N° d'ordre : 07/2004-M/IN

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université des Sciences et de la Technologie *Houari Boumédiene*
USTHB – ALGER

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

M^r^ MOHAMMEDI Samir

Sujet

Un modèle organisationnel de systèmes multi-agents pour l'Acquisition Coopérative de l'Information

Soutenu le 13-10-2004, devant le jury composé de :

Mr	M. AHMED-NACER	Professeur	USTHB	Président
Mme	H. DRIAS	Professeur	USTHB	Directeur de thèse
Mr	C. HANACHI	Maître de Conférence	Toulouse 1	Co-Directeur de thèse
Mme	F. YOUCEF ETTOUMI	Maître de Conférence	USTHB	Examinateur

Remerciements

Je souhaite en premier lieu exprimer ma profonde gratitude à M^{me} Drias Habiba qui ma accordé sa confiance. Quand il y a quelques années je fouillais dans les articles de recherche, j'étais loin de penser que j'aurais le plaisir de travailler avec l'auteur.

Je veux ensuite adresser mes plus vifs remerciements à Mr Hanachi Chiheb, qui ma prodigué conseils avisés et constants encouragements.

Je remercie Mr Ahmed-Nacer Mohamed, qui me fait l'honneur de présider ce jury.

Je tiens aussi à exprimer ma reconnaissance envers Mme Youcef Ettoumi Fatiha, qui me fait l'honneur d'être rapporteur.

Plusieurs personnes ont bien voulu consacrer un peu de leur temps à corriger les fautes de français du manuscrit de ce mémoire, je les remercie, ainsi tous mes amis, et les membres de ma famille, qui ont contribués de prés ou de loin pour que ce mémoire puisse voir le jour.

À mes parents.

Résumé

Ce mémoire présente tout d'abord un état de l'art sur les agents et les systèmes multi agents (SMA) en considérant deux niveaux: le niveau microscopique relatif aux agents et le niveau macroscopique relatif aux SMA vus comme une société d'entités coopérantes dans une organisation. Ensuite nous abordons l'acquisition coopérative de l'information (ACI), qui se présente comme une extension intelligente de la recherche d'information classique, notamment lorsque les sources d'informations sont distribuées sur un ou plusieurs serveurs, et dans un réseau à grande échelle comme l'Internet, où le terme *hétérogénéité* (conceptuelle, sémantique et celle des langages) intervient à plusieurs niveaux.

L'objectif de notre travail est de concevoir une architecture basée sur une organisation multi-agents avec des agents mobiles pour l'ACI, qui assure le fonctionnement du système, et elle doit prendre en compte les différentes hétérogénéités rencontrés lors de ce processus, car une organisation est composée d'entités coopérantes et autonomes, une multitude de fonctions (rôles) qui sont assurées. Nous associons à chaque agent un rôle spécifique, des buts, des règles d'interactions et de lui partager des sources d'informations. Ce modèle organisationnel interagit avec un modèle informationnel qui englobe une ou plusieurs ontologies définissant l'univers du discours entre agents.

Un prototype d'exécution a été développé avec la plate forme MADKIT pour valider une partie de notre proposition.

Mots clés :

Système Multi Agents, Agent mobile, Organisation multi agents, modèle AALAADIN, modèle GAIA, Acquisition Coopérative de l'Information, Ontologie, MADKIT.

Table des matières

Table des matières

Introduction générale

Introduction générale

1. Acquisition Coopérative de l'Information

L'évolution mondiale dans tous les domaines a contribué à une forte explosion en matière d'informations. Ces informations monodisciplinaire ou souvent pluridisciplinaires sont aujourd'hui sollicitées par toutes les organisations, quelles soit commerciales, économiques, militaires, éducatives (universités, bibliothèques), ou sociales. Ces organisations doivent mettre en commun toutes leurs compétences technologiques et de pouvoir partager, échanger, et collaborer les informations pour répondre aux demandes du marché et d'acquérir le meilleur profit. [Tou03].

Ceci nous ramène à penser à l'avènement Internet, qui nous offre un nouveau challenge : il y a beaucoup plus d'informations disponibles, géographiquement distribuées, et il n'y a pas d'entité centrale en charge de la gestion des informations. Dans ces conditions les questions posées ne peuvent qu'être réduites à des recherches de « mots clés » classiques. Par expérience, ce type de recherche est limité, et il convient donc de trouver une « extension » plus « intelligente » à ces types de recherche d'informations.
C'est dans ce contexte que s'est développé le thème d'**A**cquisition **C**oopérative de l'**I**nformation (**ACI**) comme l'extension désirée car :

- La question posée par l'utilisateur devient un problème décomposable en sous problèmes interdépendants ;
- La méthode de résolution de la requête sera vue comme une résolution distribuée de problème, où un ensemble de sources d'informations se fédèrent pour répondre à une question posée [Oat97];
- Les sources d'informations sont distribuées sur un ou plusieurs serveurs, et elles ne sont pas fixées ou connues par celui qui les interroge ;
- Elles sont autonomes : car elles sont conçues les une indépendamment des autres ;
- Elles sont dynamique : car elles peuvent être créées, modifiées, ou supprimées à tout moment ;
- Elles sont hétérogènes : deux types d'hétérogénéité se présentent :
  - Hétérogénéité conceptuelle : leur langage et leur système de représentation et de manipulation peuvent être différents, ainsi l'accès à ces ressources peut se faire selon des protocoles différents,
  - Hétérogénéité sémantique : où la même information est interprétée différemment et selon chaque source d'information qui la considère.

Cette « extension » plus « intelligente » est venue principalement des recherches menées dans le cadre de l'Intelligence Artificielle Distribuée, où la technologie des agents a fourni les outils conceptuels pour traiter le problème de la coopération dans l'ACI. L'ACI est devenue une branche à part des systèmes multi-agents [Oat97] [Syc98] [Klu99].

Au delà de ces avantages, l'ACI reste un domaine de recherche qui soulève beaucoup de problèmes qui lui sont spécifiques. En effet, chaque point mentionné auparavant est un problème de recherche où sa résolution reste ouverte.

2. Solutions et Approches adoptées

L'objectif de notre travail est de concevoir une architecture capable de répondre aux problèmes posés lors d'un processus d'ACI, ainsi :

- Nous définissons une approche basée sur une organisation multi-agents, qui assure le fonctionnement du système, car une organisation est composée d'entités coopérantes et autonomes, en plus, et un ensemble de fonctions (rôles) qui sont assurées. Nous associons à chaque agent un rôle spécifique, des buts, des règles d'interactions et de lui partager des sources d'informations, ceci nous résout beaucoup de problème, notamment :
  L'hétérogénéité des langages : vu que la multiplicité des langages et protocoles d'interaction rendent la fédération d'agents non basés sur le même formalisme difficile [GuFe99] ;
  L'hétérogénéité des modèles d'agents : Généralement les systèmes multi-agents sont conçus avec un seul modèle, car l'intégration dans un même système des modèles conçus selon des architectures diverses est le plus souvent délicate ;
  La nature de sources d'informations exploitées : les sources d'informations sont distribuées, autonomes, dynamiques, et elles ne sont pas connues à priori ;
- La dite organisation comporte un modèle informationnel, ce dernier englobe une ou plusieurs ontologies servant à définir l'univers du discours entre agents, ce modèle nous résout le problème d'hétérogénéité sémantique, et il fournit un outil pour décomposer une requête en sous problèmes interdépendants ;
- L'intégration des agents spéciaux « des traducteurs », qui englobent et communiquent avec les sources d'informations, pour résoudre l'hétérogénéité conceptuelle.

3. Organisation de mémoire

Ce mémoire est constitué de cinq chapitres :

Le premier est un état de l'art sur les agents, les systèmes multi agents et la société d'agent. Nous voulons détailler ce chapitre qui nous livre l'entité intelligente qui a renforcé les recherches dans ce domaine.

Le chapitre II décrit une organisation comme un cadre d'interaction en présentant deux modèles. Le premier modèle est AALAADIN, qui est un modèle basé sur trois concepts centraux celui d'agent, de groupe et de rôle. Le deuxième modèle est le modèle GAIA organisé autour d'une notion de rôle plus spécifique, et on terminera ce chapitre par une petite discussion comparative sur l'aspect architectural.

Le chapitre III présente une vue d'ensemble de la technologie d'agent mobile en tant que paradigme de construction d'applications réparties. Il situe ce modèle d'exécution distribuée par rapport au modèle Client/Serveur, nous présentons leurs avantages qu'on essayera de les utiliser. Enfin il décrit l'infrastructure système nécessaire au support d'agents mobiles, en comparant quelques plateformes d'agents mobiles sur cette infrastructure.

Le chapitre IV décrit l'architecture multi agents proposée pour l'ACI, qui prend en compte un modèle organisationnel décrit en chapitre II, et fait intervenir des agents mobiles du chapitre III pour le processus de collecte d'information. Cette organisation comprend un modèle informationnel qui décrit l'ontologie du domaine et un modèle de rôles, qu'on détaillera.

Enfin, le dernier chapitre présente une implémentation de l'architecture proposée pour une étude de cas de collecte des rapports de recherche. Cette application est réalisée à l'aide de l'environnement de développement multi agents MADKIT.

Chapitre I

Les Systèmes multi agents

Le défi de l'intelligence artificielle pour la prochaine décennie est de construire des systèmes capables d'interagir de façon productive avec : les autre systèmes, les hommes et le monde physique.

Daniel Bobrow 1990.

Chapitre I

Les Systèmes multi agents

I.1 Introduction

La communauté de l'Intelligence Artificielle Distribuée (IAD) n'a pas à nos jours donnée une définition commune d'un système multi-agents (SMA), c'est peut être parce qu'il est difficile de définir la notion d'agent.
Poser la question, qu'est ce qu'un agent pour ceux qui travaillent en IAD est aussi difficile que poser la question qu'est ce que l'intelligence pour la communauté de l'IA [Hew91]. En effet, la notion d'agent diffère selon l'axe de recherche envisagé. Ferber [Fer95] a préféré employer le terme kénétique qui regroupe l'ensemble des théories et réalisations dans ce domaine.

Au delà des multiples définitions, on distingue dans la littérature deux définitions d'agents, la première est dite "faible" et évidente, et la deuxième est dite "forte".

I.2 Agents et systèmes multi-agents

I.2.1 Définition faible de la notion d'agent

Un agent est une entité matérielle ou logicielle qui a les propriétés suivantes [WoJe94] :

- *Autonomie* : les agents sont doués d'autonomie, cela signifie qu'ils ne sont pas dirigés par des commandes venant de l'utilisateur ou d'un autre agent [Fer95].
- *Aptitude sociale* : les agents interagissent avec d'autres agents via un langage de communication pour agent [Cha94].
- *Perception* : les agents ont les capacités de percevoir leur environnement (mais d'une manière limitée).
- *Engagement* : les agents ne contentent pas seulement de répondre à leur environnement, ils peuvent, suivant leur but, prendre des initiatives [Del00].

I.2.2 Définition forte de la notion d'agent

Les prétendants de cette notion (particulièrement ceux qui travaillaient en IA), veulent implanter, en plus des propriétés citées ci-dessus, les concepts inhérents à l'homme. Donc l'agent est caractérisé avec des notions désignant des états mentaux : la connaissance, la croyance et l'intention [Sho93], l'émergence [Fer95].

L'agent est donc une entité physique ou logicielle :

- qui est capable d'agir dans un environnement (Activité) ;
- qui peut communiquer directement avec d'autres agents ;
- qui est mue par un ensemble d'objectifs propres (sous forme d'objectifs individuels ou d'une fonction de satisfaction) ;

- qui possède des ressources ;
- qui est capable de percevoir son environnement ;
- qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune) ;
- qui possède des compétences (services) qu'elle peut offrir aux autres agents ;
- qui a un comportement tendant à satisfaire ses objectifs, en tenant compte d'une part des ressources et des compétences dont elle dispose, et d'autre part de ses perceptions, de ses présentations et des communications qu'elle reçoit.

I.2.3 Autres attributs d'agents

On trouve aussi, en plus des attributs définit en haut, d'autres attributs parmi lesquels : [Gal88]

- *sincérité* : qui indique qu'un agent ne doit pas communiquer de fausses informations ;
- *la rationalité* :qui indique le fait qu'un agent agisse pour atteindre ses buts ;
- *la mobilité* : qui est la capacité qu'a un agent à se mouvoir dans un réseau.
- *La réactivité* : l'agent peut répondre aux changements de l'environnement qu'il perçoit.

