie de décision et sa recherche de satisfaction. L'interaction est le concept permettant de relier les agents d'un système. Nous l'avons présenté pour les agents communicants, les techniques et les modèles de l'interaction sont développés dans le chapitre suivant. Enfin, l'organisation est le concept permettant de désigner l'activité de l'ensemble des agents constituant le système qui forme alors une société.

Chapitre 3

Techniques et modèles d'interaction

Dans les systèmes multi-agents, une des principales, propriétés d'un agent est celle d'interagir avec les autres agents. Le concept d'interaction est l'un des aspects les plus importants de l'IAD, dans la mesure où il permet de relier les agents qui peuvent échanger des informations et s'influencer afin de constituer un système.

Cette notion d'interaction peut faire intervenir différents modèles de communications dont les agents peuvent être dotés. Mais, l'interaction ne se résume pas uniquement à une action locale de communication d'un agent vers un autre, c'est surtout un enchaînement d'actions et de communication qu'il faut mettre en place et contrôler entre agents.

Du fait que notre étude s'oriente vers les interactions dans les SMA, nous abordons, dans ce chapitre, les principes de l'interaction et sa mise en œuvre. Nous commençons par présenter le concept de communication sur le quel repose toute interaction. Dans la deuxième section, nous présentons les principaux langages de communication inter-agents à savoir le langage KQML et le langage FIPA-ACL. La troisième section présentera les protocoles d'interaction. La dernière section est consacrée pour les modèles d'interactions où le modèle d'interaction dynamique est détaillé.

1. Communication:

La communication est le principe de base qui permet de relier les agents d'un système pour qu'ils constituent un tout. Les agents communiquent dans le but d'accomplir leurs buts ou les buts de la société. Elle permet aux agents de coordonner leurs actions et leur comportement pour obtenir un système cohérent. Elle permet de réaliser l'échange d'informations, la formulation de requêtes au sein du système et par là même, elle est le support de l'interaction [Fois98].

La communication peut être définie comme une action locale d'un agent vers les autres agents, qui peut prendre deux formes; elle peut être indirecte ou directe.

Dans cette définition, les communications indirectes (ou implicites) supposent l'existences d'un support de communication commun aux agents (un environnement ou une structure de donnée partagée) qui mémorise et propage (en déformant éventuellement) l'information associée aux signaux déclenchés par la réalisation de l'action. Les communications directes (ou explicites) utilisent comme support de communication des liens directs entre les agents (envoi de message, diffusion totale ou restreinte) [Chev95].

1.1. Communication indirecte:

Dans ce type de communication deux techniques différentes sont distinguées, selon que le support de communication est passif ou actif vis-à-vis de l'information.

a. ***Communication par tableau noir :*** Cette technique utilise une structure de donnée commune aux agents dont la quelle les agents déposent, consultent et modifient les informations. L'information n'est pas modifiée par le tableau noir. Ce support est dit passif ou neutre vis à vis des informations, bien qu'il signale les évènements de modification et de dépôt d'informations aux agents.[Fois98]

b. ***Communication Via l'environnement :*** Dans cette technique, l'environnement est une structure commune aux agents. Cet environnement contient un ensemble d'objets passifs qui peuvent être perçus, crées, détruits ou modifiés par l'ensemble des agents. Les agents réagissent à leurs perceptions partielles de l'état de l'environnement et combinent leurs actions dans celui-ci, afin de modifier l'état de l'environnement et ainsi influencer la résolution collective. Par exemple, le passage d'un agent fourmi va laisser une trace (objet phéromone) dans l'environnement, qui pourra être perçu par d'autres fourmis pendant un certain temps et influer sur leur comportement. Cette technique est généralement utilisée dans des systèmes réactifs.

Une caractéristique importante de ces techniques de communication est que l'information n'est pas privée, dans le sens où l'agent réalisant l'action de communication ne désigne pas le ou les agents concernés par celui-ci. La structure commune est accessible par tous les agents et les informations sont par conséquent publiques.

1.2. Communication directe:

Ce type de communication utilise une technique d'envoi de messages pour transmettre directement les informations. Les agents sont en liaison directe et envoient leurs messages directement et explicitement aux destinateurs.

Cette technique de communication peut prendre plusieurs formes :

a. Communication par envoie de messages point à point :

Dans ce cas, l'agent émetteur (celui qui réalise l'action de communication) connaît nominativement l'agent destinataire du message et lui adresse alors directement l'information. Cet agent destinataire est le seul à recevoir l'information qui est alors privée.

b. Communication par diffusion généralisée:

Il s'agit, dans ce cas, d'un envoi simultané d'un même message à tous les agents du système. L'agent émetteur ne connaît pas nécessairement nominativement les destinataires du message, ou du moins ne le désigne pas explicitement lors de l'envoi. L'agent suppose également que parmi tous les agents du système, il en existera bien un capable de traiter ce message

c. Communication par diffusion restreinte:

Il s'agit là aussi d'un envoi simultané d'un même message mais seulement à un sous-ensemble des agents du système. L'agent émetteur ne connaît pas nécessairement nominativement tous les agents destinataires, mais doit être capable de les décrire par les caractéristiques discriminantes, ou de les atteindre grâce à une notion de groupe présente dans le système.

2. Langage de communication inter-agents (ACL):

L'intérêt des langages de communication est de réduire les communications en évitant une description des messages ad hoc et une gamme étendue de protocoles. Ces langages se focalisent essentiellement sur la manière de décrire exhaustivement des actes de communication d'un point de vue syntaxique et sémantique supportant un langage de représentation des connaissances. [Koni95]

Généralement, les langages d'interaction sont développés à partir de la théorie des actes de langage. Plusieurs tentatives de normalisation de la communication inter-agents ont été effectuées au sein de la communication multi-agents ces dernières années.

La section suivante présente les langages de communication inter-agents les plus marquants dans la communauté agent. Le langage KQML, l'approche MAGMA (le langage IL) et le langage de communication inter-agents FIPA-ACL.

2.1. Le langage de communication KQML:

Le langage de communication KQML (Knowledge Query and Manipulation Language) a été développé au sein du groupe de travail "External interface Group" du projet de recherche "Knowledge Sharing Effort" aux états unis. Ce langage de haut

niveau est fondé sur la théorie des actes de langage [Fini94]. Il a été développé dans le but de permettre aux agents cognitifs de coopérer. Il est basé sur le fait de pouvoir coder explicitement dans les messages des actes illocutoires en terme de type de message ou performatives et repose sur les états mentaux des agents. Le contenu du message est une expression spécifiée en KIF (Knowledge Interchange Format) qui utilise le format de la logique du premier ordre.

KQML est un langage de communication pour l'échange de l'information, orienté message véhiculant un contenu et une ontologie applicable. Une ontologie est une spécification ou une vue simplifiée et abstraite du domaine qui sera présenté. En d'autre terme, une ontologie définit le vocabulaire dans un domaine donné pour que les agents puissent se comprendre.

Ainsi, KQML est indépendant du mécanisme Transport (TCP/IP,…), indépendant du langage du contenu échangé (KIF, SQL, Prolog, ou autre) et indépendant de l'ontologie utilisée.

Conceptuellement, dans un Message KQML, trois couches sont identifiées: Contenu, Communication et Message.

- La couche *contenu* comporte la teneur réelle du message, utilisant un langage de représentation propre au système. KQML peut supporter n'importe quel langage de représentation, y compris des langages exprimés en ASCII et ceux exprimés en utilisant la numérotation binaire.
- La couche de *communication* code un ensemble de dispositifs au message qui décrivent les paramètres de bas niveau de communication, tels que l'identité de l'expéditeur et du destinateur et un identifiant unique associé à la communication.
- La couche *message*, qui code un message qu'une application voudrait transmettre à une autre, est le noyau du KQML. Cette couche détermine les genres d'interactions au niveau des agents dialoguant via KQML. La fonction de cette couche est d'identifier l'acte du langage ou la performative que l'expéditeur attache au contenu. Cet acte indique si le message est une affirmation, une question, une commande ou tout autre d'un ensemble de performatifs communs (type de message primitif). En outre, puisque le message est opaque à KQML, le message inclut également les questions décrivant le langage du contenu, l'ontologie qu'il utilise et une certaine description du contenu, comme un descripteur référençant une matière dans l'ontologie. Ces dispositifs permettent une analyse correcte des messages sans avoir à accéder au contenu.

Les messages du KQML ne concernent pas seulement des phrases dans un langage quelconque, mais sont enrichis d'une attitude sur le contenu (affirmation, engagement, requête, question,…)

Ce langage propose une description d'un nombre important de performatives permettant les conversations entre agents

2.1.1. Structure globale d'un message KQML:

Un message KQML est composé d'un performatif et d'un ensemble de paramètres, chacun prenant la forme d'une paire "mot-clé & contenu". Le contenu du paramètre est exprimé soit sous la forme d'un mot, soit sous la forme d'une expression dans un langage de contenu (cela dépend du type du paramètre). Les performatifs indiquent quel type d'acte de langage l'agent désire employer; ils définissent les opérations possibles que les agents peuvent demander sur les connaissances et les objectifs des autres agents.

La structure globale d'un message KQML est donnée comme suit:

```
(KQML-performative
        : Parameter <Word>
        : Parametre  <Word>
        : Parametre  <Word>

        …….

        : Content  < expression>
        ……)
```

Les performatifs peuvent être accompagnées de paramètres qui composent le contenu du message. Les paramètres sont optionnels, leur utilisation dépend du performatif utilisé. Ces paramètres sont:

Mots-clés	*signification*
: Sender	nom de l'agent qui envoi la performative
: Receiver	nom de l'agent qui reçoit la performative
: Reply-with	identificateur unique du message, en vue d'une référence ultérieure.
:in-reply-to	référence à un message auquel l'agent est en train de répondre (récupérer dans reply-with)
: Language	permet de préciser le langage dans lequel le message est exprimé dans le contenu : content
:Ontology	le nom de l'ontologie à laquelle se rapporte le contenu : content précise le vocabulaire des « Mots » utilisés dans le message.
: Content	l'information à propos ce que le performatif exprime.

Paramètre des messages KQML

En KQML, les agents communicants sont supposés implémenter selon différents langages de programmation et selon différentes méthodes et techniques de représentation des connaissances. Pour pouvoir communiquer, tous les agents considèrent que leurs partenaires de dialogue possèdent une représentation des connaissances [Wool02]. Les connaissances attribuées par chaque agent aux autres agents sont vues en KQML comme une base virtuelle de connaissances - BVC (en anglais *Virtual Knowledge Base*). Dans ce qui suit, les abréviations suivantes seront utilisées :

E - l'agent émetteur
R - l'agent récepteur
C - le contenu du message
BVC - la base virtuelle de connaissances

Les performatifs de KQML peuvent être classifiés en trois grandes catégories :

- discours - performatifs utilisés pour l'échange d'informations et connaissances (tel, untell, ask-if, ask-one, ask-all, stream-all, deny, insert, delete-one, delete-all, uninsert, undelete etc.)
- interconnexion entre les agents pour faciliter l'obtention des informations et connaissances (broker-one, recommend-one, recruit-one, broker-all, recommend-all, recruit-all, broadcast, register, unregister)
- exception - des performatifs qui modifient le flux des informations et connaissances (sorry, standby, error, ready, next, rest, discard etc.)

Les performatifs KQML et leur signification sont donnés dans le tableau suivant:

Performatif	***Signification***
`achieve`	E veut que R rende quelque chose **vrai** dans leur environnement
`advertise`	E fait connaître aux autres agents les capacités disponibles dans le traitement d'une performative
`ask-about`	E veut connaître toutes les propositions pertinentes dans les BVC de R
`ask-if`	E veut savoir si la réponse à la question précisée en C se trouve dans la BVC de R
`ask-one`	E veut que seulement R réponde à sa question C
`break`	E veut que R interrompe le flux établi par pipe
`broadcast`	E veut que R transmette à son tour la performative á tous ses connexions
`broker-all`	E veut que R collecte toutes les réponses á la performative
`broker-one`	E veut que R l'aide à répondre à une performative
`deny`	la performative ne peut pas être appliquée à E
`delete-all`	E veut que R efface toutes les propositions qui s'apparie avec C de sa BVC
`delete-one`	E veut que R efface une proposition qui s'apparient avec C de sa BVC
`discard`	E ne veut pas le reste des réponses de R à une interrogation
`eos`	termine un flux de réponses (en anglais stream) à une interrogation (en anglais eos vient de 'end of stream')
`error`	E considère le message précédent de R comme mal formé
`evaluate`	E veut que R évalue (simplifie) C
`forward`	E veut que R transmette le message vers un autre agent
`generator`	la même signification que `standby` de `stream-all`
`insert`	E demande à R d'ajouter C à sa BVC

Les performatives du KQML [Fini93]

Performatif	*Signification*
monitor	E veut que R actualise sa réponse à un stream-all
next	E veut la réponse suivante de R à un flux d'une performative
pipe	E veut que R transmette toutes les performatives futures à un autre agent
ready	E est prêt pour répondre à la performative mentionnée précédemment par R
Recommend-all	E veut tous les noms des agents qui peuvent répondre à C
Recommend-one	E veut le nom d'un agent qui peut répondre à C
recruit-all	E veut que R 'recrute' tous les agents capables de lui répondre à C
recruit-one	E veut que R 'recrute' un agent capable de lui répondre à C
register	E peut transmettre des performatives à un certain agent
reply	communique une réponse attendue
rest	E veut le reste des réponses de R à une interrogation nommée précédemment
sorry	R ne peut pas fournir plus d'information
standby	E veut que R soit prêt pour répondre à une performative
stream-about	une version réponse multiple de ask-about
stream-all	une version réponse multiple de ask-all
subscribe	E veut que R actualise par la suite sa réponse
tell	E affirme au R que C est dans la BVC de E
Transport-address	E associe un nom symbolique à une adresse de transport
unadvertise	l'action contraire de advertise
unregister	l'action contraire de register
untell	E affirme à R que C n'est pas dans la BVC de E

Les performatives du KQML (suite)

Exemples des dialogues KQML:

1. L'agent A1 demande à l'agent A2 le prix de l'imprimante HP-Jet et A2 lui répond.

```
 (ask-one
         :sender A1
         :receiver A2
         :content (val (prix HP-Jet))
         :language KIF
         :ontology imprimantes
         )
(ask-one
         : sender A2
         : receiver A1
         : content (=(prix HP-Jet) (scalar 7000 DA))
         : language KIF
         : ontology imprimantes
         )
```

2. L'agent A1 demande à l'agent A2 toutes les informations concernant l'imprimante HP-Jet, et A2 lui répond par plusieurs messages qui se terminent avec un '*eos*':

```
(stream-about
         :sender A1
         :receiver A2
         :reply-with hpj
         :language KIF : ontology imprimantes
         :content HP-Jet
         )
(tell
         :sender A2 :receiver A1
         :in-reply-to hpj
         :content (=(resolution HP-Jet) (scalar 300 dpi))
         :language KIF : ontology imprimantes
         )
(tell
         :sender A2 :receiver A1
         :in-reply-to hpj
         :content (=(prix HP-Jet) (scalar 7000 DA))
         :language KIF : ontology imprimantes
         )


(eos
         :sender A2 :receiver A1
         :in-reply-to hpj
         )
```

2.2. Le langage de communication FIPA-ACL:

Récemment, la collaboration internationale des membres d'organisations universitaires et industrielles regroupés au sein de FIPA (Foundation for intelligent physical agents) a permis de spécifier des standards dans la technologie agent et vise à favoriser l'interopérabilité des applications, des services et des équipements informatiques basés sur le paradigme agent.

Ils ont défini un certain nombre de spécifications principales d'agents. Notamment un standard de langage de communication agents ACL (Agent Communication language) [Fipa97] a été proposé et spécifié. Comme KQML, ce dernier est basé sur la théorie des actes de langage: les messages sont des actions ou des actes communicatifs, car ils sont prévus pour effectuer une certaine action en vertu de l'envoi. Les spécifications décrivent chaque acte communicatif avec une forme narrative. Elles fournissent également la description normative d'un ensemble de protocoles d'interaction de haut niveau, y compris la demande d'action, l'établissement de contrat (contract net) et plusieurs genre de ventes aux enchères.

Le langage FIPA-ACL a une syntaxe similaire à KQML, et s'appuie sur la définition de deux ensembles :

- un ensemble d'actes de communication primitifs, auquel s'ajoutent les autres actes de communication pouvant être obtenus par la composition des ces actes de base
- un ensemble de messages prédéfinis que tous les agents peuvent comprendre

FIPA-ACL possède 21 actes communicatifs, exprimés par des performatifs, qui peuvent être groupés selon leur fonctionnalité 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 primitifs ou composés. Les actes communicatifs primitifs sont définis de façon atomique, c'est-à-dire qu'ils ne sont pas définis à partir d'autres actes (dans la classification ci-dessus ils sont suivis d'une étoile - "*"). En revanche, les actes communicatifs composés sont 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.

2.2.1. Structure d'un message:

Un message comprend plusieurs éléments qui sont présentés dans le tableau suivant:

Elément	***Signification***
performative	type de l'acte communicatif
sender	l'émetteur du message
receiver	le destinataire du message
Reply-to	participant à l'acte de communication
content	le contenu du message (l'information transportée par la performative)
language	le langage dans lequel le contenu est représenté
encoding	décrit le mode d'encodage du contenu du message
ontology	le nom de l'ontologie utilisé pour donner un sens aux termes utilisés dans le *content*
Protocol	contrôle la conversation
conversation-id	identificateur de la conversation
Reply-with	identificateur unique du message, en vue d'une référence ultérieure
in-reply-to	référence à un message auquel l'agent est entrain de répondre (précisé par l'attribut `reply-with` de l'émetteur)
Reply-by	impose un délai pour la réponse

Structure d'un message FIPA-ACL

Exemple de message FIPA-ACL

```
(inform
         :sender A
         :receiver B
         :reply-with laptop
         :language KIF
         :ontology ordinateurs
         :content (=(prix HP-Jet) (scalar 150 000 DA))
         :reply-by 10
         :conversation-id conv01
         )
```

2.3. Le langage d'interaction "IL"(approche MAGMA):

Le langage d'interaction entre agents "IL" [Dema95] est basé sur une formalisation des messages sur un langage inspiré de la théorie des actes de langages. La capacité d'interaction repose sur l'implémentation de protocoles d'interactions qui sont associés aux langages d'interactions.

Le protocole d'interaction est modélisé par un graphe d'état prédéterminé dans lequel l'agent évolue en fonction des messages qu'il reçoit. Il conditionne ainsi le comportement de l'agent en le renseignant sur le type de message qu'il doit émettre à son tour. Dans cette approche, les protocoles utilisés sont transmis dans le message pour éviter tout indéterminisme local aux agents

L'exécution d'une interaction entre des agents est guidée ainsi par le protocole utilisé qui est défini, à priori, par l'ensemble des transitions possibles qui relient un ensemble d'états finis que l'agent peut occupé.

Ce langage est organisé en trois couches: communication, multi-agents et domaine d'application.

<Interaction> = <Communication> <Multi-agent> <Application>.

Le sous-langage communication est composé de champs utilisés pour le niveau de communication du système. Il décrit les différents paramètres de communications utilisés pour le routage des messages.

<communication> = <from> <to> <id> <via> <mode>

- <from> : identifiant de l'émetteur;
- <to> : identifiant du destinateur;
- <Id> : identifiant du message;
- <Via> : type de canal de communication utilisé; transfert de données à haut débit, passage de message par boite aux lettres, …
- <Mode> : le mode de communication (synchrone ou asynchrone)

Le sous-langage Multi-agent est utilisé pour expliciter l'intention de l'agent émetteur, ainsi que ses attentes. Dans ce sous-langage sont comprises les informations qui concernent le niveau multi-agent de l'interaction.

<multi-agent> = <type> <nature> <force> <protocole> <position>.

- <type> : est exprimé selon 4 types de primitifs: Present, request, answer et inform.
- <nature> : fait la distinction du type d'information échangé: connaissances (ou hypothèses), solutions possibles (ou plans) et choix (ou actions choisies).
- <force> : la force illocutoire qui définit la priorité de l'acte de communication pour le receveur (informing, information seeking, expressing, warning, persuading et commanding).
- <protocole> : indique quel est le protocole d'interaction employé dans l'interaction en cours.
- <position> : indique quel est l'état de déroulement du protocole dans l'interaction en cours.

Le sous-langage application doit être adapté au domaine d'application. L'approche choisie par les concepteurs du langage IL est de ne pas standardiser un langage pour traiter des aspects liés à l'application, mais de donner suffisamment d'éléments dans les autres sous-langages (de communication et Multi-agents) pour que l'interprétation du langage de contenu soit plus directe.

2.4. Les langages de contenu des messages:

Le contenu des messages KQML ou FIPA-ACL est exprimé dans un langage de contenu. Parmi les langages utilisés comme langage de contenu des messages, on peut cité le langage KIF (Knowledge Interface Format) basé sur la logique du premier ordre, le langage PROLOG, le langage FIPA-SL, etc… . Nous présentons dans cette partie les langages de contenu les plus utilisés, le langage FIPA-SL et le langage KIF.

2.4.1. Le langage FIPA-SL:

FIPA-SL (Semantic Language) est le langage formel employé pour définir la sémantique des performatives de FIPA-ACL [FIPA97]. SL utilise une logique multi-modale avec les opérateurs modaux pour les croyances (B), les désirs (D), les croyances incertaines (U) et les intentions (buts persistants). SL peut représenter des propositions, des objets, et des actions. Une description détaillée de SL, y compris sa propre sémantique peut être trouvée dans les spécifications de FIPA-ACL [FIPA97].

Le modèle mental d'un agent est basé sur la représentation de trois attitudes primitives: croyance, incertitude et choix (ou, dans une certaine mesure, but). Elles sont respectivement formalisées par les opérateurs modaux B, U, et C.

B_i(p) : "i croit (implicitement) que p est vrai"
U_i(p) : "i pense que p est plutôt vrai que faux "
C_i(p) : "i souhaite que p soit vrai. Pour ce faire, l'agent établit un plan d'actions lui permettant d'atteindre le but désiré p"

Les composantes des actes de communication planifiés au sein d'un agent caractérisent à la fois le but pour lequel l'acte est choisi et les conditions qui doivent être satisfaites pour que l'acte soit exécuté. Pour un acte donné, le premier est désigné

sous le nom de l'effet rationnel (*rational effect*), et le dernier comme pré-conditions de faisabilité (*feasibility preconditions*) qui sont les qualifications de l'acte. Ainsi, chaque acte de communication est défini de cette manière, ce qui permet aux agents de raisonner et d'avoir un comportement rationnel cohérent.

La grammaire de ce langage est donnée en Annexe A.

2.4.2. Le langage KIF (Knowledge Interchange Format):

L'objectif de KIF est de définir un format standard de représentation des connaissances manipulées par les agents qui soit suffisamment générique pour être utilisé dans différents domaines, mais suffisamment précis pour ne pas conduire à des interprétations différentes.

KIF est un langage logique capable de décrire des faits concrets, des définitions, des abstractions, des règles d'inférence, ou des contraintes. Il a été conçu pour être utilisé dans le cadre des agents mais aussi celui des systèmes experts ou encore des bases de données.

KIF est une version préfixée du calcul formel des prédicats du 1er ordre, étendue pour supporter définitions et raisonnements non-monotones. Il inclut une spécification de sa syntaxe et une autre pour sa sémantique.

Il peut exprimer des expressions comme par exemple le fait que la *forme.1* est plus grande que la *forme.2* :
*(> (* (largeur forme.1) (longueur forme.1))*
*(* (largeur forme.2) (longueur forme.2)))*

On peut inclure des opérateurs logiques dans les formules, comme dans l'assertion suivante qui dit que l'élévation de n'importe quel réel à une puissance paire fournie un nombre positif :

```
(⇒   (and  (nombre-réel   ?x)
           (nombre-pair   ?n))
     (>     (résultat     ?x  ?n) 0))
```

3. Protocoles d'interaction:

Dans la section précédente, nous avons présenté les langages de communication inter-agents, ces langages sont basés sur la théorie des actes de langage. La théorie des actes de langage ne prend en compte que l'acte isolé et ne traite pas les enchaînements des actes de communications entre les interlocuteurs.

Les communications inter-agents ne permettent généralement pas de faire émerger aisément des enchaînements conversationnels. Le principe adopté par la majorité des chercheurs est plutôt une description explicite des enchaînements en utilisant la notion de protocole d'interaction (ou de conversation)

La raison pour laquelle a été adopté le terme protocole d'interaction (parfois conversation) utilisé dans les SMA, est qu'il distingue clairement ces protocoles des protocoles de communication standards qui existent dans d'autres domaines comme les systèmes répartis ou les réseaux d'ordinateurs [Koni95].

Les protocoles d'interaction sont utilisés dans les SMA afin de contrôler et de structurer les échanges d'informations entre agents de façon à réduire le temps de communication et la qualité d'information échangée. Ils définissent le contexte dans lequel un message est envoyé, simplifiant la tâche du récepteur au moment de l'interprétation: autrement dit, l'acte illocutoire est explicite par l'émetteur, sans l'utilisation de protocoles, le récepteur doit inférer l'intention de l'émetteur par rapport au message.

Par ailleurs, les protocoles définissent comment un agent peut réagir à un message, ou bien ce qu'il doit attendre après avoir envoyé un message.

Un protocole d'interaction est définit comme étant un réseau de transitions qui définit à priori l'ensemble des transitions possibles liant les états que les agents occupent alternativement par rapport aux interactions échangées.[Dema94]

Dans la suite, nous présentons d'abord les différents types de protocoles d'interaction qui ont été conçus pour les systèmes multi-agents, puis nous donnons quelques protocoles notamment ceux spécifiés par FIPA.

3.1. Classification des protocoles d'interaction:

Les agents communiquent entre eux dans les systèmes pour s'échanger des informations tout en ayant comme objectifs d'accomplir des buts qui leurs sont propres ou partagés. La communication peut permettre aux agents de coordonner leurs actions et d'aboutir ainsi à des comportements cohérents du système.

La coordination est une caractéristique essentielle dans les SMA. Elle permet par exemple aux agents d'éviter de conflit d'accès aux ressources, les attentes indéfinies et l'inter-blocage. La coopération est la forme de coordination entre agents non antagonistes; à l'inverse, la négociation correspond à la coordination en univers compétitif ou entre agents "égoïstes" ou concurrentiels. Typiquement, pour coopérer avec succès, chaque agent doit maintenir un modèle des autres agents, et développer également un modèle des futures interactions (planification). La figure 3.1 ci-dessous présente une taxonomie des différentes manières dont les agents peuvent coordonner leurs activités et comportements.

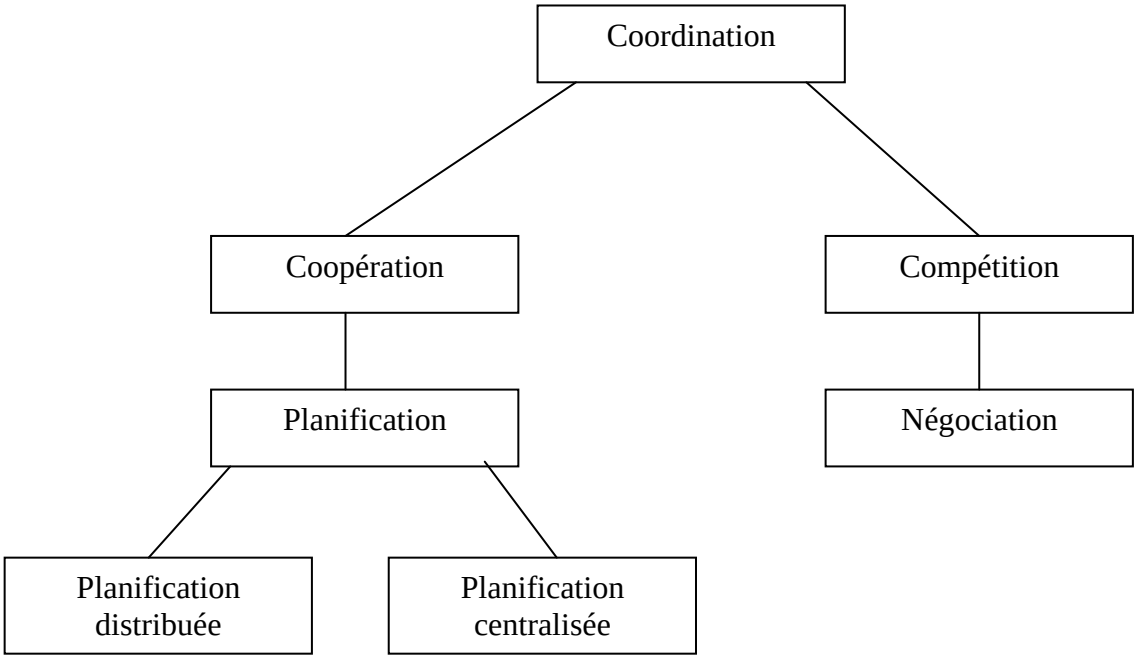


Fig. 3.1. Taxonomie des différentes formes de coordination entre agents cognitifs

L'objectif des protocoles s'inscrit dans deux types d'interactions:

- Agent concurrents: il faut maximiser l'utilité de chaque agent.
- Agent ayant des buts semblables ou problème communs: il faut maintenir des performances globalement cohérentes sans inhiber l'autonomie de chacun. Les aspects importants de ce type d'interaction sont:
    - Déterminer les buts partagés
    - Déterminer les tâches communes
    - Eviter des conflits
    - Mettre en évidence et partager la connaissance

Dans la suite, nous décrivons les différents types de protocoles d'interaction existant.

3.1.1. Protocole de coordination:

Dans des environnements à ressources limitées, la coordination se traduit par un comportement individuel visant à servir ses propres intérêts tout en essayant de satisfaire le but global du système. Les actions des agents doivent être coordonnées pour plusieurs raisons:

- il y a des dépendances entre les actions des agents.
- Aucun agent n'a suffisamment de compétences, de ressources et d'information pour atteindre tout seul le but du système complet ou son propre but.
- Eviter les redondances dans la résolution de problème.

Dans la plupart des recherches en IAD se sont concentrées sur des techniques pour distribuer le contrôle et les données. Le contrôle distribué que les agents ont un degré

d'autonomie leur permettant de produire de nouvelles actions et décider quels sont les buts à poursuivre. L'inconvénient de la distribution de contrôle et des données est que la connaissance de l'état global du système est dispersée sur tous les agents où chaque agent a seulement une perception partielle imprécise. Il y a un plus grand degré d'incertitude au sujet des actions de chaque agent ce qui rend difficile l'atteinte d'un comportement global cohérent.

Un exemple de protocole très connu pour la coordination est celui du réseau contractuel (contract-net) [Smit80]. Initialement, il a été utilisé pour l'allocation des tâches. Une session typique pour un tel protocole peut être comme suit: un agent ayant une tâche à faire exécuter (manager) l'annonce aux autres (contractants). Ces "contractants" sont des agents dans le même système qui soumettront des "offres" (pour le cas de l'allocation de tâches, les propositions d'offre sont par exemples le temps estimé par chaque agent pour exécuter la tâche) au manager. Selon, les critères de sélection du manager (par exemple le temps d'accomplissement le plus court), un "contrat" sera attribuer à un des agents. Une fois qu'un contrat est établi, les deux parties devront respecter les règles du contrat dans la suite de l'interaction et les autres agents du système cesseront la participation au processus.

Point de vue Manager	*Point de vue contractant*
Annonce d'une tâche qui doit être réalisée	Réception de l'annonce d'une tâche évaluation de sa capacité à répondre
Réception et évaluation des offres des autres agents	Réponse (offre, rejet)
Etablissement d'un contrat avec l'agent le plus adapté	Accomplissement de la tâche si offre acceptée
Réception et synthèse des résultats	Envoi des résultats

L'intérêt d'un tel protocole réside dans le fait qu'il réalise non seulement la coordination de tâches parmi les agents, mais s'assure également que l'allocation est faite de façon optimale.

3.1.2. Protocole de coopération:

Une stratégie de base partagée par plusieurs protocoles pour la coopération est de décomposer un problème en tâches puis les distribuer. Une telle approche peut réduire la complexité d'un problème: des tâches exigent moins de capacités des agents et peu de ressources. Cependant, le processus de décomposition doit prendre en considération les ressources et les capacités de chaque agent. Egalement, il pourrait y avoir des interactions entre les tâches et par conséquent des conflits entre les agents. La décomposition peut être effectuée par le concepteur, par les agents en utilisant une planification hiérarchique ou être directement liée à la nature du problème. La décomposition peut être spatiale ou fonctionnelle.

Une fois la décomposition réalisée, il faut distribuer les tâches selon les critères suivants:

- éviter la surcharge des ressources critiques.
- Assigner les tâches aux agents ayant des capacités correspondantes.
- Assigner des tâches interdépendantes à des agents proches spatialement ou sémantiquement pour limiter les coûts de communication et de synchronisation.
- Réassigner des tâches pour accomplir les plus urgentes.

Plusieurs mécanismes sont utilisés pour distribuer des tâches. En voici quelques uns:

- Mécanisme d'élection: des tâches sont attribuées aux agents suite à un accord ou vote,
- Réseaux contractuels: cycles d'appels d'offres, de propositions et attribution à un agent,
- Planification multi-agents: les agents planificateurs ont la responsabilité de la distribution des tâches,
- Structure organisationnelle: les agents ont des responsabilités pour des tâches particulières.

3.1.3. *Protocoles de négociation:*

Une forme fréquente de l'interaction qui se produit entre des agents avec des buts différents est la négociation. La négociation est un processus par lequel une décision commune est précise par deux agents ou plus où chacun d'entre eux essayant d'atteindre leurs buts ou objectifs propres. Les agents communiquent d'abord leurs positions, qui pourrait être source de conflit, et essayant ensuite de s'orienter vers un accord en faisant des concessions ou en recherchant des solutions alternatives.

Les dispositifs principaux de la négociation sont:

- le langage utilisé par les agents participants
- le protocole suivi par les agents dans le processus de négociation
- la procédure de décision que chaque agent utilise pour déterminer ses positions, concessions, et critères pour l'accord.

Les techniques de négociation peuvent être centrées environnement ou centrées agent:

- Centrées environnement: comment doit-on créer les règles de l'environnement pour que les agents interagissent de façon productive?

- Centrées agent: étant donné un environnement dans lequel un agent doit évoluer, quelle est la meilleure stratégie à suivre?

Le mécanisme de négociation doit avoir les caractéristiques suivantes:

- Efficacité (pas de perte de ressources pour aboutir à un accord)
- Stabilité (un agent ne doit pas dévier de sa stratégie)
- Simplicité (faible coût en calcul et en bande passante)
- Distribution (le mécanisme ne doit pas nécessiter un agent de décision central)
- Symétrie

Dans les systèmes multi-agents, les agents sont équipés de protocoles d'interaction, où chaque agent peut choisir sa propre stratégie. Ceci permet aux agents d'être construits par des concepteurs différents et/ou représentant chacun différentes partie du monde. On ne peut pas assumer que les agents utilisent des stratégies coopératives de négociation, mais à la place, chaque agent essayera d'employer une stratégie qui satisfasse le plus possible ses propres objectifs sans se soucier de ceux des autres agents.

3.2. Protocoles de FIPA:

Cette section présente quelques protocoles de FIPA. Cette liste de protocoles est non exhaustive et ces derniers peuvent être appliqués à n'importe quelle application. Chaque protocole est désigné par un nom prédéfini. La condition pour adhérer aux spécifications FIPA est la suivante: dans la conception d'agents communicants à base de FIPA-ACL, il n'est pas forcément nécessaire de mettre en application les protocoles standardisés de FIPA, et il est possible d'utiliser d'autres noms de protocoles. Cependant, si un nom standard de protocole est utilisé, l'agent doit se comporter conformément aux spécifications du protocole en question.
Il est à noter que, par leur nature, les agents peuvent s'engager dans des dialogues multiples, peut-être avec différents agents, simultanement. Le terme conversation est utiliser pour dénoter n'importe quel exemple particulier (instance) d'un tel dialogue. Ainsi, l'agent peut concurremment être engagé dans des conversations multiples, avec différents agents, dans différents protocoles.

Lors d'une interaction entre deux agents E et R, le choix du protocole est effectué comme suit: Envoi d'un message *inform*, par l'agent E, pour désigner le nom du protocole via le paramètre *protocol* signifiant ainsi l'utilisation du protocole par l'agent E. Si l'agent recevant un message dans le contexte protocole qu'il ne connaît pas, ce dernier renvoie un message de refus (*refuse*) en expliquant la cause.

Par convention un agent qui s'attend à un certain ensemble de réponses dans un protocole, et qui reçoit un autre message autre que ceux prédéfinis, devrait répondre avec un message *not-understood*. Pour éviter les boucles infinies de messages, il n'est pas permis de répondre à un message *not-understood* par d'autres messages *not-understood* .

Dans la suite, nous donnons une description de quelques protocoles FIPA. Les Rectangles blancs représentent l'action exécutée par l'initiateur. Et les Rectangles noirs représentent les actions exécutées par les autres participants dans le protocole

3.2.1. Protocole FIPA-request :

Le protocole FIPA-request permet simplement à un agent d'inviter d'autres agents à exécuter une certaine action. L'agent récepteur peut accepter comme refuser l’action. S'il accepte, il doit exécuter l’action et répondre par un *inform*.