I.2.4 Modèle d'agents

On trouve plusieurs types d'agents dans la littérature, et selon leur niveau d'intelligence et leur domaine d'applications, parmi ces agents, nous pouvons citer :

- Les agents communicants qui possèdent des capacités de communication complètes, mais leur capacité de raisonnement est liée au domaine d'activité. [Del00]
- Les agents de traitement dont le rôle sont d'effectuer des transformations qui sont des traitements ou des calculs sur des flots de données en entrée pour produire des flots de données en sortie. Il assure l'exécution d'un algorithme de complexité quelconque. [Att97]
- Les agents naturels qui sont des agents dotés de possibilités d'auto apprentissage (faculté d'augmentation de sa base de connaissance) par ajout de règles de fonction de l'évolution de son environnement et des événements qui peuvent se produire. [Att97]
- Les agents spécialistes ont une compétence précise et disposent d'une représentation partielle de leur environnement. Dans un système d'agents spécialistes, il existe des moyens d'affectation des tâches entre agents. [Vin93]

Mais, il y a deux conceptions qui ont donné lieu à deux écoles de pensée différente. Ces conceptions ont un grand effet dans la recherche sur les SMA's, ce sont les agents cognitifs et réactifs.

I.2.4.1 **Agents cognitifs** (école cognitive ou sociale)

Elle est la plus représenté en IAD, car elle trouve son origine dans la volonté de faire communiquer et coopérer des systèmes experts classiques. Dans ce cadre, un système multi-agents est composé d'un petit nombre d'agents ''intelligent'', chaque agent dispose d'une base de connaissance [Fer95].
Cette classe d'agent tend à simuler les aptitudes (comportement) humaines.

I.2.4.1.1 caractéristique d'un agent cognitif

On dit que les agents cognitifs sont :

- *Intentionnels* : c'est-à-dire qu'ils possèdent des buts et des plans explicites leurs permettant d'accomplir leurs buts. [Fer95].
- *Rationnels* : le mot provient de la raison, et signifie que si un agent sait qu'une de ces actions lui permet d'atteindre un de ses buts, il la sélectionne. [Kho94]
- *Adaptatifs* : c'est la flexibilité, c'est-à-dire qu'un agent est autonome, sociable (interaction avec d'autres agents), réactif (perception de son environnement et réaction), proactif (il peut prendre des initiatives de manière à stimuler l'environnement pour la réalisation de son but) [WoJe95].
- *Intelligents* : c'est l'ensemble de fonctions mentales ayant pour objet la connaissance conceptuelle et rationnelle. Un agent intelligent est un agent cognitif, rationnel, intentionnel et adaptatif.

I.2.4.1.2 architecture d'un agent cognitif

- Structure interne : La structure interne d'un agent cognitif (Fig.I.1.) comprend les éléments suivants : les croyances, les contrôles, l'expertise, les connaissances de communication et de coordination. [Bau92]

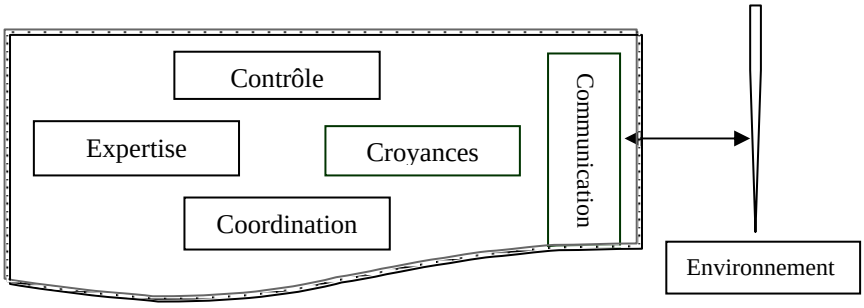


Fig. I.1. Structure interne d'un agent

- Les croyances : les croyances d'un agent sont des connaissances, pas nécessairement objectives, sur soi (méta connaissance) qui est caractérisé par son nom, son identificateur, son adresse, ses objectifs à attendre et son état interne, et sur les autres appelés ses accointances, qui est une représentation partielle (adresses des accointances et leurs buts).

- L'expertise : c'est les aptitudes de résolution de l'agent dans un domaine. Elle représente la connaissance sur la résolution d'un problème. [Fer89]

- La coopération : Elle concerne les connaissances sur les modes d'interaction entre agents. Elle est liée à la communication inter-agents.
- Le contrôle :l'agent doit déterminer l'action à entreprendre en fonction des informations, des connaissances, des intentions et des buts dont il dispose pour assurer une fonction tout en conservant son intégrité structurelle.
- La communication : Les communications sont la base des interactions entre les agents dans un système multi-agents. Car sans communication l'agent n'est qu'un individu isolé, sourd et muet.

La communication est l'un des aspects les plus important en IAD qui permet l'échange d'informations entre agents, la formulation de requête, …etc. Plus un système est important, plus il y a un besoin accru de coopération et de communication entre agents. [Vin93]

♦ Fonctionnement : Les agents cognitifs fonctionnent en respectant les trois étapes suivantes (Fig.I.2.) :

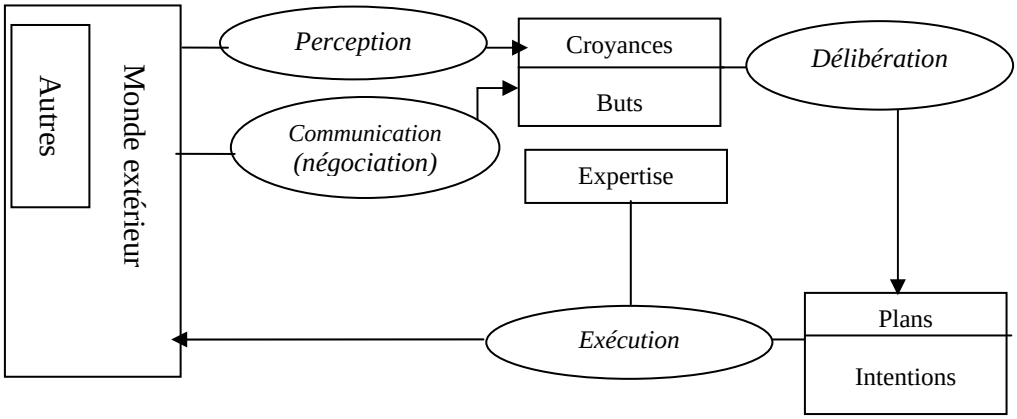


Fig.I.2. fonctionnement d'un agent cognitif.

- ***Perception*** : elle fournit les capacités d'entrée nécessaires à l'agent, et de pouvoir classer et distinguer les états du monde. Elle comprend le savoir initial de l'agent, la perception de soi et des autres et la communication avec les autres. C'est à dire que l'agent détecte la présence de toute entité initialement inconnue dans le système [Cha96] (par exemple d'autres agents, des obstacles …etc.), il reçoit aussi des messages des autres.
- ***Délibération*** : c'est la partie la plus essentielle d'un agent, c'est la que s'élaborent les objectifs, les prises de décision, la détermination de plan et la séquence d'actions qui permet de satisfaire les buts (planification). [Fer95]
- ***Exécution*** : Fournit à l'agent les capacités nécessaires de sortie (output), elle permet d'agir en effectuant certaines actions. [Del00]

I.2.4.2 Agents réactifs (école réactive ou biologique)

Cette tendance prétend qu'il n'est pas nécessairement que les agents soient intelligents individuellement pour que le système ait un comportement global intelligent, ainsi on fait coopérer un grand nombre d'agent de faible granularité pour aboutir aux objectifs fixés, ces agents ont un comportement élémentaire de type stimulus / réponse.
Les premiers travaux relatifs à cette approche ont été réalisés en 1986 par Brooks [Bro86], qui a fait coopérer plusieurs micro-robots de petite taille pour réaliser seulement une tâche donnée, et selon lui, elle est plus efficace que de travailler avec un seul gros robot.
La plate-forme MERING a été utilisée pour la conception de SMA réactif. [Fer91]

I.2.4.1.1 architecture d'un agent réactif

- *Structure Interne : Un* agent réactif peut être modélisé par un simple objet doté d'un comportement et d'un moyen de communication avec les autres agents. Il n'a pas de connaissances sur les autres, ni de capacités à raisonner sur les messages qu'il reçoit, ni même de développer des stratégies de contrôle. Ils sont donc orientés plutôt comportement que connaissance.

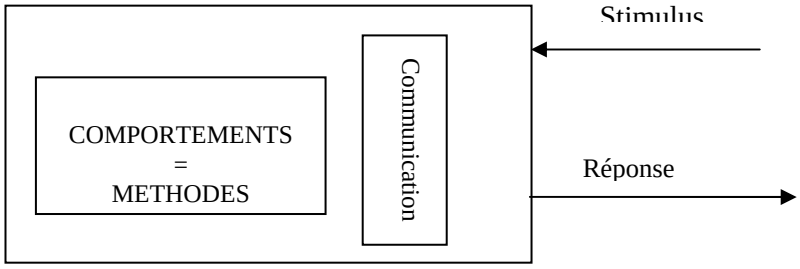


Fig.I.3. Architecture d'un agent réactif

- *Fonctionnement :* Ces agents aux comportements élémentaires sont susceptibles d'interagir qu'aux stimuli observés, leurs interactions peuvent émergées de l'intelligence.

I.2.5 Les Systèmes multi agents

I.2.5.1 Définitions

Selon Ferber et Ghallab [FeGh88], «un système multi-agents (SMA) est une communauté d'agents travaillant en commun selon des modes parfois complexes de coopération, de conflit et de concurrence pour aboutir à un objectif global : la résolution d'un problème, l'établissement d'un diagnostic …etc. ».
Le principe est que si un agent détient l'expertise qui lui permet de résoudre le problème en entier alors il le résout selon les données et les ressources dont il dispose, dans le cas contraire, il use de toute la puissance de ses moyens de communication pour arriver à une satisfaction globale. [Hat&91]

Selon Jennings, Wooldridge et Sycara [Jen&98], un SMA peut être défini comme un ensemble de « résolveurs » appelés agents qui travaillent ensemble pour résoudre des problèmes dont aucun n'a les capacités ou les connaissances individuelles pour les résoudre totalement.

Le principe adopté serait qu'aucun agent n'ait les informations ou les capacités suffisantes pour résoudre le problème : il a une vue limitée, il n'y a pas un système de contrôle global, les données sont décentralisées et la résolution est asynchrone.

I.2.5.2 Les modèles de SMA

Pour faire communiquer des entités informatiques, les systèmes d'IAD ont eu recours à différentes approches : les tableaux noirs, les acteurs et les systèmes physiquement distribués

I.2.5.2.1 Les architectures à base de tableau noir (BlackBoard)

L'architecture à base de tableau noir est l'une des plus utilisé dans les systèmes multi-agents cognitifs, originalement développée dans le cadre de l'IA pour la reconnaissance de la parole avec un système, baptisé HEARSAY. [Erm&80]

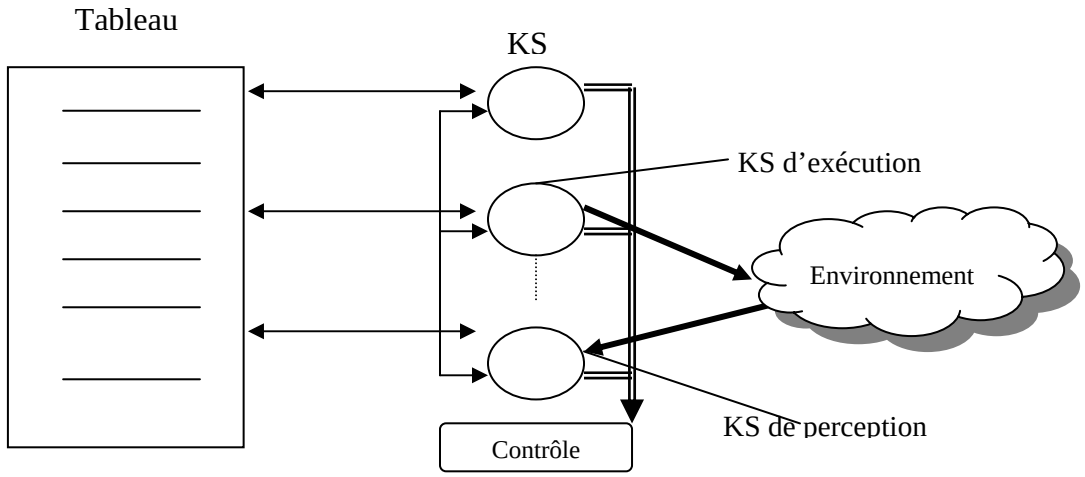


Fig.I.4. Une architecture de système à base de tableau noir.