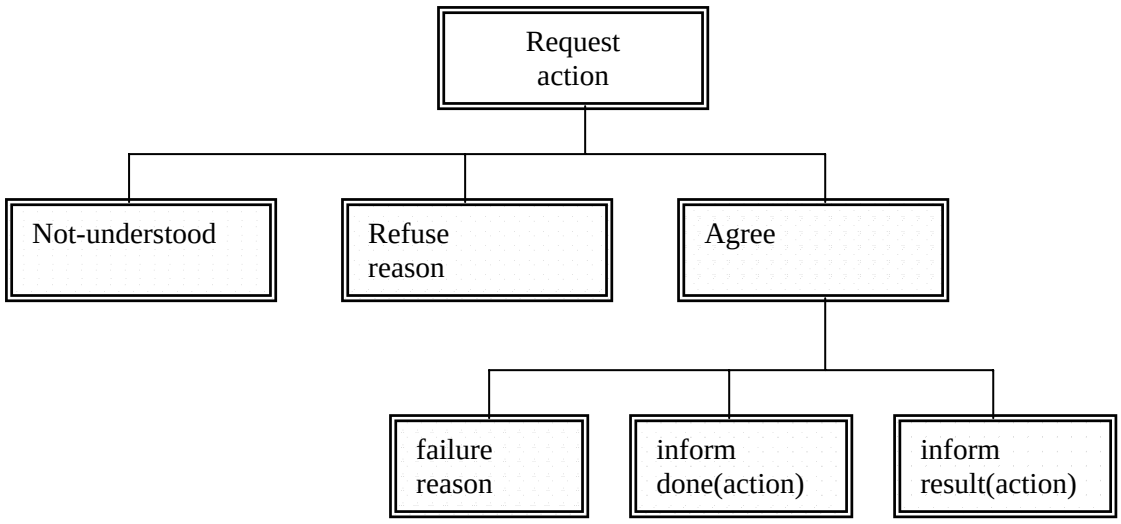


Fig 3.2. Protocole FIPA-request

3.2.2. Protocole FIPA-query :

Dans le protocole de FIPA-query, un agent cherche à connaître un objet correspondant à une description ou interroger un autre agent sur une proposition ou l'état sur 1'exécution d'une action. La demande d'être informé est une requête qui se traduit par deux actes possibles : Query-if (questionner -si une proposition est vraie) et Query-ref (demander la liste d'objets correspondant a une référence ou a une description). L'un ou l'autre acte peut être utilisé pour lancer ce protocole. Le récepteur projettera de renvoyer la réponse à la requête par l'acte inform.

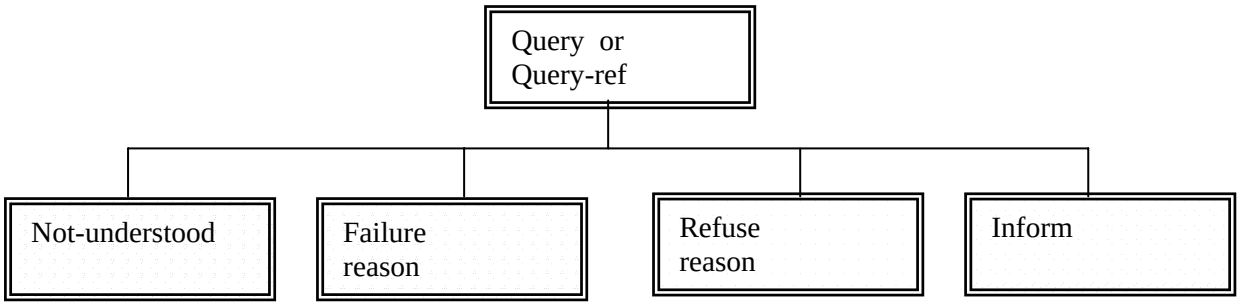


Fig. 3.3. Protocole FIPA-query

3.2.3. Protocole FIPA-request-when :

Le protocole FIPA-request-when est une autre manière d'exprimer une demande d'exécution d'une action. L'agent utilise l'acte request-when d'une action pour demander à un autre agent d'exécuter une certaine action à l'avenir une fois la condition préalable donnée devient vraie. Si 1'agent destinataire comprend la demande et ne refuse pas, il attendra jusqu'à ce que la condition préalable soit vérifiée, exécute alors l'action, après quoi il informera le demandeur que l'action a été exécutée. Notons que ce protocole est quelque peu redondant dans le sens où l'action demandée implique d'informer 1'agent demandeur dans tous les cas. Si

ultérieurement, il devient impossible d'exécuter l'action par le destinataire (occupé par d'autres taches plus prioritaires), il envoiera un message de refus au demandeur initial.

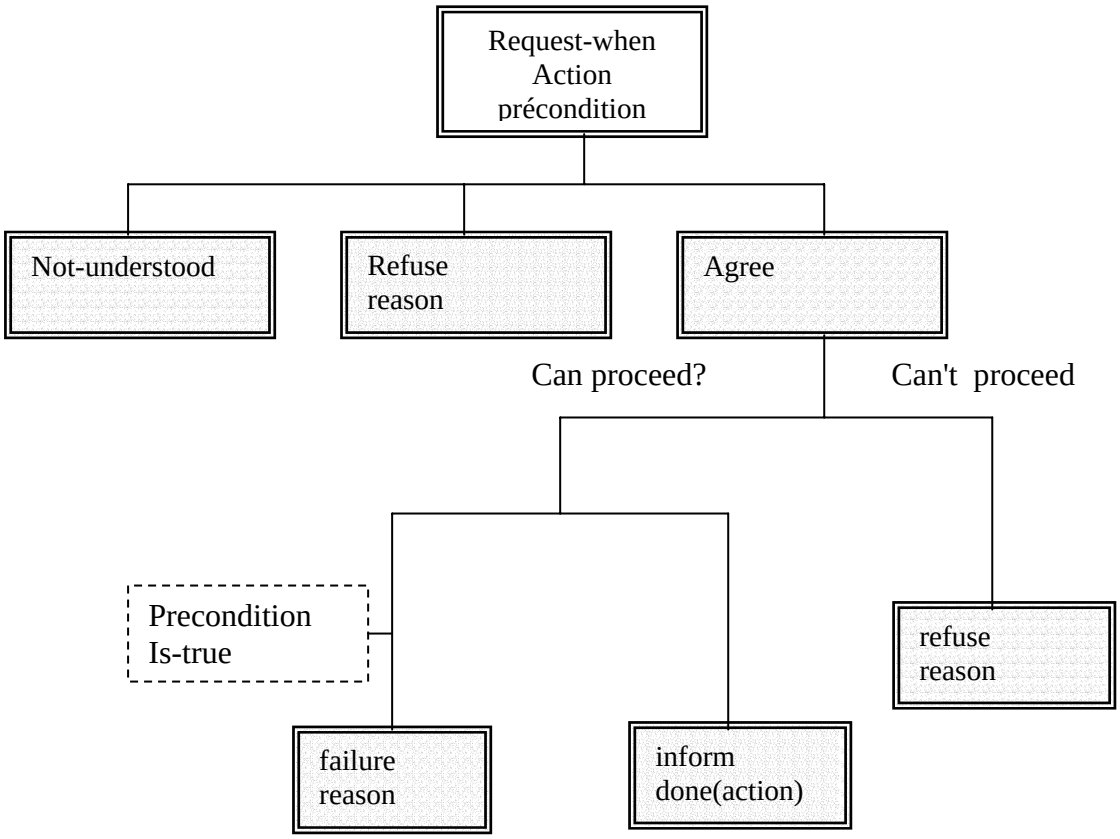


Fig. 3.4.Protocole FIPA-request-When

3.2.4. Le protocole FIPA contract-net :

Cette section présente une version du protocole Contract-Net, initialement développé par [Smit80]. FIPA-Contract-Net est une modification mineure du protocole initial car il ajoute des actes communicatifs de rejet et de confirmation. Dans le protocole de Contract-Net, un agent prend le rôle du manager. Le manager souhaite faire accomplir une certaine tâche par un ou plusieurs autres agents, et souhaite plus loin optimaliser une fonction qui caractérise la tâche. Cette caractéristique est généralement exprimée comme coût, spécifique au domaine, et pourrait être par exemple le temps nécessaire à l'accomplissement d'une tâche.

Le manager sollicite des propositions d'autres agents en émettant un appel d'offres, en décrivant la tâche et les conditions nécessaires à son exécution. Des agents recevant 1'appel sont des contractants potentiels, et peuvent fournir des propositions pour accomplir la tâche à travers l'acte *propose*. La proposition du contractant inclut les conditions préalables pour 1'exécution de la tâche, qui peuvent être le prix, le moment où la tâche sera faite, etc. Alternativement, le contractant peut refuser de faire de proposition. Une fois que le manager reçoit des réponses de tous les contractants, il évalue les propositions et fait son choix sur un, plusieurs ou aucun agent pouvant accomplir la tâche. Le ou les agents choisis seront notifies par un message d'acceptation, les autres recevront une notification de rejet. On suppose que le contractant s'engage à accomplir la tâche dès qu'il reçoit le message d'acceptation.

Une fois que le contractant ait terminé la tâche, il envoie un message d'accomplissement au manager. Notons que le protocole exige du manager de savoir le moment où il a reçu toutes les réponses. Dans le cas où un contractant ne répondrait pas, le manager peut potentiellement être en attente indéfiniment. Pour éviter ceci, la performative *Cfp* inclut une date-limite à laquelle des réponses devraient être reçues par le manager. Les propositions reçues après la date limite sont automatiquement rejetées en se justifiant par le fait que la proposition était tardive.

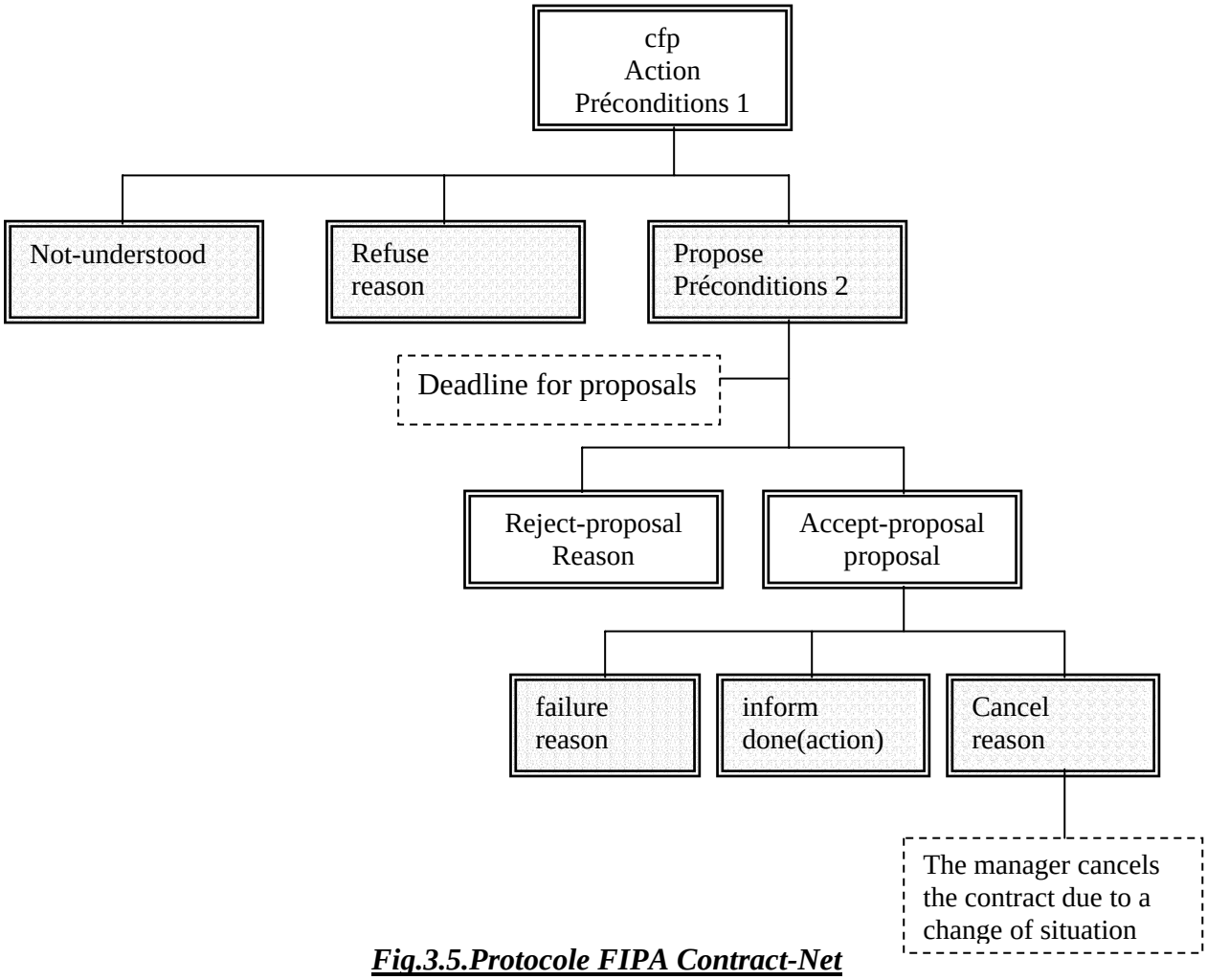


Fig.3.5.Protocole FIPA Contract-Net

3.2.5. Protocole FIPA-Iterated-Contract-Net :

Le protocole FIPA-Iterated-Contract-Net est une extension du contract-Net de base comme décrit ci-dessus. Il diffère de la version de base de FIPA-Contract-Net en permettant plusieurs itérations. Comme ci-dessus, le manager émet 1'appel aux propositions initiales avec 1'acte *Cfp*. Les contractants répondent alors par les actes *Propose*. Le manager peut alors accepter une ou plusieurs des offres, rejeter d'autres ou peut réitérer le processus en émettant une Cfp "révisée". L'objectif est de permettre au manager d'obtenir de meilleures offres des contractants en modifiant 1'appel et en demandant de nouvelles offres (d'une manière équivalente mais révisée). Le processus termine quand le manager refuse toutes les propositions, n'émet pas de nouveaux

appels en acceptant la meilleure offre de 1'itération antérieure ou tout simplement lorsque les contractants refusent de faire des offres.

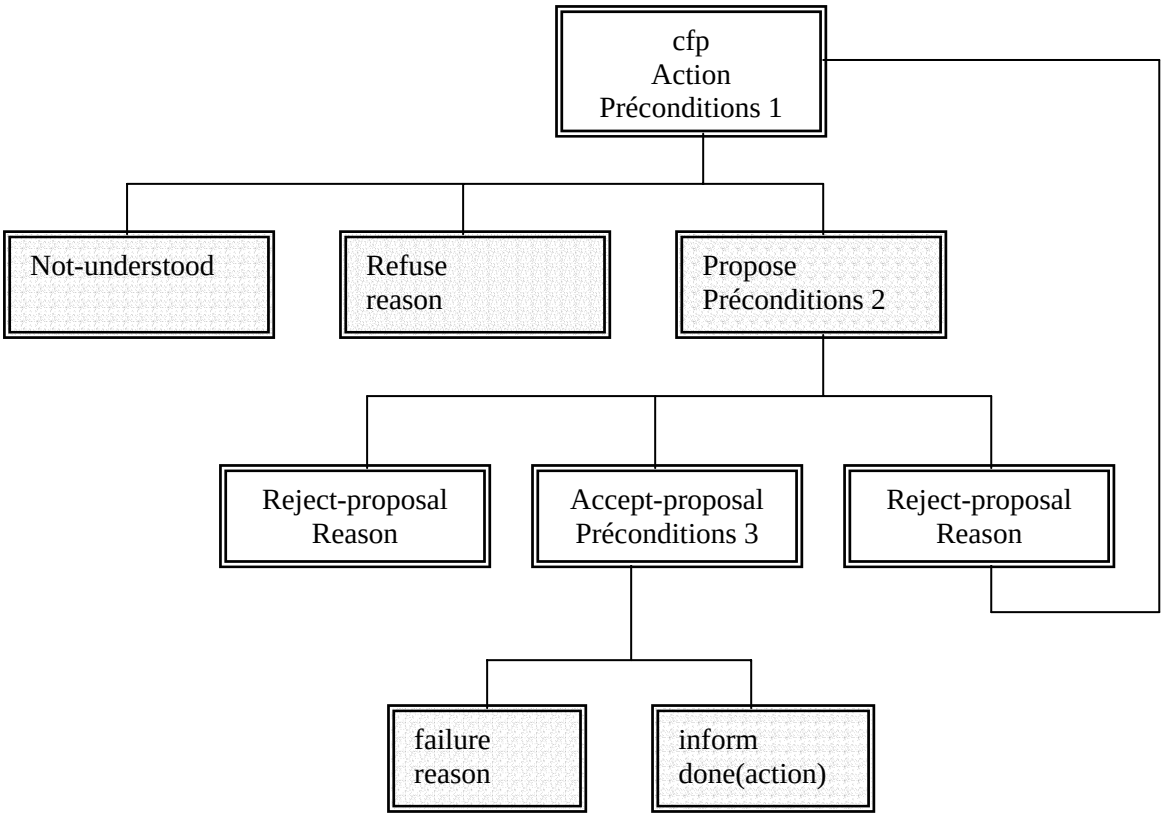


Fig. 3.6. Protocole FIPA Iterated-Contract-Net

3.2.6. Protocole FIPA-Auction-English (enchère anglaise):

Dans 1'enchère anglaise, le manager cherche à trouver le prix du marché (pratiqué) d'une marchandise en proposant initialement un prix au-dessous de celui de la valeur marchande supposée, et en augmentant alors graduellement le prix. Chaque fois que le prix est annoncé, le manager attend pour voir si des acheteurs signaleront leur bonne volonté d'accepter le prix proposé. Dès qu'un acheteur indique qu'il accepte le prix, le manager émet un nouvel appel d'offres avec un prix 1égèrement supérieur. L'enchère continue jusqu'à ce qu'aucun acheteur ne soit disposé à payer le prix proposé, auquel cas 1'enchère termine. Si le dernier prix qui a été reçu par un acheteur excède un prix de réserve (connu en privé par le manager), la marchandise est vendue à cet acheteur pour le prix convenu. Si le dernier prix accepté est moins que le prix de réserve, la vente est annulée.

Dans la figure, les appels du manager, exprimés comme l'acte général de *cfp*, sont "multi-cast " à tous les participants de l'enchère. Pour simplifier, seulement un exemple de message est présenté. Notons également qu'e dans la réalité, les acheteurs s'observent entre eux et donc chaque acceptation d'une offre est simultanément connue par tous les participants. Ceci peut ne pas être vrai dans un marché d'agents. Dans ce cas, il est possible que plusieurs agents proposent le même prix.

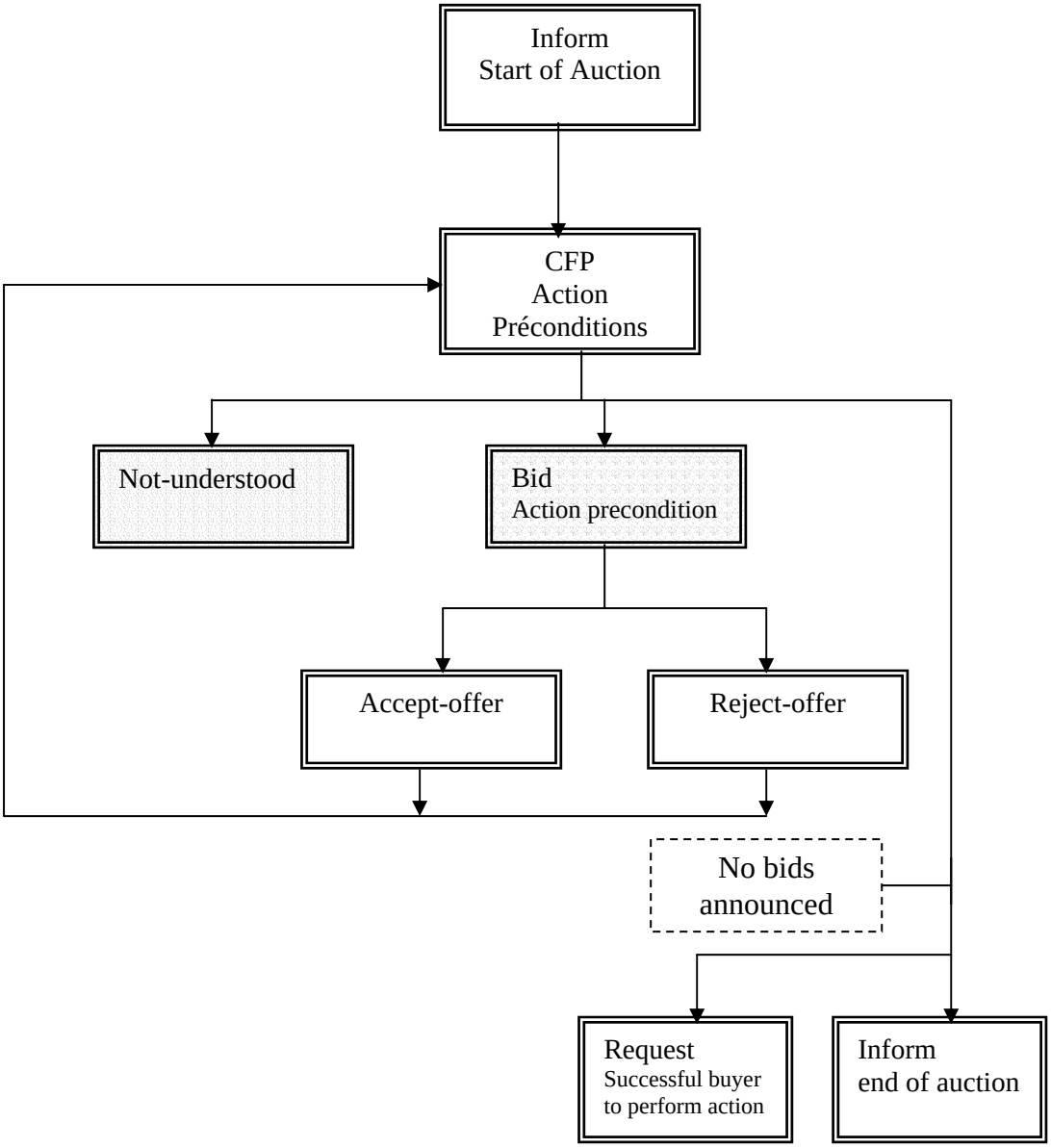


Fig. 3.7. Protocole FIPA-Auction-English

3.2.6. Protocole FIPA-Auction-Dutch (enchère hollandaise) :

Dans ce qui est appelé généralement 1'enchère hollandaise, les tentatives du manager de trouver le prix de vente d'une marchandise se concrétisent en commençant par un prix beaucoup plus haut que la valeur prévue, puis en réduisant progressivement le prix jusqu'à ce qu'un des acheteurs accepte le prix. Le manager a habituellement un prix de réserve au-dessous duquel il ne vend pas. Si 1'enchère ramène le prix de la marchandise au prix de réserve sans qu'il y ait d'acheteurs, 1'enchére se termine.

Le terme "enchère hollandaise" dérive des marchés de fleurs en Hollande, ou cette méthode est utilisée pour déterminer la valeur marchande des fleurs. Dans cette enchère quelques complexités supplémentaires apparaissent. D'abord, la marchandise peut être vendue en plusieurs parties: par exemple le manager peut vendre un lot de marchandise au prix X ou un acheteur peut intervenir et acheter seulement une partie du lot. L'enchère continue alors, avec un prix au prochain incrément au-dessous de X, jusqu'a ce que le reste du lot soit vendu ou le prix de réserve est rencontré. De telles ventes partielles des marchandises sont seulement présentes sur quelques marchés; dans d'autres, 1'acheteur doit acheter la marchandise entièrement. Deuxièmement, le mécanisme du marché de fleurs est utilisé pour s'assurer qu'il n'y a aucun conflit parmi les acheteurs vu que l'on n'accepte que la première offre.

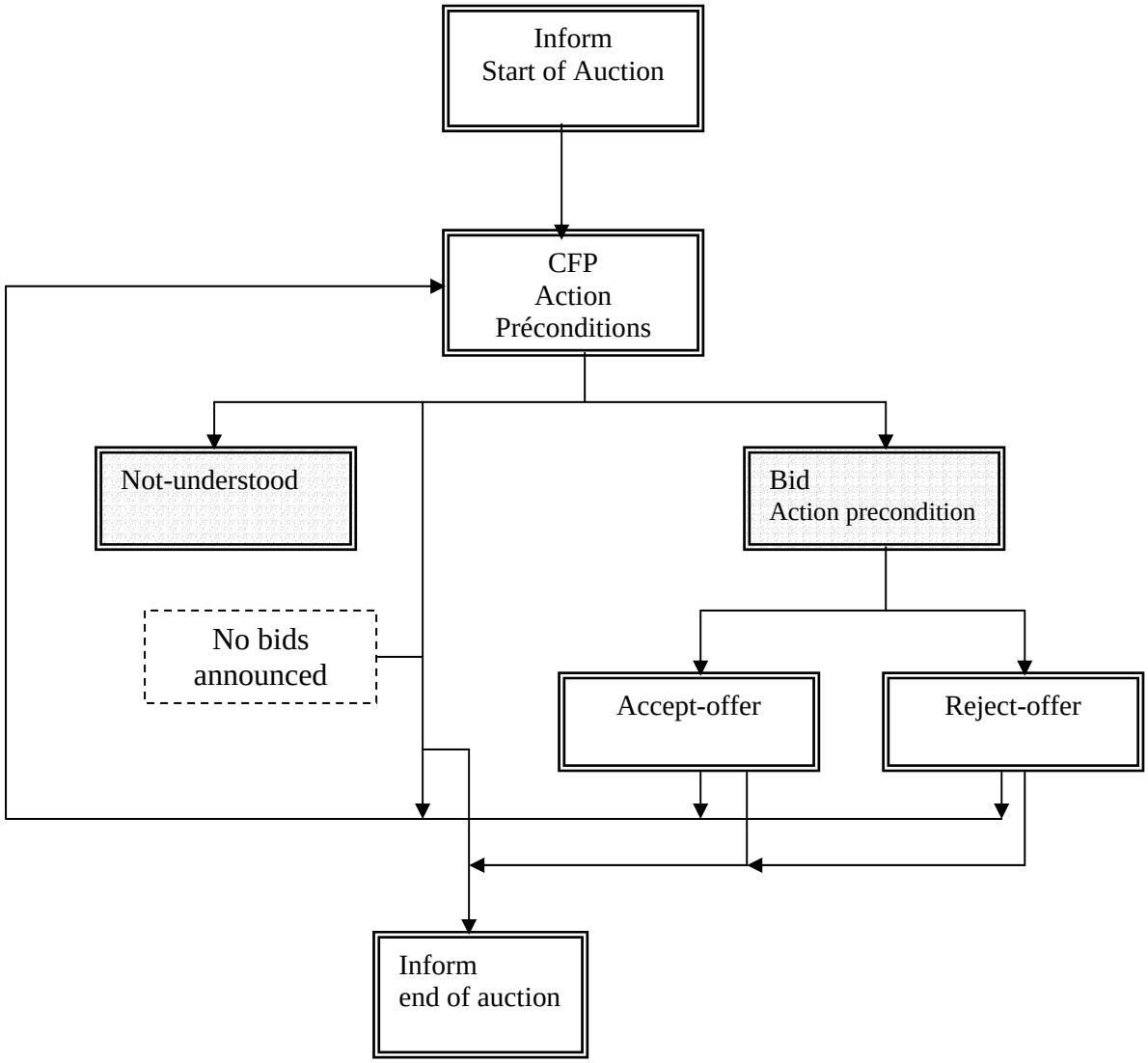


Fig. 3.8. Protocole FIPA-Auction-Dutch

4. Modèles d'interaction:

Dans cette section, nous présentons les principaux modèles utilisés dans l'interaction. Nous avons rassemblé ici ces modèles en trois groupes qui concernent la représentation des interactions, les architectures de systèmes permettant de mettre en œuvre les interactions et la dynamique des interactions.

4.1. Représentation des interactions:

Il existe plusieurs modèles qui sont utilisés pour représenter les interactions dans les SMA, comme les graphes d'état et les réseaux de petri. Ils ont souvent été inspirés d'autres domaines comme les systèmes répartis et les réseaux d'ordinateurs.

Le choix d'un modèle dépend surtout de son application et des caractéristiques telles que la représentation graphique utilisée, la sémantique du modèle et du raisonnement qu'il permet sur l'interaction représentée.

4.1.1. Représentation des protocoles par graphes d'états:

Généralement, dans la littérature, les graphes d'états sont choisis, comme modèle, pour la description des protocoles d'interaction. Les graphes d'états possèdent une représentation graphique simple et claire, ainsi qu'une sémantique bien définie fondée sur des automates d'états finis.

Dans le modèle des graphes d'états, une conversation est représentée par une séquence d'états, et par les transitions entre ces états. La transition entre les états est réalisée par l'émission ou la réception d'un message.

La représentation graphique des graphes d'états (avec des nœuds qui représentent les états et les arcs qui représentent les transitions entre états) est assez naturelle pour des interactions, comme on peut le constater dans le graphe d'états du protocole de Sian[Oude98] présenté à la figure 3.9 suivante:

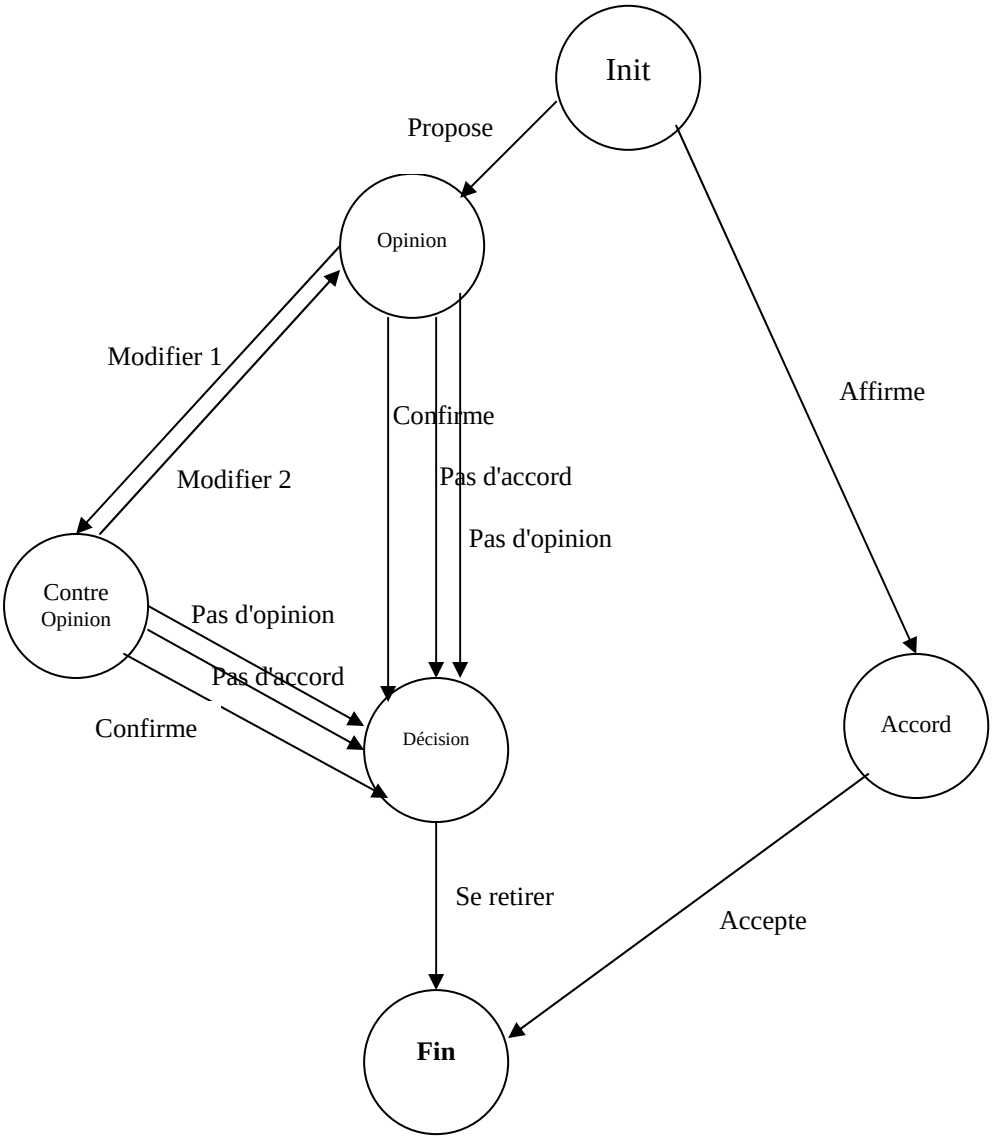


Fig.3.9. Graphe d'états du protocole Sian

La formalisation d'un graphe d'états se fait à travers un automate à états finis [Huge01], qui est défini sous la forme d'un 6-tuple **A = < E, q0, I, O S, T >**, où :

- **E = { q1, q2, …, qn }** est un ensemble fini d'états ;
- **q0** est l'état initial de l'automate;
- **I** est l'ensemble fini d'événements en entrée;
- **O** est l'ensemble fini d'événements en sortie;
- **S = E x I → O** est la fonction de génération de l'événement en sortie;
- **T = E x I → E** est la fonction de transition d'états.

4.1.2. Représentation des protocoles par le langage de description graphique AUML:

Une autre méthode de représentation graphique des protocoles d'interaction est apparue dernièrement. Il s'agit du langage AUML (Agent Unified Modeling Language) [Odel00] et [Auml00]. Ce langage tire son origine des diagrammes de séquence d'UML.

Chaque rôle d'agent est représenté par une barre verticale surmontée par une boîte qui contient le rôle de l'agent (voir figure). Une flèche permet de décrire un message d'un émetteur à un récepteur.

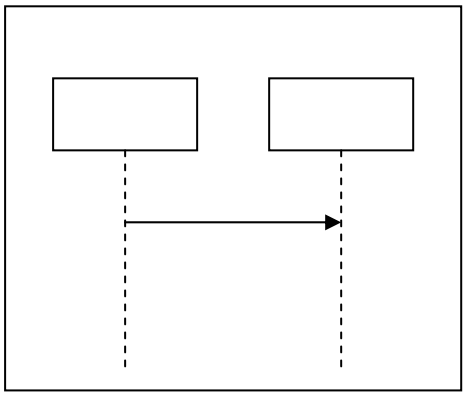


Fig.3.10. Représentation d'une interaction en AUML.

Il est possible sur un diagramme de séquence de préciser la durée de l'activation d'un message par un rectangle plus au moins grand sur la barre verticale (voir figure). Si un concepteur désire connaître l'ensemble des messages échangés après un message particulier, il recherche la barre d'activation de celui-ci et parcourt l'ensemble des flèches et des barres d'activation qui découlent de ce message.

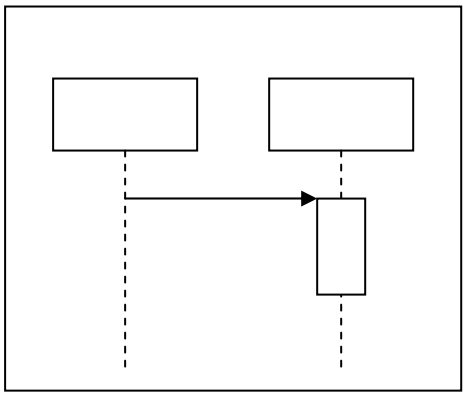


Fig.3.11. ure durée d'activation des messages en AUML

La gestion des boucles se fait par l'intermédiaire de while … loop… endloop (Cf. figure 3.12)

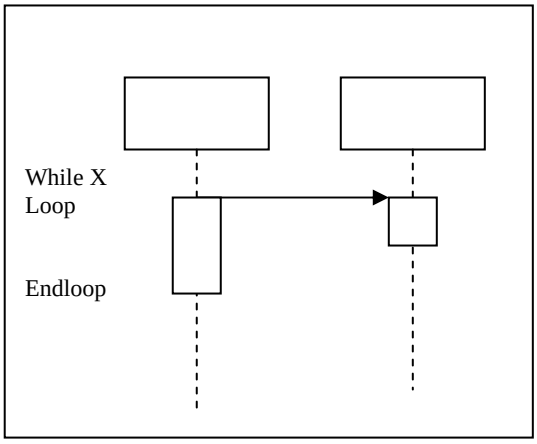


Fig.3.12. boucle en AUML

Il est même possible de trouver des boucles au niveau des messages à l'aide d'un astérisque (voir figure 3.13)

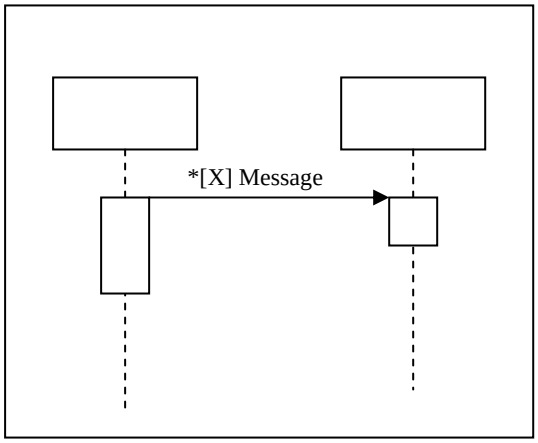


Fig.3.13. Autre représentation d'une boucle en AUML.

La condition de branchement est représentée par if…else… endif (Cf. figure 3.14)

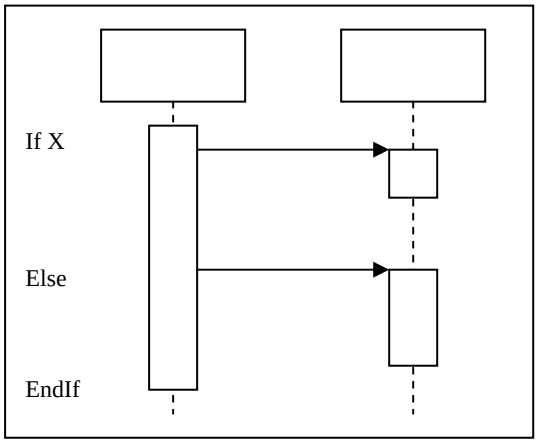


Fig.3.14. Un Branchement en AUML.