Le modèle à tableau noir est fondé sur un découpage en modules indépendants qui ne communiquent directement aucune information, mais qui interagissent en partageant des informations, ces modules appelés sources de connaissance ou KS (pour Knewledge Sources) travaillent sur un espace qui comprend tous les éléments nécessaires à la résolution d'un problème. L'architecture de Fig.I.4 comprend trois sous systèmes [Fer95] :

- La source de connaissance KS ;
- La base partagée (le tableau) : qui comprend les états partiels d'un problème en cours de résolution, les hypothèses et les résultats intermédiaires. D'une manière générale toutes les informations que s'échanges les KS ;
- Un dispositif de contrôle qui gère les conflits d'accès entre les KS, ces dernières interviennent sans être déclenché par un système de contrôle centralisé.

Le problème de contrôle dans un tableau noir [Fer95] revient à essayer de savoir ce qu'il convient de faire ensuite, c'est à dire déterminer quelle KS doit être déclenchée. Au début, avec HearsayII, le contrôle était câblé au sein d'une procédure, puis il fut donnée sous forme d'un ensemble de règles, mais ce n'est qu'avec le système BB1 de Hayes-Roth que le

contrôle a acquit ses lettres de noblesse, en considérant que le contrôle est un problème important pour qu'il dispose de son propre tableau [Hay85].

Les avantages de ce type d'architecture sont indéniables : une remarquable souplesse pour décrire les modules et articuler leur fonctionnement, il est possible d'implémenter n'importe quelle structure d'agent en terme d'éléments de tableau et de KS. Son principal inconvénient est du fait que son contrôle très centralisé (le module de contrôle qui ordonne l'ordre dans lequel les KS seront activés) et de leurs manques de mémoire locale.

Le tableau noir se présente comme une sorte de méta-architecture c'est à dire une architecture pour implémenter des architectures.

I.2.5.2.2 Les systèmes à acteurs

Un acteur est une entité informatique qui se compose de deux parties : une structure qui compose l'adresse de tous les acteurs qu'il connaît (ces accointances) et qu'il peut les envoyés des messages, et une partie active, le script, qui décrit son comportement lors de la réception un message (Fig.I.5.)[Hew73][Agh86].
Le comportement de chaque acteur, qui s'exécute indépendamment, et en parallèle avec les autres, se résume à un ensemble d'action réduit : envoyer des messages, créer des acteurs et modifier son état, et c'est suffisant pour pouvoir exprimer n'importe quel calcule parallèle.

La communication entre acteurs s'effectue par des envois des massages asynchrones. [Jen&98]

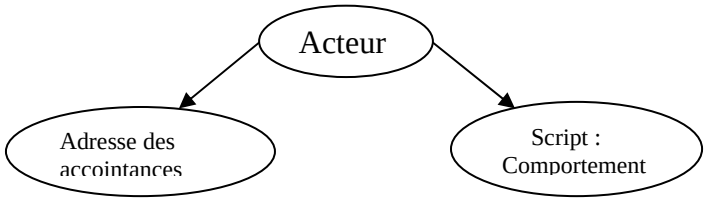


Fig.I.5. Modèle d'acteur.

I.2.5.2.3 Les systèmes physiquement distribués

Les systèmes proposés en planification multi-agents centralisée présentent des options intéressantes pour la résolution des conflits et des convergences vers une solution globale. Cependant, ils ont tous les mêmes inconvénients liés à la centralisation du contrôle au niveau d'un seul agent.
Les différents agents travaillant d'une façon concurrentielle dans le réseau doivent envoyer leurs actions au coordinateur, afin qu'il résolve les conflits éventuels et gère leurs interactions.

La charge de communication est très importante : pour cette raison les chercheurs ont pensé à des systèmes à planification distribuée. Ces systèmes sont caractérisés par un ensemble d'agents complexes, dotés de grandes capacités de raisonnement, et distribués sur des sites géographiquement répartis. Ceci nécessite la constitution de mécanismes de coordination très élaborés entre agents.

I.3 Sociétés d'Agents

Dans les applications multi-agents, les entités sont autonomes et chacune de ces entités ont des buts différents. Ceci peut conduire à une incohérence du système lors de la résolution du problème, les agents sont organisés en société régie par des comportements globaux structurés. Ils sont structurés grâce aux théories sociales pour la résolution d'un problème posé.

Dans cette partie, nous nous intéressons au comportement global d'une société d'agents et aux théories sociales requises pour la résolution de problèmes dans un univers multi-agents à savoir l'organisation, la coopération entre les agents, les moyens de communication, la résolution des éventuels conflits dus à la diversité de buts des agents faisant intervenir la coordination et la négociation et enfin l'émergence d'intelligence dans une interaction entre agents réactifs.

I.3.1 L'Organisation Sociale

L'organisation sociale d'un système multi-agents est la manière dont le groupe est constitue, a un instant donné, pour pouvoir fonctionner. Elle décrit l'ensemble des composants fonctionnels du système, leurs natures, leurs responsabilités et leurs besoins en ressources ainsi que les liens de communication entre les agents. Cette organisation peut être statique ou dynamique.

Selon Gasser [Gas90], une société d'agents est constituée de trois éléments : un ensemble d'agents, un ensemble de taches a réaliser et un ensemble d'objets associés à l'environnement. Un agent peut prendre la responsabilité d'effectuer une tâche s'il en a la capacité. Il prend alors un rôle dans le groupe. La réalisation d'une tâche suppose la manipulation d'objets de l'environnement.

On étudiera en détail cette théorie d'organisation en donnant quelque modèle de conception en chapitre 2.

I.3.2 La Coopération

De par le fait que les SMA sont conçus dans le but de résoudre un problème de manière distribuée, il apparaît que la coopération est l'une des caractéristiques essentielles dans un SMA. Aussi elle est la forme d'interaction la plus étudiée.

Dans un SMA, chaque agent renferme une expertise qui ne lui permet pas en général de résoudre seul le problème posé. Il ne peut que participer partiellement à la résolution, et donc soit aider un autre agent à résoudre un sous problème, soit de demander à un autre de l'aider à résoudre un sous problème. Ceci introduit la notion de coopération entre les agents et pose la question : qui fait quoi, quand, par quel moyens et sous quelles conditions ?

I.3.2.1 Définitions

La coopération est un accord entre agents permettant la synchronisation des activités des agents, le parallélisme des calculs et le partage des ressources. [Sca&96]

La coopération dans un SMA est développée par la volonté des agents de participer à l'accomplissement d'un objectif commun. Elle se rapporte à l'activité globale d'un groupe

d'agents car l'aboutissement de la résolution distribuée d'un problème est le résultat de l'interaction coopérative entre les différents agents. [Del00]

Des agents coopèrent ou sont en situation de coopération si les conditions suivantes sont vérifiées : [LaLe93]

- l'ajout d'un nouvel agent permet d'accroître les performances du groupe ;
- l'action des agents sert à éviter d'éventuels conflits.

Ainsi, un agent dispose de moyens qui lui permettent de s'informer sur l'état de l'environnement et des moyens de modifier l'état de cet environnement.

Les objectifs de la coopération entre agents est d'améliorer le mode d'évolution de la résolution (par les agents) en termes : [Bau92]

- de validité et de rationalité des informations échangées et des comportements ;
- d'efficacité des stratégies de résolution employées (alternance entre comportement opportuniste et comportement dirigé par les buts) ;
- de suppression des efforts inutiles de synchronisation ;
- de cohésion entre planification locale et globale.
- de rééquilibrage dynamique de la charge du travail (attribution de tâches selon disponibilité) ;

Un agent doit disposer – en plus de la connaissance reflétant son degré d'implication dans cette dynamique collective (croyances, buts, intentions, engagements, modèle de soi et d'autrui) - d'un certain nombre de compétences nécessaires pour la coopération. Il doit pouvoir [LaLe93]:

- mettre à jour le modèle du monde environnant,
- intégrer des informations venant d'autres agents,
- interrompre un plan pour aider d'autres agents,
- déléguer la tâche qu'il ne sait pas résoudre à un autre agent dont il connaît les compétences.

Ces caractéristiques forment les qualités essentielles d'un agent coopératif.

I.3.2.2 Les modèles de coopération

Selon qu'il existe une hiérarchie ou non entre les agents, nous distinguons les modes de coopération suivantes :

I.3.2.2.1 Coopération en hiérarchie

I.3.2.2.1.1 le mode commande

L'agent superviseur décompose le problème en sous problèmes qu'il envoie aux agents exécutants. Ces derniers, après avoir résolu leur problème, renvoient le résultat au superviseur qui choisira la «meilleure » réponse.

I.3.2.2.1.2 Le mode compétition

L'agent superviseur décompose le problème en sous problèmes et envoie la liste des sous problèmes à tous les agents exécutants. Ceux-ci consultent la liste et en résolvent chacun,

un ou plusieurs. A la fin, ils renvoient leurs solutions respectives au superviseur qui décident des solutions adoptées.

I.3.2.2.1.3 le mode appel d'offres

La répartition de sous problèmes se fait comme dans les autres modes. Chaque agent envoie pour sa part un appel d'offre après avoir consulté la liste. L'agent superviseur, à son tour, envoie les sous problèmes aux agents qu'il aura choisi. C'est seulement, en ce moment que les agents exécutants résolvent leurs sous problèmes et envoient leurs résultats au superviseur.

I.3.2.2.2 coopération sans hiérarchie

I.3.2.2.2.1 Coopération par partage de tâches

Le contrôle est dirigé, ici, par les buts. Le problème est réparti parmi tous les agents qui travaillent parallèlement, disposant des compétences et des ressources nécessaires. Chaque agent représente une tâche qu'il s'est engagé à exécuter. Donc à chaque engagement, les buts de l'agent changent et puisque les agents sont autonomes, on dit que le contrôle est dirigé par les buts.

I.3.2.2.2.2 Coopération par partage de résultats

Dans ce mode de coopération, le contrôle est dirigé par les données. Aucun agent ne peut résoudre une tâche individuellement. Il nécessite donc l'expertise des autres, donc ils s'échangent des solutions partielles à la demande. Chaque agent est représenté par une source de connaissance.

I.3.3 Le Contrôle

Le mécanisme de contrôle dans un SMA doit permettre de résoudre des problèmes de plusieurs types [Vin93] :

- après avoir résolu un problème posé, le résultat peut influencer la décision de l'autre ;
- les buts d'un agent et ceux du système sont antagonistes ;
- plusieurs agents ont besoin d'une ressource qui n'est pas partagée ou qui est limitée ;
- on peut arriver à une situation de conflit puisque dans certains systèmes les solutions des agents sont exclusives ;

Dans les systèmes où le contrôle est centralisé, des problèmes importants qui pourraient se poser sont la résistance aux pannes de tout le système. Si l'agent contrôleur s'effondre, tout le système s'écroule.

Dans les systèmes où le contrôle est réparti entre plusieurs agents des problèmes de coordination des actions peuvent surgir.

Un mécanisme de contrôle efficace doit pouvoir choisir l'agent le plus apte à une étape donnée de la résolution. Par conséquent, il est rapide (en terme d'action), économique (gestion des ressources et nombre de communications effectuées) et il doit satisfaire le plus de contraintes dans un problème où la solution est sur contrainte.

Selon Vincent [Vin93], le contrôle ou encore conduite du raisonnement représente les activités d'organisation du système et de répartition des rôles et tâches parmi les agents afin que le système aboutisse à une solution. Il concerne la planification et le suivi des tâches lors de la résolution.

Le contrôle peut être conduit de différentes manières. Il peut être :

- par un seul (moniteur) qui gère l'ensemble des agents. Il doit être capable de choisir à tout instant lors de la résolution l'agent qui sera chargé de faire évoluer vers la solution ;
- par un groupe d'agents appelés «agents de contrôle » dont le rôle de chacun consiste en la gestion d'un sous-groupe d'agents ;
- par tous les agents parmi lesquels les connaissances de contrôle sont distribués et par conséquent la négociation est accentuée pour la résolution .

Suivant le nombre d'agents ayant une activité de contrôle sur le domaine, le contrôle sera centralisé ou décentralisé

I.3.3.1 contrôle centralisé

Un seul agent se charge du contrôle donc décide des actions à entreprendre, gère les conflits et les ressources. Tous les autres agents sont sous sa commande et n'entament une tâche que s'il le leur demande. L'autonomie de l'agent «résolveur » ne se fait valoir qu'après l'allocation de la tâche. En outre, l'agent « moniteur » se réserve le droit d'interrompre la tâche d'un agent (en ne prenant plus en considération la solution rendue par celui-ci).

Ce processus simple permet de conduire et de prédire les interactions des différentes entités du système autour du «moniteur ». Le contrôle est alors dit centralisé et implique un système très efficace s'il n'est pas surchargé. Il est aussi assez facile à mettre en œuvre.

Ce type de contrôle pose des problèmes dans la communication et le traitement, car cet agent constituera un goulot d'étranglement. Sa fiabilité est aussi compromise car s'il est en panne, il entraîne tout le système. On trouve ce modèle dans les systèmes à tableau noir.

I.3.3.2 distribution du contrôle

La question «quel agent fait quoi et quand ?» peut être considérée comme étant la principale question en IAD. Ainsi, la distribution du contrôle se fait selon trois caractéristiques [Vin93] : le degré de coopération entre les agents, le type d'organisation adapté et la manière de la mise en œuvre de cette coopération.

Premièrement, la distribution des activités du contrôle peut entraîner que les agents aient des buts locaux antagonistes entre eux ou avec le système. Un agent est coopératif s'il est capable de changer ses objectifs locaux pour satisfaire ceux d'un autre agent ou ceux du système. Mais cette distribution pourrait ne pas être bénéfique puisque la coopération peut être très coûteuse en terme de communication.

Deuxièmement, le type d'organisation utilisé est étroitement lié au contrôle. Les relations et la répartition des rôles par le contrôle définissent les types d'organisation. Dans une organisation hiérarchisée, les agents renfermant plus d'information guident (contrôlent) ceux qui en ont moins. Les relations d'autorité du supérieur au subordonné surgissent entre agents [Vin93]. Elles seraient d'autant plus importantes si l'autorité revient aux agents qui ont une vue globale du système. Un problème appelé principe de rationalité limité ou d'éventail de contrôle se passe alors. Cela veut dire qu'un agent ne peut avoir d'autorité que sur un nombre

limité d'autres agents. Dans les entreprises humaines, on l'appelle éventail de subordination ou surface de contrôle. On trouve ce modèle dans les systèmes d'acteurs.

I.3.4 La Communication

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

I.3.4.1 Protocoles de communication

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

Les primitives de base d'un protocole de communication sont les suivantes [LaLe93]:

- établissement d'une connexion entre deux entités ;
- identification du nœud destinataire dans un réseau de communication ;
- envoi de données ;
- réception de données ;
- définition d'un type de communication : synchrone ou asynchrone.

Les protocoles de communication sont les règles nécessaires à la communication, ils permettent de structurer et d'uniformiser les interactions inter-agents. Un protocole de communication peut être basé sur une architecture standard de communication, en l'occurrence, Internet ou Osi.

I.3.4.2 Modes de communication

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

I.3.4.2.1 Communication par envoi de messages

Les agents sont en liaison directe et envoient leurs messages directement et explicitement au destinataire. La seule contrainte est la connaissance de l'agent destinataire : « Si un agent A connaît l'agent B alors il peut entrer en communication avec lui » [LaLe93].
Les systèmes fondés sur la communication par envoi de messages relèvent d'une distribution totale à la fois de la connaissance, des résultats et des méthodes utilisées pour la résolution du problème. Les systèmes d'acteurs en sont l'illustration parfaite.

Ce type de communication concerne aussi bien les agents cognitifs que les agents réactifs. Les agents échangent des informations en envoyant et en recevant des messages. Ces échanges peuvent se faire explicitement entre deux agents (point à point), d'un agent à un groupe d'agents (Multicast) ou d'un agent à tous les autres agents (broadcast). Pour cela, les agents disposent d'un même protocole de communication. Ils ont, en outre, une base de connaissance locale sur leurs accointances (les autres agents). Donc un agent peut à tout

moment, s'il le juge nécessaire faire des envois de message qui peut être une information ou une requête. Nous nous trouvons dans un cas où il y a une distribution des connaissances.

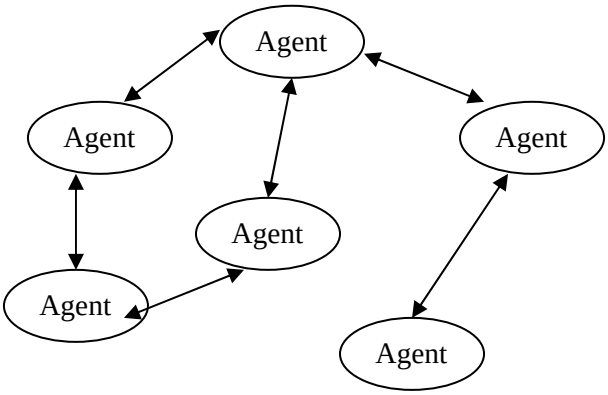


Fig. I.6 : Communication par envoi de messages

I.3.4.2.2 Communication par partage d'informations

Ce type de système de communication concerne les agents les plus évolués notamment les agents cognitifs. Il est mis en évidence par le fait que les agents communiquent à travers une structure de données contenant les données initiales du problème et au fur et à mesure qu'on avance vers la solution, on y stocke les solutions partielles des différents agents. Cette structure est souvent appelée Blackboard tous les agents ont accès à la structure de donnée ; ils y prennent les informations qui les intéressent, les traitent puis déposent leur solution dans la structure. De cette manière, les agents ne sont pas au courant de l'existence de leurs paires.

Les agents sont anonymes en ce sens qu'il n'est pas nécessaire à un agent de savoir qui a émis telle ou telle information. Les agents ne sont en relation directe qu'avec le tableau noir dont ils sont connectés par le système de contrôle. Par conséquent, il est plus facile de les supprimer ou d'en ajouter dans le système car seul le BB doit en être informé.

Un problème qui doit être relevé est le fait qu'un libre accès au BB amène les agents à traiter un grand nombre de solution partielle. Donc la dynamique de résolution est organisée en niveau.

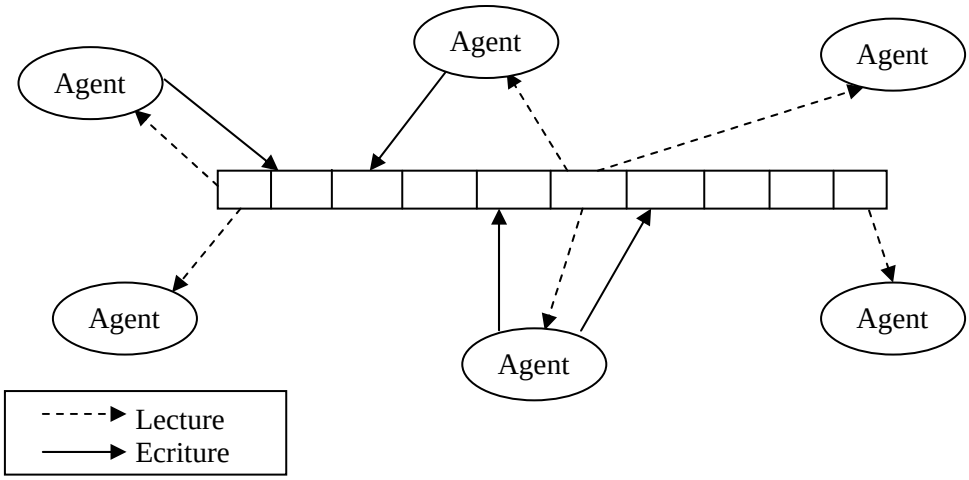


Fig. I.7 : Communication par partage d'information

I.3.4.2.3 Délégation

Les accointances sont spécifiques à chaque agent. A la réception d'un message, l'agent vérifie si l'expéditeur du message fait partie de ses accointances. Si l'expéditeur est inconnu, l'agent ne va pas traiter le message, mais le déléguer à un autre susceptible de s'y intéresser. Pour cela, il consulte les informations qu'il possède sur les modèles des autres agents.

La délégation est un mécanisme asynchrone et non bloquant, permettant à chaque acteur, ne pouvant pas traiter un message, de le déléguer à un autre acteur, appelé PROXY [LaLe93]. Le PROXY prend en charge la tâche qui lui a été déléguée et l'acteur déléguant est donc immédiatement prêt à traiter un autre message. Le PROXY connaît toutefois l'acteur déléguant, pour l'interroger dans le cas où il aurait besoin d'information supplémentaire.

La délégation est un moyen de partage de connaissances et de comportement plus souple que la notion d'héritage. Un acteur peut se choisir dynamiquement de nouveaux PROXYs, alors que le partage de connaissances établi par le graphe d'héritage est statique.

I.3.5 La Résolution de conflits

Les agents coopératifs ont besoin d'éviter autant que possible les situations conflictuelles pour résoudre un problème, Chaque agent a des buts et des objectifs qui lui sont propres et ils peuvent être en conflit avec ceux des autres. Dans le souci de résoudre le problème, une résolution de conflit est nécessaire. Ils concernent la négociation et coordination d'actions. La première s'occupe de la manière dont les agents s'échangent des informations et la seconde traite de la façon dont les actions des différents agents doit être organisées dans le temps et dans l'espace de manière à réaliser les objectifs [Fer95].

I.3.5.1 La négociation

La négociation correspond au dialogue entre agents afin de concilier des vues incohérentes et d'obtenir un compromis sur la manière dont les agents doivent agir ensemble. [Del00]

K.Sycara [Syc89] la définit comme étant un processus itératif qui implique l'identification des itérations potentielles par communication ou par raisonnement sur les états constants et intentions des autres agents du système et la modification de ces intentions de manière à éviter les interactions nuisibles et créer les situations coopérantes.

La négociation a été introduite dans le but de conjuguer les intérêts individuels des agents souvent conflictuels de manière à satisfaire les contraintes du problème en désignant les sources du conflit de telle sorte qu'on y pallie sans pour autant modifier les buts des agents.

La négociation est un processus adaptatif (instant du déclenchement du conflit inconnu), sûr car l'indisponibilité d'un agent ne bloque pas le processus et efficace car les communications sont structurées de telle sorte qu'elles minimisent les échanges [Vin93]. La communication et l'organisation influencent beaucoup le type de négociation en fonction des politiques. La négociation permet aux agents d'avoir un contrôle distribué sur les données incohérentes (du domaine) et d'avoir un contrôle sur les stratégies (changement) de résolution (allocation de tâches). La négociation est caractérisée par: [LaLe93]

- Un faible nombre d'agents impliqués dans le processus, la plupart des travaux ne mettant en œuvre que deux agents négociants ;
- La négociation implique au moins le protocole d'actions suivantes : proposer, évaluer, corriger et accepter une solution ;
- Le processus peut se dérouler de manière distribuée ou centralisée. Dans ce dernier cas, un agent est impliqué en tant qu'agent négociateur, il est toujours le même ou désigné en fonction du contexte.
- Afin de permettre les échanges d'information, un langage commun est nécessaire ;
- Les agents peuvent avoir besoin du modèle des autres agents pour négocier ;
- La négociation ne consiste pas forcément à trouver un compromis à partir des positions initiales mais peut également consister à modifier les croyances de l'autre agent pour faire prévaloir son point de vue.

I.3.5.2 La coordination

La résolution d'un problème par un ensemble d'agents doit être temporairement coordonnées pour qu'elle soit utilisée par d'autres agents.

La coordination est définie comme étant l'ensemble des activités supplémentaires qu'il est nécessaire d'accomplir dans un environnement comprenant plusieurs agents et qu'un agent seul poursuivant les mêmes buts n'accomplirait pas. [Vin93].

La coordination est l'activité qui permet aux agents de considérer toutes les tâches (aucune tâche n'est ignorée) et de ne pas dupliquer le travail [LaLe93].

La coordination est construite par les agents en fonction de leurs différents objectifs et en conciliant les contraintes locales des agents tout en permettant la résolution d'un problème global.

Les systèmes d'allocation de tâche et de coordination tels que la médiation (médiateurs ou courtiers reliant les agents) et le réseau contractuel basé sur le marché récupèrent les résultats d'un agent et cherche celui à qui il doit affecter la tâche suivante. Il y a là une certaine planification (distribuée).