Il faut ajouter à cela trois connecteurs qui interviennent pour représenter l'envoi de manière concurrente (figure 3.15.a), le choix de zéro ou plusieurs messages parmi une liste (figure 3.15.b) et le choix d'un message parmi une liste (figure 3.15.c)

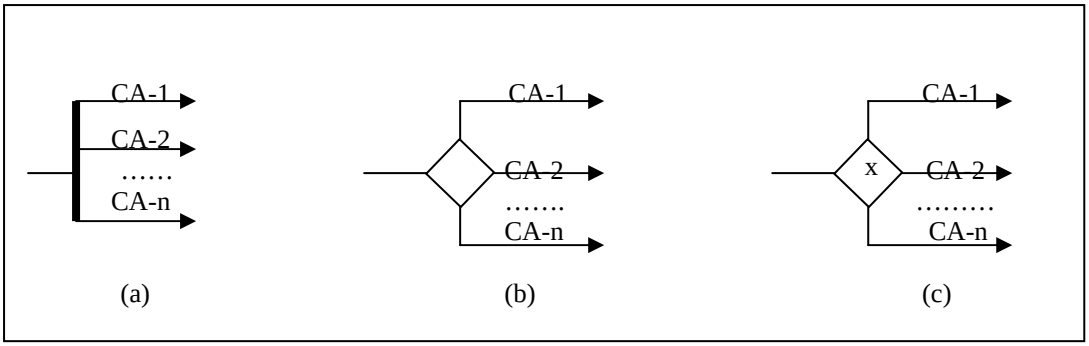


Fig.3.15. les connecteurs en AUML.

Le protocole Fipa-Request-When dans le langage AUML est représenté à la figure.

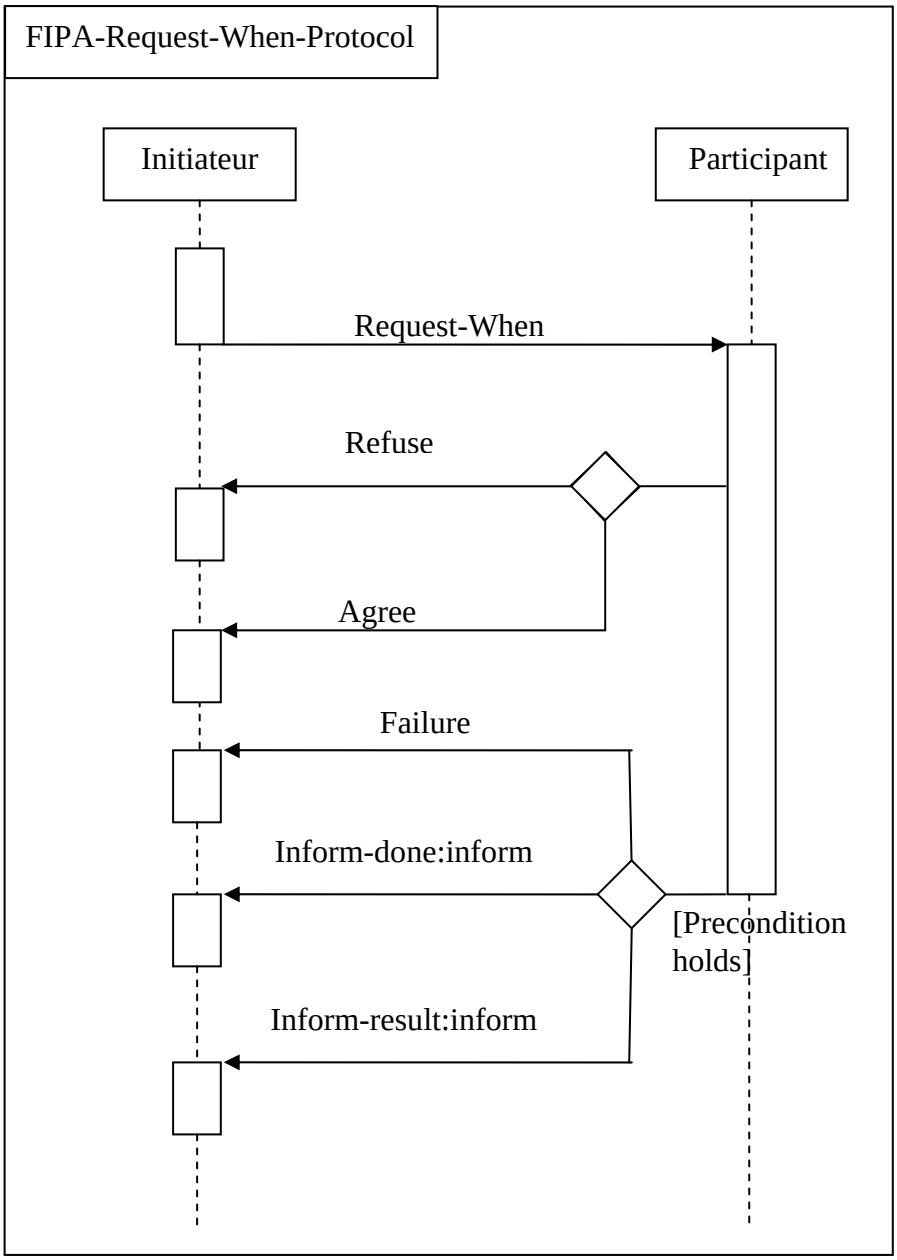


Fig.3.16. Le protocole Fipa-Request-When en AUML

Sur le cadre du diagramme, nous trouvons (dans le sens des aiguilles d'une montre) le nom du protocole et parfois une boite contenant de haut en bas, les rôles des agents impliqués, un commentaire et l'ensemble des messages utilisés dans le protocole d'interaction. L'astérisque, qui suit certains messages, fait référence au fait que le message pourra être émis 0 ou plusieurs fois.

Un autre modèle qui est souvent utilisé pour la représentation des interactions est les réseaux de Petri [Huge01]. Ce modèle présente un nombre plus important d'outils pour évaluer un protocole, mais par contre, sa représentation graphique est plus complexe et grandit vite au fur et à mesure que le protocole devient plus étendu ou dès que sont pris en compte plusieurs agents. Etant donné que notre intérêt est d'avoir une représentation graphique des protocoles et non de les concevoir ou de les analyser, nous estimons que la représentation par le langage AUML est plus adéquate.

4.2. Architecture de système pour la mise en œuvre d'une interaction:

Le modèle d'interaction utilisé dans un SMA peut imposer quelques restrictions au niveau de l'architecture du système. Ces contraintes prennent la forme de quelques composants (services) qui doivent être présents dans le système et de la façon dont ils interagissent au sein du système.

Les cas les plus répandus d'architecture de système SMA déterminés par le modèle d'interaction sont les facilitateurs (surtout dans le contexte de KQML) et les médiateurs. La plateforme FIPA [Fipa97] peut être considérée comme une architecture SMA déterminée par le modèle d'interaction. Les facilitateurs et les médiateurs sont présentés dans les paragraphes qui suivent.

4.2.1. Facilitateurs:

Les facilitateurs sont des agents qui fournissent des services en ce qui concerne la coordination d'autres agents. Ils possèdent un répertoire des services et capacités des agents et sont capable d'aider dans l'aiguillage du flux d'informations et dans la mise en relation des agents. Leur utilisation simplifie les connaissances dont les agents ont besoin à propos d'autres agents; ils peuvent se cantonner à faire appel aux facilitateurs, qui à leur tour, iront fournir des informations sur les autres agents ou sur les services dont ils ont besoin.

Les facilitateurs peuvent être définis comme étant un agent qui exécute plusieurs services de communication utile [Feni94]. Il peut par exemple maintenir un registre des services de noms, retransmettre des messages aux services des noms, faire le routage des messages en se fondant sur leur contenu, fournir le gestionnaire d'appariement entre les fournisseurs d'information et les clients, ou bien encore fournir les services de traduction et médiation.

Les facilitateurs peuvent être des agents intelligents ou simplement des pages jaunes.

Le concept de facilitateur a été particulièrement utilisé dans le contexte de KQML. Le langage lui-même prévoit l'existence d'agents qui sont capables d'informer des capacités des autres agents. Le langage prévoit des performatifs dédiés à l'échange d'informations à propos des capacités des agents et leurs besoins (par exemple les performatifs `subscribe` et `monitor`).

Il existe plusieurs façons d'utiliser les facilitateurs:
a. comme conseiller : l'émetteur demande au facilitateur les coordonnées des agents susceptibles de répondre à ses demandes (cf. Figure 3.17);

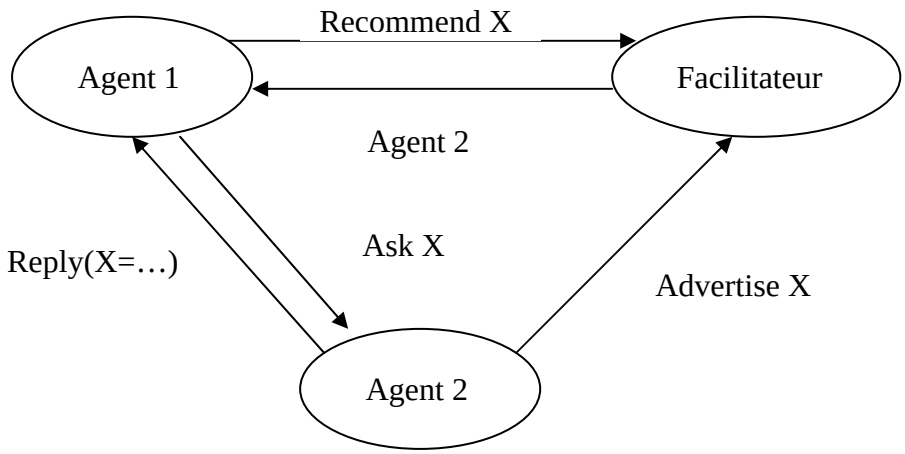


Fig. 3.17. Facilitateur comme conseiller

c. *comme recruteur* : l'émetteur envoie au facilitateur une demande, qui la renvoie aux destinataires correspondants, qui répondent alors directement à l'émetteur (cf. Figure 3.18)

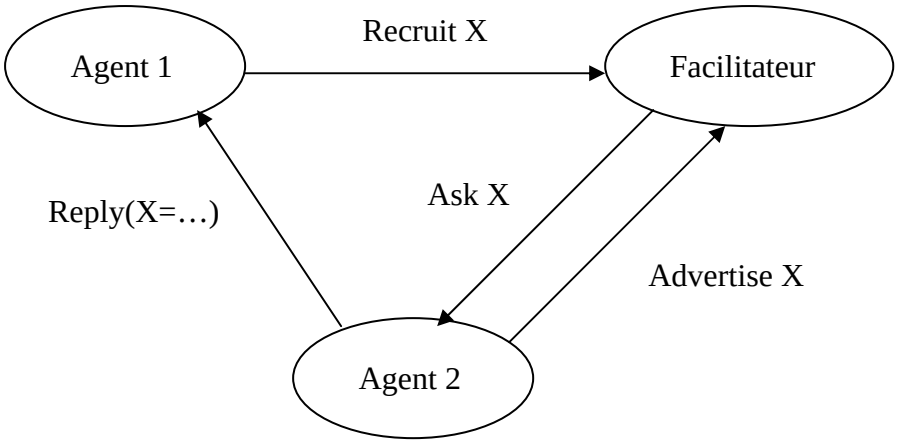


Fig.3.18. Facilitateur comme recruteur

c. comme courtier (« broker ») : le facilitateur s'interpose toujours entre l'émetteur et le récepteur (cf. Figure 3.19).

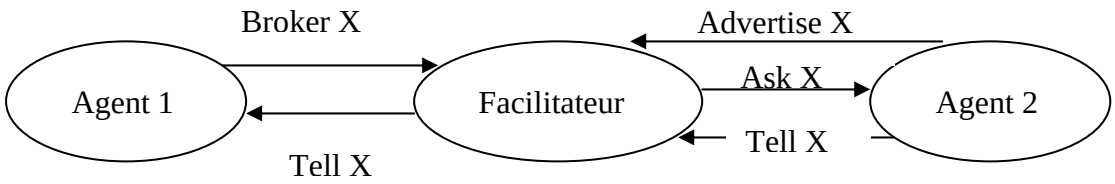


Fig.3.19. Facilitateur comme courtier

L'utilisation des facilitateurs facilite la gestion des connaissances à propos des services disponibles dans le système, néanmoins elle introduit d'autres problèmes puisqu'il faut savoir à quel facilitateur s'adresser pour obtenir une certaine information. L'utilisation d'un gestionnaire d'appariement (matchmaker) permet de résoudre ce problème au prix de la centralisation du facilitateur.[Ribe01]

Une variante de l'utilisation des facilitateurs consiste à utiliser seulement un facilitateur centralisé. Tous les messages sont envoyés à ce facilitateur central, qui les adresse vers l'agent approprié sans que l'émetteur ait besoin de le nommer explicitement. Cette approche cache la structure du système aux agents. Du point de vue de l'agent, il semble que le facilitateur possède une énorme base de connaissances, alors qu'il connaît seulement l'emplacement des connaissances. Dans le contexte de KQML, cela est connu sous le nom de *base de connaissances virtuelle*. Ce mécanisme rend plus simple les changements dans la structure du système minimisant les effets secondaires.

Le fonctionnement des gestionnaires d'appariement (matchmaker) est fondé sur un partenariat coopératif entre les fournisseurs et les consommateurs d'informations et de services, assistés par un facilitateur intelligent. Le fonctionnement des gestionnaires d'appariement se déroule en trois étapes :

- d'abord les fournisseurs d'informations et de services annoncent leurs capacités auprès du gestionnaire d'appariement, de cette façon ils jouent un rôle actif dans la recherche des consommateurs ;
- les consommateurs d'information et de services envoient leurs requêtes au gestionnaire d'appariement ;
- le gestionnaire d'appariement compare les annonces et les requêtes reçues et essaie d'identifier celles qui sont compatibles. Quand il trouve une paire qui se marie, il avertit les fournisseurs et consommateurs ou, selon la façon dont la requête a été posée, l'adresse directement au destinataire.

4.2.2. Médiateurs:

Il existe plusieurs approches à l'application du concept de médiateur dans les SMA. La plus utilisée est d'utiliser un agent médiateur qui s'interpose entre un service ou une source d'informations et les agents qui iront l'utiliser, de façon à présenter aux agents une interface plus adaptée à leurs besoins et à leur niveau.

Le médiateur médie les interactions entre différents agents appelés collègues [Kolp02]. Un initiateur adresse au médiateur une requête de service au lieu de directement l'adresser à d'autres collègues pouvant jouer le rôle de fournisseur du

service requis. Le médiateur possède des modèles d'accointance des collègues et coordonne la coopération entre eux. Inversement, chaque agent collègue possède un modèle d'accointance du médiateur. Alors qu'un courtier fait simplement la correspondance entre les participants fournisseurs de services et le client, initiateur de la requête, un médiateur encapsule les interactions et communication et maintient à jour continuellement les modèles d'accointance entre initiateurs et performeurs.

Le médiateur peut jouer le rôle d'un ambassadeur. Un ambassadeur médie un service requis par un acteur étranger à un agent local et gère la réponse. Si l'accès est autorisé, l'agent étranger peut soumettre des messages à l'ambassadeur pour traduction [Kolp02]. Le contenu du message est traduit en accordance avec l'ontologie standard du système dont fait partie l'agent local. Les messages traduits sont transmis aux agents locaux cibles. Les résultats de la requête sont transmis en retour à l'agent étranger et traduits en sens inverse.

4.3. Le modèle d'interaction dynamique:[Ribe01][Ribe98]

Le modèle d'interaction dynamique est un modèle où le mécanisme d'interaction joue un rôle dynamique, c'est-à-dire une partie de la gestion de l'interaction est déplacée de l'agent vers le mécanisme d'interaction. Ce modèle permet d'augmenter les possibilités d'interaction des agents sans pour autant augmenter la complexité des agents eux-mêmes. En effet l'augmentation des capacités du mécanisme d'interaction peut amener à concevoir des SMA où les agents peuvent avoir des interactions assez élaborées sans être eux-mêmes très complexes. Et l'augmentation des capacités d'interaction aide les agents à interagir même s'il s'agit d'agents hétérogènes (par rapport à l'interaction, quant les agents utilisent des bases d'interactions différentes c'est-à-dire des langages d'interaction entre agents, des langages de contenu ou des ontologies différentes).

Les modèles d'interaction statiques, prennent en charge uniquement le transport des messages. Ils sont dits statique par rapport au comportement du message, qui est passif. Le modèle d'interaction dynamique est un modèle où le mécanisme d'interaction joue un rôle actif dans le traitement des messages cela veut dire qu'au contraire de ce qui se passe avec les modèles d'interaction statiques, il agit sur le message. Dans un tel modèle le message évolue entre son émission et sa réception, permettant au message d'agir afin de s'assurer qu'il arrive à destination, ainsi que pour s'assurer qu'il soit correctement énoncé et interprété.

Le modèle d'interaction dynamique est fondé sur le partage de l'interaction entre l'agent et le mécanisme d'interaction. La distribution des tâches concernant l'interaction entre l'agent et le mécanisme d'interaction permet à l'agent de se concentrer sur le sujet de l'interaction et de laisser de côtés les questions qui portent sur comment et avec qui interagir.

Dans le MID, le *mécanisme d'interaction* représente l'ensemble des services nécessaires à l'interaction ainsi que les *messages actifs*. Parmi les services qu'offre le mécanisme d'interaction on peut citer:

- Création et suppression des messages actifs,
- gestion de la bibliothèque des protocoles d'interaction,
- etc**…**

4.3.1. Les Messages Actifs:

Les messages actifs jouent un rôle central dans l'interaction, ils reçoivent la responsabilité de mener à terme une interaction.

Un message actif peut être regardé comme étant un agent spécialisé dans la livraison de messages. Cette livraison est prise en compte dans un sens plus large puisqu'elle peut comprendre plusieurs échanges de messages entre le message actif et d'autres agents. Le message actif peut, par exemple, avoir besoin de rechercher des informations (comme par exemple sur l'organisation des agents, ou sur les protocoles d'interaction utilisés) avant de délivrer le message, soit pour définir le(s) destinataire(s), soit pour composer le message.

Le message actif interagit à deux niveaux différents : avec son émetteur (avec qui les échanges sont au niveau des états mentaux) et avec les autres agents du système.

Fig.3.20. Interfaces d'interaction d'un message actif

1. Tâches déléguées aux Messages Actifs:

Les tâches qui peuvent être déléguées aux messages actifs sont:

- *le routage,*
- *le suivi des protocoles d'interaction,*
- *la prise en compte de l'organisation*
- *la composition des messages.*
- *la traduction entre langages,*

D'une certaine façon, toutes les tâches contribuent à la composition du message, néanmoins quelques unes (comme le routage, la prise en compte de l'organisation et le suivi des protocoles) vont plutôt dans la direction de l'obtention des données nécessaires à l'élaboration du message. La traduction intervient après la composition.

Les tâches qui peuvent être déléguées au Mécanisme d'Interaction forment le *noyau des messages actifs*. Une fois que l'agent peut être libéré de ces tâches, il peut dédier plus de temps à poursuivre ses objectifs.

a. Le routage:

Le routage peut être réalisé de plusieurs manières : *directement*, quand l'agent émetteur choisit lui-même à qui adresser les messages ; *indirectement*, quand l'agent émetteur a besoin de consulter le mécanisme d'interaction afin d'obtenir les informations nécessaires pour pouvoir choisir à qui adresser les messages ; *fondé sur le contenu*, quand l'agent émetteur n'indique pas le destinataire dans le message, il est

identifié par le Mécanisme d'Interaction à partir du contenu du message ; *par diffusion*, quand le message est envoyé à tous les agents connus.

b. Suivi des Protocoles:

L'utilisation des protocoles d'interaction est prise en charge par les messages actifs. La gestion et le suivi des protocoles d'interaction sont complètement délégués au mécanisme d'interaction.

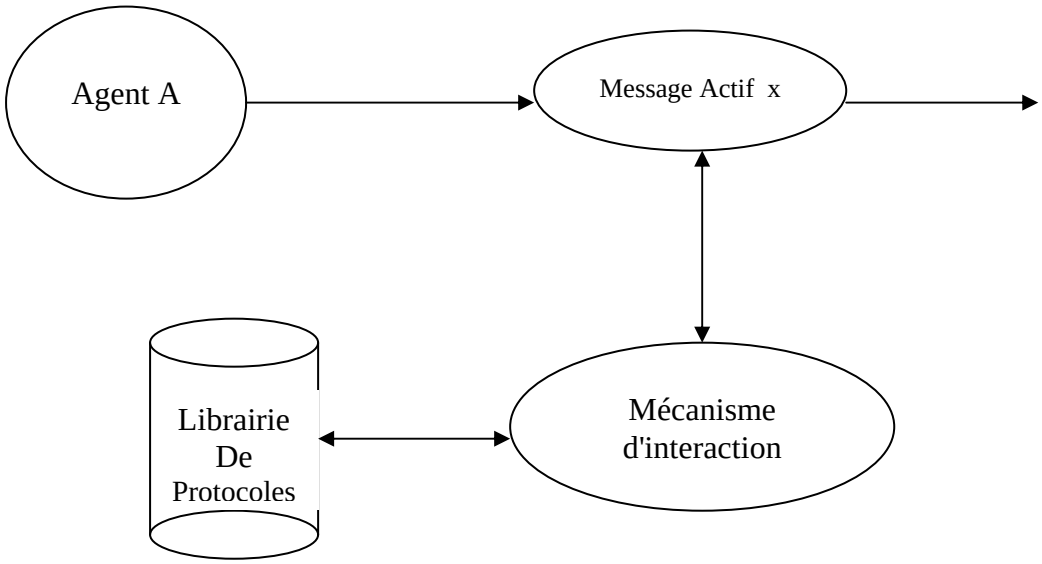


Fig.3.21. Les messages actifs et les bibliothèques de protocoles

Après avoir choisi le protocole d'interaction, le message actif ira l'utiliser pour composer le message et poursuivre les échanges suivants.

c. Prise en compte de l'organisation:

Les connaissances à propos de l'organisation ont une très grande influence sur l'interaction. Quels sont les autres agents du système, quelles sont leurs capacités et quel est leur rôle dans le système, sont des informations qui affectent directement l'interaction.

Dans les SMA, il existe deux types d'informations à propos des organisations qui sont exploités par les messages actif : structurelle et fonctionnelle. Le premier groupe traite des regroupements des agents, les rôles joués au sein des groupes et les relations existantes entre ces rôles. Le deuxième groupe traite des connaissances sur les autres agents, leurs capacités, ressources, besoins, etc. Ces deux types d'information sont importants pour l'interaction.

d. Composition des Messages:

La composition des messages par les messages actifs est faite à partir de schémas de conversation, qui consistent en des structures qui indiquent comment écrire un message dans un contexte donné.

Les schémas de conversation font le lien entre les états mentaux, échangés entre l'émetteur et le message actif, et les actes de langage échangés entre le message actif et le(s) destinataire(s) du message.

L'ensemble de schémas de conversation que possède un message actif détermine son *type*, c'est-à-dire quelles sont ses interfaces avec son créateur et avec les autres agents. L'interface avec les autres agents indique quel est(sont) le(s) langage(s) d'interaction que l'agent peut utiliser à travers ce type de message actif.

e. Traduction:

Les messages actifs, donnent la possibilité de faire interagir des agents hétérogènes (par rapport à l'interaction) entre eux au sein d'un même système. Cela est possible à travers la traduction effectuée sur les langages d'interaction. Cependant, il existe deux langages qui sont pris en compte par la traduction : le langage de communication entre agents (aussi appelé langage d'interaction et qui traite des aspects multi-agents) et le langage de contenu (qui porte sur le sujet de l'interaction). Comme exemple, dans le premier cas, un agent utilisant KQML peut avoir besoin de s'adresser à un agent qui lui, utilise FIPA-ACL ; pendant que, dans le deuxième cas, un agent, utilisant KIF pour s'exprimer, peut échanger des messages avec un agent qui lui, utilise de la logique des prédicats du 1er ordre.

Le Tableau suivant présente les types de traduction qui peuvent s'avérer nécessaire pour faire interagir des agents utilisant des bases d'interaction différentes, avec quelques exemples.

Niveau de traduction nécessaire aux SMA	*Exemples*
Entre Langage de Communication entre Agents – LCA	KQML x IL, FIPA-ACL x KQML
Entre Ontologies	(dépendant de l'application)
Entre Langages de Contenu	Logique des prédicats du 1er ordre x KIF …

Les messages actifs sont capables d'identifier quels sont les langages et ontologies utilisés par les agents et demander au mécanisme d'interaction les traductions quand il le faut. La traduction est toujours réalisée au départ, ce qui réduit les cas ou l'agent est incapable de lire le message.

2. Architecture d'un système avec le MID:

Les architectures possibles, qui peuvent être utilisée pour l'implémentation des messages actifs sont:

- Le message actif peut être utilisé comme un médiateur,
- Le message actif est reçu directement par l'agent récepteur.
- Le message actif est reçu par un autre message actif engendré par l'agent récepteur.

Médiateur:

Le message actif peut être utilisé comme un médiateur entre deux agents. Il doit comprendre la méthode d'interaction des agents participants.

Dans la Figure 3.22, il est possible d'observer le message actif qui s'interpose entre les agents *A* et *B*, toutes les interactions entre les deux agents passent à travers le message actif, qui fait les adaptations nécessaires pour que les deux agents se comprennent.

Fig.3.22. Le Message Actif va comme un médiateur

Dans cette approche, le message actif n'est pas une partie de l'agent, mais un autre composant intermédiaire placé entre les deux agents.

Avec réception directe:

Cette approche résout partiellement le problème de délégation vers le message actif, dans le sens de l'envoi, mais pour la réception l'agent doit traiter lui-même le message comme dans les cas sans message actif.

A la Figure 3.23, chaque agent engendre un message actif au moment où il désire commencer une interaction; par contre il traite directement les messages en provenance d'autres agents. Les numéros placés sur les transitions indiquent l'ordre dans lequel les messages sont échangés. Quand il existe plusieurs possibilités, le numéro est suivi d'une lettre ; par exemple, après avoir reçu le message (2), l'agent *B* peut répondre directement (dans le cas d'une réponse simple), où créer un message actif pour poursuivre l'interaction.

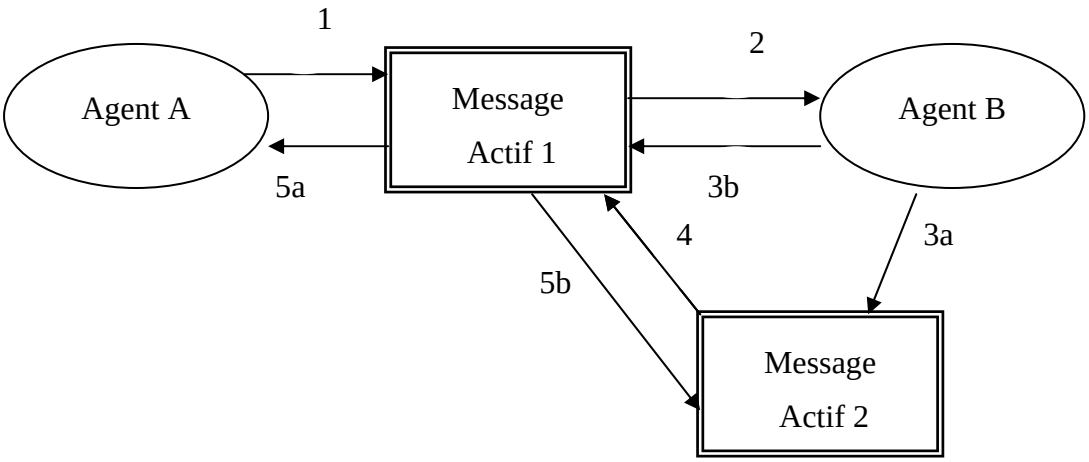


Fig.3.23. Message Actif avec réception directe

Avec réception indirecte:

Cette approche est la plus complète : au début d'une nouvelle interaction l'agent crée un nouveau message actif qui ira s'occuper de cette interaction.

A la Figure 3.24 ci-dessous il est possible d'observer qu'au moment où l'agent *B* reçoit le message provenant de l'agent *A*, il déclenche la création du message actif et lui transmet le message reçu de *A* pour qu'il s'en occupe. La suite de l'interaction est gérée par les messages actifs attachés aux agents.

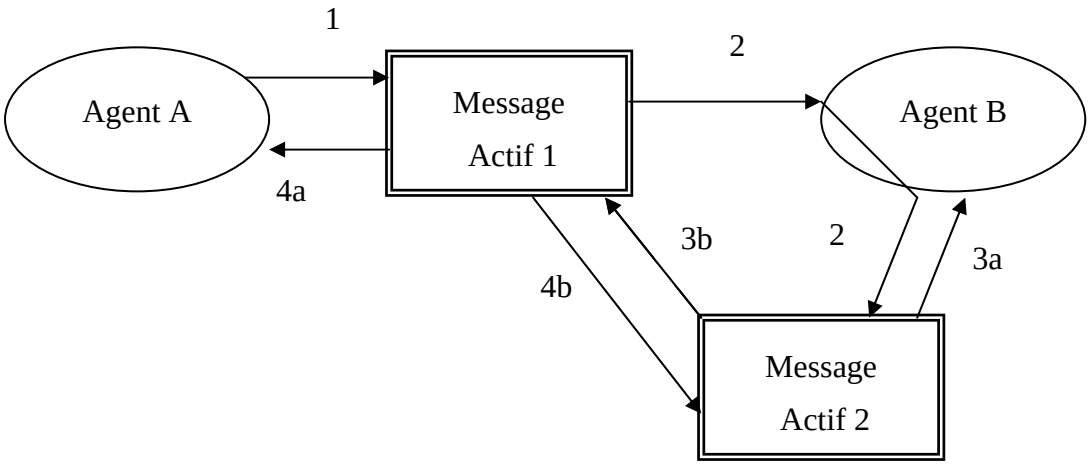


Fig.3.24. Message Actif avec réception indirecte

5. Conclusion:

Ce chapitre vient de présenter les différents techniques et modèles permettant de mettre en place l'échange d'informations et de donner aux agents la possibilité d'interagir.

La communication qui constitue un concept fondamental du paradigme multi-agents [Deck87] est la brique de base de l'interaction. Nous avons montré que cette communication pouvait prendre différentes formes, mais quelle que soit la forme prise, il s'agit d'actions locales fiables entreprises par les agents.

Les techniques multi-agents considèrent que l'interaction nécessite l'adoption d'un langage d'interaction (langage de communication inter-agents) afin d'assurer une compréhension mutuelle des agents composants le système [Dema95]. Les langages d'interaction les plus utilisés, cités dans la deuxième section de ce chapitre, sont le langage KQML, FIPA-ACL et le langage IL.

La notion d'interaction englobe et combine plusieurs actions de communication. Cette combinaison se traduit par des enchaînements conversationnels qui sont réglementés par les protocoles d'interaction. Les protocoles d'interaction sont utilisés par les agents afin de contrôler et de structurer les échanges d'informations entre eux.

La dernière section de ce chapitre est consacrée aux modèles d'interaction. La première partie de cette section, présente les modèles de représentation des protocoles

d'interaction, à savoir les graphes d'états et le langage AUML, ce dernier sera employé dans la description des protocoles d'interaction utilisés dans les chapitres suivants. Une deuxième question traitée, dans cette section, concerne l'influence du modèle d'interaction sur l'architecture du SMA. La dernière question traitée est l'aspect dynamique de l'interaction. Nous avons présenté le MID, un modèle d'interaction où la gestion de l'interaction est déplacée de l'agent vers le mécanisme d'interaction.

Chapitre 4

Plateformes de développement des SMA

La nouvelle technologie des agents et SMA connaît un vif succès chez les industriels pour qui elle promet des outils de conception et d'implémentation flexibles et adaptatifs. Pourtant, la difficulté de réaliser des SMA due à la multiplicité et la complexité des concepts et composantes rend l'utilisation des outils existants très difficile par les "non-spécialistes" des SMA, et même par les "non-auteurs". Ces systèmes ont tous les problèmes des systèmes distribués et concurrents mais aussi d'autres problèmes liés à leurs caractéristiques telles que la flexibilité des interactions et à la dynamique des organisations.

Afin de réaliser une opérationnalisation plus accessible des systèmes multi-agents, des travaux ont tenté de réutiliser des architectures et des modèles existants pour construire des environnements de développement de ces systèmes. Les environnements de développement ou les plateformes multi-agents sont nécessaires pour renforcer le succès de la technologie multi-agents. Les plateformes multi-agents permettent aux développeurs de concevoir et réaliser leurs applications sans perdre de temps à réaliser des fonctions de base pour la création et l'interaction entre agents et éliminent, dans la plupart des cas, la nécessité d'être familier avec les différents concepts théoriques des systèmes multi-agents.

Il existe un nombre important d'environnements de développement des applications orientées agents: il y a aussi bien des produits commerciaux que des logiciels dans le domaine public. On peut trouver une liste complète de ces plateformes à l'adresse *www.agentlink.org/ressources/agent-software.php*

Parmi les plateformes fournies comme logiciels libres, il y quelques plateformes plus connues pour avoir été utilisées dans le développement de plusieurs applications : JADE, MASK et MADKIT pour les agents cognitifs. Il faut noter que cette liste n'est pas unique, et qu'il y a aussi d'autres plateformes qui ont été utilisées avec beaucoup de succès pour bâtir diverses applications.

Ce chapitre est organisé de la façon suivante: dans la première section, nous présentons le standard des plateformes FIPA. Dans la deuxième section, nous présentons d'une manière synthétique, quelques plateformes existantes.

1. La norme FIPA pour les Systèmes Multi-Agents:

La FIPA ("Foundation for Intelligent Physical agent") est une fondation qui réunit un grand nombre d'industriels et de partenaires universitaires et qui a pour but de développer des standards dans la technologie multi-agents à partir des besoins identifiés par ses membres. Parmi les technologies choisies pour la standardisation, la plateforme et le langage FIPA-ACL sont importants pour nos travaux.

Les premiers documents de spécification de la norme FIPA [Fipa97], 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 plateforme multi-agents (figure 4.1) où ils identifient les rôles de quelques agents clés nécessaires pour la gestion de la plateforme, et spécifient le contenu du langage de gestion des agents et l'ontologie du langage. Les spécifications FIPA peuvent être trouvées dans *www.Fipa.org*

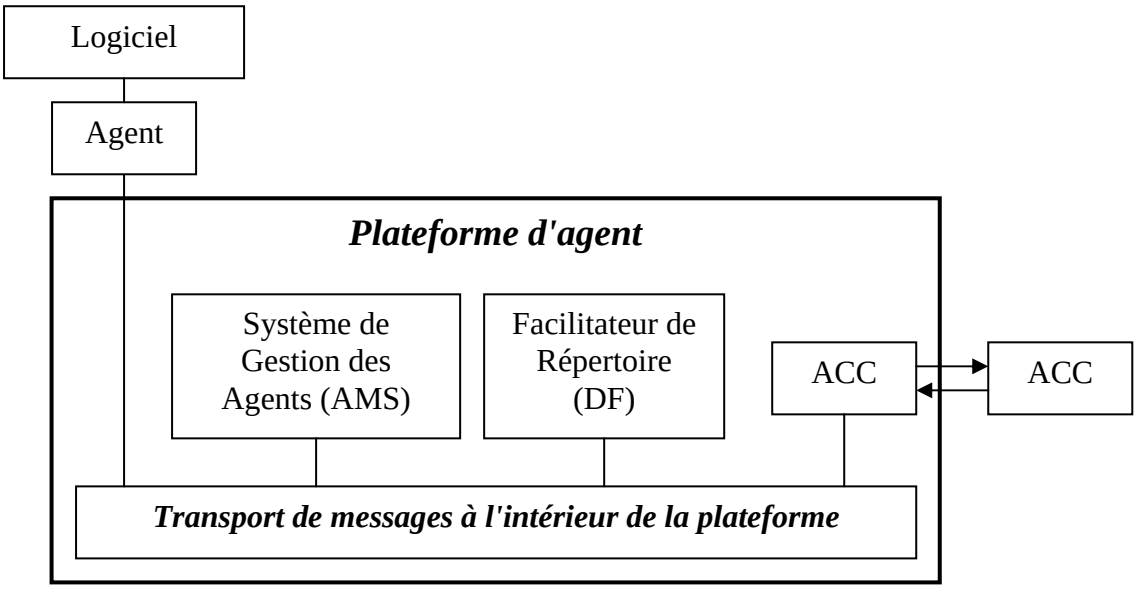


Fig.4.1. Le modèle de référence pour une plateforme multi-agents FIPA

Dans cette figure, on voit qu'il y a trois rôles principaux dans une plateforme 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: la création des agents, leur exclusion, décider si un agent peut s'enregistrer dynamiquement et la migration des agents dans la plate-forme 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 plateformes multi-agents.

- Le *Facilitateur d'Annuaire* (Directory Facilitator - DF) est l'agent qui fournit un service de pages jaunes à la plateforme multi-agents. Les agents peuvent publier leurs compétences (services) et consulter celles des autres agents. Chaque plate-forme doit avoir au moins un DF.