Les agents coopèrent avec une négociation et une coordination d'action qui leur permet d'éviter au maximum les conflits. C'est ce principe qui est adopté dans le modèle du Partial Global Planning (PGP). Les agents s'échangent leurs plans partiels respectifs permettant à chacun de modéliser les activités des autres.

I.3.6 L'Emergence

L'émergence est un concept relatif aux systèmes réactifs. C'est un phénomène complexe et très peu compris, il est en cours d'exploration dans les systèmes multi-agents réactifs ; il est aussi étudié par beaucoup de chercheurs en physique, en biologie et en systématique.

Une fonctionnalité qui émerge est une fonctionnalité qui n'existait pas au départ et qui n'a pas été explicitement programmé, elle est le résultat d'interaction entre deux ou plusieurs comportements [Fer95]. Prenons cet exemple introduit dans [LaLe93] : si on veut qu'un robot (en marche) suive un mur (comportement suivre le mur), on définit deux comportements (antagonistes) : être attiré par le mur et être repoussé par le mur. De l'interaction de ces deux comportements va émerger un nouveau comportement : suivre le mur.

Dans ce cas, on parle d'émergence de comportement, mais il existe aussi d'autres formes d'émergence telles que l'émergence de rôle, l'émergence de stratégie, l'émergence de contrôle et émergence de structures de coordination.

I.4 Conclusion

Nous avons vu dans ce chapitre, les différents concepts qui entrent en jeu lors de la mise en œuvre d'un Système Multi Agents. En fait, nous avons remarqué que ce sont des concepts souvent interdépendants. En effet, la communication est à la base de toutes les transactions entre agents. Or, la résolution distribuée de problème nécessite une certaine organisation et une coopération entre agents, c'est-à-dire l'échange d'information relative aux solutions partielles. Le contrôle doit mener le processus de résolution et ainsi, assurer la coordination et une négociation pour éviter ou résoudre les conflits occasionnés par les buts différents agents.

Nous avons aussi abordé le sujet de l'émergence d'intelligence dans les systèmes à agents réactifs. C'est un concept qui n'est pas très maîtrisé dans la pratique. Il se trouve toujours au stade de recherche.

Chapitre II

Modèles organisationnels pour la conception des Systèmes multi agents

Du point de vue de la réflexion sur la distribution des tâches et de l'interaction cohérente entre les agents dans la résolution de problème distribué ou dans un système multi agents, le problème de base est une question d'organisation, à savoir, décider quel agent fera quoi et quand.

Les Gasser 1988.

Chapitre II

Modèles organisationnels pour la conception des Systèmes multi agents

II.1 Introduction

Les problèmes que rencontrent les systèmes multi agents peuvent être classés en deux types : Le premier concerne les problèmes classiques de l'IA qui ont pris une autre dimension dans le contexte multi agents [LaLe93] :

- La modélisation et la répartition de la connaissance entre les agents (comment formuler, comment décomposer….) ;
- La prise en compte de plusieurs agents lors de la génération de plan ;
- La gestion de conflit entre agents (coopération et négociation) et la maintenance de la cohésion dans leurs décisions et leurs plans (coordination) ;
- La communication entre agents (langage et protocole) ;

Le second concerne les problèmes liés à l'IAD, notamment à l'organisation d'un groupe d'agent, l'architecture à adopter et le problème d'un agent au sein d'une communauté en terme de capacité social, d'hétérogénéité de langage et protocole….

Pour résoudre, notamment le deuxième problème, Les Gasser, lors d'une conférence invitée à ICMAS'95[Gas95], s'était fait l'avocat d'un enrichissement de l'IAD par les théories de l'organisation. Son système MACE [Gas&87] utilisait explicitement le concept de rôles pour définir un SMA, par exemple, dans le protocole de réseau contractuel, des agents ayant un rôle de gestionnaire faisaient de propositions à des agents ayant le rôle d'offrant. M.Fox [Fox 81] a également adopté dans ces travaux une vue organisationnelle.

Plusieurs travaux sont centrés sur le concept de rôle, mais ces dernières années, il existe deux méthodologies qui sont les plus répondus dans l'univers des SMA, le premier basé sur trois concepts : l'agent, le groupe et le rôle (appelé AALAADIN), et la deuxième méthodologie de conception a une notion de rôle plus spécifique tels que les : permissions, responsabilités, protocoles… (appelé GAIA). Dans ce chapitre, on étudiera chaque modèle en détail.

II.2 Généralités

II.2.1 Définitions

En adaptant [Gas92], nous définissons l'organisation comme un cadre d'activité et d'interaction, mis en place par la définition de groupes, rôles et leur relation.

Selon Ferber [Fer95], une organisation peut être définie comme un agencement de relation entre composants ou individus qui produit une unité ou système. L'organisation lie d'une façon interrelationnelle des éléments ou événement ou individus divers qui dès lors

deviennent les composants d'un tout. Elle joue avec l'interaction l'un des concepts de base des systèmes multi agents.

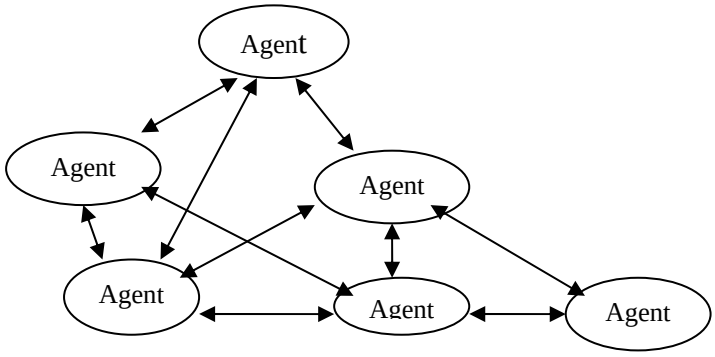
Une organisation est une structure («pattern ») décrivant comment les membres de l'organisation sont en relation et interagissent afin d'atteindre un but commun : les sorties («output ») de l'organisation [Fox81]. Dans ce cas, la structure de l'organisation est fonction de l'environnement dans laquelle elle évolue, des ressources disponibles et de la nature des sorties. Les ressources sont souvent appelées information et le rôle de chaque agent référencés comme des tâches. L'introduction de la «connaissance organisationnelle » dans un SMA est due d'une part à la décomposition du problème donc à la coopération et d'autre part à la coordination dans la société d'agents. Fox pense que la création des structures organisationnelles est due à la capacité limitée des possibilités humaines ou des systèmes artificiels.

II.2.2 Structure d'organisation

Il existe deux architectures d'organisation pour les sociétés d'agents, la structure horizontale et la structure verticale [LaLe93].

II.2.2.1 Structure horizontale

Dans de telles sociétés, tous les agents sont au même niveau, il n'y a pas d'agents maîtres et d'agents esclaves. C'est le cas, par exemple, d'un groupe d'agents ayant des spécialités différentes travaillant pour la résolution d'un même problème.



FigII.1. Structure horizontale

II.2.2.2 Structure verticale

Dans une architecture verticale, les agents sont structurés par niveaux. Dans un même niveau on retrouve localement une structure horizontale. Le fonctionnement des agents dans de telles sociétés est le suivant : l'agent reçoit le problème à résoudre d'un autre agent qui lui est supérieur dans la hiérarchie, il le décompose en :

- des sous-problèmes auxquels il peut répondre localement,
- des sous-problèmes qu'il pourrait résoudre en coopèrent avec les autres agents du même niveau que lui,
- des sous-problèmes qu'il fait suivre aux agents du niveau inférieur dans la hiérarchie.

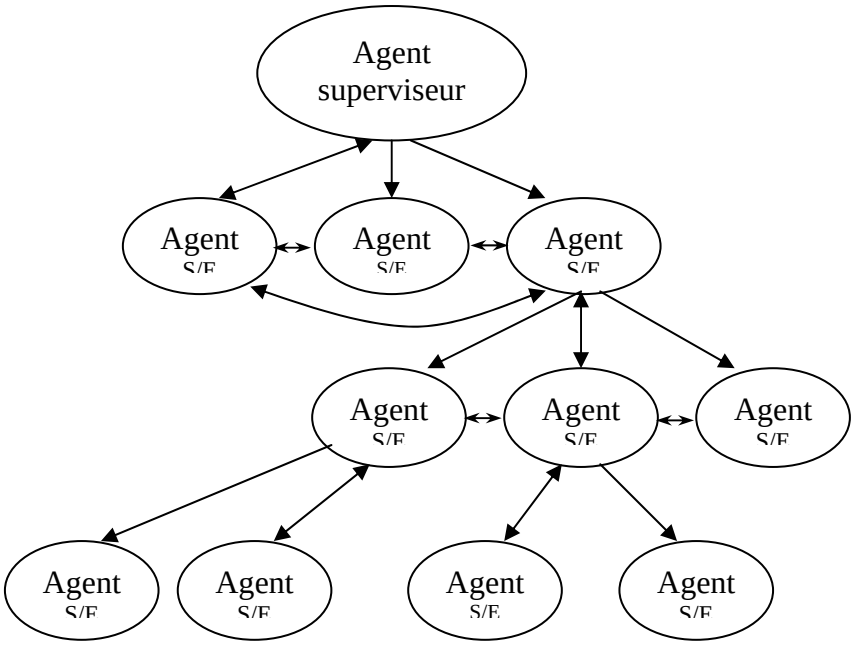


Fig II.2. Structure verticale

II.3. Le Modèle AALAADIN

Ce modèle a vu le jour au laboratoire d'informatique, de robotique et de microélectronique de l'université de Montpellier II (LIRMM) en 1998 par Jacques Ferber et Olivier Gutknecht, il a été décrit de façon plus complète dans [FeGu98].

Le modèle AALAADIN est basé sur trois concepts, à savoir l'agent, le groupe et le rôle, il permet d'exprimer et d'analyser divers systèmes multi agents en utilisant avant tout les concepts organisationnels. AALAADIN situe l'analyse de SMA à deux niveaux : [GuFe99]

-- Le niveau descriptif correspond aux concepts primaires d'agent, de groupe et de rôle. C'est à ce niveau que se décrit une organisation multi agents effective.

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

II.3.1. Niveau descriptif

La figure (Fig.II.1.) présente un diagramme UML de ce modèle basé sur les trois concepts : Agent, Groupe et Rôle.

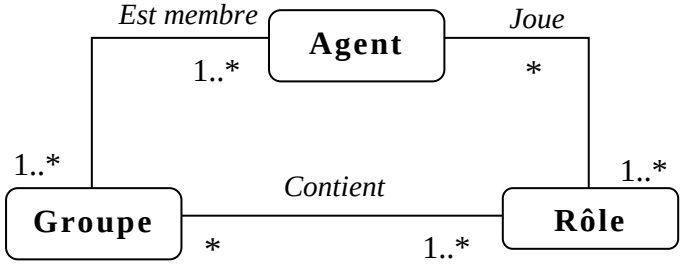


Fig.II.1. Les trois concepts centraux

II.3.1.1. Agent

Quasiment aucune contrainte n'est posée sur l'architecture interne de l'agent, ou sur le modèle particulier de l'agent. Il est décrit comme une entité autonome communicante qui joue des rôles au sein de différents groupes.
La très faible sémantique associée à l'agent dans ce modèle est volontaire, le but est de laisser toute liberté au concepteur pour choisir l'architecture interne appropriée à son domaine applicatif.

II.3.1.2. Groupe

Le groupe est la notion primitive de regroupement d'agents. Chaque agent peut être membre d'un ou plusieurs groupes d'une façon simpliste, un groupe peut être vu comme un moyen d'identifier par regroupement un ensemble d'agents, plus classiquement, associé au rôle, il définira la structuration organisationnelle d'un SMA usuel, les différents groupes peuvent se recouper librement.

Par exemple (FigII.2.), si on prend un agent humain, il peut participer à divers groupe, un étudiant en P.G (Post Graduation), dans le même temps il est chercheur au laboratoire LRIA, il effectue le métier d'enseignant et dispute pendant son temps libre des matchs de football. Ce que nous constatons dans ces situations, est la variété des situations d'interactions, car chaque communauté a ses propres fonctions et langages

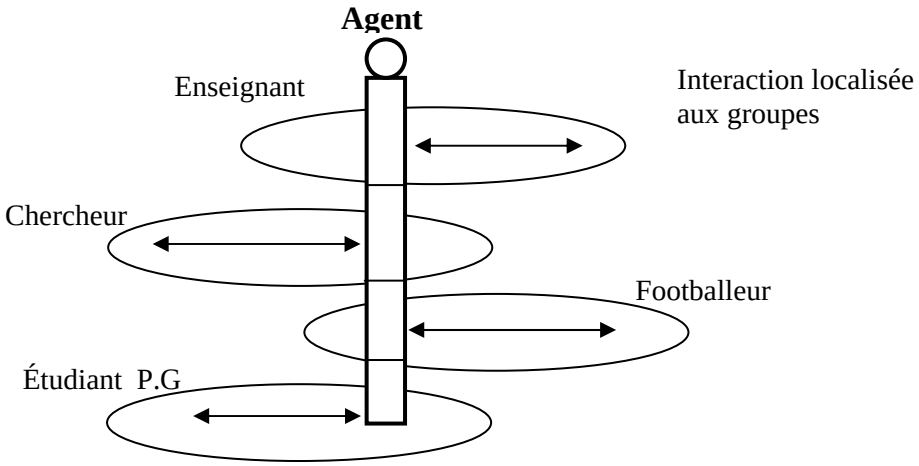


Fig.II.2. Le principe de groupe

II.3.1.3. Rôle

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.
A chaque rôle est associée une fonction de rôle qui défini les contraintes que devra respecter l'agent pour pouvoir jouer le rôle en question.

II.3.1.4. Caractéristiques

– 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 locales à un groupe.

– un agent *a* peut entrer dans un groupe *g* pour y jouer un rôle *r* si la fonction d'acceptation booléenne $f_{gr}(a)$ est évaluée à vrai. Les auteurs n'ont pas défini les mécanismes pour contrôler cet accès, mais néanmoins quelques fonctions d'admission à un rôle ont été décrites dans [FeGu98] :

- **Acceptation ou refus automatique**, où un agent voit ses demandes d'admission systématiquement acceptées ou rejetées.
- **Admission conditionnée par les compétences**. un rôle est confié à un agent seulement s'il possède un certain jeu de compétences.
- **Admission contrainte par l'architecture** pour se voir accepté dans un groupe, il faut que l'agent ait certains pré-requis sur sa structure interne.
- **Admission négociée.** Un agent devra entrer dans un dialogue initial avec l'agent ayant le rôle de gestionnaire du groupe pour déterminer ses droits au rôle.

II.3.2. Niveau organisationnel

La figure (Fig.II.3.) nous présente quelques termes supplémentaires qui ne sont pas représenté directement dans une organisation multi agents. [FeGu98] :

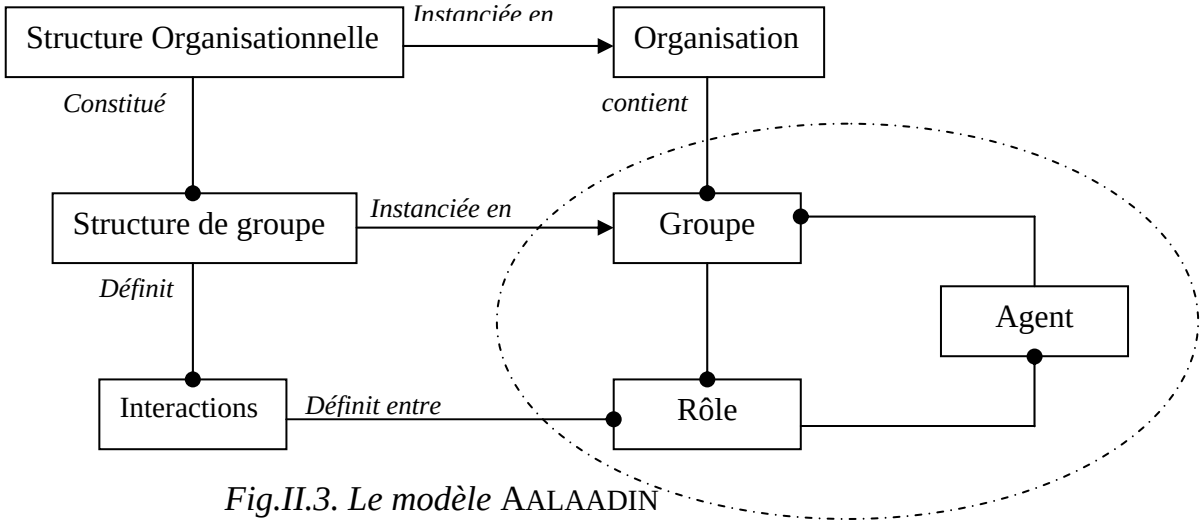


Fig.II.3. Le modèle AALAADIN

II.3.2.1. Structure de groupe

Une structure de groupe est une description abstraite d'un groupe. Elle identifie tous les rôles et les interactions qui peuvent survenir au sein d'un groupe. Elle se définit comme un tuple: S = < R , G , L > où :

- 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 spécifiant l'ensemble des interactions entre rôles deux à deux. G: R x R ⟶ L.

L'orientation correspond au rôle initiant l'interaction
- L est le langage d'interaction. C'est le formalisme choisi pour décrire chaque interaction.

II.3.2.2. Structure organisationnelle

La structure organisationnelle est un ensemble de structures de groupe définissant un modèle d'organisation multi agents. Elle est peut être vue comme la spécification globale du problème initial. Les différentes structures de groupe mises en œuvre 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 définit 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 ou chaque arc Sa Sb réunit deux rôles r_1 r_2 , où (S_a , S_b)$\in S^2$, avec r_1 et r_2 étant des rôles définis dans les structures de groupes *Sa* et *Sb* respectivement.

Un représentant entre deux structures de groupes S_a et S_b est un agent qui aura simultanément le rôle $r_{a,i}$ dans le groupe S_a et le rôle $r_{b,j}$ dans le groupe S_b.

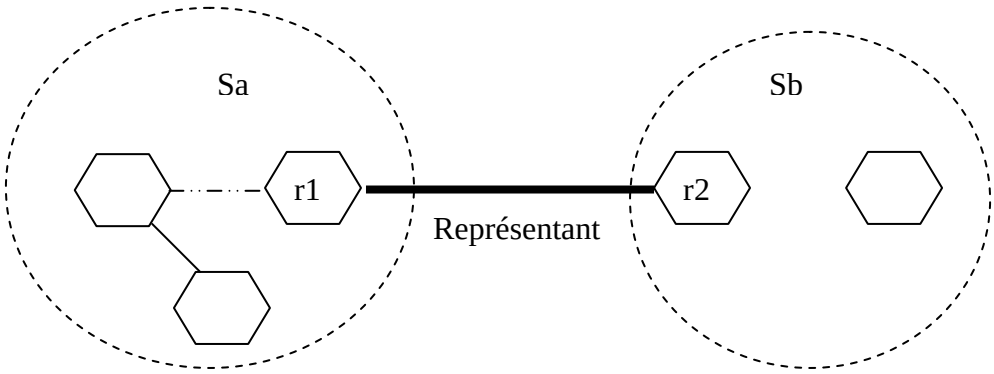


Fig.II.4. Structure organisationnelle

II.3.2.3. Le rôle de "gestionnaire de groupe"

Pour s'assurer qu'un groupe d'agents restent dans leurs modèle durant leurs cycles de vie, un rôle important et particulier a été défini dans un groupe, c'est le rôle de gestionnaire de groupe. Ce rôle correspond à la responsabilité de gestion des demandes d'admission dans le groupe et de tenue de rôle, ainsi que de leurs éventuelles révocations. Il peut être vu comme l'agent responsable de l'évaluation
des fonctions d'acceptations définies, et le contrôlé de la cohérence du groupe actuel de point de vue de la définition de structure de groupe [GuFe98] .

II.3.3. Exemple

Étant donné un exemple d'une communauté d'agent représentant une structure de recherche de site Web contenant des informations désirées par des utilisateurs, on définit trois groupes (Fig.II.5.), le lien entre les deux modèles d'interaction est réalisé par le rôle de Broker qui est un représentant dans le groupe utilisateurs via un rôle de Broker.

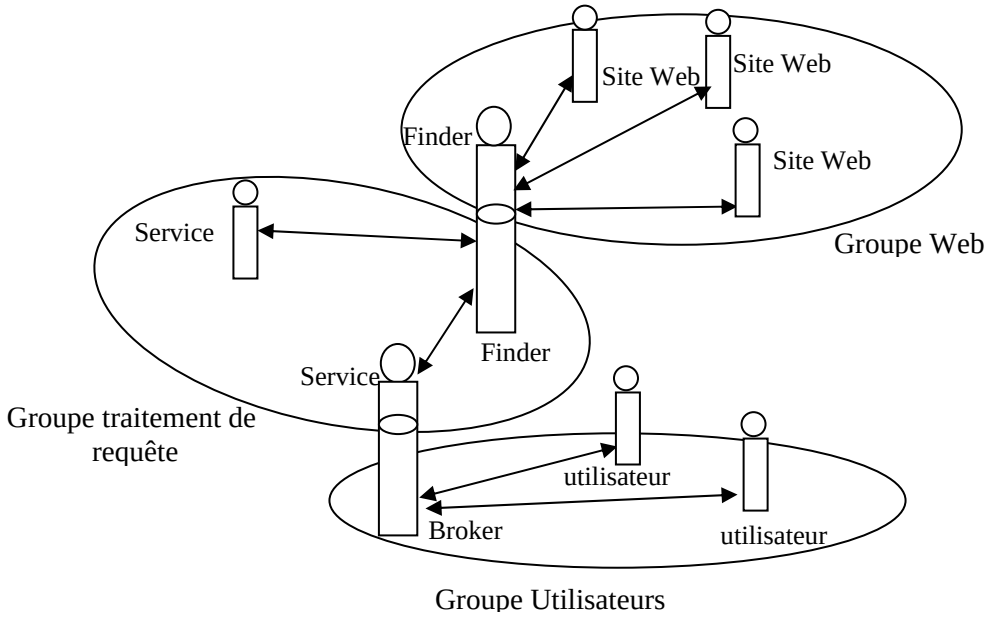


Fig.II.5. La structure organisationnelle du « Recherche d'informations».

Le Finder interagit avec l'intermédiaire de ces services pour chercher un site Web adéquat.
Une organisation réelle possible découlant de cette structure organisationnelle est montrée en figure ci-après (Fig.II.6.):

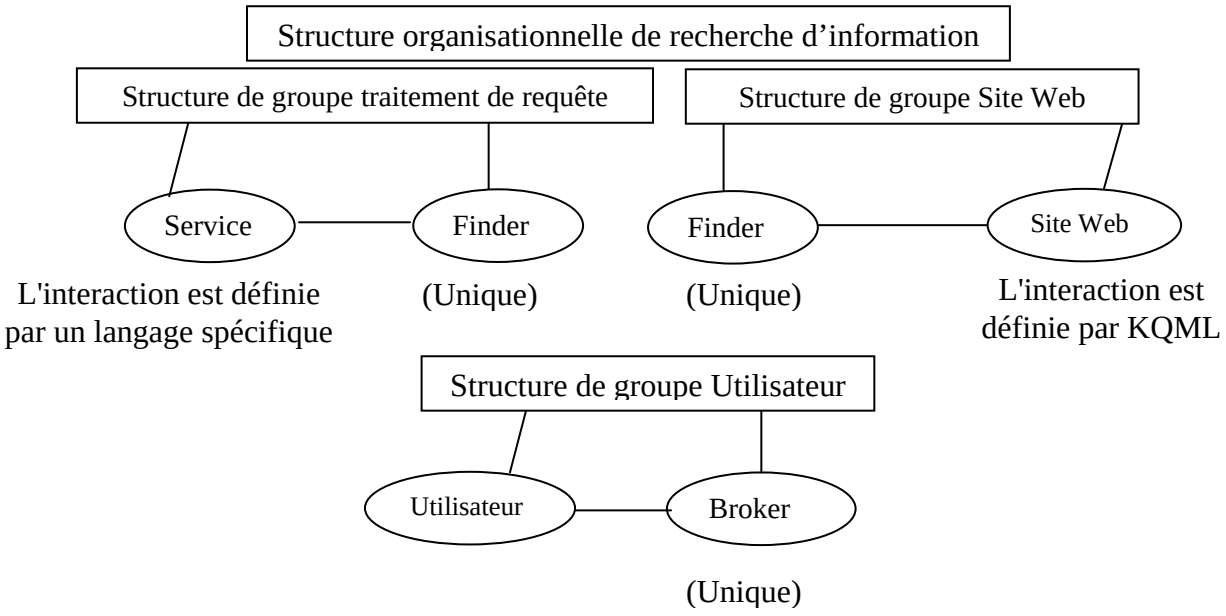


Fig.II.6. Une Organisation de la recherche d'information

Trois structures de groupe sont définies, un *groupe utilisateurs* qui contient les agents *utilisateurs*, un *groupe traitement de requête* qui contient un agent *service* qui a le rôle de fournir un annuaire des sites Web pour l'agent *Finder*, ce dernier est le responsable de la recherche d'information, le lien d'interaction entre ces deux groupes est tenu par un agent qui joue deux rôles, un rôle *Broker* dans le *groupe Utilisateur*, et un rôle de *service* dans le groupe *traitement de requête,* qui fournit au rôle *Finder* la requête d'un *utilisateur*. Le troisième groupe intitulé *groupe Site Web*, contient plusieurs sites Web et l'agent *Finder*.
Le rôle *Finder* est membre des deux groupes en même temps, ce qui implique un lien direct d'un dialogue entre les deux groupes, il communique avec le groupe de *Traitement de requête* avec un langage spécifique (ACL FIPA par exemple), et il est capable de communiquer avec le langage KQML dans le *groupe clients*.