Le standard spécifie aussi le **Langage de Communication d'Agents** (Agent Communication Language – ACL -voir la description de FIPA-ACL dans le Chapitre 3- section 2). La communication des agents est basée sur l'envoi de messages. Le langage FIPA-ACL est le langage standard des messages et impose le codage, la sémantique et la pragmatique des messages. La norme n'impose pas de mécanisme spécifique pour le transport interne de messages. Plutôt, puisque les agents différents pourraient s'exécuter sur des plateformes différentes et utiliser technologies différentes d'interconnexion, FIPA spécifie que les messages transportés entre les plateformes devraient être codés sous forme textuelle. On suppose que l'agent est en mesure de transmettre cette forme textuelle. La norme FIPA préconise des formes communes pour les conversations entre agents par la spécification de protocoles d'interaction, qui incluent des protocoles simples de type requête-réponse, mais aussi des protocoles spécifiques aux agents comme le réseau contractuel et les enchères anglaises et hollandaises.

Pour qu'un agent soit compatible avec la FIPA par rapport au langage d'interaction, il doit satisfaire à quelques exigences :

1. L'agent doit envoyer un message « *not-understood* » s'il reçoit un message qu'il ne comprend pas ou qu'il n'est pas capable de traiter. D'autre part, les agents doivent être prêts à traiter ce type de réponse venue d'autres agents.

2. L'agent peut implémenter un sous-ensemble des types de messages et de protocoles prédéfinis.

3. Si un agent utilise un acte de communication faisant partie des actes prédéfinis, il doit être implémenté correctement par rapport à sa spécification.

4. Les agents peuvent utiliser de nouveaux actes communicatifs, mais ils doivent s'assurer de leur compréhension et être sûrs qu'il n'existe pas un autre acte avec le même sens.

5. L'agent doit être capable de générer des messages grammaticalement bien formés, correspondant à ce qu'il souhaite envoyer. D'autre part il doit être capable de traduire une chaîne de caractères bien formée dans le message correspondant.

Dans la plateforme FIPA, les agents sont organisés en domaines. Un *Domaine* est un groupement logique d'agents et de services défini par l'appartenance à un répertoire (géré par un DF). Un agent doit s'inscrire auprès du DF pour appartenir au domaine. Les agents peuvent appartenir à plusieurs domaines au même moment.

2. Aperçu de quelques plateformes Multi-agents:

Nous présentons dans cette partie, un aperçu de quelques plateformes multi-agents. Nous ne prétendons pas donner ici une liste exhaustive de l'ensemble des plateformes disponibles actuellement. Nous avons en effet choisi de sélectionner les plateformes les plus significatives pour le développement d'agents logiciels, et d'en donner une brève description. La première plateforme MADKIT fondée sur le modèle Aalaadin, la deuxième Mask fondée sur l'approche Voyelle de MAGMA et la dernière JADE qui est une plateforme conforme aux spécifications FIPA.

2.1. La plateforme MADKIT [Madk03]:

MadKit est une plate-forme développée par le Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM) de l'Université Montpellier II. MADKIT est libre pour l'utilisation dans l'éducation. MADKIT est écrit en Java et est fondé sur le modèle organisationnel Alaadin. Il utilise un moteur d'exécution où chaque agent est construit en partant d'un micro-noyau. Il y a un environnement de développement graphique qui permet facilement la construction des applications.

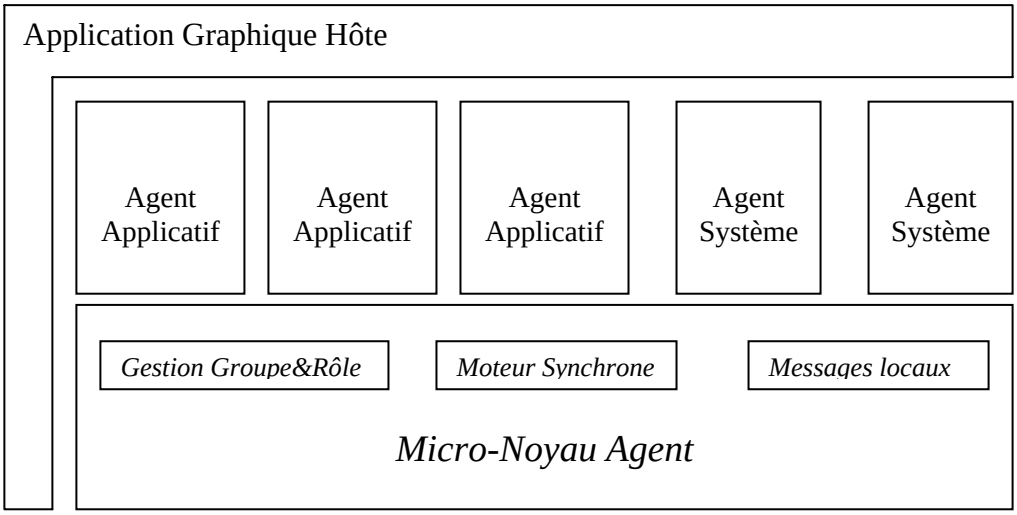


Fig.4.2. Architecture générale de MADKIT

Le principe essentiel de MADKIT est d'utiliser partout où cela est possible la plateforme pour sa propre gestion et exécution. Tous les services hors ceux assurés par le micro-noyau sont implémentés par des agents à part entière.

Le Micro-noyau prend en charge les fonctions suivantes :

Gestion des groupes et rôles locaux: le noyau a la responsabilité de maintenir une information correcte sur les membres des groupes et les rôles tenus. Il vérifie également si les requêtes faites sur le système de groupes et rôles sont acceptables.

Gestion du cycle de vie des agents: Le noyau gère également le lancement (et éventuellement l'arrêt) des agents et maintient les tables de références sur les objets

d'implémentation. Il est également le gestionnaire des informations administratives sur les agents (possesseur, modalités de créations, ...) et donne un identifiant garanti unique à chaque agent. Cet identifiant, l'AgentAddress, est forgé à partir de l'adresse du noyau MadKit et l'identification de l'agent sur le noyau ; il peut être rapproché de la notion de GUID de FIPA. La forme de cet identifiant peut également être redéfinie pour faciliter l'intégration avec d'autres plateformes agent.

Passage de message local: Le noyau a la responsabilité de l'aiguillage et de la distribution de messages entre agents locaux (s'exécutant sur le noyau). Le passage de message au plus bas niveau revient à de simples échanges de références pour pouvoir facilement implémenter différentes sémantiques de passage de message à un niveau supérieur.

Les fonctions associées à tout agent dans MADKIT sont :

Cycle de vie: L'agent dispose de quatre états (création, activation, exécution, et destruction), et a la possibilité de démarrer d'autres agents sur le noyau local (et les désactiver par la suite).

Communication: La communication est implémentée sous forme de passage de message asynchrone, soit d'agent à agent identifiée par leur AgentAddress ou leur groupe et rôle, soit sous la forme d'une diffusion à tous les teneurs d'un rôle dans un groupe donné.

Organisation: Tout agent dispose de primitives permettant d'observer son organisation locale (connaître les groupes et rôles courants) et d'y agir (prise de rôle, entrée et retraits de groupes).

Outils: La classe de base des agents permet également de manipuler une éventuelle interface graphique associée à l'agent, les flots d'entrée/sortie, etc.

2.2. La plateforme Mask:[Dema97]

MASK (Multi-Agent System Kernel) est un environnement de développement de SMA. Cette plateforme est Développée dans l'équipe MAGMA du laboratoire LEIBNIZ/IMAG/CNRS de Grenoble. Elle est fondée sur l'approche Voyelle (AEIO) et constitue le support logiciel de cette méthode.

L'objectif est de fournir des bibliothèques d'agents (A), de manipulation d'environnement (E), d'interaction (I) et d'organisation (O) ainsi que les outils d'aide a la programmation.

La plateforme MASK est composée de boites à outils couvrant les différents aspects du paradigme multi-agents:

- La boite à outils Agent fournit a l'utilisateur des éditeurs d'agents instances de modèles d'agents prédéfinis ou permet au concepteur de modèles d'ajouter de nouveaux modèles avec leurs éditeurs associés.

- La boite à outils Environnement fournit à l'utilisateur des bibliothèques de fonctions pour manipuler les environnements (simulés) où évoluent les agents. Le concepteur

peut définir de nouveaux environnements et associer les bibliothèques correspondantes. Cette boite dépend très fortement des applications.

- La boite à outils Interaction est chargée de la présentation des fonctions d'interaction. Cette boite se décompose en deux parties :

- une bibliothèque de fonction
- un éditeur d'interaction.

- La boite à outils Organisation se décompose comme la boite interaction en deux parties :

- un ensemble d'outils de modélisation de l'organisation fournissant des primitives d'exploitation,
- des éditeurs permettant d'instancier des organisations sur le SMA. le modèle "RESO" permet de représenter une organisation. Ce modèle propose une interface graphique de construction et de la représentation d'une organisation de SMA.

2.3. La plateforme JADE: [Jade03] [Belli02] [Belli04]

JADE (Java Agent Development Framework) est une plateforme multi-agents développée en Java par CSELT (Groupe de recherche de Gruppo Telecom, Italie) qui a comme but la construction des systèmes multi-agents et la réalisation d'applications conformes à la norme FIPA. JADE comprend deux composantes de base : une plateforme agents compatible FIPA et un paquet logiciel pour le développement des agents Java.

L'architecture de la plateforme est basée sur la coexistence de plusieurs Machines Virtuelles (VM) Java et la communication se fait par la méthode RMI (Remote Method Invocation) de Java entre machines virtuelles (VMs) différentes. Chaque VM est un réceptacle d'agents qui fournit un environnement d'exécution complet pour l'exécution des agents et permet d'avoir plusieurs agents qui s'exécutent simultanément sur un même hôte. Chaque réceptacle d'agents est un environnement multi-threads d'exécution composé d'un thread d'exécution pour chaque agent et, en plus, des threads créés à l'exécution par le système RMI pour envoyer des messages. Un récipient spécial joue le rôle du frontal de la plateforme ; il contient les agents de gestion et représente la plateforme toute entière pour le monde extérieur.

La plateforme JADE offre une interface graphique utilisateur (GUI) pour la gestion à distance qui permet de contrôler et superviser les états des agents, par exemple arrêter et remettre en marche un agent. L'interface graphique permet aussi de créer et de commencer l'exécution d'un agent sur un hôte éloigné, à condition qu'un réceptacle d'agents s'exécute déjà sur cet hôte. L'interface elle-même a été implémentée comme un agent, appelé RMA (Remote Monitoring Agent). Toute la communication entre les agents et l'interface (GUI) et toute la communication entre cette interface et l'AMS est faite par ACL via une extension ad hoc de l'ontologie des agents de gestion FIPA.

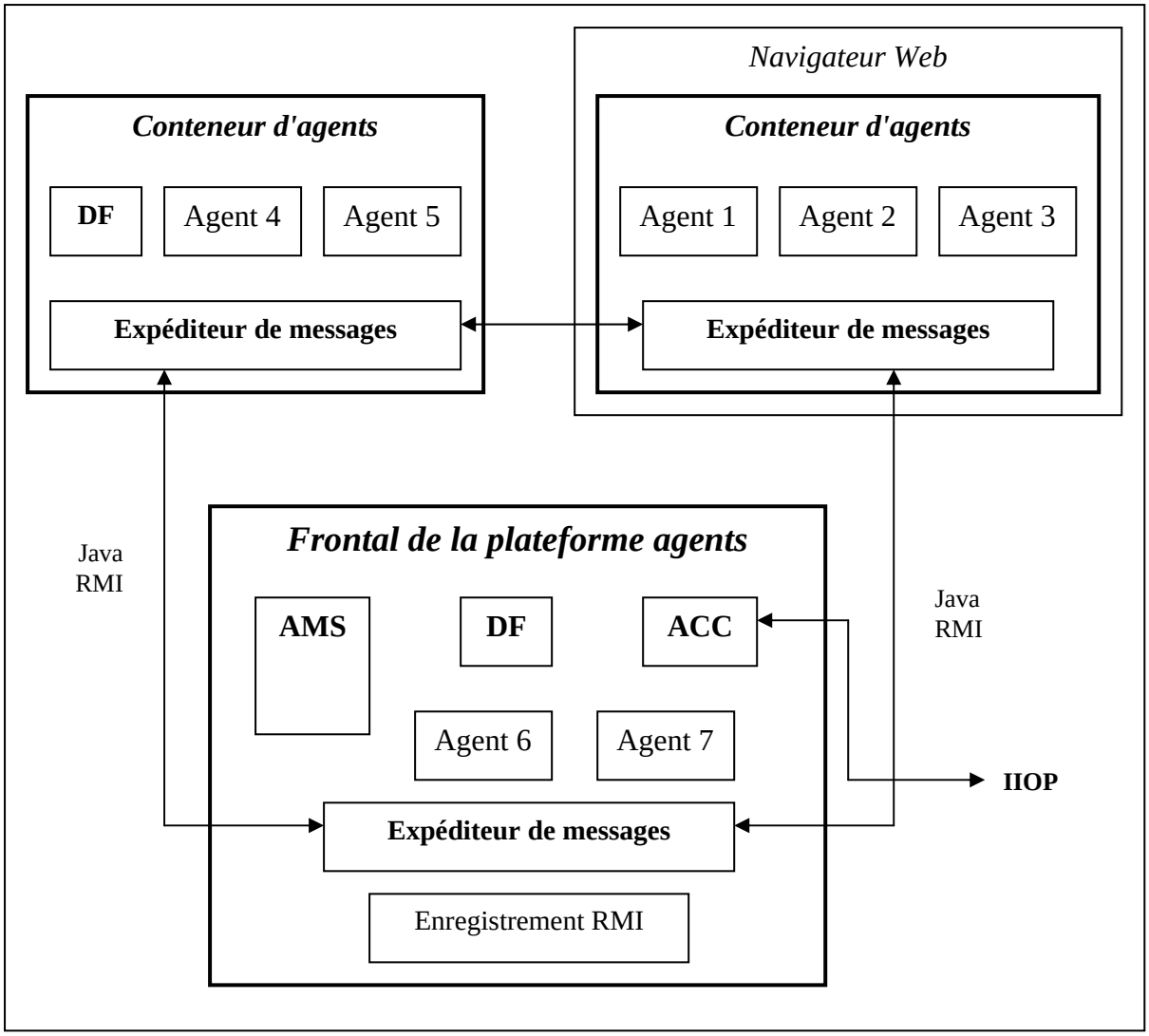


Fig.4.3. Architecture de la plateforme multi-agents JADE

3. Conclusion:

La notion de plateforme est liée à l'implémentation des systèmes multi-agents : elle constitue un réceptacle au sein duquel les agents peuvent évoluer. Les plateformes sont un environnement permettant de gérer le cycle de vie des agents et dans lequel les agents ont accès à certains services (pages jaunes, transport de messages, etc…).

Dans un premier temps, nous avons présenté la norme FIPA des systèmes multi-agents, qui permet de faciliter l'interopérabilité des agents et des systèmes multi-agents provenant de différents développeurs. Dans la deuxième section de ce chapitre, nous avons passé en revue quelques plateformes SMA, plus de plateformes pourront être trouvées sur les sites www.AgentLink.org et www.AgentBuilder.com . Nous n'avons pas énuméré toutes les plateformes existantes mais seulement citer celles qui nous paraissent les plus intéressantes et sur lesquelles nous nous sommes appuyés pour le développement de notre plateforme.

La Fipa a produit depuis 1997 un ensemble de spécifications qui s'étendent des langages de communications (Agent Communication Languages) aux langages de contenu (Content Language) ainsi qu'aux protocoles d'interaction.

La notion de plateforme est utilisée comme une infrastructure de communication et de gestion des agents. Ces fonctionnalités sont apportées au travers de services de plateformes accessibles soit aux agents de la plateforme, soit aux autres plateformes.

Partie II:

Mise en œuvre et test du modèle d'interaction dynamique.

Chapitre 5

Conception de la plateforme MAP

Dans ce chapitre, nous présentons le développement de la plateforme MAP (pour Multi-Agents Plateform). MAP est un est un environnement de développement de systèmes multi-agents. Cette plateforme est développée selon les spécifications de FIPA, elle est basée sur le modèle d'interaction dynamique (MID présenté dans le chapitre 3- section 4).

Le choix du modèle d'interaction MID se justifie par l'avantage qu'offre ce modèle par rapport aux modèles statiques; au lieu d'intégrer (dupliquer) le module d'interaction au corps de chaque agent, comme c'est le cas dans les modèles d'interaction statiques, dans le MID, la gestion de l'interaction est déléguée au mécanisme d'interaction MI. Nous pensons que ce modèle est bénéfique dans la mesure où il soulage l'agent de certaines tâches de l'interaction et les transférer au MI. Ce mécanisme peut manipuler dynamiquement le message de différentes manières: trouver le bon destinataire, donner une meilleure représentation du message et s'assurer que le message sera bien interpréter par le destinateur (ajout des informations supplémentaires au message d'origine, tel que le protocole utilisé etc…). Ainsi l'utilisation du MID facilite l'interaction entre agents hétérogènes (en terme de langages). Ce modèle permet à l'agent de se concentrer au sujet de l'interaction et laisser de côté les questions qui portent sur comment et avec qui interagir. Il décharge l'agent d'un certains nombre de responsabilité ce qui permet l'utilisation d'agents plus simples.

En plus de la mise en œuvre du modèle d'interaction dynamique, cette plateforme: (i) fournit une base logicielle expérimentale pour les besoins de recherche de l'équipe (intégration et test de nouvelles techniques et modèles multi-agents), (ii) fournit une infrastructure d'exécution répartie d'applications multi-agents afin de permettre aux utilisateurs de développer de façon simple leurs propre systèmes multi-agents sans avoir à se préoccuper des problèmes de gestion des communications et d'interaction entre agents.

Pour le développement de cette plateforme, nous nous sommes inspirés de l'architecture abstraite de FIPA et principalement sur les travaux réalisés dans la plateforme JADE qui implémente une plateforme conforme aux spécifications FIPA et offre un environnement distribué.

La première section de ce chapitre présente l'architecture de la plateforme MAP que nous avons développé ainsi que la description des différents composants de cette architecture. Dans la deuxième section, nous présentons le fonctionnement de cette plateforme.

1. Architecture de la plateforme MAP:

Tout comme la plateforme JADE [Belli01], la plateforme MAP est conçue sur les spécifications de FIPA [FIPA97]. Elle reprend donc l'architecture de l'Agent Management Reference Model proposé par FIPA. Les services de bases proposés par le noyau de la plateforme sont le Directory Facilitator (DF), l'Agent Management System (AMS) et le Message Transport System (MTS). Un nouveau composant est intégré au noyau de la plateforme, le Mécanisme d'interaction permettant la mise en oeuvre du modèle d'interaction dynamique.

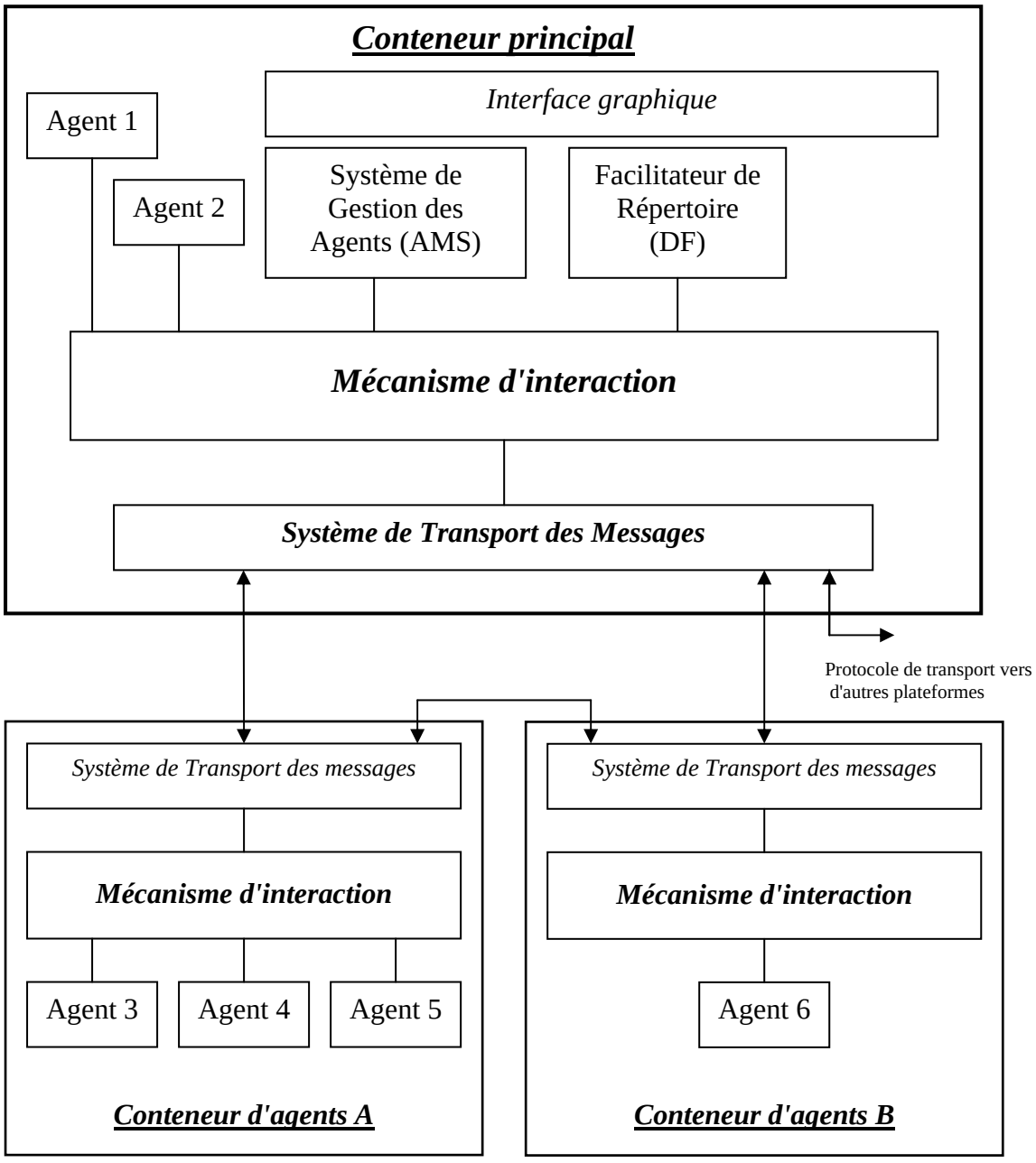


Fig.5.1. Architecture de base de la plateforme MAP

L'architecture de base de la plateforme MAP est illustrée dans la figure 5.1 ci-dessus. La plateforme MAP est composée d'un conteneur principal d'agents et de plusieurs conteneurs d'agents distribués. Chaque conteneur d'agents permet de gérer localement un ensemble d'agents. Le conteneur principal est un récipient spécial, il contient les agents de gestion (AMS, DF, MTS et le MI) et représente la plateforme toute entière.

Les agents AMS, DF et système de Transport des Messages sont des agents systèmes, ils constituent le noyau du système, ils sont démarrés au même moment que la plateforme.

1.1. L'agent de gestion des agents (AMS):

L'AMS est un composant important car il assure la cohérence des entités de la plate-forme et maintient un répertoire contenant les adresses de transport des agents de la plate forme. Ce service est de type "page blanche" car il fait la correspondance entre un agent et son identificateur unique, appelé AID (Agent IDentifier). Il gère le cycle de vie complet des agents. Il peut leur donner naissance, les stopper ou de les migrer d'un conteneur vers un autre. Il offre des services aux agents comme le nommage; l'affectation d'un identificateur unique (AID Agent IDentifieur) à un agent lors de son lancement. Il peut être interrogé par un agent pour l'obtention de descriptions sur d'autres agents telle que l'adresse.

Les services offerts par l'AMS sont:
- Enregistrement d'agents
- Suppression d'agents
- modification de description d'agents
- recherche d'agents
- Avoir-description d'agents (local, externe / conteneur, état d'un agent)
- Nommage : Affectation d'un identificateur unique (AID) à un agent
- Enregistrement de conteneurs d'agents
- Gérer le cycle de vie d'un agent:

- Suspendre un agent
- Terminer l'exécution d'un agent
- Créer un agent
- Reprendre l'exécution d'un agent
- Invoquer un agent
- Exécuter un agent

Le cycle de vie d'un agent FIPA définit quel est son état à un moment donné. Il peut être : *initialisé, actif, suspendu, en transit* ou *en attente.*

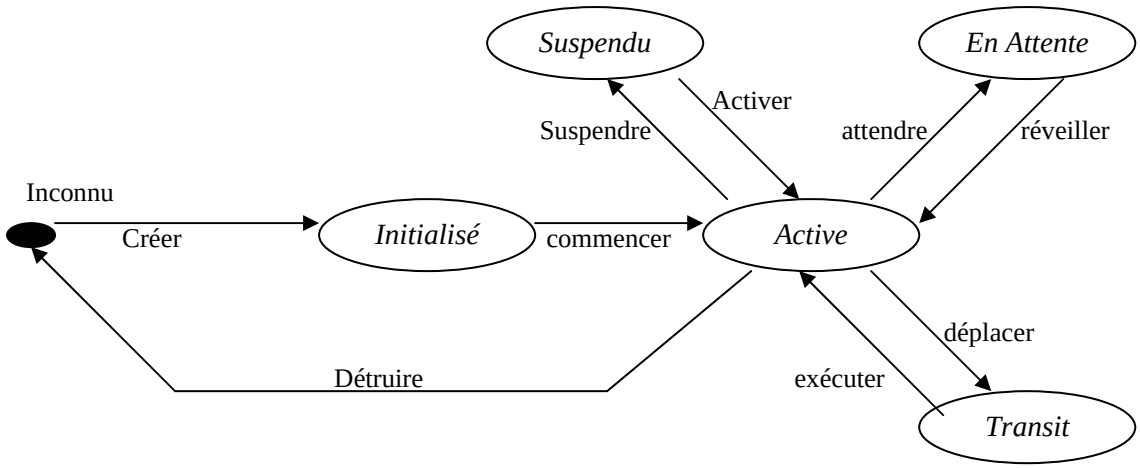


Fig.5.2. Cycle de vie d'un agent

Le répertoire des agents utilisé par l'AMS a la structure suivante:

Identificateur de l'agent	Adresse de l'agent	Etat	Langage ACL

La table des conteneurs a la structure suivante:

Nom du conteneur	Adresse

1.2. Facilitateur de Répertoire (DF):

Le facilitateur de répertoire (DF) est un composant qui fait office d'annuaire. C'est un service de "pages jaunes" qui permet de mettre en relation les agents avec leurs compétences. Ainsi, un agent peut enregistrer ses compétences dans le DF ou interroger celui-ci pour connaître les agents possédant une compétence précise. Selon les normes FIPA, il est possible d'avoir plusieurs DF sur une même plate forme mais dans notre implémentation, la plate-forme dispose d'un DF unique. Le DF est un des éléments centraux lors de la délégation de l'interaction au mécanisme d'interaction : il permet aux messages actifs de savoir quels sont les fournisseurs d'un service (tâche à réaliser ou produit à vendre) et quels sont les agents clients d'un service. Le DF rassemble dans une table les agents par compétences possédées. Cette table est remise à jour par envoi de message des agents lorsque ceux-ci apprennent de nouvelles compétences (ou à leur création bien entendu). Cependant pour que les informations contenues dans le DF soient fiables, il faut que les agents qui seront détruit informent le DF pour les supprimés du répertoire. Les agents ont la possibilité de contacter des DF distants sur une autre plateforme afin de se renseigner sur les agents de cette plate-forme (les DF ne se contactent pas entre eux).

En plus du service de pages jaunes, le DF gère l'organisation des agents, et il est capable de répondre à des questions sur ce sujet (organisation): l'ensemble d'agents par groupe, le groupe d'agents assurant un service, les services assurés par un groupe, etc…

Les services offerts par le DF sont:

- Enregistrement des agents ainsi que les services qu'ils offrent ou dont ils ont besoin
- Recherche des agents offrant un service
- Recherche des agents demandant un service
- Modification des services assurés par un agent
- Suppression d'agents du répertoire.

L'annuaire utilisé par le DF a la structure suivante:

Identificateur de l'agent	Rôle/ Service	Offre/demande	Ontologie

La table des groupes utilisée par le DF a la structure suivante:

Identificateur du Groupe	Identificateur d'agent	Type d'agent (chef/membre)

1.3. Le Système de Transport des Messages (MTS):

Le Système de Transport des Messages MTS (Message Transport System) fournit un mécanisme pour le transfert de messages entre agents. Les agents concernés peuvent être locaux sur un unique conteneur, sur différents conteneurs ou sur une autre plateforme. Sur un unique conteneur, le MTS est fourni par un Agent Communication Channel (ACC). L'Agent Communication Channel (ACC) propose un service directement aux agents de la plateforme. L'ACC peut accéder aux informations des autres services de la plateforme (comme l'AMS ou le DF) pour permettre d'effectuer ses tâches de transport de messages. L'ACC opère l'envoi physique de messages selon l'emplacement de l'agent destinataire; il peut être:

- Intra plateforme, Intra container;
- Intra plateforme, Inter containers;
- Inter Plateformes compatibles FIPA.

1.4. L'interface graphique (GUI):

L'interface graphique est considérée comme un agent, elle est en liaison directe avec l'agent AMS, l'agent DF et le Système de Transport des Messages. Cette interface permet de:

- Consulter et superviser les états des agents,
- Visualiser la liste des agents, leur états et leur adresses
- Visualiser la liste des conteneurs connectés et leur adresses
- Afficher la table des agents par rôles et par groupes (organisation des agents)
- Afficher l'échange de messages entre agents
- Comptabiliser le nombre de messages effectuer par agents.
- Inspecter les Yellow Pages (services enregistrés)
- Surveiller les échanges de messages dans la plateforme
- Surveiller le cycle de vie d'un agent

1.5. Le Mécanisme d'interaction:

Le mécanisme d'interaction est le mécanisme qui prend en charge la gestion de l'interaction. Il est invoqué chaque fois qu'un agent désire initié une interaction. Il permet de:

- créer les messages actifs,
- gérer le cycle de vie des messages actifs,
- orienter les messages reçus vers les messages actifs correspondants,
- gérer les conversations entre agents,
- fournir aux messages actifs une bibliothèque de protocoles.

1.5.1. Création des messages actifs:

Le message actif est créé par le Mécanisme d'Interaction au moment où l'émetteur envoie un message, le contenu de ce message est utilisé pour initialiser le message actif. Le message actif est chargé de mener cette interaction jusqu'à sa fin. Une fois l'interaction est terminée, le message actif sera automatiquement détruit par le mécanisme d'interaction.

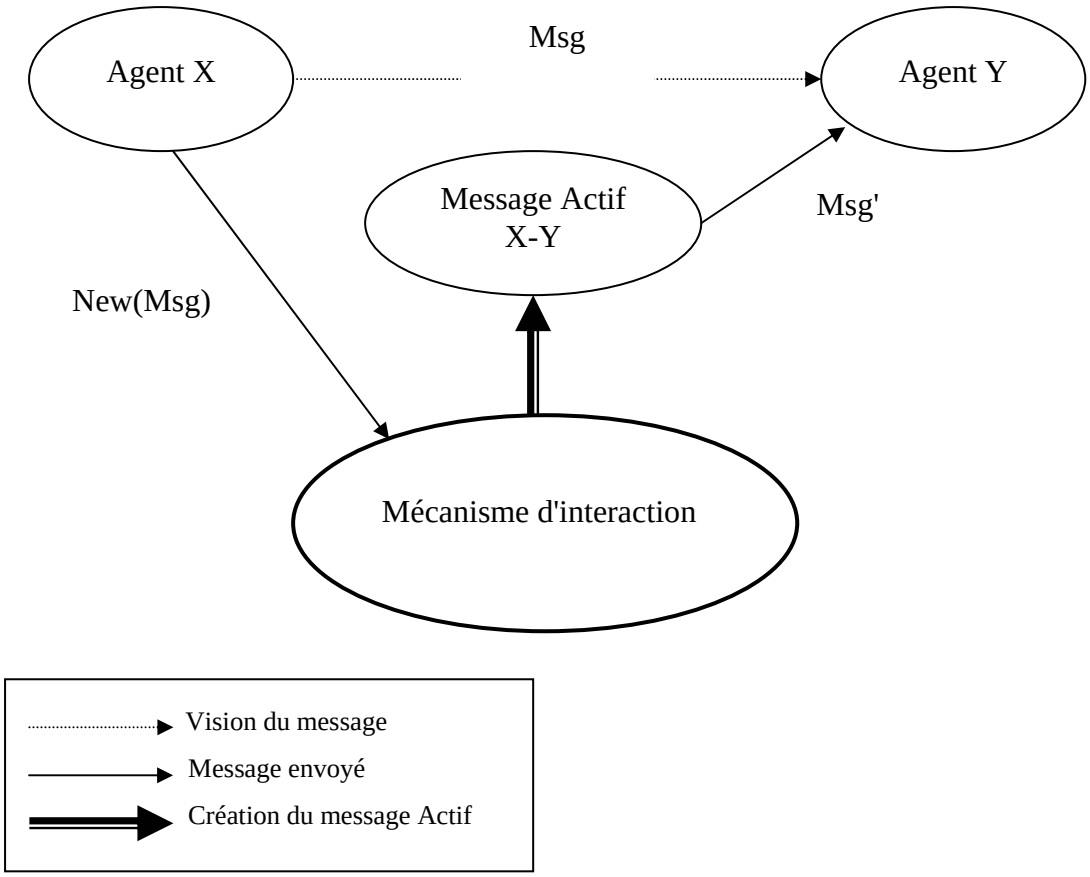


Fig.5.3. Création d'un message actif

Le message actif assure les fonctions suivantes:

- Le routage
- La composition des messages
- Le choix et le suivi des protocoles
- Traduction
- La prise en compte de l'organisation.

a. le routage:

Dans le cadre des SMA, le routage comprend: la détermination du destinataire d'un message et le chemin que le message doit suivre pour y arriver. Le message actif traite la première partie du routage, la deuxième partie est prise charge par le Système de Transport des messages.

La sélection du type de routage qui sera utilisé par le message actif prend en compte les données fournies par l'émetteur. Les types de routage possibles sont :

1. Si le destinataire est nommé par l'émetteur : **direct**, le message actif n'agit pas sur le routage, il utilise l'information fournie par l'émetteur.

2. Si l'émetteur fournit le rôle du destinataire : **indirect**, le message actif consulte le DF pour obtenir l'identification de(s) l'agent(s) qui correspond(ent) au rôle donné. S'il existe plusieurs candidats : il peut envoyer le message à un des agents ou à tous les agents.

3. Si l'émetteur fournit le nom d'un groupe d'agents : **indirect**, le message actif consulte le DF pour obtenir la description du groupe. A ce moment là, il a besoin particulièrement du point d'entrée dans le groupe (le(s) agent(s) responsable(s) de la réception des messages au sein du groupe). Il existe plusieurs possibilités, dépendant de la structure du groupe :

- S'il existe un responsable pour la réception des messages (un point d'entrée dans le groupe), il est choisi comme destinataire ;
- S'il existe plusieurs points d'entrée, ou s'il n'en existe aucun (dans ce cas tous les membres du groupe sont considérés comme des points d'entrées), le message peut se comporter de deux façons :
  - le message actif crée un nouveau message actif adressé à chacun des agents ;
  - au lieu de créer un message actif pour chaque agent, le message actif peut envoyer un seul message à tous les agents. Cette approche offre une économie de ressources.

4. Si l'émetteur sollicite la diffusion du message: **diffusion**, le message actif obtient la liste des agents connus et l'envoi se passe comme dans le cas précédent.

5. Si le destinataire n'est pas nommé : **fondé sur le contenu**, le message actif utilise le contenu pour identifier le sujet en question et le DF pour trouver le(s) bon(s) destinataire(s). Le message actif utilise le corps du message (performatif et paramètres). Par exemple, dans le message "… inform X…", le message actif doit chercher auprès de la DF quels sont les agents intéressés par X ensuite les contacter.

b. Composition des Messages:

La composition des messages par les messages actifs est faite à partir de schémas de conversation, qui consistent en des structures qui indiquent comment écrire un message dans un contexte donné.

Exemple de schéma de conversation:

```
Schéma: Demande (request)
Arguments: ( content, destinataire )
LCA: FIPA-ACL
Composition: ( type: request,
protocol: NegociationProtocol,
content: argument(content),
to: argument(destinataire) ).
```

Le message *Request (Achat Imprimante HP 3420, matériel informatique)* reçu par un message actif, à partir du schéma de conversation donné ci-dessus, le message actif engendre le message FIPA-ACL suivant:

(request

:sender Agentx
:receiver A,B,C
:protocol contract-net
:language Fipa-Sl
:reply-with conv1
:ontology matériel-informatique
:content (Achat imprimante HP 3420)

Pour réaliser une interaction, le message actif envoie un (ou plusieurs) messages(s), qui doi(ven)t être préalablement composé(s). Les messages sont écrits dans un Langage de Communication entre Agents (KQML ou FIPA-ACL), qui est composé de plusieurs informations (champs). Pour définir le contenu de chaque champ, le message actif dispose des informations qui lui permettront de choisir le contenu pour ce champ. Les tableaux ci-dessous donnent les informations nécessaires pour remplir chaque champ d'un message pour les langages KQML et FIPA- ACL.