Cet exemple illustre qu'un agent appartenant à deux groupes différents, peut simultanément interagir avec deux schémas de communications différents en même temps et facilement. On voit aussi qu'un agent prend part à plusieurs groupes et peut jouer de multiples rôles simultanément.

II.3.4 Processus méthodologique

Comme toute construction d'une application, le processus s'effectue en trois phases : Analyse, Conception et Exécution.
Voici les trois étapes de ce processus [GuFe98] :

Analyse
1. Identifier les groupes.
2. Choisir un modèle organisationnel spécifique adapté à chaque structure de groupe isolée.

Conception
3. Spécifier la structure organisationnelle dans le modèle choisi.
4. Identifier les entités supplémentaires éventuelles.
5. Décrire les schémas d'interactions.

Exécution
6. Définir les architectures individuelles.

Pour isoler des groupes, il nous faut disposer de critères de formation d'un groupe. Le premier critère, qui est le seul pré-requis est l'existence d'un mécanisme de communication commun au sein du groupe, puisque la définition du groupe étant centrée sur l'interaction entre les membres, d'autres critères d'isolation d'un groupe sont soit inhérents à l'application (activité commune), ou à l'architecture ou au paradigme de communication (propriété commune).

La deuxième étape est la séparation des rôles au sein des groupes ainsi isolé. La spécification de la structure de groupe se fait alors dans un des modèles relatifs à l'application, les contraintes de l'application entre rôles s'établissent dans l'expression des fonctions de rôles.
Les scénarios d'interactions sont précisés entre les rôles identifiés. Ces modèles de protocoles sont exprimés dans le formalisme modèle organisationnel dans lequel a été exprimée la

structure de groupe, (Ce formalisme peut être des diagrammes de séquence d'UML, des scénarios ACL ou KQML, ou un réseau de Pétri définissant le protocole…).
Enfin le choix de l'architecture est alors contraint par les rôles possibles pouvant prendre un type d'agent particulier, et pour l'implémentation des différents schémas d'interactions.

II.4. Le modèle GAIA

GAIA est avant tout, une méthodologie de conception d'un système multi agents organisationnel basé uniquement sur le concept de rôles, proposés par Michael Wooldridge, Nicholas Jenings et David Kenny en 1999 [Woo&99]. Dans cette approche la notion de rôle est précisé par des propriétés spécifiques : permissions, responsabilités, activités et protocoles.
Les modèles utilisés par GAIA sont spécifiés dans la figure (Fig.II.7.). GAIA empreinte quelque terminologie et notation de de la méthode FUSION [Col&94] et décompose le processus en deux phases : analyse et conception.

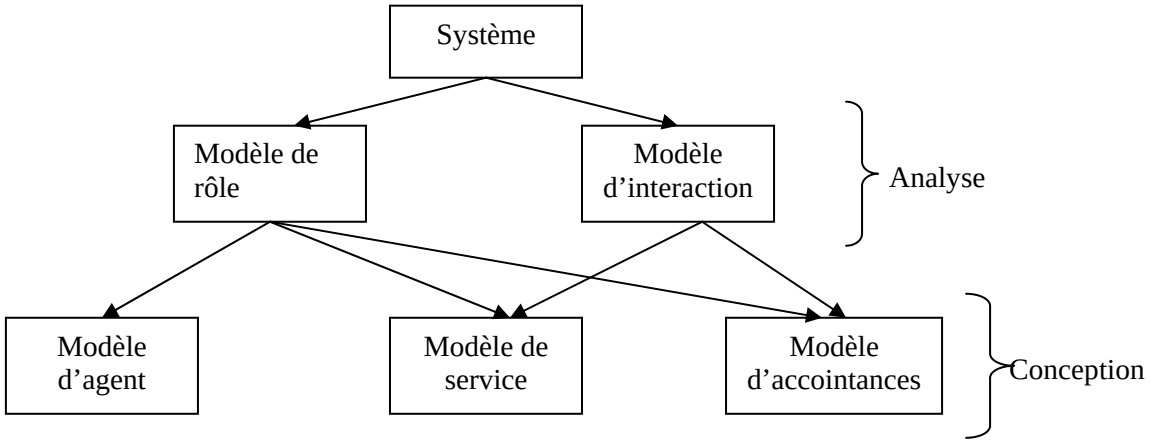


Fig.II.7. les modèles de Gaia.

II.4.1. Phase d'analyse

L'objectif de cette étape est de développer une compréhension de l'organisation et de ces structures, l'organisation est vue comme une collection d'objets de rôles, qui ont une certaine relation entre eux.

II.4.1.1. Le modèle de rôle

Le modèle de rôle identifie les rôles clés d'une organisation, le rôle est vu comme une fonction de description abstraite d'une entité, il est caractérisé par deux types d'attributs :

- *Les permissions* (droits associés avec le rôle) : Un rôle est associé à une permission relative à la quantité de ressources qui peuvent être exploitées durant l'exécution d'un rôle.
- *Les responsabilités* : Un rôle est créé pour faire quelque chose, donc il a une certaines fonctionnalités, ces dernières sont représentées par cet attribut.

A. ***Les permissions*** : les permissions associées à un rôle ont deux aspects :
   i)- Elles identifient les ressources utilisées pour effectuer le rôle, intuitivement, elles nous montrent ce qu'on peut utiliser pour effectuer le rôle.

ii)- Elles situent les limites d'une ressource dans lequel le rôle exécuteur doit fonctionner, intuitivement, elles nous montrent ce qu'on ne peut pas utiliser pour effectuer le rôle.

Généralement les permissions sont reliées à n'importe quelles ressources, cependant dans GAIA on pense aux ressources comme une information ou une connaissance que l'agent peut avoir [W00&99].

Pour effectuer un rôle, un agent doit être capable d'accéder à des informations, quelques rôles produisent des informations (le mot clé dans GAIA est generate), d'autres ont un accès aux informations mais sans faire une modification (reads) et enfin d'autres peuvent modifié les informations (changes).

B. ***Les responsabilités*** : Les fonctions d'un rôle sont définit par ces responsabilités, ces dernières peuvent être divisés en deux catégories : responsabilités de vivacités et celles de sécurités.

Les responsabilités de vivacités (liveness responsabilities) indiquent que quelque chose sera fait, ainsi l'agent qui effectuera ce rôle restera vivant (en vie). Dans GAIA cette propriété est spécifié via une expression de vivacité qui définit le cycle de vie d'un rôle, cette expression est une expression régulière, elle nous montre la trajectoire d'exécution à travers différents protocoles et activités associé avec le rôle, sa forme générale est :

nom_Rôle = Expression.

Avec *nom_Rôle* est le nom de rôle où les responsabilités de vivacités sont définit et expression est l'expression de vivacité qui définit les responsabilités de vivacités du *nom_Rôle*.

Le tableau (Tab.II.1.) illustre les opérateurs utilisés dans les expressions de vivacités :

Opérateur	**interprétation**
x.y	x suivit de y
x|y	x ou y
x*	Plusieurs occurrences de x (0 possibles)
x+	au moins une occurrence de x
x^{w}	infinité d'occurrence de x
[x]	x est optionnel
x||y	x et y

Tab.II.1. Les opérateurs d'expressions de vivacités

Les composants atomiques des expressions de vivacités sont les activités et les protocoles.

Une activité est semblable en quelque sorte à une méthode dans le terme orienté objet, ou une procédure dans un langage comme pascal. Elle correspond à une action qu'un agent doit exécuté, sans impliquer une interaction avec un autre agent, d'autre part les protocoles sont des activités qui exigent des interactions avec d'autre agents (pour différencier les deux termes, les activités seront soulignées).

Dans plusieurs cas, il est insuffisant de spécifier un rôle uniquement par ses propriétés de vivacités, car en effectuant le rôle, l'agent doit également satisfaire un certain invariant, par exemple dans une application d'E-commerce un agent ne doit pas dépenser sur un article plus

d'argent qu'il ne le lui a été attribué. Ces invariants sont appelés les conditions de sécurité, car elles sont confrontées à quelques conditions indésirables.

II.4.1.2. Le modèle d'interaction

Dans une organisation multi agents, il existe des dépendances et des relations entre plusieurs entités, en effet, ces interactions sont au centre des fonctionnalités d'un système. Dans GAIA, le modèle d'interaction représente les liens qu'on peut avoir entre les rôles, ce modèle contient un ensemble de définitions de protocole, un pour chaque type de d'interaction inter rôle. Ici le protocole est vu comme un schéma d'interaction qui définie formellement chaque séquence d'exécution. Chaque définition d'un protocole a les attributs suivants :

- Résolution (purpose) : un texte bref décrivant la nature de l'interaction.
- Initiateur : le(s) rôle(s) responsable(s) pour le début de l'interaction.
- Répondeur : le(s) rôle(s) où l'initiateur veut agir.
- Données : informations utilisées par le rôle initiateur.
- Résultats : informations fournie par le répondeur durant l'interaction.

II.4.2 Exemple

Pour introduire les concepts de la phase d'analyse, on suppose un exemple de rôle *Utilisateur* pour l'exemple précédent de recherche d'information. Ce rôle permet d'introduire les informations pour rechercher des sites Web qui correspondent à une requête introduite par un utilisateur, les permissions suivantes sont une simple illustration associée avec le rôle *Utilisateur* :

Reads	source	// nom du site Web.
Changes	connexion	// la connexion existe ou pas.
generate	mots-clés	// l'information à rechercher.

Ces spécifications indiquent qu'un agent effectuant ce rôle a une permission pour accéder à la valeur *source*, il a une autre permission de lire et de modifier la valeur de *connexion* et de générer l'information *mot-clés.* Noter que ces permissions ont un lien avec les connaissances que doit avoir un agent. En effet, *connexion* est une représentation qui correspond à une valeur dans un monde réel.

L'expression de vivacité qui mentionne la responsabilité du rôle est :

$$Utilisateur = (\underline{VérifierConnexion}.TransmettreRequête.\underline{AttendreResult}.RecevoirResult.\underline{Afficher})^{+}$$

Cette expression indique que le rôle *Utilisateur* doit exécuter l'activité de *vérifierConnexion* suivi par le protocole *TransmettreRequête,* ensuite l'activité *AttendreResult*, ensuite du protocole *RecevoirResult*, et enfin l'activité *Afficher*. Cette séquence d'exécution est répétée au moins une fois (Opérateur +).
L'agent effectuant ce rôle doit s'assurer toujours que la connexion existe et que le résultat de la recherche *source* n'est pas vide, pour que l'expression de vivacité puisse avoir une durée finie, cet invariant s'exprime :

(*connexion*==Vrai) && (*source*<> {})

On peut donc définir le modèle de rôle dans une Table (Tab.II.2.). (Dans GAIA chaque rôle est représenté dans une table)

Nom rôle : *Utilisateur*		
Description Le rôle introduit des informations pour la recherche des sites Web.		
Protocoles et activités *VérifierConnexion*, *TransmettreRequête*, *AttendreResult*, *RecevoirResult*, *Afficher*.		
Permissions		
Reads	source	// nom du site Web.
Changes	connexion	// la connexion existe ou pas.
generate	mots-clés	// l'information à rechercher.
Responsabilités **De vivacités** : *Utilisateur* = (*VérifierConnexion*.*TransmettreRequête*.*AttendreResult*.*RecevoirResult*.*Afficher*)$^+$ **De sécurité** : (*connexion*==Vrai) && (*source*<>{})		

TabII.2 Schéma du rôle Utilisateur.