Champs	*Source des Connaissances*
Type	Il peut être obtenu à partir du message original (qui a engendré le message actif)
Sender	Défini à l'émission du message
Receiver	Fourni par l'émetteur ou le message actif à partir de la DF et l'AMS
From	Ces champs ne sont pas utilisés par les messages actifs (dans la composition des messages)
To	
in-reply-to	Gérés automatiquement par le mécanisme d'interaction (par rapport aux protocoles d'interaction utilisés)
reply-with	
Language	Identifie le langage de contenu utilisé (Fipa-Sl0)
Ontology	Identifie l'ontologie utilisée
Content	A partir du message original

<u>Champs du Message x Sources de Connaissances - KQML</u>

Champs	*Source des Connaissances*
Type	Il peut être obtenu à partir du message original (qui engendre le message actif)
Sender	Défini à l'émission du message
Receiver	Fourni par l'émetteur ou le message actif à partir de la DF et l'AMS
Content	A partir du message original
reply-with	Gérés automatiquement par le mécanisme d'interaction (par rapport aux protocoles d'interaction utilisés)
in-reply-to	
reply-by	
conversation-id	
Language	Identifie le langage de contenu utilisé (Fipa-Sl0)
Ontology	Identifie l'ontologie utilisée
Protocol	Il peut être indiqué explicitement dans le message

<u>Champs du Message x Sources de Connaissances- FIPA-ACL</u>

c. Choix et Suivi des Protocoles:

Les protocoles d'interaction utilisés par les messages actifs sont stockés dans une bibliothèque de protocoles. La gestion des protocoles d'interaction est assurée par le mécanisme d'interaction. Et puisqu'il existe un mécanisme d'interaction dans chaque conteneur, les protocoles d'interaction sont partagés entre les agents connectés à la plateforme.

Le message actif choisit le protocole d'interaction à utiliser à partir de la requête envoyée par l'agent émetteur et le schéma de conversation correspondant.

Par exemple, dans une négociation, un agent peut envoyer un message "*negotiate (AgentX, achat (impimante hp 845 c,7500 Da))*", à partir de ce message et des schémas de conversation, le message actif identifie qu'il faut utiliser un protocole de négociation (par exemple le *contract- net*).

Le message actif utilisera le protocole d'interaction choisit pour composer les messages et poursuit les échanges de messages selon le déroulement du protocole jusqu'à sa fin.

d. Traduction:

La plateforme MAP donne la possibilité aux agents d'utiliser les langages de communication entre agents (ACL) le langage KQML et le langage FIPA-ACL. Et le Fipa-Sl0 comme langage de contenu.

Afin de permettre aux agents hétérogènes (utilisant des langages ACL différents) de communiquer entre eux, les messages actifs offre le service de traduction entre langages. A la composition du message, le message actif consulte l'AMS pour savoir quel est le langage ACL utilisé par l'agent récepteur, s'il s'agit d'un langage différent que celui utilisé par l'émetteur, le message actif convertit le message vers le langage utilisé par le récepteur.

La traduction est toujours réalisée au départ (à l'émission du message), ce qui permet aux agents récepteurs de lire et traiter directement le message.

e. Prise en compte de l'organisation:

Les agents ne font pas référence directement à l'organisation dans leurs messages, c'est le message actif qui adapte ces messages à l'organisation. Les messages actifs utilisent les connaissances organisationnelles offertes par la DF, pendant le routage et la composition des messages.

Les requêtes que les messages actifs peuvent poser à la DF, à propos de l'organisation, sont données ci-dessous:

Search_groupe <service/rôle> : donne comme réponse le groupe d'agents offrant <service>

Search_agent <service/Rôle> : donne comme réponse un agent offrant <service>

1.5.2. Cycle de vie d'un message actif:

Un message actif possède un cycle de vie plus court que le cycle de vie de l'agent, il est créé par le mécanisme d'interaction pour mener une interaction, il est supprimé à la fin de cette interaction.

Le cycle de vie d'un message actif est le suivant :

1. création (déclenchée par l'envoi d'un message par l'agent émetteur) ;
2. routage ;
3. suivi des protocoles ;
4. composition du message ;
5. interaction (action d'envoi et de réception des messages);
6. suppression (fin des activités).

1.5.3. Orientation des messages:

A la réception d'un message, le Mécanisme d'Interaction détermine à partir du contenu de ce message s'il s'agit d'une interaction déjà en cours, auquel cas, le message sera orienté vers le message actif correspondant. Dans le cas où il s'agit d'une nouvelle interaction, le mécanisme d'interaction, créé un message actif qui prendra en charge cette interaction.

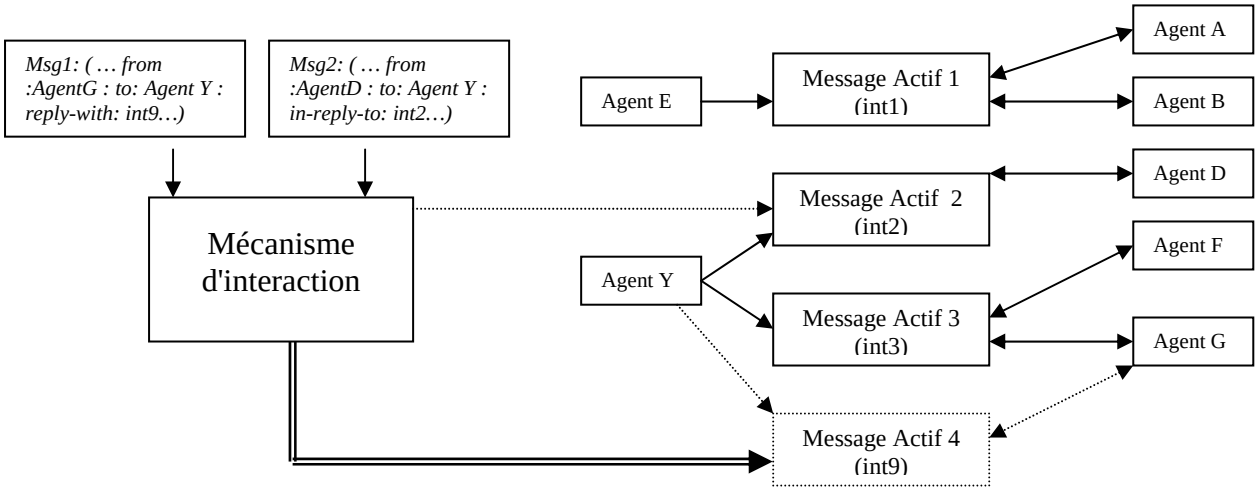


Fig. 5.4. Orientation des messages par le Mécanisme d'interaction

Dans l'exemple donné à la figure ci-dessus, l'agent Y mène deux interactions à travers le message actif 2 et le message actif 3. Et comme le message reçu Msg1 porte sur une nouvelle interaction (int9), le Mécanisme d'interaction crée le message actif 4 pour menée cette interaction entre l'agent Y et l'agent G. et le message reçu Msg2 qui porte l'interaction (int2) est orienté vers le Message actif 2 (chargé de cette interaction).

2. Fonctionnement de la plateforme:

2.1. Schéma général de fonctionnement de la plateforme:

La plateforme multi-agents MAP est un système autonome composée d'un conteneur principal et de plusieurs conteneurs d'agents. Ces conteneurs sont distribués à travers un réseau d'ordinateurs. Chaque conteneur d'agents gère localement un ensemble d'agents. Il fournit un environnement d'exécution pour l'exécution des agents et permet d'avoir plusieurs agents qui s'exécutent simultanément sur un même hôte. Il traite tous les aspects de la communication : répartition des messages reçus, routage des messages selon le champ de destination (*:receiver*) et dépôt des messages dans la file de messages du mécanisme d'interaction. Pour les messages vers l'extérieur, le conteneur d'agents, à travers le Système de transport des messages, maintient assez d'information pour chercher l'emplacement de l'agent récepteur et pour choisir une méthode de transport convenable pour expédier le message. Chaque conteneur d'agents dispose d'un Mécanisme d'interaction permettant de gérer les interactions entre agents.

La figure 5.5 suivante présente le schéma général de fonctionnement de la plateforme. Le Mécanisme d'interaction joue un rôle central, il prend en charge toutes les interactions initiées ou reçues par les agents (applicatifs ou systèmes). Il crée un message actif pour chaque agent en interaction avec un ou plusieurs agents, ce message actif est détruit une fois l'interaction est terminée.

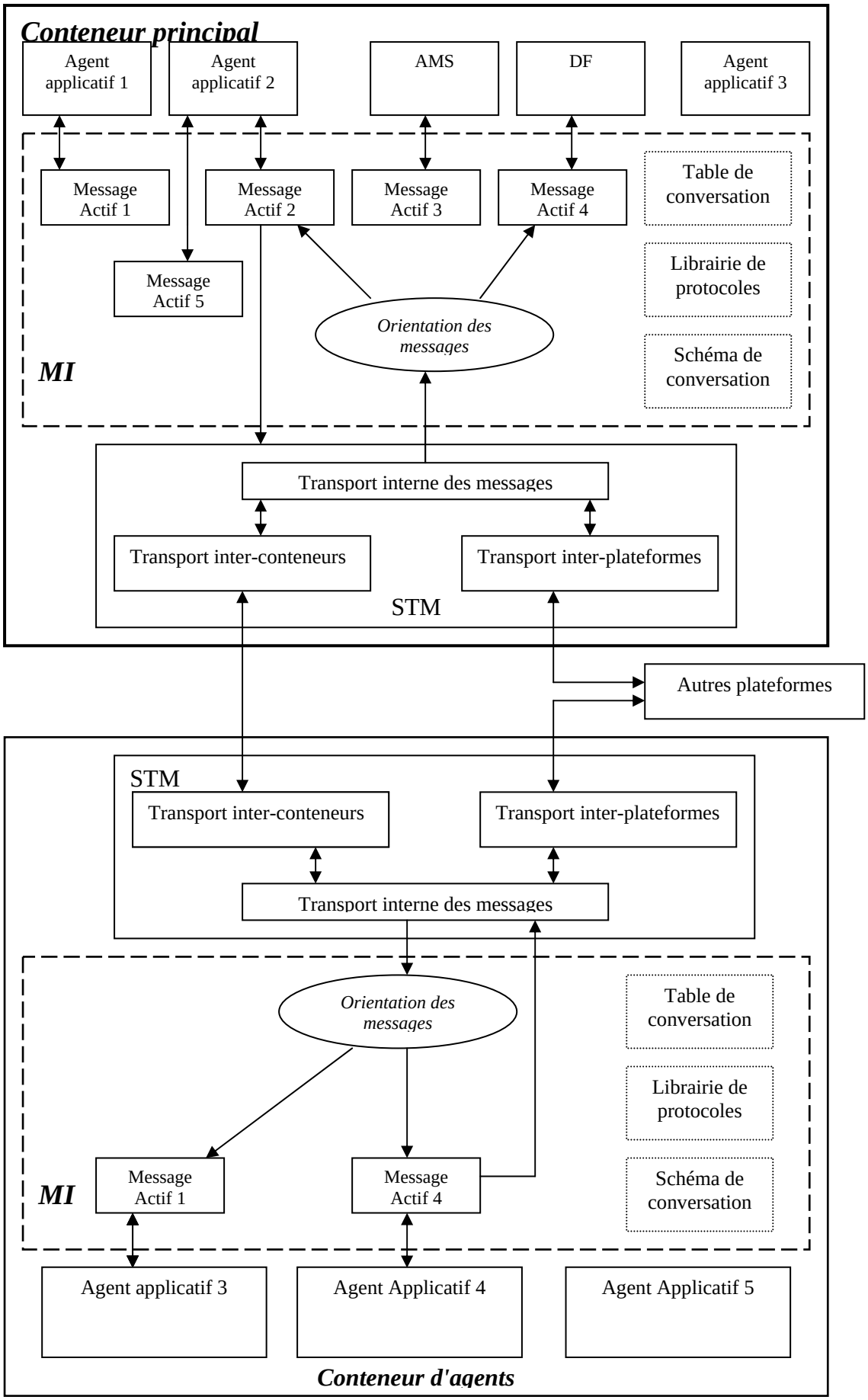


Fig.5.5. Schéma général de fonctionnement de la plateforme

2.2. Fonctionnement du mécanisme d'interaction:

Le fonctionnement du mécanisme d'interaction est présenté par le diagramme d'activité UML dans la figure 5.6. Ce processus est exécuté en boucle par le mécanisme d'interaction.

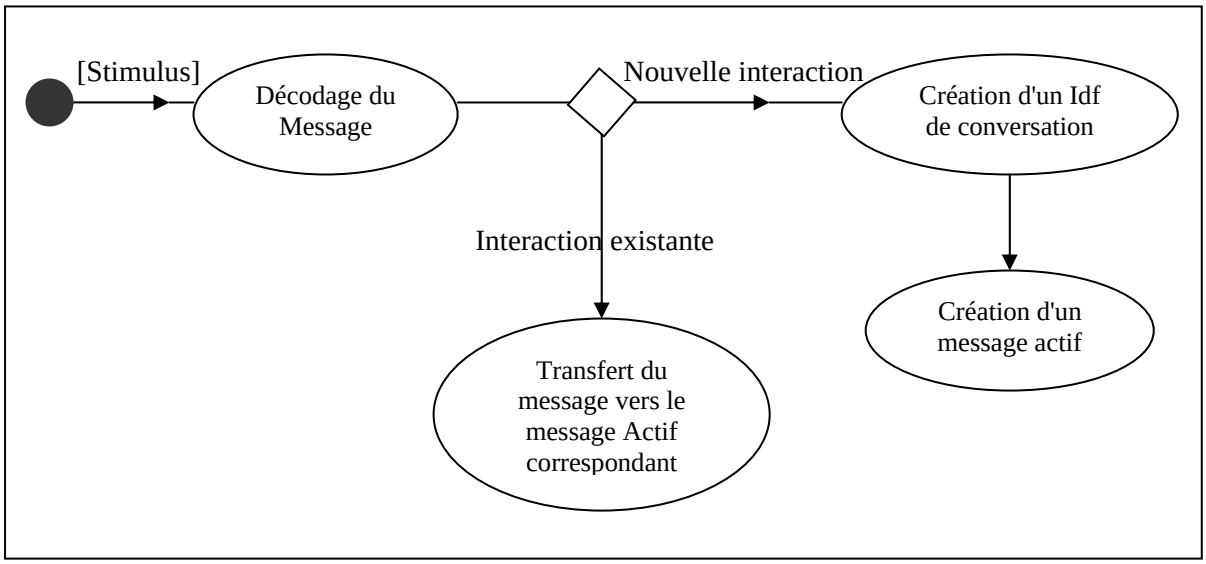


Fig.5.6. Processus de suivi des conversations

Le stimulus d'activation est bien sûr la réception d'un message. Après décodage du message reçu, le mécanisme d'interaction oriente ce message vers le message actif correspondant s'il s'agit d'une interaction déjà en cours, sinon le MI crée un nouveau message actif qui prendra en charge cette interaction. Afin de discriminer les messages reçus et de distinguer les différentes interactions, le MI affecte un identificateur de conversation unique pour chaque message actif, pour cela, il maintient une table de conversations permettant de gérer les différentes interactions en cours. La structure de cette table est la suivante:

Identificateur d'agent	Identificateur du message actif	Conversation identificateur

De cette façon, le MI permet aux agents de suivre plusieurs interactions simultanément, même si ces interactions sont régies par le même protocole d'interaction.

Outre la table de conversations, le MI maintient une librairie de protocoles d'interaction FIPA, et un ensemble de schémas de conversation possibles. Ces derniers sont mis à la disposition des messages actifs.

2.3. Fonctionnement des messages actifs:

Nous présentons dans cette partie comment les tâches qui caractérisent un message actif doivent se succéder. Le schéma ci-dessous (Figure 5.7) présente le déroulement d'un message actif.

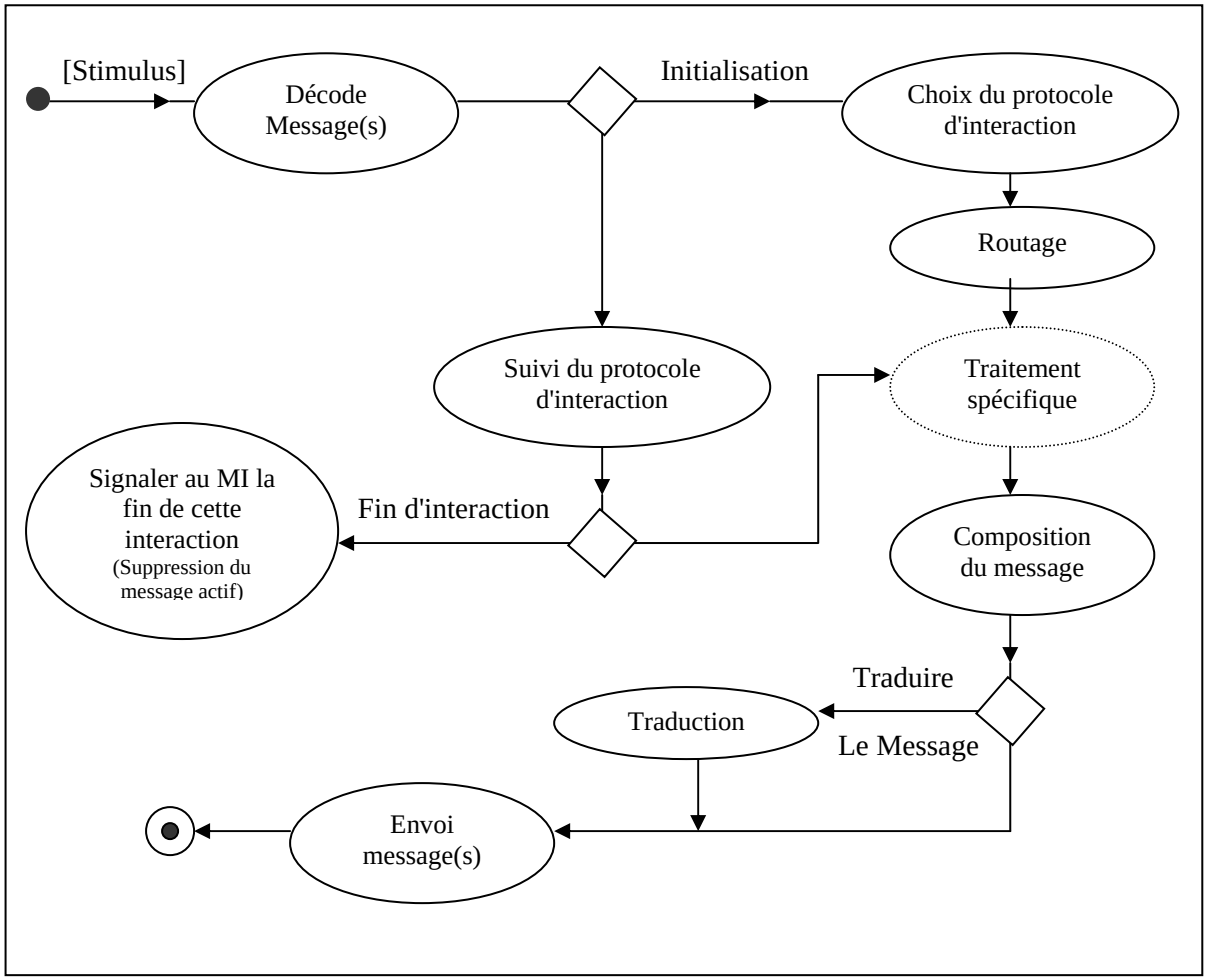


Fig.5.7. Fonctionnement du message Actif

Le message actif est créé par le Mécanisme d'Interaction selon deux cas:

- si l'agent est l'initiateur d'une interaction, il doit envoyer un message initial au MI qui sera utilisé pour créer et initialiser un nouveau message actif,
- si l'agent n'est pas l'initiateur de l'interaction, dans ce cas le MI crée un nouveau message actif qui sera initialisé par le contenu du message reçu.

Quand une interaction aboutit à sa fin (atteinte d'un état final du protocole d'interaction), le message actif est supprimé automatiquement, et il est retiré de la table des conversations.

2.4. Exécution des agents applicatifs sur la plateforme:

2.4.1. Enregistrement d'un agent:

Un agent applicatif peut s'exécuter sur le conteneur principal ou sur un conteneur d'agents distribué. A son lancement, l'agent est dans l'état d'initialisation, et pour

pouvoir se connecter à la plateforme; c'est-à-dire communiquer avec les agents résidents dans la plateforme, l'agent doit réaliser les tâches suivantes:

- s'enregistrer auprès de l'agent AMS
- publier ses services et/ou ses besoins auprès de l'agent DF.

La figure ci-dessous, décrit dans le formalisme UML, le déroulement de l'inscription d'un agent auprès de l'agent AMS.

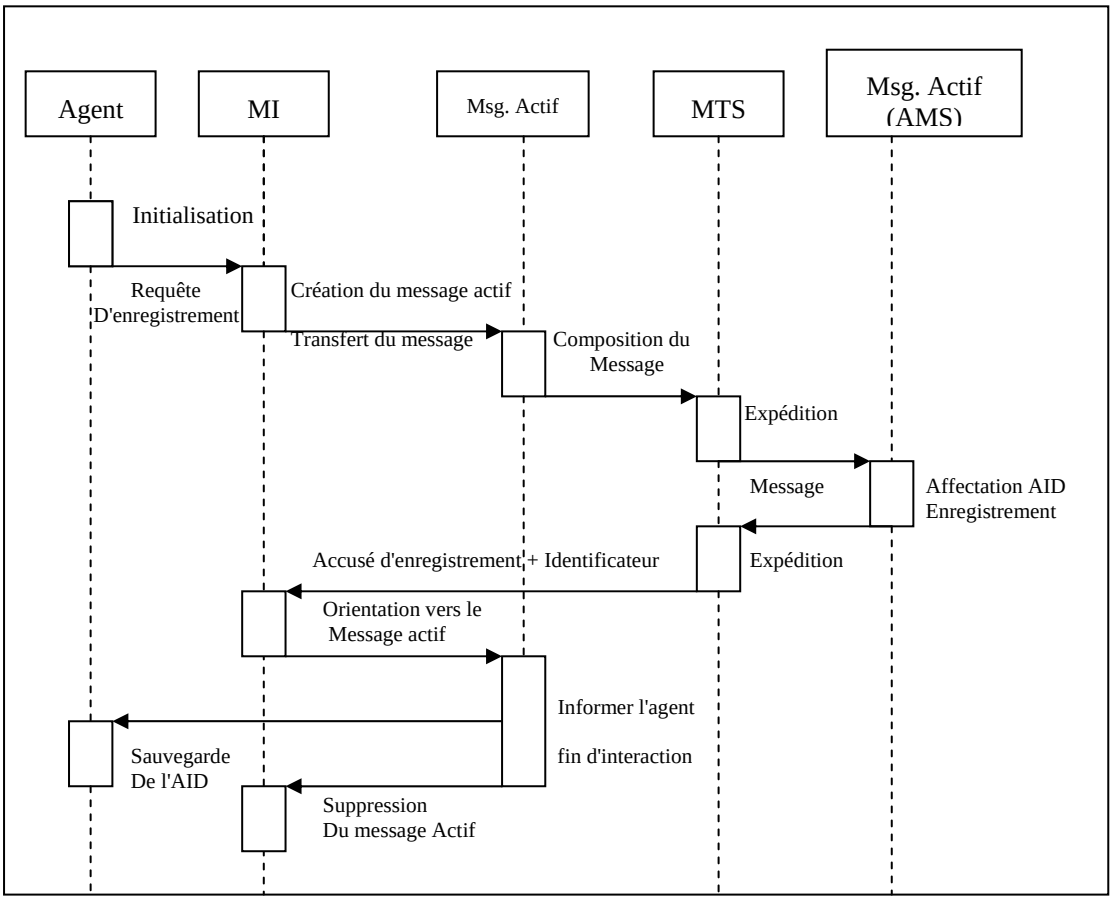


Fig. 5.8. Enregistrement d'un agent dans la plateforme

Dans le diagramme ci-dessus, l'interaction du côté de l'agent AMS, se passe de la même manière que pour l'agent, c'est-à-dire les messages passent d'abord par le MI, du conteneur principal, puis au message actif correspondant à l'AMS.

Par exemple, si un agent "Agent1" demande de s'enregistrer dans la plateforme, le message actif correspondant compose et adresse le message FIPA-ACL suivant:

```
(request
  :sender
    (agent-identifier
      :name agent1
      :addresses (192.168.0.1))
  :receiver (set
    (agent-identifier
      :name ams
      :addresses (192.168.0.0))
  :ontology agent-management
  :language Fipa-Sl0
  :in-reply with int1
  :protocol fipa-request
```

```
:content
  "( register
      (agent-identifier
            :name agent1
            :addresses (192.168.0.1))
    )" )
```

Le protocole d'interaction utilisé, par le message actif, pour enregistrer un agent est le protocole FIPA-Request. La représentation de ce protocole est donnée dans la figure suivante.

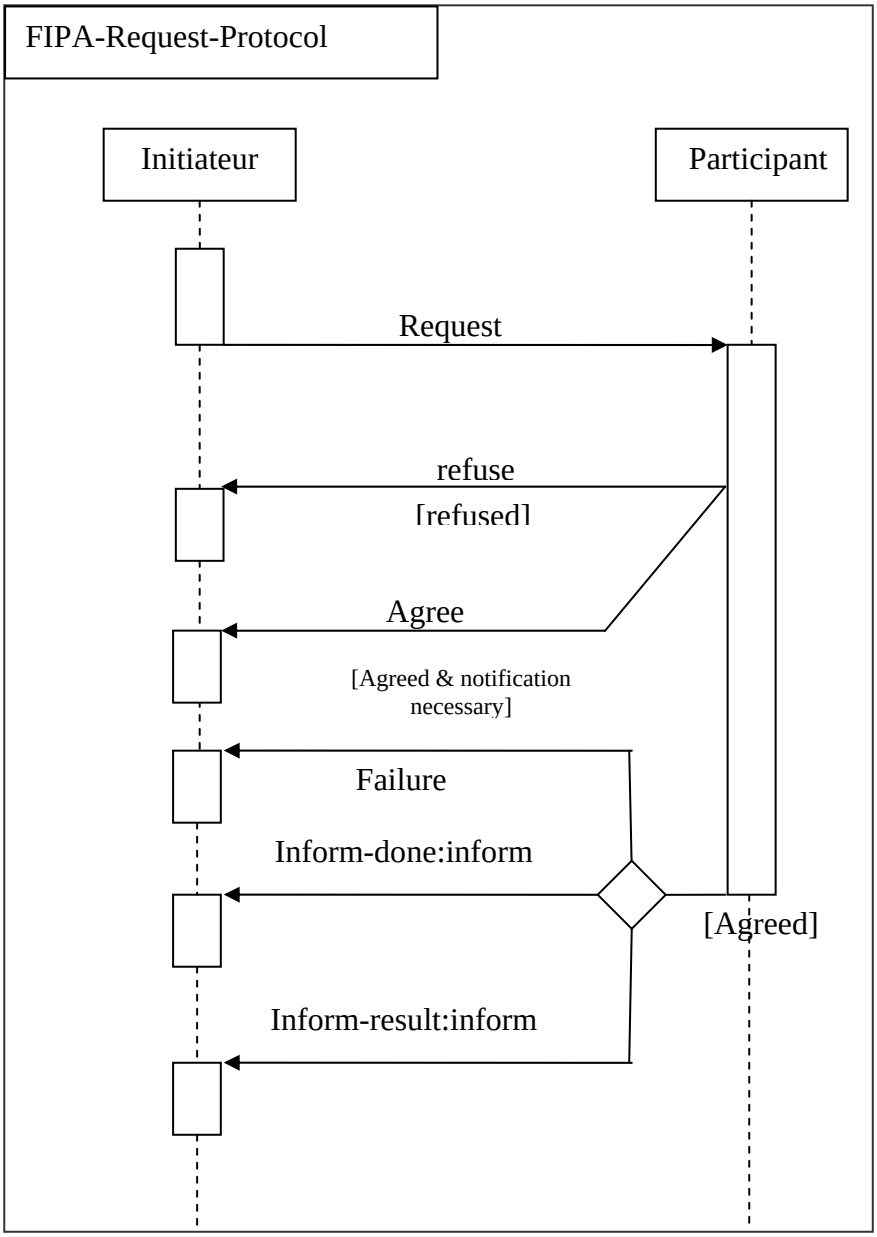


Fig. 5.9. Protocole FIPA-Request

L'enregistrement d'un agent par l'AMS est effectué selon les étapes suivantes:

a - associe un Identificateur Unique (AID) à cet agent. Cet identificateur est un nom de type name@HAP où HAP (Home Agent Platform) correspond au nom de la plateforme suivi de l'adresse IP de la machine hôte (ex: AgentA@ConteneurA.194.254.17.91).

b - place cet agent en état activé
c - enregistre cet agent dans le répertoire

L'enregistrement d'un agent auprès du facilitateur de répertoire (DF) s'effectue de la même manière que pour son enregistrement auprès de l'AMS. Dans ce cas, l'agent communique, au facilitateur de répertoire (DF), son identificateur ainsi que les services qu'il offre et les services qu'il demande.

2.4.1. Sortie d'un agent de la plateforme:

La sortie d'un agent (fin d'activité) doit être signalée auprès des agents AMS et DF. Elle s'effectue de la même manière que pour l'enregistrement, sauf que dans ce cas, l'action demandée est de supprimer (*deregister*) l'agent du répertoire de l'AMS et du répertoire du DF.

2.5. Exemple d'interaction (cas d'appel d'offre):

Nous présentons dans cette partie un exemple d'interaction, permettant d'illustrer le déroulement des interactions entre agents dans la plateforme. Nous considérons le cas d'un appel d'offre annoncé par un agent initiateur pour l'achat d'un produit. Dans ce cas, l'agent initiateur et les agents participants utiliseront le protocole d'interaction FIPA-Contract-Net pour contrôler leurs interactions.

Dans ce protocole, un agent assume le rôle d'initiateur qui souhaite voir un certain service (produit, tâche à réaliser, etc…) accompli par un ou plusieurs autres agents participant. Pour ce faire, l'initiateur publie, à destination des agents participant, un appel à propositions (call for proposal) qui spécifie le service à effectuer ainsi que toutes les conditions que l'initiateur souhaite. Les participants vérifient s'ils peuvent satisfaire aux conditions préalables. Si cela est possible, ils répondent à l'initiateur par un acte de proposition. Alternativement, un participant peut refuser de faire une proposition. Quand un moment-butoir (deadline) défini est dépassé, l'initiateur évalue toutes les propositions reçues et choisit l'agent qui accomplira le service ou peut toutes les refuser L'initiateur envoie alors un acte d'acceptation de proposition aux participants dont la proposition est choisie, les autres reçoivent un acte de rejet de proposition. Lorsque l'initiateur accepte la proposition, l'agent participant acquière un engagement pour accomplir le service. Quand ce dernier a accompli le service, il envoie un message d'accomplissement à l'initiateur. Il est à noter que ce protocole exige de l'initiateur de savoir quand il a reçu toutes les réponses des participants. Si un participant ne réussit pas à lancer un acte de proposition ou un acte de refus, l'initiateur peut potentiellement être mis en attente indéfiniment. Pour éviter ce problème, l'appel à proposition se compose d'un moment-butoir avant lequel les réponses doivent être reçues par l'initiateur. Les propositions reçues après le moment-butoir sont automatiquement rejetées avec comme justification que la proposition a été faite trop tardivement.

La figure 5.10 suivante illustre en détails le protocole ContractNet FIPA en AUML (Agent Unified Modeling language) Les termes restent en anglais pour des raisons de cohérence de vocabulaire par rapport aux spécifications FIPA.

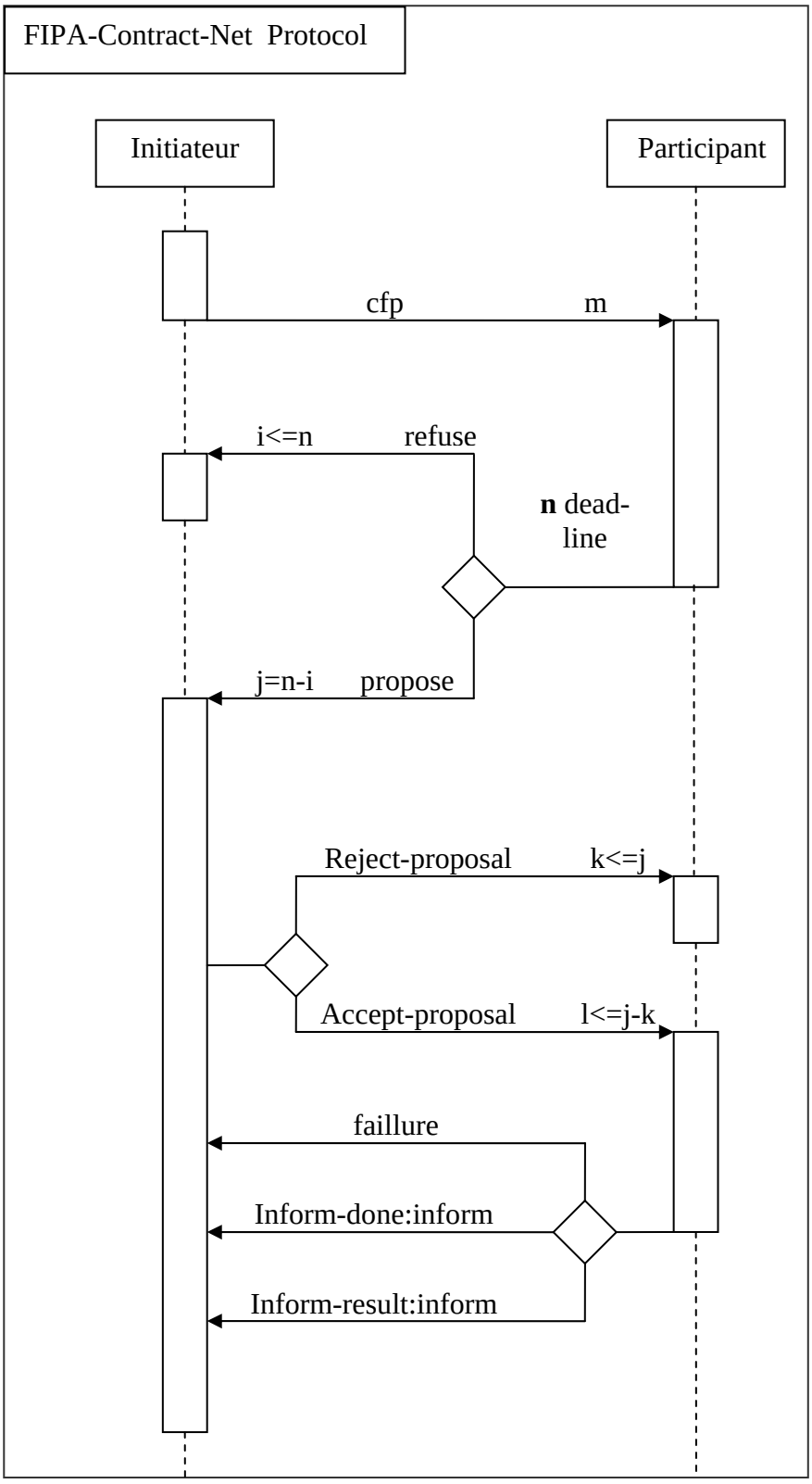


Fig. 5.10. Le protocole FIPA-Contract-net

Dans cet exemple, nous considérons quatre agents Agent1, Agent2, Agent3 et Agent4. L'Agent1 est l'initiateur souhaitant acquérir un produit P1 avec le moindre prix sur le marché. Celui-ci enverra un appel à propositions aux participants, qui sont l'agent2, Agent3 et l'Agent4, fournisseurs de ce produit. Les participants vérifient

dans leurs bases de données (croyances) s'ils peuvent répondre à cette demande. Si c'est le cas, ils envoient leurs propositions à l'initiateur (Agent1). Le déroulement de cet appel d'offre initié par l'agent1 est illustré dans la figure suivante:

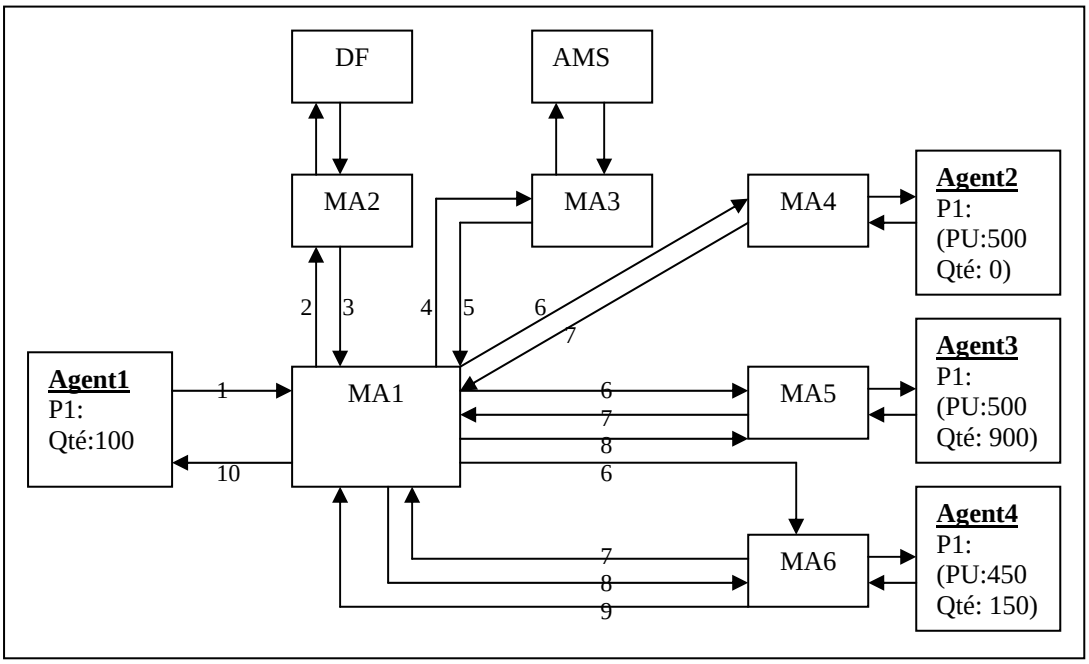


Fig. 5.11. Déroulement des interactions dans le cas d'appel d'offre

Au départ, l'Agent1 dépose sa requête d'appel d'offre auprès du mécanisme d'interaction. Cette requête comprend la désignation du produit (P1), la quantité demandée, l'ontologie utilisée et le deadline des offres. Le MI, crée alors le message actif (MA1) qui sera chargé de cette interaction. Le MA1 choisit le protocole d'interaction FIPA-contract-net pour suivre cette interaction. Le MA1 consulte le DF pour avoir la liste des agents fournisseurs du produit P1 (Agent2, Agent3 et Agent4). Le MA1 récupère les adresses de ces agents auprès de l'AMS, compose le message contenant l'annonce, et envoi ce message aux agents participants. Pour répondre à cette annonce, les messages actifs MA4, MA5 et MA6 sont crées, respectivement pour les agents Agent2, Agent3 et Agent4. Et Comme il s'agit d'un appel à propositions, ces messages actifs utiliseront le protocole d'interaction FIPA-contract-net pour répondre. Le MA1 reçoit les propositions où Agent2 refuse de proposer, Agent3 et Agent4 propose leurs prix. Il répond à l'Agent3 par un rejet de proposition "reject-proposal" et à l'Agent4 par une confirmation de proposition "accept-proposal". L'Agent4 notifie cette confirmation en établissant un contrat. Le MA1 informe l'Agent1 que l'Agent4 est sélectionné comme le fournisseur le moins-disant du produit P1.

La demande d'appel d'offre, initié par l'agent1, est donnée par le message FIPA-ACL suivant:

```
(cfp
        :sender Agent1
        :language Fipa-Sl0
        :ontology ontol1
```

```
         :content ( Price
                    :product P1
                    :Quantity 100
                    :?prix)

         :reply-by 10
)
```

Le message FIPA-ACL composé par MA1 puis adressé aux agents Agent2, agent3 et Agent4 est le suivant:

```
(cfp
         :sender Agent1
         :receiver Agent2
         :language Fipa-Sl0
         :ontology ontol1
         :content (Price
                    :product P1
                    :Quantity 100
                    :?prix)
         :reply-by 10
         :conversation-id conv01
         )
```

Le message FIPA-ACL composé par MA5 et MA6 pour proposer leur prix est le suivant:

```
(propose
         :sender Agent3
         :receiver Agent1
         :language Fipa-Sl0
         :ontology ontol1
         :content (Deal
                    :product P1
                    :Quantity 100
                    :price 450 )
         :conversation-id conv01
         )
```

Le message FIPA-ACL composé par MA4 pour refuser de proposer est le suivant:

```
(refuse
         :sender Agent2
         :receiver Agent1
         :language Fipa-Sl0
         :ontology ontol1
         :content (Price
                    :product P1
                    :Quantity 100
                    :?prix)
         :conversation-id conv01
         )
```

3. Conclusion:

Dans ce chapitre, nous avons présenté la plateforme MAP. Le développement de cette plateforme nous a permis de mettre en œuvre le modèle d'interaction dynamique MID. Ce modèle permet de partager l'interaction entre l'agent et le mécanisme d'interaction. Ce partage se fait sous la forme d'une délégation par l'agent au mécanisme d'interaction; le mécanisme d'interaction, à travers les messages actifs prend en charge l'interaction. En adoptant le MID, cette plateforme est caractérisée par sa flexibilité, car elle décharge l'agent d'un certains nombre de responsabilités, ce qui permet l'utilisation d'agents plus simples et de n'imposer aucun modèle d'agents.

Nous avons développé une architecture qui repose sur la spécification d'architecture abstraite de FIPA, cette spécification définie les éléments architecturaux et leurs relations, ainsi que des règles de conformité pour les agents et les plateformes.

La plateforme MAP va permettre de développer des applications multi-agents. Le fonctionnement de cette plateforme est testé en réalisant quelques applications qui seront présentées dans le chapitre 7.

Chapitre 6

Implémentation de la plateforme MAP

Nous avons présenté dans le chapitre précédent l'architecture et le fonctionnement de la plateforme MAP. Dans ce chapitre, nous allons reprendre l'étude de la plateforme, mais cette fois-ci du point de vue implémentation.

Nous commencerons par faire introduire le langage de programmation, que nous utilisons pour implémenter la plateforme. Dans la deuxième section, nous présentons l'implémentation des différents composants de la plateforme, à savoir le mécanisme d'interaction, l'AMS, le DF, le MTS, et l'interface graphique.

1. Environnement d'implémentation:

Le premier choix que nous avons dû effectuer fut celui du langage d'implémentation. Nous avons choisi la plateforme Java, comme pour la majorité des plateformes muti-agents, pour l'ensemble des raisons suivantes :

Le paradigme de programmation orienté objet a été choisi pour implémenter la plateforme, car la notion d'objet est proche de celle d'agent dans les systèmes multi-agents. En effet, les connaissances et le comportement des objets sont séparés au niveau de l'implémentation, ce qui facilite le contrôle de l'état interne des objets. Comme les agents, les objets communiquent également par envoi de messages.

Ce qui distingue un agent d'un objet est essentiellement l'attitude intentionnelle de l'agent. En effet, les objet n'ont pas de but, leur comportement est déterminé uniquement à partir des messages q'ils reçoivent. Les objets sont donc définis au niveau implémentation et les agents à un niveau conceptuel, ce qui fait que le paradigme de programmation orienté objet est particulièrement adapté pour implémenter des modèles multi-agents [Ferb97].

Bien qu'il y ait certaines similarités entre les approches orientées objet et celles orientées agent, il y a aussi plusieurs différences importantes. Premièrement, les objets sont généralement passifs : ils ont besoin de recevoir un message avant de devenir actifs. Deuxièmement, bien que les objets encapsulent l'état et le comportement, ils n'encapsulent pas le choix de l'action. Troisièmement, l'approche

orientée objet ne fournit pas un ensemble de concepts assez riche et les mécanismes nécessaires pour modéliser les systèmes complexes à un niveau suffisant d'abstraction.

Nous avons utilisé le langage Java car il présente des avantages significatifs par rapport à d'autres langages de programmation par objets tel que C++:

- java est un langage mixte, compilé (génération du bytecode) et interprété;
- il est portable, car le même bytecode java peut être exécuté sur différentes machines virtuelles java JVM (Java Virtuel Machine);
- java est un langage très fiable, car il permet la détection d'erreurs au moment de la compilation et de l'exécution;
- la gestion des processus légers existe dans la version standard du langage, ce qui le rend bien adapté à la problématique multi-agents, car la notion de concurrence est facilement programmable;
- la gestion de la mémoire est automatique (utilisation du garbage collector), simplifiant significativement la tâche du programmeur
- java permet le développement d'applications distribuées et hétérogènes (programmation en réseau);
- java est un langage de programmation adapté à internet.

Par ailleurs, java offre la possibilité d'interroger des méthodes d'objets à distances à travers le RMI (Remote Method Invocation). Le principe de fonctionnement consiste à écrire une interface java définissant l'ensemble des méthodes de l'objet qui pourront être invoquées à travers le réseau. Cette interface va dériver de l'interface java : *Remote*, et sera implémentée par l'objet héritant lui même de la classe java : *UnicastRemoteServer*. L'objet aura alors deux parties, la partie distribuée provenant de l'interface et le reste qui sera sa partie locale. Une fois l'objet distant codé, il est compilé avec le compilateur *rmic*, qui va créer deux classes : une classe *stub* du coté de la machine appelante et une autre nommée *skeleton* pour la machine appelée. Une fois cette étape terminée, l'objet est prêt à être distribué. Le serveur RMI (le *rmiregistry*) doit être lancé et un programme java ira inscrire une instance de notre objet nouvellement créé auprès de ce serveur. Dès cet instant, l'objet est distribué et accessible à travers le réseau.

2. Modèle d'agents

Un agent est une unité autonome et communicante. Il ne doit pas se limiter à réagir aux événements externes, mais il doit être aussi capable de prendre l'initiative de nouveaux actes communicatifs d'une façon autonome. Son autonomie nécessite donc que chaque agent ait un thread (processus) de contrôle au moins. Et afin de permettre aux agents de recevoir et traiter des messages asynchrones, chaque agent est représenté par deux threads, l'un gérant l'exécution de l'agent et l'autre traitant ses messages asynchrones. Cependant, un agent peut engager, à travers le mécanisme d'interaction, des conversations simultanées multiples, tout en poursuivant d'autres activités qui n'impliquent pas d'échanges de messages.

La classe *Agent* représente une super-classe commune pour tous les agents définis par l'utilisateur. Du point de vue du programmeur, un agent applicatif est

simplement une classe Java qui étend la classe de base *Agent*. Cela permet à l'agent d'hériter un comportement fondamental (qui traite toutes les tâches liées à la plateforme, telles que l'enregistrement, la communication avec le mécanisme d'interaction, etc.), et un ensemble de méthodes qui peuvent être appelées pour implémenter les tâches spécifiques à l'agent, par exemple envoi et réception des messages, utilisation des protocoles d'interaction standard, etc.

3. Structure de Message:

Les messages sont définis par héritage à partir d'une classe de base Message qui ne définit que la notion d'émetteur et destinataire. Une bibliothèque de messages de base (voir figure) est fournie et permet l'envoi de chaînes, d'objets sérialisés sous forme de messages conformes aux spécifications KQML et FIPA-ACL.

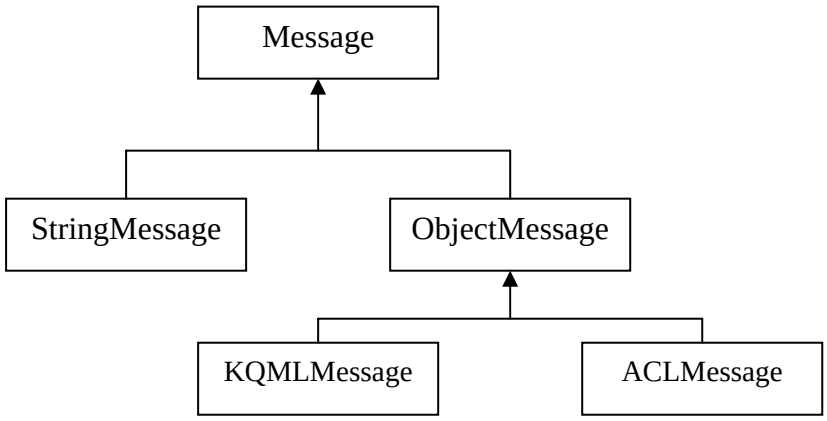


Fig.6.1. Hiérarchie des messages

La classe ACLMessage (respectivement KQMLMessage) représente les messages, dans le langage de communication entre Agents FIPA-ACL (respectivement KQML), qui peuvent être échangés entre agents. Elles contiennent un ensemble d'attributs (Performative, sender, receiveir, language, content,…).

Pour envoyer un message, les agents ou bien les messages actifs créent un nouvel objet, instancié à partir de ACLMessage ou KQMLMessage (selon le langage utilisé par le destinataire). Les attributs de cet objet sont remplis par les valeurs appropriées, puis la méthode *Send*() est lancée pour envoyé cet objet au système de transport des messages (MTS). De la même façon, la méthode *Receive()* est lancée pour recevoir un message.

Les messages sous forme de chaînes de caractères, sont utilisés pour communiquer avec les autres plateformes FIPA.

4. Les agents Système:

A l'heure actuelle, un débat a lieu au sein de FIPA sur le sujet de "l'agentification" des trois services de l'AMS, DF et ACC (ou MTS). Initialement dans les spécifications FIPA'97, ces trois services étaient des agents à part entière, au sens FIPA. Ils communiquaient en FIPA-ACL et étaient des intermédiaires obligés.

Par exemple, pour envoyer un message d'un agent sur une plateforme à un autre sur une plateforme distante, il faut déjà faire suivre le message à l'ACC local. Celui-ci transmettrait alors ce message à son homologue. Ce mécanisme a été critiqué pour son inefficacité et sa lourdeur : une interaction n'étant pas quelque chose d'anodin du point de vue traitement, on double quasiment toutes les interactions par des interactions contraignantes pour des tâches purement administratives.

Voici par exemple le message nécessaire pour envoyer un message d'un agent agent-a à un autre agent-b si l'on observe scrupuleusement la norme :

```
(request
      :sender agent-a
      :receiver acc
      :content (action acc
                  (forward
                  (request
                        :sender agent-a
                        :receiver agent-b
                        :content
                        ......
                  )))
:protocol fipa-request
:Language FIPA-ACL
:language sl0)
```

L'approche actuelle est donc de considérer l'ACC comme un service non-nécessairement agent, qui peut être décrit en termes d'objet, à l'usage des agents de la plateforme et qui puisse éviter le déploiement d'une architecture de traitement lourde pour ce service de base.

Dans la suite, nous présentons l'implémentation des trois services de bases de la plateforme, à savoir, l'agent AMS, l'agent DF et le système de transport de message MTS.

4.1. L'agent AMS:

L'AMS est défini par héritage de la classe *Agent*. Il implémente un ensemble de méthodes statiques permettant de réaliser les services offerts par l'AMS. Ces méthodes sont:

- Nommer: Attribution d'identificateur d'agent
- Enregistrer: Enregistrement d'agents
- Modifier: Modification de description d'agent
- Rechercher: Recherche de description d'agent
- Détruire: Destruction d'agent
- Enregistrer_Conteneur: Enregistrement de conteneurs
- Détruire_Conteneur: Déconnexion de conteneurs

4.2. L'agent DF:

De la même façon que l'agent AMS, l'agent DF est obtenu par héritage de la classe Agent. Les méthodes définies pour cet agent sont:

- Enregistrer: enregistrement des agents ainsi que les services qu'ils offrent ou dont ils ont besoin

- Rechercher_fournisseur: recherche des agents offrant un service
- Rechercher_Client: recherche des agents demandant un service
- Modifier: modification des services assurés par un agent
- Supprimer: suppression d'agents du répertoire.
- Structurer: organisation des agents en groupes suivant les services qu'ils offrent.

4.3. Le système de transport des messages MTS:

Dans le but de simplifier la transmission, les messages internes (sur la même plateforme) sont transférés codés comme des objets et non comme des chaînes de caractères. Quand l'expéditeur ou le récepteur n'appartient pas à la même plateforme, le message est automatiquement converti au format de chaîne de caractères spécifié par la FIPA. De cette façon, la communication est cachée au programmeur d'agents, qui a seulement besoin de traiter la classe d'objet Message.

La communication peut être réalisée de trois façons différentes, selon l'emplacement du destinataire:

a. Intra plateforme, Intra container : envoi des messages directement vers le Mécanisme d'Interaction situé sur le même container, en utilisant la sérialisation des objets messages

b. Intra plateforme, Inter containers : envoi vers le Système de Transport des messages situé dans le conteneur dans lequel l'agent destinataire s'exécute, en utilisant l'invocation de méthodes à distance RMI (Remote Method Invovation)

c. Inter Plateformes : envoi des messages, sous forme de chaînes de caractères, vers le destinataire situé dans une autre plateforme FIPA en utilisant un protocole de transport entre plateformes tel que IIOP (Internet Inter-ORB Protocol), http (Hypertext Transmission Protocol),…).

Le MTS reçoit les messages à partir d'un autre MTS situé sur un autre conteneur ou à partir d'un message actif qui s'exécute sur le même conteneur. Ces messages seront redirigés directement vers le Mécanisme d'interaction local.

5. Le Mécanisme d'Interaction:

La figure suivante représente le traitement effectué par le mécanisme d'interaction sur les messages reçus.

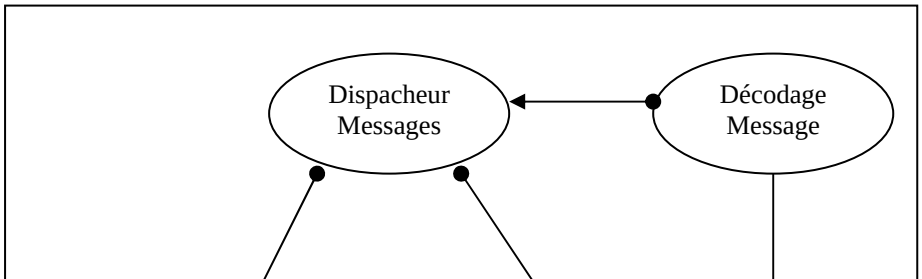


Fig.6.2. Traitement des messages par le MI

Le MI intercepte les messages entrants dans la file d'attente interne de messages. Ces messages sont reçus soit à partir d'un agent initiant une interaction, ou à partir du système de transport des messages MTS. En utilisant l'identifiant de conversation comme clef, le MI redirige les messages automatiquement vers le Message actif (gestionnaires de conversations) associé. Les messages initiant une nouvelle interaction sont utilisés pour la création et l'initialisation d'un nouveau Message actif.

Le traitement sur les messages reçus est effectué par le cycle suivant:

```
Cycle
        Recevoir (Message)
        Décoder (Message) // récupérer les champs Conv-id : Cv-id et  receiver: ag_r
        Rechercher (ag_r,Cv-id , Table-conversation,Trouve)
        Si Trouve
           Alors  Send(Message, Message Actif correspondant)
          Sinon Créer(NouvelleConversation-id, NouveauMessageActif, ag_r)
     Fsi;
Fin_Cycle
```

La création d'un message actif est effectuée comme suit:

a. Obtention d'un identificateur de conversation : Cet identificateur est obtenu par concaténation d'un numéro séquentiel (géré par chaque MI) à l'adresse IP de la machine hôte, il est de type Conv-NumSeq-AIP (exemple Conv-5-192.168.0.1). De cette manière, l'identificateur de conversation est unique dans la plateforme.

b. Insertion d'une ligne dans la table de conversations.

c. Lancement du thread du message actif.

5.1. Messages actifs:

Lorsqu'un message actif est créé, il est initialisé par les paramètres suivants:

- Identificateur de l'agent, pour lequel ce message actif est créé,
- Identificateur de conversation (Conv-id),
- Le message initial (émis ou reçu par l'agent).

Le traitement effectué par le message actif, lors de son initialisation, est donné selon les étapes suivantes:

1. *Choix du protocole*: parmi la liste de protocoles du Standard FIPA, le protocole d'interaction, qui sera utilisé pour contrôler cette conversation, est choisi, selon deux cas :

    a. Si l'agent correspondant à ce message actif, est initiateur de cette interaction, le protocole est choisi en fonction de la performative utilisée dans le message initial. La correspondance des performatives et des protocoles d'interaction est donnée dans le tableau suivant:

Performative	*Protocole d'interaction FIPA*
Request	Fipa-Request-protocol
Query ou Query-ref	FIPA-query-Protocol
Cfp	FIPA-Contract-Net- Protocol FIPA-Iterated-Contract-Net-Protocol
Inform (début de l'enchère)	FIPA-Auction-English-Protocol FIPA-Auction-Dutch-Protocol

    b. Si l'agent est un participant à cette interaction, le protocole est choisi selon le paramètre *Protocol* du message initial.

2. *Routage :* la fonction de routage est déclenchée dans le cas où l'agent, correspondant à ce message actif, est l'initiateur de cette interaction. Selon que le destinateur est nommé ou non dans le message initial, le routage est effectué comme il a été décrit dans la section (Chapitre 05- 1.5.1.a), il permet de désigner les participants de la conversation soit directement, soit à travers le rôle qu'ils jouent, soit à travers le groupe auquel ils appartiennent. Autrement dit, le routage peut être fait d'une façon *directe* ou *indirecte*.

3. *Suivi des Protocoles*: Après avoir choisi le protocole d'interaction, le message actif, selon que l'agent engendrant ce message actif, est initiateur ou participant à cette interaction, la partie correspondante du protocole (initiateur ou participant) est instanciée et exécutée en ayant comme paramètres les informations données

initialement par l'agent. Par exemple, dans le cas d'une négociation, l'agent initiateur fait un appel à propositions (cfp) d'un produit P, le message actif correspondant à cet agent, choisit le protocole Fipa-contract-net, et exécute la partie "initiateur" de ce protocole, les paramètres utilisés sont la désignation du produit et le deadline des propositions.

4. *La composition des messages* utilise des schémas de conversation qui sont obtenus à partir du Mécanisme d'interaction au fur et à mesure qu'ils sont nécessaires. La figure suivante représente le schéma de conversation utilisé au moment d'un appel à propositions d'un agent.

Schéma negotiate
Arguments (content , destinataire, reply)
FIPA-ACL
Composition (type : cfp ,
protocol : FIPA-Contract-net ,
Language: Fipa-Sl0
Reply-by: argument(reply)
content : argument(content) ,
to : argument(destinataire))

Schéma de conversation : négociation.

5. *La traduction* intervient, après la composition du message, dans le cas où l'agent destinataire utilise un langage d'interaction différent. Pour le moment, dans le prototype implémenté, tous les agents utilisent le même langage (d'interaction et de contenu) de telle façon qu'il n'a pas été nécessaire de faire appel aux services de traduction.

5.2. Protocoles d'interaction:

Fipa spécifie un ensemble de protocoles d'interaction, qui peuvent être utilisés comme des modèles standard pour établir des conversations entre agents. Dans toute conversation entre agents, deux rôles sont distingués; le rôle initiateur (l'agent initiant une conversation) et le rôle participant (l'agent engagé à une conversation après avoir été contacter par un autre agent). Le MI fournit pour chaque protocole d'interaction FIPA, deux classes, une classe pour le rôle initiateur de conversation et une deuxième classe pour le rôle de participant à la conversation.

Par exemple, le protocole Fipa-Request est défini par les classes suivantes:

```
public class FipaRequestInitiator {
    static MessageActif actif;
    protected String ConversationId=new String();
    protected static ACLMessage msg;
    static boolean finished;
    static Session session = new Session();
    public FipaRequestInitiator(MessageActif actif,ACLMessage
                                msg,boolean finished) {
           this.actif=actif;
           this.msg=msg;
           this.finished=finished;
         this.sendInitiation(this.msg);
     }
```

```
… … …
    protected void SuiviSequence(ACLMessage reply) {
                  if (this.session != null) {
                        int perf = reply.getPerformative();
                    if (session.update(perf)) {
                      switch (session.getState()) {
                      case Session.POSITIVE_RESPONSE_RECEIVED:
                           handleAgree(reply);
                      case Session.NEGATIVE_RESPONSE_RECEIVED:
                             handleRefuse(reply);
                              break;
                      case Session.RESULT_NOTIFICATION_RECEIVED:
                             this.actif.SendMessage(reply);
                             handleResultNotifications(reply);
                              break;
                      default:
                        // erreur de suivi du protocole
                      }
                       if (session.isCompleted()) {
                         this.finished=true;
                       }
                    }
                  }
    }
static class Session implements Serializable{
      static final int INIT = 0;
      static final int POSITIVE_RESPONSE_RECEIVED = 1;
      static final int NEGATIVE_RESPONSE_RECEIVED = 2;
      static final int RESULT_NOTIFICATION_RECEIVED = 3;
      private int Etat = INIT;
      boolean update(int perf) {
          switch (Etat) {
          case INIT:
            switch (perf) {
            case ACLMessage.AGREE:
                Etat = POSITIVE_RESPONSE_RECEIVED;
                return true;
            case ACLMessage.REFUSE:
            case ACLMessage.NOT_UNDERSTOOD:
                Etat = NEGATIVE_RESPONSE_RECEIVED;
                return true;
            case ACLMessage.INFORM:
            case ACLMessage.FAILURE:
                Etat = RESULT_NOTIFICATION_RECEIVED;
                return true;
            default:
                return false;
            }
          case POSITIVE_RESPONSE_RECEIVED:
            switch (perf) {
            case ACLMessage.INFORM:
            case ACLMessage.FAILURE:
                Etat = RESULT_NOTIFICATION_RECEIVED;
                return true;
            default:
                return false;
            }
          default:
            return false;
          }
      }
```

….

La Classe FIPA-Request-participant:

```
public class FipaRequestParticipant {
  private final static int WAITING_MSG_STATE = 0;
  private final static int PREPARE_RESPONSE_STATE = 1;
```

```
  private final static int SEND_RESPONSE_STATE = 2;
  private final static int PREPARE_RES_NOT_STATE = 3;
  private final static int SEND_RESULT_NOTIFICATION_STATE =4;
  private final static int RESET_STATE = 5;
  private int state = WAITING_MSG_STATE;
  private String replymsg,responsemsg;
  private Map store = new HashMap();
  private boolean finished = false;
  MessageActif_Request_Partricipant requestparticipant;
  ACLMessage msg;
… … …
public final void action(ACLMessage reply){

  switch(state){
      case WAITING_MSG_STATE:{
            ACLMessage request = reply;
            if(request != null){
              state =SEND_RESPONSE_STATE;
              store.put(replymsg,request);
              this.requestparticipant.Send(request);
            }
            else
             // Message vide
            break;
      }
      case SEND_RESPONSE_STATE:{
          ACLMessage response = reply;
          ACLMessage receivedMsg =(ACLMessage)
          store.get(replymsg);
          if(response != null){
            response = arrangeMessage(receivedMsg,response);
            this.requestparticipant.Send(response);
            if(response.getPerformative() == ACLMessage.AGREE)
                state = SEND_RESULT_NOTIFICATION_STATE;
            else
                this.requestparticipant.finished=false;
          }else{
                  this.requestparticipant.finished=false;
          }
          break;
      }
      case SEND_RESULT_NOTIFICATION_STATE:{
          ACLMessage resNotification = reply;
          if(resNotification != null){
            ACLMessage receivedMsg =
           (ACLMessage)store.get(replymsg);
      this.requestparticipant.Send(arrangeMessage(receivedMsg,
      resNotification));
     this.requestparticipant.finished=false;
          }
          break;
      }

      }
    }
```

Lors d'une interaction de type *Request*, l'une des deux classes Fipa-Request-initiator ou Fipa-Request-participant est instanciée, par le message actif, selon que l'agent engendrant ce message actif est initiateur ou participant à cette interaction.

6. Interface Graphique (Graphic User Interface):

L'interface graphique est implémentée sous forme d'un agent. Elle communique avec le reste des composants de la plateforme pour recueillir des

informations. Par exemple, pour afficher les agents qui s'exécutent dans un conteneur, l'interface interroge l'agent AMS pour avoir la liste des agents.

L'interface offre un outil traceur de messages permettant de vérifier les différents transferts de messages entre agents. Cette interface permet également de visualiser la table de conversations de chaque Mécanisme d'interaction. Et il est envisageable d'intégrer d'autres outils permettant d'effectuer des contrôles sur le fonctionnement de la plateforme, tel que l'évaluation de la durée d'une interaction et le nombre de messages effectués pour une interaction, la communication par le langage FIPA-ACL entre utilisateurs et agents, etc…

La figure suivante présente l'interface graphique de la plateforme.

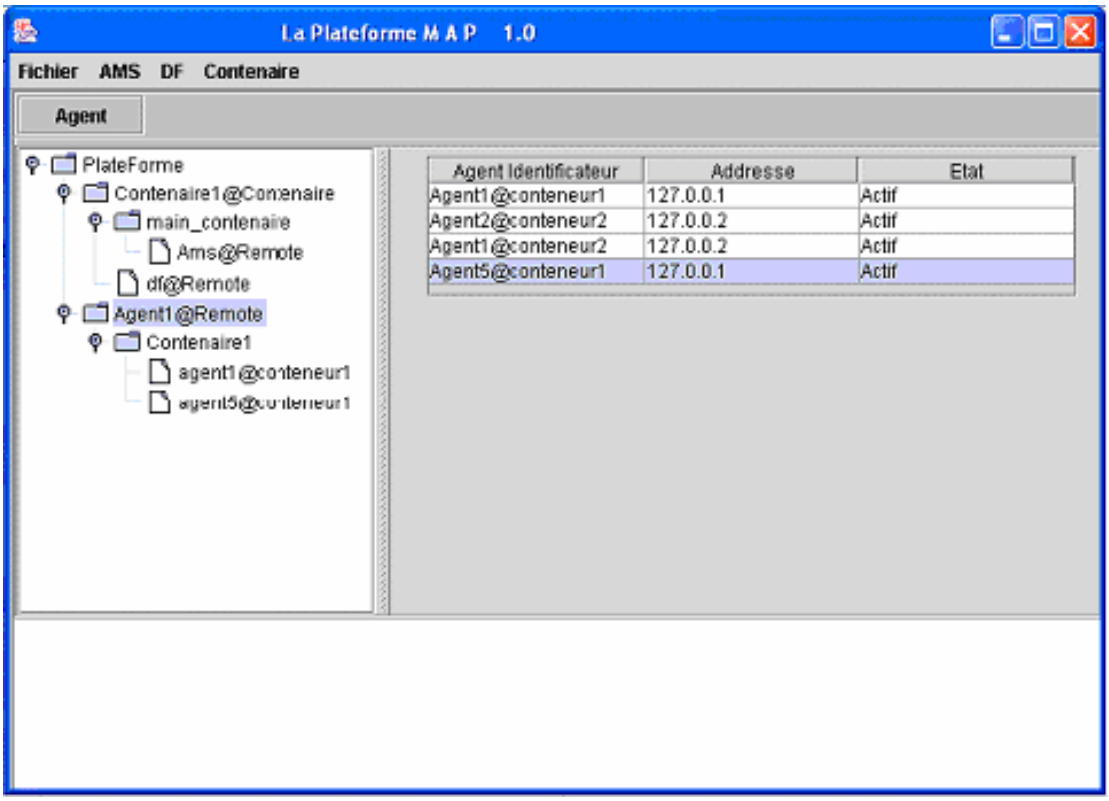


Fig.6.3. Interface graphique

7. Conclusion:

Dans ce chapitre, nous avons présenté l'implémentation de la plateforme MAP basée sur le modèle d'interaction. Lors de la réalisation de cette plateforme, nous avons identifié que la principale difficulté rencontrée dans le développement d'un système multi-agents, est liée à la complexité induite par la gestion et le suivi des conversations.

L'accent est mis sur le Mécanisme d'Interaction, lors du développement de cette plateforme, en effet, le Mécanisme d'interaction joue un rôle central dans la plateforme, il permet de gérer et de suivre les interactions (Messages Actifs). Pour cela, nous avons dû faire quelques restrictions qui vont être prises en compte ultérieurement, ces restrictions sont les suivantes: d'abord, le service de traduction n'est pas encore mis en œuvre, car nous n'avons considéré qu'un langage de

communication entre agents le FIPA-ACL, et comme langage de contenu, nous avons considéré le langage SL0. Cependant, la conception permet une éventuelle extension des services de la plateforme, telles que la migration des agents d'un conteneur vers un autre et la communication avec d'autres plateformes FIPA (à travers le protocole IIOP) n'est pas encore mise en oeuvre.

Par ailleurs, nous possédons aujourd'hui un prototype opérationnel du logiciel qui nous permet de développer des petites applications. Le bon fonctionnement de cette plateforme fut validé par la réalisation de quelques applications de type Toy-Problems présentées dans le chapitre suivant.

Chapitre 7

Applications

Le fonctionnement de la plateforme MAP sera validé par la réalisation de quelques applications benchmarks, ces applications sont sous forme de Toy Problems. Les Toy-Problems sont des petits programmes montrant une ou plusieurs caractéristiques de la plateforme comme la distribution, l'auto-adaptation, la dynamicité, l'ouverture, etc. Ces implémentations permettent également de tester la fiabilité de la plateforme. Un certain nombre d'exemples types ont été proposés par le groupe ASA [ASA02]. Ils permettent de comparer le nombre de lignes de code nécessaires pour l'implémentation de chacun d'entre eux sur différentes plateformes et comparer ainsi le niveau d'abstraction du langage. Les applications que nous avons choisis pour tester le bon fonctionnement de la plateforme, sont : la vente aux enchères, le toy-problem des proies et prédateurs et le jeu Pièrre-Papier-Ciseau.

Le présent chapitre est organisé en trois sections. La première section présente l'application de la vente aux enchères, la deuxième section est consacrée pour l'implémentation de deux exemples d'applications des toy-problems, le jeu PPC et les proies et prédateurs.

1. Vente aux enchères:

1.1. Vente aux enchères et systèmes multi-agents:

A partir de l'évolution et de la croissance du commerce électronique, le terme "vente aux enchères en ligne" devient aussi de plus en plus fréquent et important. Dans l'objectif de développer des applications simples pour comparer des plateformes SMA, le Groupe ASA [ASA02] propose un scénario de vente aux enchères pour un système multi-agents.

Dans ce scénario, on a le choix entre plusieurs modèles pour la réalisation d'une vente aux enchères, comme par exemple : le modèle anglais (plus traditionnel), le modèle hollandais (descendant) et le modèle " mieux disant " (offre fermé - premier prix, normalement utilisé dans les marchés public). Pour cette première approche multi-agents du problème, nous nous concentrons sur le modèle anglais, plus simple et plus connu. Dans ce modèle, le commissaire-priseur (Vendeur) établit le prix minimal initial pour un item à vendre, chaque participant fait une offre et tous les participants sont libres d'augmenter leur offre jusqu'au moment où personne ne veut plus surenchérir. La vente aux enchères est finie et le participant ayant soumis la meilleure offre, achète l'item pour le prix correspondant à son offre. Une stratégie très simple pour le comportement des participants dans une vente aux enchères est de faire des offres d'une valeur légèrement supérieure à l'offre courante (l'intérêt est toujours d'acheter l'item au prix le plus bas) et s'arrêter quand sa valeur limite est atteinte.

Un système multi-agents pour les enchères peut être imaginé, avec un agent commissaire-priseur et plusieurs agents participants (acheteurs). Les agents acheteurs achètent un seul objet à la fois (ça n'empêche pas l'agent CP d'avoir une liste d'objets à vendre et chaque agent acheteur sa liste d'objets qui peuvent l'intéresser).

1.1.1. Rôle et compétences de l'agent vendeur ou commissaire-priseur (CP) :

L'agent commissaire-priseur est le responsable de la vente aux enchères, donc il contrôle le cycle de participation à ses enchères. Le cycle d'une vente aux enchères est, grosso modo, divisé en deux parties :

a) La première partie, ou partie initiale, ressemble un peu au modèle descendant hollandais. Cette partie du cycle commence toujours avec l'agent CP qui décrit l'item à vendre et établit son prix minimal initial. Ensuite il attend les offres des acheteurs, pendant un certain temps. Au bout de cette période si cette valeur initiale n'a été acceptée par aucun des agents acheteurs, l'agent CP relance la proposition avec une réduction du prix initial (par exemple -10%) et attend qu'un acheteur se présente et accepte cette nouvelle valeur (un message d'offre). Cette boucle peut se répéter encore plusieurs fois

jusqu'au moment où, soit l'agent CP décide de retirer l'item de la vente, parce que personne ne s'intéresse à cet objet, soit un agent acheteur accepte cette proposition, et dans ce cas la deuxième partie du cycle commence.

b) La deuxième partie du cycle d'une vente aux enchères se caractérise par une compétition entre les agents acheteurs, chacun avec le but d'acheter l'item en vente pour la valeur la plus basse, dans les limites de ses disponibilités (valeur limite ou plafond).

Dans cette partie, l'agent CP va contrôler la participation des agents acheteurs aux enchères, il va mettre à jour de façon continue la variable " offre courante " par rapport à la dernière offre et il va contrôler la temporisation de toute les activités dans les enchères (par exemple, combien de temps il faut attendre avant de déclarer la vente faite (" coup de marteau "), etc.). Au moment où l'agent CP ne reçoit plus aucune nouvelle offre, il relance encore une ou deux fois l'offre courante. Si aucune nouvelle offre n'est lancée dans cette période, il décide que la vente est finie et déclare l'agent acheteur vainqueur, l'objet et le prix d'achat.

Dans notre application, le prix minimal (mise à prix) pour chaque item n'est pas négocié par les participants mais il est fixé par le vendeur initialement. Et si aucun participant n'est intéressé par ce prix, l'item correspondant est retiré automatiquement de l'enchère. Dans ce cas, le cycle des ventes aux enchères se résume en une seule partie, qui est la compétition des agents acheteurs pour l'achat d'un produit au meilleur prix.

1.1.2. Rôle et compétences des agents acheteurs :

Dans ce modèle l'agent acheteur représente un utilisateur humain qui lui définit ses intérêts (type d'item ou item spécifique) et établit les limites des ressources financières que l'agent a à sa disposition par item (valeur limite ou plafond). Chaque agent acheteur participant à une vente aux enchères, en ignorant les autres participants, dès qu'un item l'intéresse commence sa participation aux enchères par une offre. Ensuite, il suit l'évolution des enchères et augmente la valeur de son offre jusqu'à son plafond.

Lors de la compétition entre les agents acheteurs, le comportement d'un agent acheteur sera orienté par deux caractéristiques principales : le temps de réflexion qu'il faut attendre avant d'augmenter une offre et la décision sur le delta d'enchère qu'il va utiliser (de quelle valeur il va augmenter l'offre courante pour faire une nouvelle offre).

Evidemment, l'idéal est d'avoir des agents acheteurs avec comportements différents uns des autres, comme par exemple :

- Un agent peut être très statique, avec temps de réflexion et un delta d'enchère définis d'avance et constants (indépendamment de l'item en vente) ;
- Autres agents pourront être plus dynamiques où leurs temps de réflexion varie en fonction d'un ordre de priorité d'achat reçu de l'utilisateur (que peut aussi varier en fonction du plafond fixé par un item spécifique) ou en

fonction de la variation de l'offre en cours (la différence entre l'offre courante actuel et l'offre antérieure). Les deltas d'enchère pourront varier selon une fonction dynamique proportionnelle à l'ordre de priorité d'achat et inversement proportionnelle à la proximité du plafond ;

- On peut aussi avoir un agent acheteur très intelligent avec un comportement orienté selon l'historique des enchères. Il peut garder des informations sur les enchères précédentes (l'agent acheteur qui a gagner, item acheté, prix d'achat, profil de comportement, etc.) et les réutiliser pour changer sa propre stratégie d'achat dans les enchères suivantes (un agent apprenant).

L'agent acheteur doit aussi gérer son argent, ce qui veut dire que, dans le cas ou il achète un item pour un prix plus bas que la valeur limite qu'il avait pour cet item, la différence peut être réutilisée dans l'achat d'un autre item (l'item prioritaire suivant).

Une caractéristique importante qu'il faut prendre en compte est que l'annonce d'une offre interrompt le processus de réflexion de chaque agent et relance un nouveau processus de réflexion.

1.2. Implémentation de l'application des ventes aux enchères:

Chaque utilisateur (vendeur ou acheteur) possède un agent qui le représente vis-à-vis des autres agents (cf. Figure7.1). L'agent, que nous avons nommé Agent Vendeur (respectivement Agent Acheteur), est le responsable de la vente (respectivement Achat) de son utilisateur et doit négocier avec les autres agents; il reste actif tout le temps, même quand l'utilisateur n'est pas connecté. L'interface entre l'utilisateur et son agent est faite à travers le module d'interface, aussi nommé agent interface. L'agent interface est un programme qui peut être active seulement quand l'utilisateur est connecté

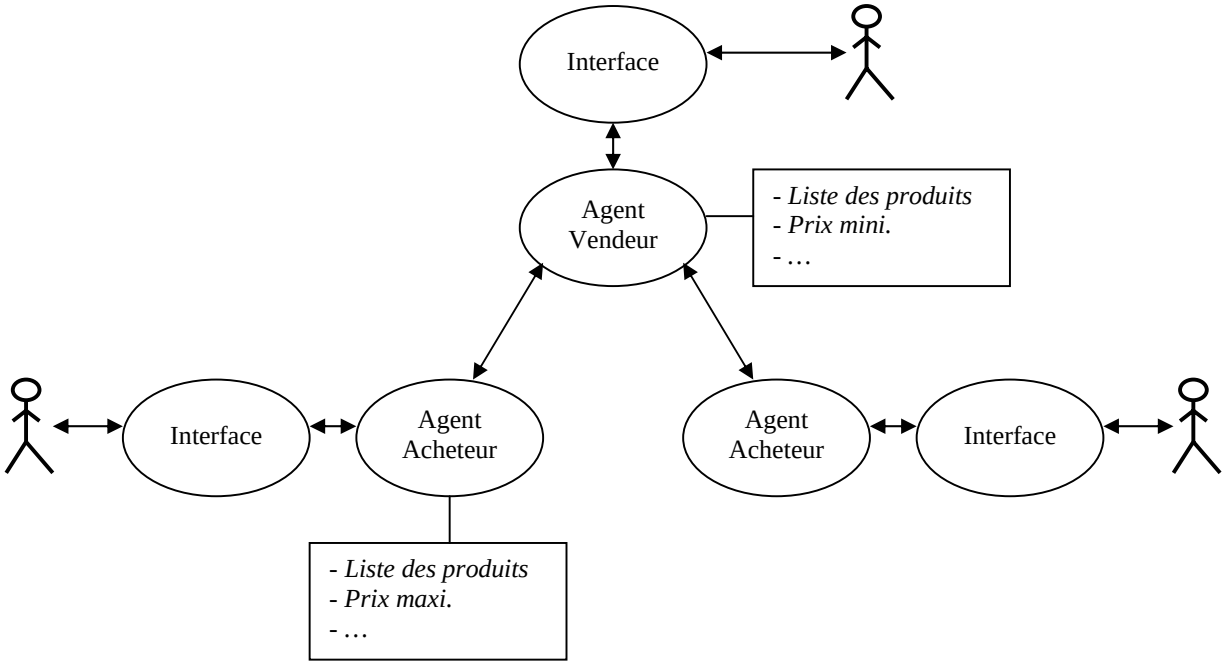


Fig.7.1. Schéma général de la vente aux enchères

1.2.1 Déroulement d'une enchère:

Le déroulement d'une enchère suit le processus décrit par les étapes suivantes:

- D'abord, l'utilisateur configure l'agent vendeur à travers son interface graphique, et choisi les paramètres de cette enchère, comme le prix minimal, la quantité et le produit à vendre ainsi que la date de l'enchère et le délai de réponse;
- Chaque utilisateur, participant à l'enchère, configure l'agent acheteur (client) à travers son interface graphique, et choisi les produits qui l'intéressent, la quantité, le prix maximal et le type du delta enchère utilisé pour chaque produit;
- Quand la date de l'enchère est arrivée, l'agent vendeur démarre l'enchère. Et il exécute le cycle suivant:

Répéter

- Sélectionner un produit P de la liste des produits à vendre;
- L'agent vendeur déclenche la recherche d'acheteurs potentiels, il va alors interroger le facilitateur de répertoire pour retrouver les participants intéressés par le produit P;
- Annonce le début de l'enchère;
- Offre_courante=PrixMin(P);
// suivre le protocole d'enchère anglaise, partie initiateur
Répéter
-Il prépare le message de cette enchère, en renseignant :
- Les participants à cette enchère;
- Le contenu du message, c'est-à-dire, informer les participants de l'offre_courante;
- envoi des messages aux participants
- Attente jusqu'à réception de toutes les réponses ou écoulement du délai d'attente (time-out);
-Si aucune offre alors fin;
Sinon Choisir la meilleure offre
Informer l'offrant par accept-prposal;
Mettre à jour l'offre_courante;
Fsi;
Jusqu'à fin (vente du produit ou retrait de ce produit)
-informer les participants de la fin d'enchère du produit courant;
- Si il existe une offre alors retenir l'agent ayant proposé
l'offre_courante;
Sinon Le produit est retiré des enchères;

Jusqu'à fin de la liste des produits;

- Chaque Agent acheteur exécute en cycle la boucle suivante jusqu'à fin de l'enchère:

Répéter
Recevoir offre sur un produit P;
Si Offre<prix_max_P Alors Surenchérir en augmentant l'offre courante de Delta enchère; puis proposer l'offre;
Jusqu'à vente du produit en cours ou son retrait;

La figure 7.2 représente l'interface graphique de l'agent vendeur. L'utilisateur choisit la liste des produits à vendre et fixe le prix minimum de vente de chaque produit. L'utilisateur fixe également la date du début d'enchère et le time-out. Après avoir initialiser son agent vendeur, l'utilisateur déclenche l'agent vendeur en appuyant sur le bouton *Lancer*.

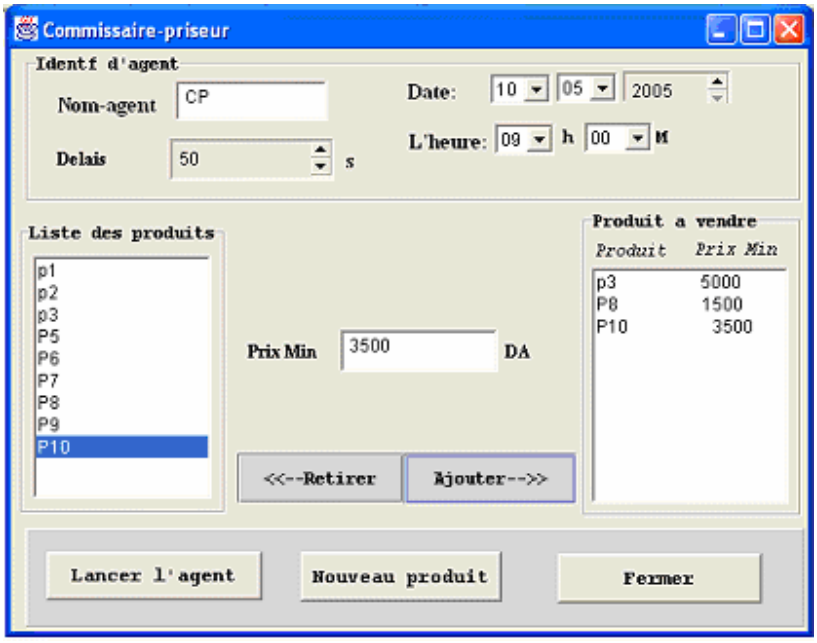


Fig.7.2. Interface d'initialisation de l'agent vendeur

L'interface graphique utilisée par un acheteur, est présentée dans la figure 7.3 ci-dessous. Elle permet à l'utilisateur de sélectionner la liste des produits à acheter ainsi que le prix max de chaque produit. L'utilisateur choisit également la politique de l'enchère où le delta d'enchère peut être un pas fixe, un taux fixe ou un Rapport variable selon l'offre courante et le prix max du produit.

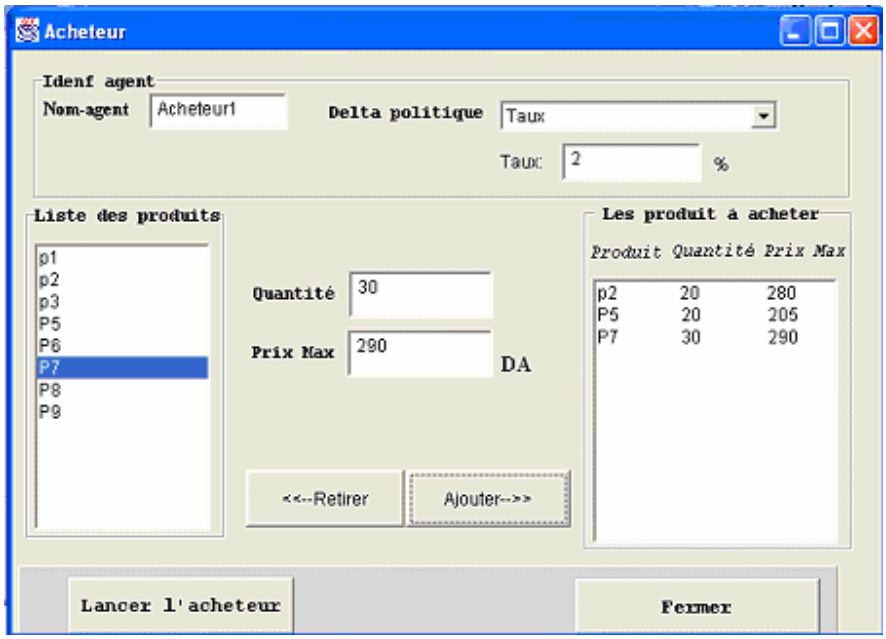


Fig.7.3. Interface d'initialisation de l'agent acheteur

2. Les Toy-Problems :

Dans la présente section, nous présentons deux toy-problems qui sont en cours d'implémentation sur la plateforme MAP. Ces problèmes sont le jeu Pièrre-Papier-Ciseau et le jeu Proies-prédateurs

2.1. Le jeu PPC:

Le jeu Pierre-Papier-Ciseaux est bien connu : c'est un jeu à deux joueurs où les joueurs jouent simultanément et ont trois coups possibles : Pierre, Papier ou Ciseaux.

Le résultat de chaque coup est :

- les deux joueurs jouent la même chose : pas de point
- Pierre contre Papier : Papier gagne (et marque 1 point)
- Pierre contre Ciseaux : Pierre gagne (et marque 1 point)
- Ciseaux contre Papier : Ciseaux gagne (et marque 1 point)

Une partie dure 10 coups.

Le but de ce problème est de jouer à PPC avec un système multi-agent. Un arbitre, appelé Banque, est nécessaire pour coordonner la partie puisque les deux coups des joueurs doivent être simultanés. Un des joueurs doit pouvoir être humain.

Dans cette application, trois types d'agents sont distingués:

- L'agent joueur humain: à travers son interface graphique, l'agent humain peut sélectionner un joueur avec qui il veut jouer. Une fois, l'adversaire est choisi, la partie est démarrée. A chaque coup, l'agent choisit et clic sur l'un des boutons qui correspond à pierre, papier ou ciseau. En ce moment le coup est communiqué à l'agent arbitre.

- L'agent joueur logiciel: est un agent qui choisit aléatoirement pierre, papier ou ciseau, puis à chaque coup, il communique son choix à l'agent Arbitre.

- L'agent Arbitre: est un agent qui reçoit pour chaque partie le choix effectué par chaque joueur, et comptabilise les points marqués pour chaque agent ayant gagné un coup.

Le schéma suivant (Figure 7.4) explique les relations existantes entre ces types d'agents:

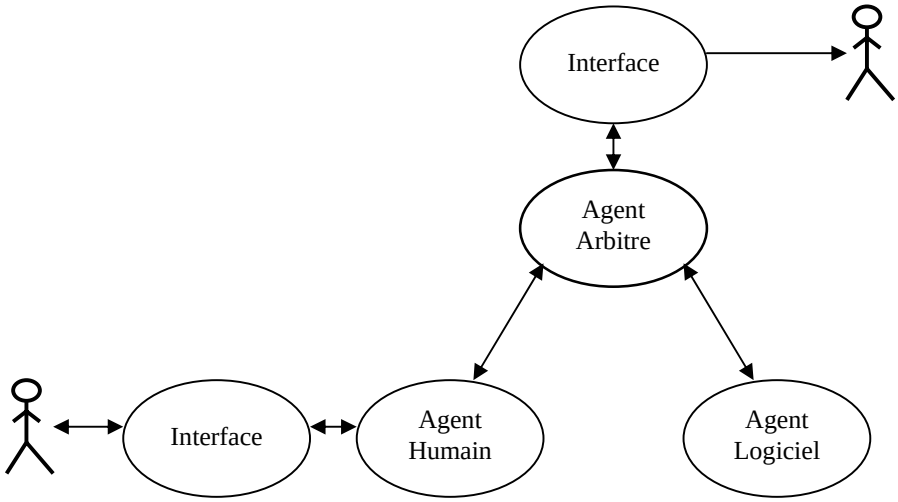


Fig.7.4. Schéma général du jeu PPC

Une fois les agents joueurs sont lancés, l'utilisateur, à travers son interface graphique, sélectionne son adversaire parmi les joueurs logiciels disponibles, et lance la partie. Après chaque délai T, les deux joueurs communiquent leurs choix à l'arbitre, qui à son tour compare les deux choix, et mis à jour le score. Après 10 coups joués par les deux joueurs, la partie est finie et l'arbitre communique le score aux deux joueurs.

L'interface graphique utilisée par l'agent joueur humain est présentée dans la figure suivante(Figure 7.5).

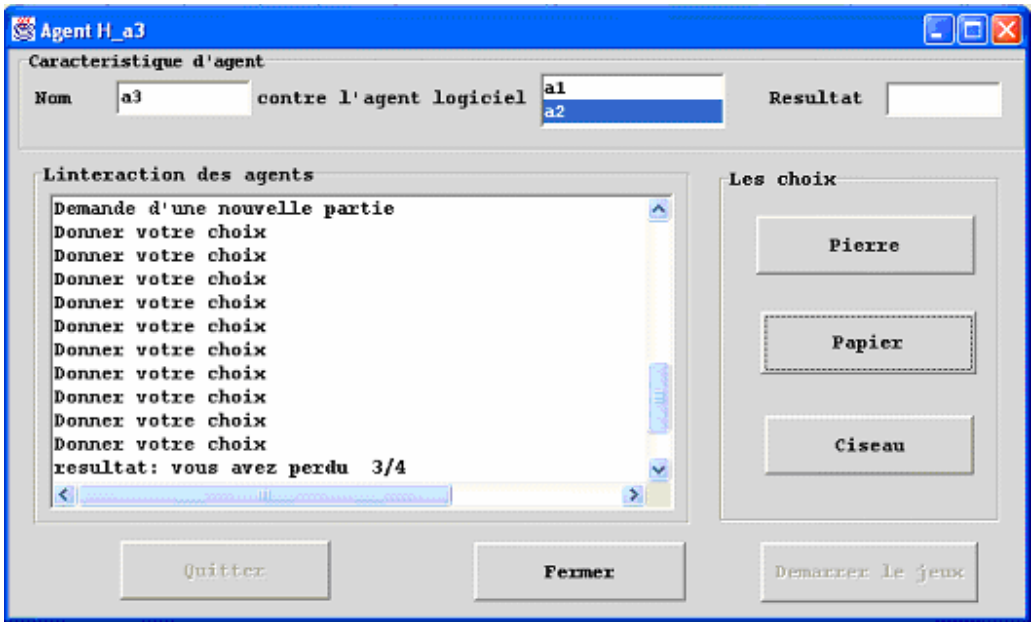


Fig.7.5. Interface graphique du joueur Humain

A tout moment, on peut consulté le score maintenu par l'agent arbitre à travers l'interface suivante (Figure 7.6):

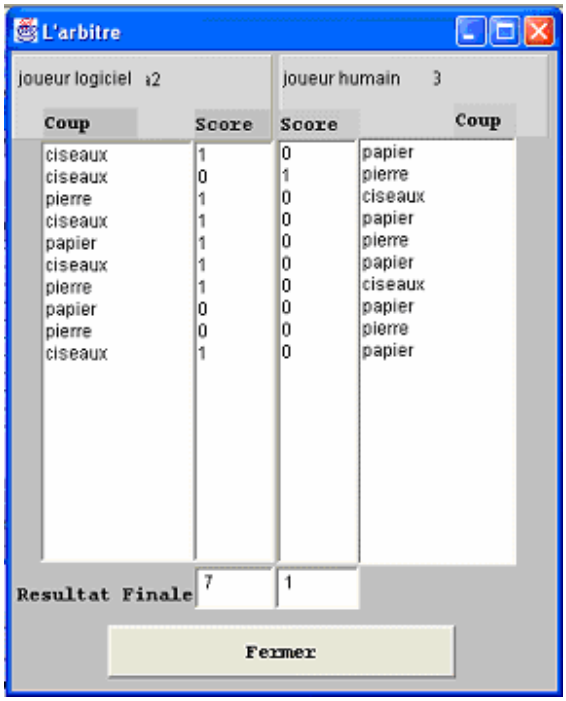


Fig.7.6. Interface de l'agent Arbitre

2.2. Proies-Prédateurs:

Les proies et prédateurs sont des entités actives et autonomes et évoluent dans un environnement spatial fini. Chaque agent (Proie ou Prédateur) a une quantité de ressources limites. Les prédateurs tentent d'encercler la proie pour la neutraliser avant qu'elle rejoigne les limites de l'environnement.

L'environnement spatial est représenté par une grille à deux dimensions. A un instant donné, chaque agent occupe une position déterminée dans la grille. Cette position est différente de la position occupée par les autres agents au même instant. Le déplacement des agents sur la grille s'effectue suivant l'axe vertical ou l'axe horizontal. Tous les agents évoluent à la même vitesse. Les prédateurs ont une perception limitée. Un prédateur ne perçoit que les agents qui se situent dans son champ de perception (ce champ est généralement circulaire).

La poursuite commence lorsque une proie rentre dans le champ de perception d'un prédateur. Quand à la proie, elle n'a pas de perception et son déplacement est aléatoire.

Dans cette application, on distingue trois types d'agents:

- Agent proie: il est initialisé par une position aléatoire et libre sur la grille. Il se déplace aléatoirement dans la grille, après chaque période T. il est automatiquement éliminé dans les deux cas suivants: - neutraliser par un prédateur ou bien il déborde les limites de la grille.
- Agent prédateur: initialement, un agent prédateur est placé dans une position aléatoire et libre sur la grille. Son but est de chasser un maximum de proie, pour ce faire, il contacte l'agent grille pour avoir le contenu des cases adjacentes. En fonction de son champ de perception, il décide quelle est sa prochaine position. Si aucune proie n'est visible, il se déplace aléatoirement. Il maintient une variable Nbproies qui s'incrémente une fois ce prédateur capte une proie.
- Agent interface (grille): l'agent interface est chargé de visualiser à tout instant (après chaque déplacement d'un agent proie ou prédateur) les emplacements des agents sur la grille. Initialement, la grille est vide. Cet agent est consulté par les agents prédateurs pour décider de leurs prochains déplacements, et par tous les agents à leur lancement pour avoir un emplacement aléatoire vide sur la grille.

Dans cette application, le système est composé d'un agent grille et de plusieurs agents proies et prédateurs. Le schéma général de cette application est présenté dans la figure 7.7 suivante.

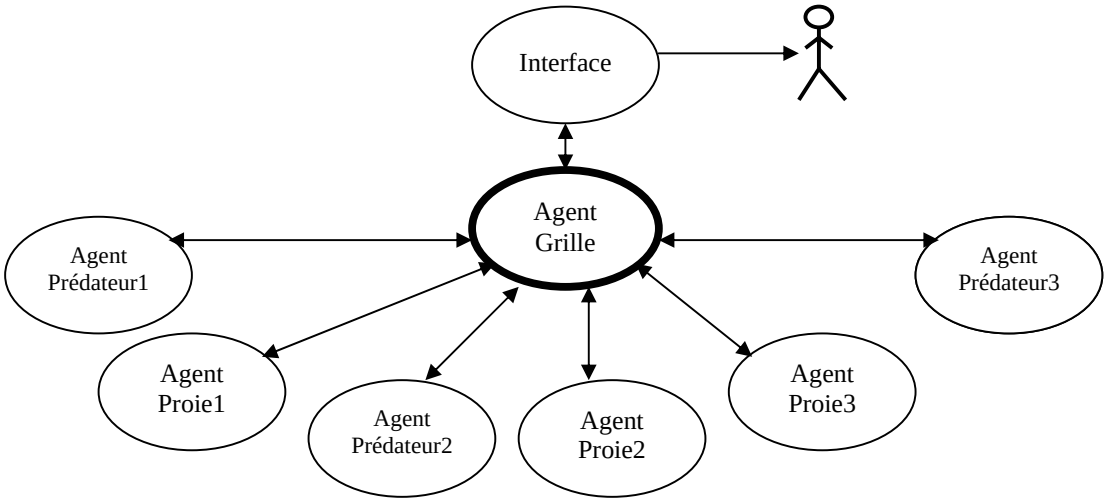


Fig.7.7. Schéma général de l'application Proies-prédateurs

3. Bilan et Conclusion:

Pour le moment, seules l'application des ventes aux enchères et le jeu PPC qui sont mise en œuvre, le reste des applications est en cours de développement. Dans ce qui suit, nous discutons les résultats obtenus pour les applications ventes aux enchères et le jeu PPC.

Le bon fonctionnement de la plateforme a été validé par le développement de ces applications. Après une phase de déboggage (environ 30 % du temps de développement), le prototype s'est comporté correctement dans les différents tests effectués.

L'analyse des traces d'exécution, obtenus à partir du traceur de messages de l'interface graphique, des messages envoyés, montre que les interactions se déroulent comme prévu, et que le MI, ainsi que le reste des composants de la plateforme, fonctionnent correctement. Dans ce qui suit, nous donnons des exemples d'exécution pour la vente aux enchères et pour le jeu PPC.

Plutôt que d'inclure tous les messages ici, il nous semble plus utile de les retranscrire sous forme de diagrammes de séquence UML, plus lisibles.

3.1. Exemple d'exécution pour l'enchère:

La figure 7.8 montre une trace pour une enchère, dans laquelle le vendeur émet une offre à un prix initial de 10, et l'acheteur 1 est prêt à payer 18, tandis que l'acheteur 2, ainsi que les autres, ne peuvent offrir plus de 12, le delta d'enchère utilisé par les agents est fixe et est égale à 1. L'enchère va se dérouler en quatre itérations (on n'a représenté que deux pour plus de clarté), après quoi le *Message actif* propose au vendeur l'offre gagnante. Puis ce dernier l'accepte, cette acceptation est confirmée par le Message actif, qui va ensuite finaliser la transaction avec le vainqueur de l'enchère. Seuls le vendeur, le *Message actif* représentant le vendeur, et deux acheteurs, sont représentés. En réalité, lors de certaines applications le nombre des participants s'élève à plusieurs dizaines, mais le principe ne change pas.

De même, comme chaque agent interagit via son message actif et que le vendeur interagit avec le facilitateur de répertoire DF pour avoir la liste des acheteurs, il est difficile de représenter ces derniers, car la figure serait trop chargée. Néanmoins, nous avons donné, en Annexe B, un exemple d'exécution où nous explicitons les différents messages échangés entre agents.

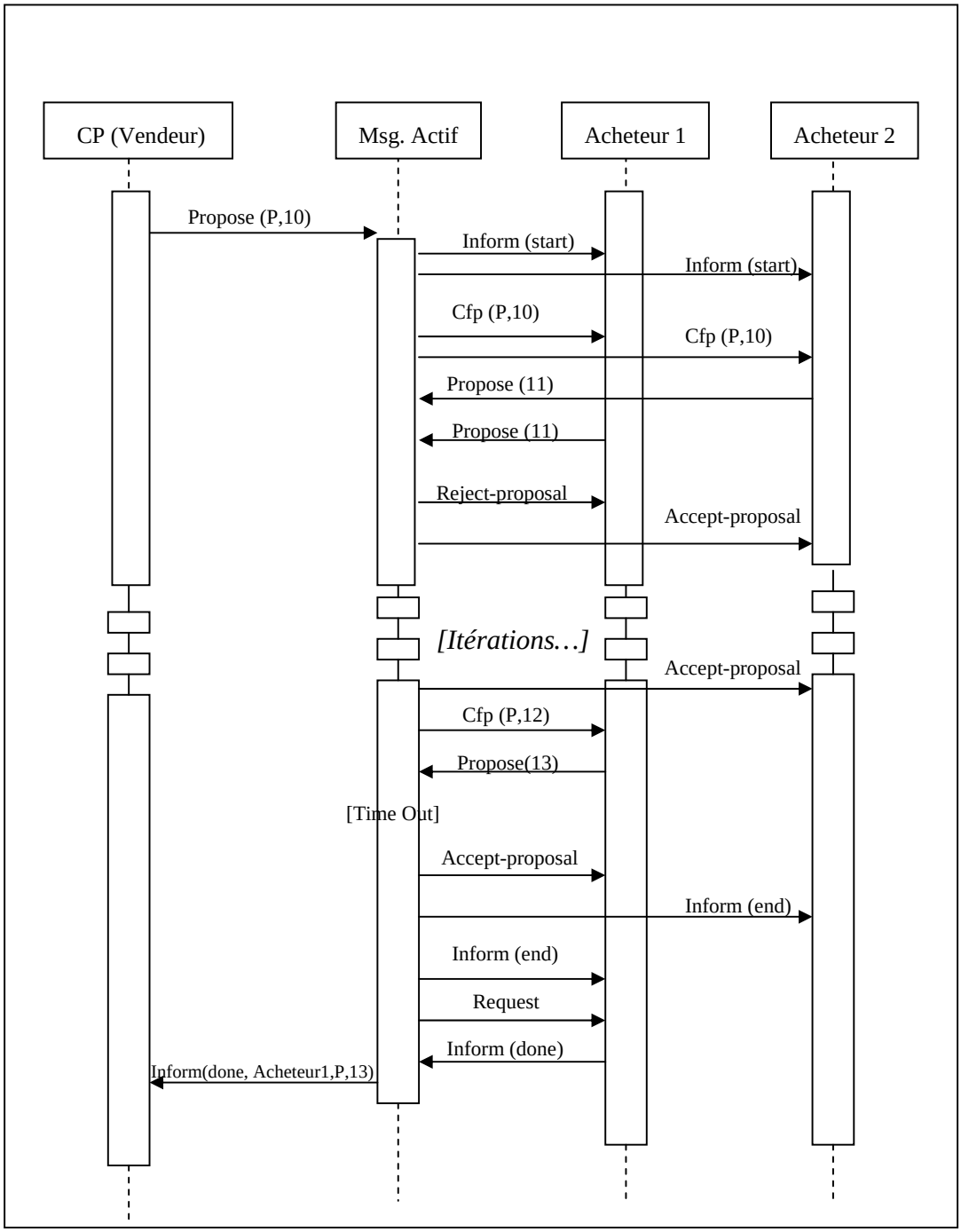


Fig.7.8. Déroulement d'enchère

3.2. Exemple d'exécution pour le jeu PPC:

Dans cet exemple, nous considérons un joueur humain et un joueur logiciel. Une fois la partie lancée par le joueur humain, l'arbitre reçoit le choix effectué par les joueurs au bout de chaque délai T, il annonce le score après que les joueurs ont joué 10 coups (choix). Pour plus de clarté, les messages actifs correspondant aux agents ne sont pas représentés sur la figure 7.9. La partie va se dérouler en dix itérations (on a représenté que deux).

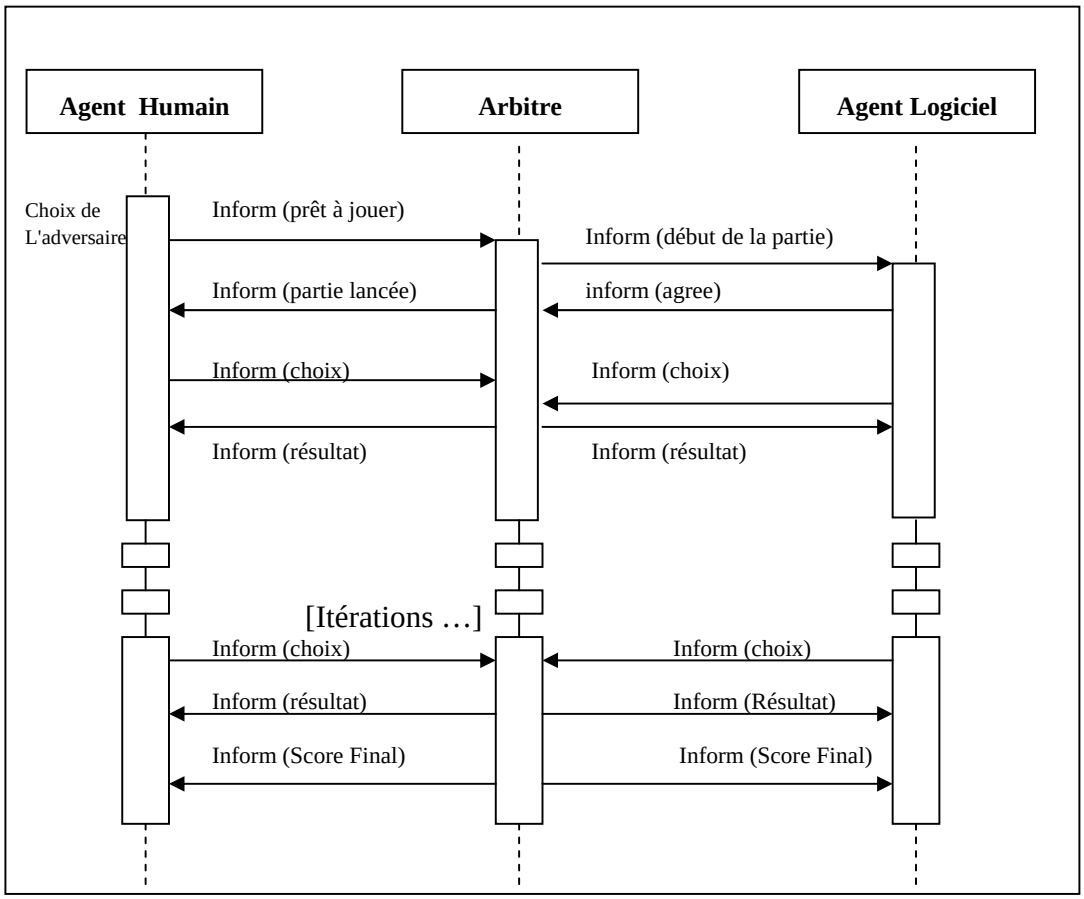


Fig.7.9. Déroulement du jeu PPC

3.3. Conclusion:

Les applications développées ont utilisé les services qu'offrent la plateforme, tels que le suivi et la gestion des interactions, le service des pages jeunes (DF) et pages blanches (AMS), le transport des messages intra et inter conteneurs etc… . En se servant de ces services les agents utilisés dans ces applications sont devenus très simples.

La mise en œuvre de ces applications sur la plateforme nous a permis d'exploiter les ressources que la plateforme met à notre disposition, et de remarquer le potentiel du Mécanisme d'Interaction basé sur le MID.

Les avantages qu'offrent la plateforme, basée sur le modèle d'interaction dynamique, pour le développement d'applications sont les suivants:

- la simplification des agents, en gardant la richesse des interactions
- la structuration des interactions des différents agents
- la possibilité d'interagir d'une façon uniforme et indépendante de la nature de l'acteur considéré (soit un utilisateur, soit un agent artificiel)
- les agents n'ont pas besoin de connaître la structure du système (puisque c'est le mécanisme d'interaction qui s'en occupe)

Par ailleurs, l'utilisation de la plateforme permet aux développeurs de concevoir et de maintenir facilement leurs applications.

Chapitre 8

Conclusion générale

Les systèmes multi-agents (SMA) proposent une approche intéressante pour le développement de systèmes à plusieurs composantes autonomes pouvant coopérer. Dans ce domaine, de nombreux modèles d'agents, d'environnement, d'interactions et d'organisations sont élaborés par les chercheurs pour la construction d'un système multi-agents. Le travail présenté dans ce mémoire est concentré sur le concept d'interaction. L'objectif de cette étude est double, d'une part, recenser les principaux travaux sur les interactions et en proposer une synthèse. D'autre part, mettre en oeuvre le modèle d'interaction dynamique (MID) en développant une plateforme multi-agents fondée sur ce modèle puis tester ce modèle en réalisant des applications sur cette plateforme.

Pour cela, nous avons organisé ce mémoire en deux parties. La première partie est consacrée à une étude bibliographique établie sur l'univers multi-agents. Nous avons introduit dans le deuxième chapitre les différents concepts de base du paradigme multi-agents. Ces concepts sont l'agent, l'interaction et l'organisation. L'agent, quel que soit son type, est l'entité de base de toute approche multi-agents. L'interaction est le concept permettant de relier les agents d'un système et l'organisation est nécessaire pour structurer les interactions qui interviennent entre les différentes entités du système.

Une synthèse des travaux effectués sur les interactions est présentée dans le troisième chapitre. La première question, traitée dans ce chapitre, porte sur les principes et concepts de base permettant de mettre en place une interaction entre agents, à savoir la communication et les langages de communication entre agents. La deuxième question porte sur les protocoles d'interaction qui permettent de contrôler et de structurer les échanges d'informations entre agents. La troisième question porte sur les outils utilisés pour la représentation des protocoles, où le langage AUML a été choisi pour être employer dans la description des interactions. La quatrième question traitée dans ce chapitre, concerne l'influence du modèle d'interaction sur l'architecture

du système multi-agent. La dernière question traitée est l'aspect dynamique de l'interaction, le traitement du message et son envoi sont normalement fait entièrement par l'agent, cependant il existe des travaux qui proposent une évolution du modèle d'interaction, comme par exemple le Modèle d'Interaction Dynamique.

Le quatrième chapitre de ce mémoire a été consacré aux plateformes multi-agents. Nous avons présenté la norme FIPA des systèmes multi-agents, et dans la suite, nous avons présenté les plateformes les plus marquantes et celles dont nous nous sommes inspirés pour développer notre plateforme.

Dans la deuxième étape de ce travail, nous avons conçu et réalisé une plateforme multi-agents basée sur le modèle d'interaction dynamique selon les spécifications FIPA. Cette plateforme permet de développer des applications multi-agents résolvant des problèmes variés.

Dans le chapitre 5, nous avons proposé une architecture qui repose sur l'architecture abstraite de FIPA et qui offre à travers le Mécanisme d'Interaction la prise en charge de la gestion des interactions. Cette plateforme offre les caractéristiques suivantes:

- La plate-forme multi-agents compatible FIPA, qui inclut le Système de Gestion d'Agents (AMS), le Facilitateur d'Annuaire (DF), et le Système de transport des messages.
- La plateforme d'agents distribuée. La plateforme d'agents peut être distribuée sur plusieurs hôtes.
- Le Mécanisme d'interaction permettant de prendre en charge la gestion des interactions menées par les agents. A chaque fois qu'un agent initie ou participe à une interaction, Le mécanisme crée un message actif qui sera chargé de gérer cette interaction. Ce message actif est détruit à la fin de cette interaction.
- Le transport léger de messages ACL sur la même plateforme d'agents. Dans le but de simplifier la transmission, les messages internes (sur la même plate-forme) sont transférés codés comme des objets Java et non comme des chaînes de caractères. Quand l'expéditeur ou le récepteur n'appartient pas à la même plateforme, le message est automatiquement converti au format de chaîne de caractères spécifiés par la FIPA. De cette façon, la conversion est cachée au programmeur d'agents, qui a seulement besoin de traiter la classe d'objets Java.
- Une bibliothèque de protocoles d'interaction compatibles FIPA.
- L'enregistrement automatique d'agents dans le Système de Gestion d'Agents (AMS).
- Un service d'attribution de noms compatible FIPA ; quand la plateforme est lancée, un agent obtient un identificateur unique (Globally Unique IDentifier - GUID).
- Une interface graphique utilisateur pour gérer plusieurs agents.

A la suite de ce travail, la plateforme a été réalisée en utilisant le langage Java. Nous avons ensuite testé la plateforme en développant plusieurs applications

benchmarks de type Toy-problems, présentées dans le septième chapitre. Les résultats d'exécutions de ces applications, nous ont montré le bon déroulement des interactions ainsi que le fonctionnement correct du Mécanisme d'Interaction. Par ailleurs, le développement de ces applications sur cette plateforme, nous a permis de conclure que c'est grâce au MID que les agents développés pour ces applications sont devenus très simples car la gestion de l'interaction est déléguée au mécanisme d'interaction. Et que ces agents n'ont pas besoin de connaître la structure du système (adresses, les autres agents,…) car c'est le mécanisme d'interaction qui s'en occupe.

Enfin, cette étude m'a permis d'approfondir mes connaissances sur les SMA, la modélisation logicielle et les technologies réseau. C'était à la fois un approfondissement technique et une introduction à la recherche.

De nombreuses extensions sont possibles au travail que nous venons de présenter. Nous nous contenterons ici des principales.

Tout d'abord, nous envisageons d'intégrer les restrictions qui ont été posées lors de l'implémentation de cette plateforme, à savoir la traduction des langages de communication entre agents, la prise en charge d'autres langages ACL tel que le KQML, et langages de contenu tel que KIF, ainsi que la migration des agents dans les conteneurs d'agents. Il est également souhaitable, de permettre la communication de cette plateforme avec d'autres plateformes FIPA.

Dans le déroulement de notre travail nous avons été confrontés à une question qui portait sur une extension du modèle d'interaction dynamique. Nous proposons l'utilisation du code mobile pour l'implémentation des messages actifs car Il nous semble que les messages actifs peuvent, dans certains cas, avoir une meilleure performance s'ils se trouvent proches des agents avec lesquels ils doivent interagir.

Un autre aspect qui peut être exploré est que la délégation de la gestion de l'interaction au mécanisme d'interaction n'est effectuée que par demande fournie par l'agent. Dans cette approche, les agents vont gérer localement leurs interactions, et s'ils décident de déléguer la gestion de leurs interactions au MI, ils le feront par demande "Request" au près du Mécanisme d'interaction, qui à son tours va créer un Message actif qui sera chargé de cette interaction.

Bibliographie

[Akni98] S. AKNINE, S. PINSON, M. ZACKLAD, Un Système d'Agents Récursifs pour l'Aide au Travail Coopératif. JFIADMAS 98.

[Akni00] S.Aknine, Modèles et méthodes de coordination dans les systèmes multi-agents. Thèse de doctorat université Paris IX Dauphine. Décembre 2000.

[Amig03] M. Amiguet. MOCA : un modèle componentiel dynamique pour les systèmes multi-agents organisationnels. Thèse de doctorat université Neuchâtel. 2003.

[Asa02] http://www-poleia.lip6.fr/~guessoum/asa/benchmarks.html

[Auml00] AUML. The Agent Unified Modeling language, http://www.auml.org/

[Ayac85] AYACHE, J. M. , COURTIAT, J-P. , DIAZ, M. , JUANOLE, G. Utilisation des réseaux de Petri pour la modélisation et la validation de protocoles. In: Technique et Science

[Baei96] C. BAEIJS; Y. DEMAZEAU; L. ALVARES. SIGMA: Application of Multi-Agent Systems to Cartographic Generalisation; In : Proceedings of the seventh European Workshop on modeling Autonomous Agents in Multi-Agent World, MAAMAW '96, LNAI 1038, Eindhoven, The Netherlands, 1996, Springer.1996.

[Baei96a] C. BAEIJS, Y. DEMAZEAU. Les Organisations dans les Systèmes Multi-Agents Toulouse 4èmes Journées du GDR-PRC IA, Février 1996.

[Baei96b] C. BAEIJS A Survey on Organisational Aspects in Multi-Agent Systems

[Baei98] BAEIJS, C. Fonctionalité Emergente dans une Société d'Agens Autonomes - Etude desAspects Organisationnels dans les Systèmes Multi-Agents Réactifs. Thèse de Doctorat INPG.Grenoble, Novembre, 1998.;

[Baro94] BARON, M. *EIAO, quelques repères.* Technologie de l'information, culture et société,n°65, Editions Harmattan, (1994). Article en ligne, http://terminal.enscachan.fr/Terminal/65multimediabaron.html.;

[Beal94] BEALE, R. & WOOD, A. Agent-Based Interaction. In: *People and Computers IX:Proceedings of HCI'94*, Glasgow, UK, August 1994, pp. 239-245.

[Bela94] O. BELAKHDAR et J. AYEL. *Modelisation et Expression des protocoles d'Interactiondans les Systèmes Multi-Agents*. Seconde Rencontre Nationale de Jeunes Chercheurs e Intelligence Artificielle, Marseille, 1994.

[Belli02] Bellifemine, F., Caire, G., Trucco, T., Rimassa, G., « Jade Programmer's Guide », JADE 2.5,2002.

[Belli03] BELLIFEMINE F., CAIRE G., POGGI A., RIMASSA G., « JADE, A white paper », September 2003,
http ://exp.telecomitalialab.com/upload/articoli/V03N03Art01.pdf

[Belli04] BELLEFEMINE F., CAIRE G., TRUCCO T., RIMASSA G., « JADE programmer's guide (JADE3.2) », rapport, 2004, TILab S.p.A.

[Bois90] BOISSIER, O. & DEMAZEAU, Y. ASIC: An Architecture for Social and Individual Control and Its Application to Computer Vision. In : Proceedings of the second European Workshop on modeling Autonomous Agents in Multi-Agent World, MAAMAW '90, Odense, Denmark, Aug 3-5, 1990.

[Bouz97] K. BOUZOUBA, B.MOULIN. Rapport sur le langage d'Interaction IL. Mémo de Recherche.1997

[Bouz97a] K. BOUZOUBA, B.MOULIN. La négociation des relations sociales dans les conversations multi-agents. In: 5ème Journées Francophones d'Intelligence Artificielle Distribuée et Systèmes Multi-Agents (JFIADSMA), Nice, France, Avril, 1997. pp. 47-62.

[Bouz97b] K. BOUZOUBA, B.MOULIN. L'implicite dans les communications multi-agents. In: 5ème Journées Francophones d'Intelligence Artificielle Distribuée et Systèmes Multi-Agents (JFIADSMA), Nice, France, Avril, 1997.pp. 63-75.

[Bouz97c] K. BOUZOUBA. Implicite, relations sociales et dialogisation : vers l'amélioration des modes de communication d'agent – Application au projet POSTAGE.

[Bouz98] K. BOUZOUBA, B.MOULIN. KQML+ : pour que les agents s'échangent des actes de discours de la conversation humaine. Rapport de Recherche DIUL-RR-9801. Département d'Informatique. Faculté des Sciences et de Génie. Université Laval. Janvier 1998

[Bras94a] BRASSAC C.; L'interaction inter-agents : non littéralité et processualité; In: 2èmes Journées Francophones d'Intelligence Artificielle Distribuée et Systèmes Multi-Agents (JFIADSMA), Voiron, Mai 1994.

[Bras94b] BRASSAC C.; Modéliser l'enchainement conversationnel; Rochebrune, Autonomie et interactions fonctionnelles; Janvier 1994.;

[Bras94c] BRASSAC C., CHEVRIER V., Vers un réexamen du statut de l'interaction dans les systèmes multi-agents; *Pragmatiques*; éditions de l'Harmattan; 1994

[Bras95] BRASSAC C. Pesty, S. Coopération dand les Systèmes Multi-Agents: Comportement ou Conduite? International Workshop on Decentralized Intelligent and Multi-Agent Systems (DIMAS'95), Krakow, Pologne, Novembre 1995.

[Bras96] BRASSAC C., DE ALMEIDA J., GREGORI N., SAINT-DIZIER V.; La logique interlocutoire peut-elle être un outil pour le paradigme de la distribution en intelligence artificielle ? In: 4èmes Journées Francophones d'Intelligence Artificielle Distribuée et Systèmes Multi-Agents (JFIADSMA), Port Camargue, Avril, 1996

[Bras97] BRASSAC, C. L'intercompréhension, modélisations et validation.1997

[Chai97] CHAIB-DRAA, B. Causual Reasoning in Multi-Agent Systems. To appear in In : Proceedings of the eigth European Workshop on modeling Autonomous Agents in Multi-Agent World, MAAMAW '97. 1997.

[Chai98] B. CHAIB-DRAA. Industrial Applications of Distributed AI. In: *Readings in Agents.* Edited by HUHNS, M. N. ; SINGH, M. P. . Morgan Kaufmann, San Francisco, 1998. pp. 31-35.

[Chai01] B. Chaib-draa, I. Jarras et B. Moulin. Systèmes multiagents : Principes généraux et applications. Article. 2001.

[Chev95] V. Chevrier., C. Brassac. Non littéralité et diffusion de l'information dans les systèmes multi-agents. Actes de la conférence Européenne sur les sciences cognitives.Saint-Malo. Avril 1995.

[Chic98] G. CHICOISNE, P-M. RICORDEL, S. PESTY, Y. DEMAZEAU. Outils et pistes pour la pratique du dialogisme entre agents. In: 6èmes Journées Francophones d'Intelligence Artificielle Distribuée et Systèmes Multi-Agents (JFIADSMA), Abbaye des Prémontrés, Pont-à-Mousson, France, 1988. pp.

[Chic98a] G. CHICOISNE. Conversation et relations sociales pour des agents moins artificiels. Rapport de DEA Sciences Cognitives, Institut National Polytecnique de Grenoble, 1998.

[Chic99] G. CHICOISNE; S. PESTY, *Modèle de conversation et agents rationnels socialement corrects*, Atelier Thématique TALN 1999 "La langue dans l'interaction homme-machine", pp91- 104, 1999.

[Chic02] G. Chicoisne. Dialogue entre agents naturels et agents artificiels. Application aux cominautés virtuelles. Thèse de doctorat INP Grenoble. Décembre 2000.

[Coen94] COEN, M. H. SodaBot: A Software Agent Environment and Construction System. Artificial Inteligence Technical Report 493. MIT, June 1994.

[Cohe97] COHEN, P. R. ; LEVESQUE, H. J. Communicative Actions for Artificial Agents. In: Software Agents, edited by BRADSHAW, J. AAAI Press/The MIT Press, Menlo Park, 1997. pp. 419-436.

[Deck87] K. Decker. Distributed problem solving techniques: A servey. IEEE Transactions on systems, Man, and cybernitics,SMC-17(5):485-519. septembre 1987.

[Dema93] DEMAZEAU, Y. *La plate-forme PACO et ses applications.* 2ème Journée Nationale du PRC-IA sur les Systèmes Multi-Agents, PRC-IA, Montpellier, Décembre 1993.

[Dema94] Y. DEMAZEAU, O. BOISSIER, J. L. KONING. Using Interaction Protocols to Control Vision Systems. IEEE In: *International Conference on Systems, Man and Cybernetics,* San Antonio, 1994.

[Dema94a] Y. DEMAZEAU, O. BOISSIER, J. L. KONING. Interaction Protocols in Distributed Artificial Intelligence and Their use to Control Robot Vision Systems. In: *International Conference on AI and Information Control of Robots,* Smolenice, SL, Sept. 1994.

[Dema95] Y.DEMAZEAU. From Interactions to Collective Behaviour in Agent-Based Systems. In: *Proceedings of the 1st European Conference on Cognitive Science.* Saint-Malo, France, Avril 1995.

[Dema96] Y. DEMAZEAU, A. C. R. COSTA. Populations and Organizations in Open Multi- Agent Systems. In: *Proceedings of the First National Conference on Parallel and Distributed Artificial Intelligence.* Hyderaband, India, july 1996.

[Dema97] Y. DEMAZEAU et all. *Retraite scientifique de l'Equipe MAGMA.* (transparences). Chartreuse de Valbonne. Septembre 8-10, 1997.

[Dema97a] DEMAZEAU, Y. *Steps towards Multi-Agent Oriented Programming*. (slides), I International Workshop on Multi-Agent Systems, IWMAS '97, Boston. 1997.

[Dema98] Y. DEMAZEAU, J. FERBER. *Introduction to Multi-Agent Systems.* Tutorial Notes, Paris, July 3, 1998.

[Dema01] DEMAZEAU Y., VOYELLES, Grenoble, France, Habilitation à Diriger les Recherches de l'INP Grenoble, Laboratoire LEIBNIZ. Avril 2001

[Drog93] DROGUL, A. *De la simulation multi-agent à la résolution collective de problèmes. Une étude de l'émergence de structures d'organisation dans les systèmes multi-agents.* Thèse de Doctorat de l'Université Paris 6. Novembre 1993.

[Ferb95] J. FERBER. Les Systèmes Multi-Agents - Vers une Intelligence Collective. Inter Editions, Paris, 1995.

[Ferb97] J. FERBER, O. GUTKNECHT. A meta-model for the analysis and design of organizations in multi-agent systems. Rapport de Recherche LIRMM 97189, Décembre 1997.

[Ferb98] J. FERBER, O. GUTKNECHT. A meta-model for the analysis and design of organizations in multi-agent systems. 3rd International Conference on Multi-Agent Systems -ICMAS´98, IEEE Computer Society Press, Paris, July, 1998. pp. 128-135.

[Fini92] T.FININ, R.FRITZON, D. McKAY. A Language and Protocol to Support Intelligent Agent Interoperability; CE & CALS Washington '92 Conference.

[Fini93] FININ T. et all Specification of the KQMLAgent-Communication Language. Draft. June 1993.

[Fini94] T.FININ, R.FRITZON, D. McKAY, R. McENTIRE. KQML as an Agent-Communication Language. Proceedings of the 3rd International Conference on Information and Knowledge Management; ACM Press, November 1994.

[Foise98] R.Foisel, Modèle de réorganisation de systèmes multi-agents: une approche dscreptive et opérationnelle. Thèse de doctorat université Henri Poincaré- Nancy I. Novembre 1998.

[Gues03] Z.Guessoum, Modèles et architectures d'agents et systèmes multi-agents adaptatifs, HDR – Université Pierre et Marie-Currie, 2003.

[Gutk97a] O. GUTKNECHT, J. FERBER. MadKit: Organizing heterogeneity with groups in a platform for multiple multi-agent systems. Rapport de Recherche LIRMM 97188, Décembre 1997.

[Gutk98] O. GUTKNECHT. J. FERBER, Un méta-modèle organisationnel pour l'analyse, la conception et l'exécution de systèmes multi-agents. In: 6èmes Journées Francophones d'Intelligence Artificielle Distribuée et Systèmes Multi-Agents (JFIADSMA), Abbaye des Prémontrés, Pont-à- Mousson, France, 1988. pp. 267-280

[Gutk00] GUTKNECHT O., FERBER J., MICHEL F., « MadKit : Une expérience d'architecture de plate-forme multi-agent générique », PESTY S., SAYETTAT C., Eds., *JFIADSMA' 00*, St Jean-la-Vêtre, Loire, France, Octobre 2000, Hermès, p. 223-236.

[Gutk00a] O.GUTKNECHT. Proposition d'un modèle organisationnel générique de systèmes multi-agents Examen de ses conséquences formelles, implémentatoires et méthodologiques. Thèse de doctorat université Montpellier II. Septembre 2001.

[Hann99] HANNOUN, M. ; BOISSIER, O. ; SICHMAN, J. S. ; SAYETTET, C. *MOISE : Un modèle organisationnel pour la conception de systèmes multi-agents*. In: Acted des 7èmes Journées Francophones d'Intelligence Artificielle Distribuée et Systèmes Multi-Agents (JFIADSMA), Saint-Gilles, Ile de La Réunion, 1999.

[Hann00] HANNOUN M., BOISSIER O., SICHMAN J. S., SAYETTAT C., __Moise : Un modèle organisationnel pour la conception de systèmes multi-agents, *Actes des JFIADSMA*, 2000.

[Huge98] M.P. HUGET. Applicabilité des systèmes multi-agents hiérarchiques au domaine de la généralisation cartographique. Rapport de DEA, septembre 1998.

[Huge99] M.-P. HUGET, S. PINSON. *Une Typologie des Modèles d'Agents.* Document 110 Cahier du LAMSADE, Université de Paris Dauphine, janvier 1999.

[Huge01] M. Huget, Une ingénierie des protocoles d'interaction pour les systèmes multi-agents. Thèse de doctorat Université Paris IX Dauphine, Juin 2003.

[Jade03] Plateforme JADE: Java Agent Development Framework, 2003. http://jade.cselt.it/

[Jenn94] N. R. JENNINGS, T. WITTIG. ARCHON: Theory and Practice, Distributed Artificial Intelligence: Theory and Praxis (eds. N. M. Avouris and L. Gasser), Kluwer Academic Press, 1992, 179-195.

[Jenn94a] N. R. JENNINGS. The ARCHON System and its Applications, Second International Working Conference on Cooperating Knowledge Based Systems (CKBS-94) (Invited Paper), Keele, UK, 1994, 13-29.

[Jenn98] N. R. JENNINGS, M. J.WOOLDRIDGE. Applications of Intelligent Agents. In: N.JENNINGS. and M. J. WOOLDRIDGE (Eds). Agent Technology: Foundations, Applications, and Markets. pp. 3-28. Springer-Verlag, Berlin, 1998.

[Jouv03] D. Jouvin. Délégation de rôles et arfchitectures dynamiques de systèmes multi-agents conversationnels. Thèse de doctorat, université de Lyon. Décembre 2003.

[Jouv03a] Jouvin D., Hassas S., « Architectures dynamiques de systèmes multi-agents conversationnels », Journée francophone des systèmes multi-agents, 2003.

[Kolp02] Manuel Kolp, Thi Thuy Hang Hoang,. Développement de patrons de conception à ancrage social pour architectures multi-agent, rapport de recherche Université Catholique de Louvain. 2002

[Koni95] J-L. KONING, Y. DEMAZEAU, B. ESFANDIARI, J. QUINQUETON. Quelques perspectives d'utilisation des Langage et Protocoles d'Interaction dans le contexte des Télécomunications. In: 3èmes Journées Francophones d'Intelligence Artificielle Distribuée et Systèmes Multi-Agents (JFIADSMA), Chambéry – St Baldoph, 1995.

[Madk03] Plateforme MADKIT, 2003 http://www.madkit.org

[Mazo01] H. Mazouzi, Ingénierie des protocoles d'interaction: des systèmes distribués aux systèmes multi-agents. Thèse de doctorat université Paris Dauphine. Octobre 01.

[Moul96] MOULIN, B. ; CHAID-DRAA, B. *An Overview of Distributed Artificial Intelligence*. In: O'HARE, G. ; JENNINGS, N. (eds.) *Foundations of Distributed Artificial Intelligence*. Wiley- Interscience Publications. pp. 3-55. 1996.

[Labi93] S. Labidi, W. Lejouad : "De l'intelligence artificielle distribué aux systèmes multi-agents".Rapport de recherche INRIA France 1993.

[Odel00] J. Odel. H.V.D Parunak. Et B. Bauer. Representing agent interaction protocols in UML. First international workshop on agent-oriented software Engineering 2000.Berlin forthcoming ISCE 2000.

[OMG 01] OMG, CORBA 2.4.2 Interceptors chapter, février 2001, OMG Document formal/01-02-57, http://www.omg.org

[Oude97] OUDEYER, P.Y. Vers une sémantique des protocoles d'interaction dans les systèmes multi-agents : le formalisme POS. Rapport de stage de Magistère Informatique et Modélisation, Ecole Normale Supérieure de Lyon, juillet 1998.

[Pest97] S. PESTY, C. BRASSAC, P. FERRENT. Ancrer les agents cognitifs dans l'environnement. In: 5ème Journées Francophones d'Intelligence Artificielle Distribuée et Systèmes Multi-Agents (JFIADSMA), Nice, France, Avril, 1997.pp.

[Ribe98] A. M. RIBEIRO ; Y. DEMAZEAU. A Dynamic Interaction Model for Multi-Agent Systems. In : F. GARIJO, C. LEMAITRE (eds) Proceedings of the II Iberoamerican Workshop on Distributed Artificial Intelligence and Multi-Agent Systems. pp. 27-36. Toledo, 1-2 oct, 1998.

[Ribe99] A. M. RIBEIRO ; Y. DEMAZEAU. *Passive and Active Interaction Mechanisms.* Twentyfourth International Conference ICT&P'99 - INFORMATION & INTERACTION. Plovdiv, Bulgaria, June 8-12, 1999.

[Ribe01] Un Modèle d'Interaction Dynamique pour les systèmes Multi-Agents, Thèse de Doctorat université Joseph Fourier- Grenoble 1, 2001.

[Rico98] P. M. RICORDEL. *Extension du modèle d'interaction dialogique à n agents.* Rapport de magistère 3ème année, Université Joseph Fourrier - Intitut National Polytechnique de Grenoble, été 1998.

[Rico98a] P.M. RICORDEL. Influences mutuelles "Organisation / Interaction" – Une étude du dialogisme. Rapport de DEA Informatique, Université de Savoie. 1998.

[Rico99] P.M. RICORDEL ; S. PESTY ; Y. DEMAZEAU, *About Conversations between Multiple Agents*, First International Conference of Central Eastern Europe on Multi-Agent Systems (CEEMAS), St Petersbourg, Juin 1999.

[Rico99a] P. M. RICORDEL. *Plates-formes Multi-Agents.* Présentation MAGMA 23 février 1999.

[Rico01] P.M. RICORDEL. Programmation Orientée Multi-Agents: Développement et Déploiement de Systèmes Multi-Agents Voyelles. Thèse de doctorat INP Grenoble. Octobre 2001.

[Rico02] RICORDEL P.-M., DEMAZEAU Y., « La plate-forme VOLCANO : modularité et réutilisabilité pour les systèmes multi-agents », Numéro spécial sur les plates-formes de développement SMA. Revue Technique et Science Informatiques (TSI). 2002.

[Sava01] M. Savall. M. Itmi - J.P. Pécuchet. Présentation de la plateforme Phoenix construite sur le modèle d'agents YAMAM. Rapport de recherche Laboratoire PSI - INSA de Rouen. 2001.

[Sava03] M. Savall. Une architecture d'agents pour la simulation- le modèle YAMAM et sa plateforme Phoenix. Thèse de doctorat de l'INSA Rouen. 2003.

[Sear69] SEARLE, J. Speech Acts. Cambridge University Press, 1969.

[Secq03] Y. Secq. Une méthodologie pour les systèmes multi-agents ouverts. Thèse de doctorat de l'université des sciences et technologies de Lille. Décembre 2003.

[Shoh93] SHOHAM, Y. Agent-oriented programming. Artificial Intelligence, V 60, N 1, pp 51-92. 1993.

[Sian91] SIAN, S. S. Adaptation based on cooperative learning in multi-agent systems. In: Decentralized A.I. - 2. Eds: DEMAZEAU, Y. ; MÜLLER, J-P. . Elsevier Science Publisher.1991.

[Smit80] SMITH, R. G. The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. IEEE Transactions on Computers, V C-29 N 12. December 1980.

[Kolp02] Manuel Kolp, Thi Thuy Hang Hoang,. Développement de patrons de conception à ancrage social pour architectures multi-agent, rapport de recherche Université Catholique de louvain. 2002

[Verc04] L. Vercoutier. MAST: un modèle de composant pour la conception des SMA. JFSMA 2004.

[Wood94a] WOOD, A. Agent-Based Interaction. Six month Report, University of Birmingham, May 1994

[Wood94b] WOOD, A. Towards a Medium for Agent-Based Interaction. Thesis Proposal, University of Birmingham, October 1994.

[Wool95] Wooldridge, M. et N. R. Jennings. Agent theories, architectures, and languages. Dans Wooldridge, Jennings (ed), Intelligent Agents, Springer Verlag, 1995. p.1-22.

[Wool00] M. Wooldrige. Reasoning about Rational Agents, The MIT Press, 2000.

[Wool02] M. Wooldridge. An Introduction to Multiagent Systems. John Wiley and Sons, 2002.

Lien vers des pages sur le World Wide Web

[Bedou] BEDOU, I. ARCHON. http://www.lirmm.fr/~bedou/archon.html .

[CSLI] CSLI Agent-Oriented Programming - Applying Software Agents to Software Communication, Integration, and HCI. http://www-csli.stanford.edu/csli/.

[FIPA96a] FIPA. Agent Communication Language. http://www.fipa.org

[FIPA96b] FIPA. Agent Management. http://www.fipa.org

[FIPA96c] FIPA. Agent/Software Interaction. http://www.fipa.org

[FIPA96d] FIPA. Design Test: Personal Travel Agent. http://www.fipa.org

[FIPA97a] FIPA. FIPA 97 Specification - Part 1 - Agent Management (23 octobre 1998). http://www.fipa.org

[FIPA97b] FIPA. FIPA 97 Specification - Part 2 - Agent Communication Language (23 octobre 1998).http://www.fipa.org

[FIPA97c] FIPA. FIPA 97 Specification - Part 3 - Agent/Software Interaction. http://www.fipa.org

[FIPA97d] FIPA. FIPA 97 Specification - Part 4 - Application Design Test: Personal Travel Agent. http://www.fipa.org

[FIPA97e] FIPA. FIPA 97 Specification - Part 5 - Application Design Test: Audio/Video Entertainment and Broadcasting. http://www.fipa.org

[FIPA97f] FIPA. FIPA 97 Specification - Part 6 - Application Design Test: Network Management & Provisioning. http://www.fipa.org

[FIPA97g] FIPA. FIPA 97 Specification - Part 7 - Application Design Test: Personal Assistant. http://www.fipa.org

[FIPA98a] FIPA. FIPA 98 Specificaton - Part 1 - Agent Management (23 octobre 1998). http://www.fipa.org

- AUML. The Agent Unified Modelling language, http://www.auml.org/
- FIPA: Foundation for Intelligent Physical Agents. Specifications. 1997. http://www.fipa.org
- Plateforme JADE: Java Agent Development Framework, 2003. http://jade.cselt.it/
- Java Expert System Shell (JESS) http://herzberg.ca.sandia.gov/jess/
- Plateforme ZEUS http://www.labs.bt.com/projects/agents/zeus
- Plateforme MADKIT, 2003 http://www.madkit.org
- Plateforme SWARMS http://www.swarm.org/index.html
- Une présentation des plateformes et environnements pour la construction des systèmes multi-agents http://epfl.ch/~iagents/Slides/lecture2_v1.1.ppt
- Liste plateformes multi-agents http://www.agentlink.org/resources/agent-software.php
- Liste plateformes multi-agents commerciales http://www.agentbuilder.com/AgentTools/commercial.php
- Liste plateformes multi-agents logiciel libre http://www.agentbuilder.com/AgentTools/academic.php

Annexe A

Le Langage de contenu des messages FIPA SL0.

Le langage Fipa SL0 [FIPA97] est un sous ensemble minimal du langage Fipa SL utilisé comme langage de contenu dans les messages FIPA-ACL. Il permet la représentation des actions, la détermination des résultats, l'accomplissement d'une action et des propositions logiques simples. Il est représenté par la grammaire suivante:

```
Content              = "(" ContentExpression + ")".
ContentExpression    = ActionExpression / Proposition.
Proposition          =  Wff.
Wff                  = AtomicFormula /  "(" ActionOp  ActionExpression  ")".
AtomicFormula        = PropositionSymbol / "(" "result"          Term  Term ")"
                       / "(" PrédicateSymbol  Term +  ")" /  "True"  / "False" .
ActionOp             = "done".
Term                 = Constant / Set / Sequence / FunctionalTerm / ActionExpression.
ActionExpression     = "(" "action" Agent  Term ")".
FunctionalTerm       = "(" FunctionalSymbol Term* ")"
                       / "(" FunctionalSymbol Parameter* ")".
Parameter            = ParameterName  ParameterValue.
ParameterValue       = Term.
Agent                = Term.
FunctionalSymbol     = String.
PropositionSymbol    = String.
PrédicateSymbol      = String.
Constant             = NumericalConstant / String / DateTime.
Set                  = "(" "set" Term* ")".
Sequence             = "(" sequence" Term* ")".
NumericalConstant    = Integer / Float.
```

Les définitions lexicales de SL0 sont données comme suit:

```
String                  = Word  | ByteLengthEncodedString | StringLiteral.
ByteLengthEncodedString = "#" DecimalLiteral+ "\"" <byte sequence>.
Word                    = [~ "\0x00" - "\0x20", "(", ")", "#", "0" - "9", ":", "-", "?"]
                          [~ "\0x00" - "\0x20", "(", ")"]*.
ParameterName           = ":" String.
Sign                    = [ "+" , "-" ].
Integer                 = Sign? DecimalLiteral+ | Sign? "0" ["x", "X"] HexLiteral+.
Dot                     = "."
Float                   = Sign? FloatMantissa FloatExponent?
                          | Sign? DecimalLiteral+ FloatExponent.
FloatMantissa           = DecimalLiteral+ Dot DecimalLiteral*
                          | DecimalLiteral* Dot DecimalLiteral+.
FloatExponent           = Exponent Sign? DecimalLiteral+.
Exponent                = ["e","E"].
DecimalLiteral          = ["0" - "9"].
HexLiteral              = ["0" - "9", "A" - "F", "a" - "f"].
StringLiteral           = "\""( [~ "\""]   | "\\\"" )*"\"".
```

DateTime = Sign? Year Month Day "T" Hour Minute Second MilliSecond TypeDesignator?.
Year = DecimalLiteral DecimalLiteral DecimalLiteral DecimalLiteral.
Month = DecimalLiteral DecimalLiteral.
Day = DecimalLiteral DecimalLiteral.
Hour = DecimalLiteral DecimalLiteral.
Minute = DecimalLiteral DecimalLiteral.
Second = DecimalLiteral DecimalLiteral.
MilliSecond = DecimalLiteral DecimalLiteral DecimalLiteral.
TypeDesignator = ["a" - "z" , "A" – "Z"].

Sémantique de FIPA-SLO:

ContentExpression: le contenu d'une expression est utilisée comme contenu d'un message ACL. Il peut être:

1) Une proposition logique à laquelle on peut assigner une valeur logique pour un contexte donné. Précisément, c'est une Wff "Well formed formula" utilisant les règles décrites dans la règle de production Wff. Une proposition est utilisée dans un acte communicatif "Inform".
2) Une action qui peut être réalisée. Cette action peut être simple ou composée construite en utilisant les opérateurs de séquence et d'alternative. Une action est utilisée comme contenu d'expression quand l'acte communicatif est "Request" ou un de ses dérivés.

Well formed formula (formules bien formées): Une wff est construite à partir de formules atomiques ce qui signifie qu'elle est déterminée par la sémantique du domaine de représentation ou "done Action expression" qui signifie que "done Action expression" est vraie.

Atomic formula (formule atomique): Représente une expression qui a une valeur logique dans le domaine. Trois formes sont définies:

i) une "propositional symbol" donnée peut être défini dans le domaine du langage qui peut être vraie ou fausse.
ii) certains prédicats sont définis sur aucun ou plus arguments, chacun d'eux est un terme
iii) deux termes sont reliés par l'opérateur binaire "Result"

La représentation Fipa-SL0 ne définie pas la signification des symboles dans les formules atomiques et ceci est du sort de la représentation du domaine du langage et de l'ontologie.

Result predicate: Un besoin commun est de déterminer le résultat de l'accomplissement d'une action ou l'évaluation d'un terme. Pour faciliter cette opération, un prédicat binaire standard "Result" est introduit dans le langage.

Termes: Les termes sont eux même atomiques ou recursivement construit comme un terme fonctionnel "functionnal term" dans le quel un "functor" est appliqué à zéro ou plusieurs arguments.

Functionnal Term : Un functionnal term référencie un objet via une relation fonctionnelle (notée par "functionnal symbol") avec d'autres objets (qui sont les paramètres ou termes) au

lieu d'utiliser le nom direct de l'objet, par exemple (Père de Ali) au lieu de Mohamed; sachant que Mohamed est le père de Ali.

Deux formes syntaxiques peuvent être utilisées pour exprimer un terme fonctionnel:

a. Le terme fonctionnel est suivi d'une liste de termes qui ne sont rien d'autre que les arguments. La sémantique des arguments est déterminée par leurs positions.

Exemple: Divide 10 2 : 10 est dividende et 2 est diviseur

b. dans la seconde forme, chaque argument est précédé de son nom.

Exemple : Divide: dividend 10: divisor 2

La première forme est utilisée dans les termes fonctionnels dont les quels l'ontologie ne spécifie pas les noms des paramètres, tandis que la seconde est utilisée dans le cas où l'ontologie doit spécifier les noms de paramètres sans leurs positions. La seconde est très appropriée pour représenter les descriptions où le symbole fonctionnel est interprété comme un constructeur d'objet dont les paramètres sont les attributs de cet objet.

Exemple:
(vehicule : color red : maxspeed 200:
owner (person : name luis : nationality French))

Action expression: Action expression est un sous-ensemble spécial de termes. Une action est introduite par le mot réservé "action" et identifie l'agent réalisant cette action et un terme dénotant l'action qui sera réalisée.

Annexe B

Exemple d'échanges possibles de messages entre agents lors d'une vente aux enchères.

Dans cet exemple d'application de vente aux enchères, nous considérons un système multi-agents pour les enchères, composé d'un agent commissaire-priseur (CP) et de quatre (04) agents acheteurs/participants (A1, A2, A3 et A4). Dans le déroulement de cet exemple, les participants achètent un seul objet à la fois (ça n'empêche pas l'agent CP d'avoir une liste d'objets à vendre et chaque agent acheteur sa liste d'objets qui peuvent l'intéresser).

Pour simplifier, on suppose que tous les produits sont de la même ontologie ONT.

Initialisation de l'agent CP:

Produit	Quantité	Prix minimum
P1	500	100
P2	250	50
P3	50	200
P4	900	80
Date d'enchère : 16.04.2005 à 10 heure Délai d'attente : 15 milisecondes		

Les agents acheteurs sont initialisés comme suit:

Agent A1 Delta d'enchère = 2		
Produit	*Quantité*	*Prix max*
P2	250	55
P4	650	75

Agent A2 Delta d'enchère = 1%		
Produit	*Quantité*	*Prix max*
P1	200	95
P2	200	60
P4	700	75

Agent A3 Delta d'enchère = (prix_max – offre_courante)/5		
Produit	*Quantité*	*Prix max*
P1	400	105

Agent A4 Delta d'enchère = 2%		
Produit	*Quantité*	*Prix max*
P1	500	120
P2	200	52
P4	900	77

Dès que la date de l'enchère est arrivée, l'agent CP démarre l'enchère par la vente du premier produit P1 et fait une requête (à travers son message actif) auprès de l'agent DF pour avoir la liste des agents intéressés par ce produit. Ces agents sont A2, A3 et A4, puis il informe ces agents du début de l'enchère, et de la mise à prix du produit P1 qui est 100. Le déroulement de l'enchère du produit P1 est présenté par la séquence des messages donnée ci-dessous.

. . .

(From: CP To: Type: inform Action: Auction Args: (P1))

(From: CP To: DF Type: Request Date: 00 Conversation-id: C1 Action: Search-participant Args: (P1))

(From: DF To: CP Type: Agree Date: 02 Conversation-id: C1 Action: Search-participant Args: (P1))

(From: DF To: CP Type: inform Date: 03 Conversation-id: C1 Action: Result Args: (A2 A3 A4))

(From: CP To: A2 Type: inform Date: 06 Conversation-id: C2 Action: Auction-start)

(From: CP To: A3 Type: inform Date: 06 Conversation-id: C2 Action: Auction-start)

(From: CP To: A4 Type: inform Date: 06 Conversation-id: C2 Action: Auction-start)

(From: CP To: A2 Type: CFP Date: 10 Conversation-id: C2 Action: propose Args: (P1 100))

(From: CP To: A3 Type: CFP Date: 10 Conversation-id: C2 Action: propose Args: (P1 100))

(From: CP To: A4 Type: CFP Date: 10 Conversation-id: C2 Action: propose Args: (P1 100))

(From: A3 To: CP Type: inform Date: 13 Conversation-id: C2 Action: offer Args: (P1 101)

(From: A4 To: CP Type: inform Date: 15 Conversation-id: C2 Action: Offer Args: (P1 102))

[time-out]

(From: CP To: A4 Type: inform Date: 26 Conversation-id: C2 Action: Accept-offer Args: (P1 102))

(From: CP To: A3 Type: inform Date: 27 Conversation-id: C2 Action: Reject-offer Args: (P1 101))

(From: CP To: A3 Type: CFP Date: 28 Conversation-id: C2 Action: propose Args: (P1 102))

(From: A3 To: CP Type: inform Date: 30 Conversation-id: C2 Action: offer Args: (P1 102.6))

(From: CP To: A3 Type: inform Date: 32 Conversation-id: C2 Action: Accept-offer Args: (P1 102.6))

(From: CP To: A4 Type: CFP Date: 34 Conversation-id: C2 Action: propose Args: (P1 102.6))

(From: A4 To: CP Type: inform Date: 37 Conversation-id: C2 Action: Offer Args: (P1 104.6))

(From: CP To: A4 Type: inform Date: 40 Conversation-id: C2 Action: Accept-offer Args: (P1 104.6))

(From: CP To: A3 Type: CFP Date: 41 Conversation-id: C2 Action: propose Args: (P1 104.6))

(From: A3 To: CP Type: inform Date: 45 Conversation-id: C2 Action: offer Args: (P1 104.68))

(From: CP To: A3 Type: inform Date: 47 Conversation-id: C2 Action: Accept-offer Args: (P1 104.68))

(From: CP To: A4 Type: CFP Date: 49 Conversation-id: C2 Action: propose Args: (P1 104.68))

(From: A4 To: CP Type: inform Date: 52 Conversation-id: C2 Action: Offer Args: (P1 106.77))

(From: CP To: A4 Type: inform Date: 54 Conversation-id: C2 Action: Accept-offer Args: (P1 106.77))

(From: CP To: A3 Type: CFP Date: 55 Conversation-id: C2 Action: propose Args: (P1 106.77))

[time-out]

(From: CP To: A4 Type: inform Date: 71 Conversation-id: C2 Action: Successful Args: (P1 106.77))

(From: CP To: A2 Type: inform Date: 73 Conversation-id: C2 Action: End-auction Args: (P1 106.77))

(From: CP To: A3 Type: inform Date: 73 Conversation-id: C2 Action: End-auction Args: (P1 106.77))

(From: CP To: A4 Type: inform Date: 73 Conversation-id: C2 Action: End-auction Args: (P1 106.77))

(From: Mactif To: CP Type: inform Date: 75 Conversation-id: C2 Action: Auction Args: (P1 A4 106.77))

***Remarque**:* une autre approche consiste à déclencher la vente aux enchères de tous les produits, proposés par l'agent CP, en parallèle. Cette approche permet de créer plusieurs messages actifs chargés du déroulement de la vente aux enchères des différents produits. Ainsi un acheteur peut s'intéresser à plusieurs produits proposés par les messages actifs engendrés par l'agent CP.