Considérons maintenant le protocole *TransmettreRequête*, qui est décrit dans le rôle *Utilisateur*. Ce protocole est initié par le rôle *Utilisateur* et prend part avec le rôle *Broker*, le rôle initiateur saisi l'information appelé *mots-clés* et le rôle *Broker* informe la valeur trouvée *source* (Fig.II.8.)

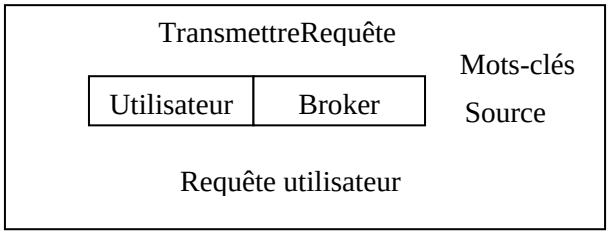


Fig.II.8. Définition du protocole TransmettreRequête.

II.4.3. Phase de conception

Le processus de conception dans GAIA entraîne la génération de trois modèles (Fig.II.7), le modèle d 'agents, le modèle de service et le modèle d'accointances.

II.4.3.1 modèle d'agents

Le but de ce modèle est de se documenter sur les types d'agents qui seront utilisés dans le système, et les instances d'agents qui les réaliseront dans l'exécution. Les types d'agents peuvent être mis en correspondance un à un avec les rôles, mais ce n'est pas toujours le cas, le concepteur peut associer plusieurs rôles à un même agent pour lui permettre d'atteindre son objectif.
Les instances d'agents qui apparaîtront dans le système ont une certaine notation dans ce modèle (TabII.3), par exemple la notation n reflète qu'on doit avoir exactement n agent de ce type pour effectuer le rôle.

notation	signification
N	Exactement *n* instances
m.. n	Entre *m* et *n* instances
*	*0* et plus d'instances
+	*1* et plus d'instances

Tab.II.3. Notation d'instances.

II.4.3.2 modèle de service

Comme son nom l'indique, le but de ce modèle dans GAIA, est d'identifier les services associés à un rôle, et de spécifier quelques propriétés pour que l'agent puisse les faire. Les propriétés d'un service sont les entrés, les sorties, les pré-conditions et les post-conditions.
Les entrées et les sorties sont dérivées des protocoles et des activités d'un rôle, et les pré et post-conditions sont à leur tour dérivées des responsabilités d'un rôle.

II.4.3.3 modèle d'accointances

Le dernier modèle dans GAIA est le modèle d'accointances. Ce modèle défini simplement la communication qui existe entre les agents, il ne défini pas les messages émis ou reçus, mais uniquement l'existence d'une communication.
Le modèle est un graphe orienté, où les nœuds correspondent aux agents et les arcs correspondent au communication existante. Par exemple un arc a $\longrightarrow$b indique que l'agent a envoi des messages à l'agent b.

Revenant à notre exemple du rôle *Utilisateur*, il existe trois autres rôles : le rôle *Finder* qui est le responsable de la recherche, un rôle de *Service* qui se présente comme des pages jaunes et, et les sites Web à rechercher.

Selon les notations du tableau précédant (*Tab.II.3)*, le modèle d'agents sera représenté en Fig.II.9. Le tableau Tab.II.4 nous montre le modèle de service, tendis que la figure Fig.II.10 nous donne une vue sur la communication entre agents (modèle d'accointances).

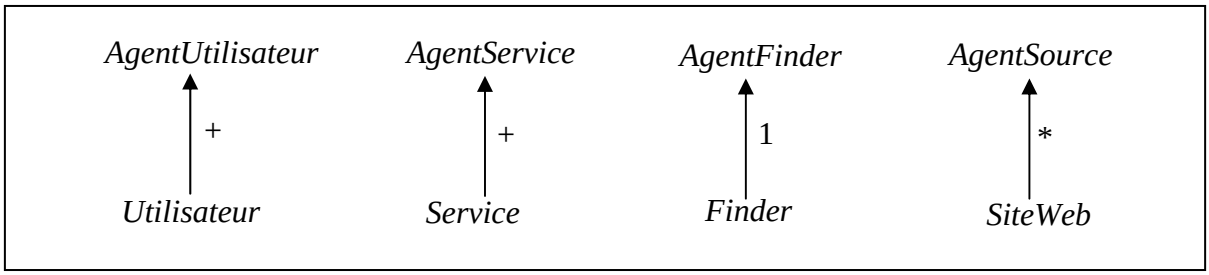


Fig.II.9. Modèle d'agents.

Services	Entrées	Sorties	Pré-conditions	Post-conditions
Transmettre la requête	mots-clés	source	connexion=Vrai	--
Recevoir le résultat	source	source	source<> {}	--
Vérification de la connexion	---	connexion	--	Mise à jour de connexion
Attendre le résultat	source	--	--	--
Affichage du résultat	source	--	source<> {}	--

*Tab.II.4. Modèle de Service***.**

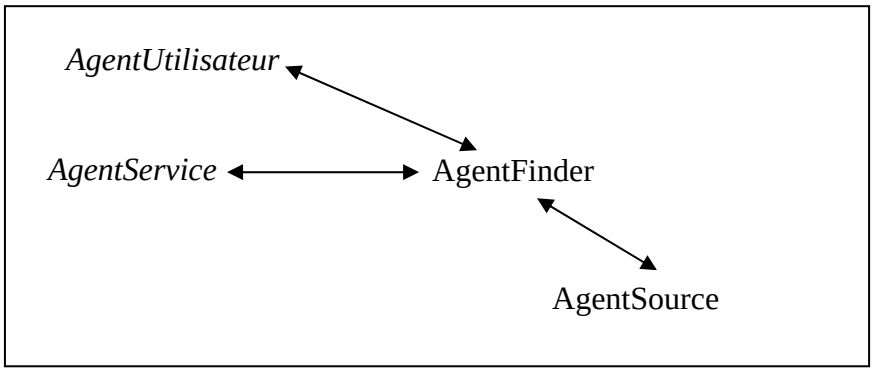


Fig.II.10. Modèle d'accointances.

II.4.4. Processus Méthodologique

Pour une meilleure conception d'un système organisationnel, voici la méthodologie GAIA présentée dans [Woo&99]:

1. Identifier les rôles dans le système.
   Sortie: un prototype de modèle de rôle (une liste de rôle clé qui intervenant dans le système)
2. Pour chaque rôle, identifier les protocoles associés, (c'est à dire le schéma d'interaction entre les rôles)
   Sortie: le modèle d'interaction.
3. En utilisant le modèle d'interaction, élaborer le modèle de rôle.
   Sortie: le modèle de rôle.
4. Itérer (1) … (3) le plus possible
5. Créé un modèle d'agent (les rôles sont des agents, puis raffiner l'hiérarchie et utiliser les notations pour les instances)
6. Développer le modèle de service en utilisant les protocoles et les responsabilités des rôles.
7. Développer le modèle d'accointances en utilisant le modèle d'interaction et le modèle d'agent.

II.5. Conclusion

Les Systèmes multi agents sont eux mêmes des organisations artificielles, ou des sociétés computationnelles, le terme société désigne une association d'agents organisés autour de ressources communes (connaissances, informations,…) de règles d'interactions et de buts. Dans cette perspective, la vue organisationnelle est appropriée aux concepts de description de niveau macro, qui est la société, en terme de comportement, et des caractéristiques interne d'agents participant (le niveau micro).[GuFe99]

Ces abstractions constituent un outil qui facilite la conception et modélise les règles, structures, et comportement social du développement des systèmes.

Contrairement à AALAADIN qui possède un modèle opérationnel mise en oeuvre à l'aide de la plate forme MADKIT (pour Multi Agents Development KIT) incluant les concepts centraux de l'organisation : agents, groupes et rôles, GAIA est considéré comme la première méthodologie qui a abordé l'aspect organisationnel dans les systèmes logiciels complexes orientés agents.

Les deux modèles considèrent que le rôle est une représentation abstraite d'une fonction, le rôle est vu comme un concept clé pour une conception organisationnelle, la notion de rôle dans AALAADIN peut s'affiner par l'ajout d'attributs complémentaires telle que (les permissions et les responsabilités de GAIA), ce qu'on appelle extension par raffinement. Dans AALAADIN le rôle est défini par des services (ou actions) mis à la disposition des autres rôles (externes) sans décrire la procédure à suivre pour réaliser ces actions, ni les relations entre elles, au contraire de GAIA qui propose de définir le rôle par ces actions internes, externes et les règles de disponibilités et d'enchaînement (par ordre temporel) de ces actions.

Cette approche est plus significative puisqu'elle définit plus clairement les contraintes comportementales des rôles et elle est plus proche de l'implémentation pour le développement des agents. Ainsi, les deux modèles n'introduisent pas la notion de but associé à l'agent, en effet, les buts ne sont pas définis dans la description de l'organisation.

Si on prend le modèle d'agents de GAIA, il représente les types et les instances d'agents qui seront utilisés dans le système, et le modèle d'accointances définit la communication existante entre agents. Par exemple si une application désire une dizaine d'agents A, une quinzaine d'agents B et un lien d'accointance entre les deux agents, on voit bien que la communication s'accroît rapidement dans le système, car toutes les instances d'agent A peuvent communiquer avec toutes les instances d'agent B. En revanche dans AALAADIN l'interaction est organisée au sein de groupes, où chaque groupe contient les instances d'agents, et le modèle d'accointances est représenté par un agent appartenant aux deux groupes (ou plus) qui assure l'interaction entre les groupes, Cette organisation permet de prendre en compte plus facilement l'hétérogénéité des langages d'interactions. Ce dernier point n'est pas abordé dans GAIA, qui ne prévoit l'hétérogénéité que dans les modèles d'agents.

Enfin, Les protocoles sont vus comme des schémas d'interactions qui doivent être identifiés et modélisés durant la phase d'analyse. Le formalisme utilisé par GAIA (modèle d'interaction) est malheureusement reconnu insuffisant [SiHa01].

Notons cependant que Zambonnelli.F a étendu le modèle GAIA pour prendre en compte l'exploitation et le fonctionnement des agents dans un environnement ouvert [Zam&01].

Chapitre III

La mobilité dans les systèmes multi agents

Chapitre III

La mobilité dans les systèmes multi agents

III.1 Introduction

Le modèle client/serveur est certainement le modèle le plus utilisé pour la construction d'applications réparties, il a été conçu pour s'appliquer à des environnements de taille limitée. Cependant la généralisation des réseaux à grande échelle a changé ce contexte. D'une part, la taille du domaine potentiel d'exécution des applications réparties s'est accrue considérablement. d'autre part les interactions avec des sites inconnus n'appartenant pas au domaine d'administration des utilisateurs sont de plus en plus fréquentes (applications de commerce électronique ou de recherche d'information) [IsBe02].

Récemment, la recherche en systèmes répartis a vu l'émergence d'un nouveau modèle pour la structuration d'applications réparties : la programmation par agents mobiles [Fug&98]. Les agents mobiles visent principalement des applications réparties sur des réseaux à grande distance, car ils permettent de déplacer l'exécution vers les serveurs et de diminuer ainsi le coût d'accès à ces serveurs. Ceci constitue des avantages pour l'ACI qu'on essayera de les utilisés pour le développement de notre architecture.

Les agents mobiles empruntent au domaine de l'Intelligence Artificielle les concepts d'autonomie, de réactivité et d'adaptation en y ajoutant celui de la mobilité.

III.2 Définition

Un Agent mobile est un programme qui représente l'utilisateur dans le réseau, qui peut migrer d'un nœud à un autre d'une façon autonome [Fug&98]. (De manière analogue à une activité humaine dans la journée : visiter des places, utiliser des services et se déplacer).

Selon [IsBe02], l'agent migrant sur un site devient une entité autonome, qui peut suivant l'exécution du code, se déplacer sur un autre site, accumuler des résultats, communiquer avec d'autres agents, signaler un évènement, etc.., sans qu'une connexion permanente soit maintenue avec le site initiateur. L'objectif est de naviguer en fonction de l'application.

Nous définissons un agent mobile comme étant un programme informatique répondant à la définition d'un agent (notamment les propriétés d'autonomie, communication, activité et réactivité) et capable de se déplacer d'un site à un autre dans un réseau pour accomplir sa tâche.

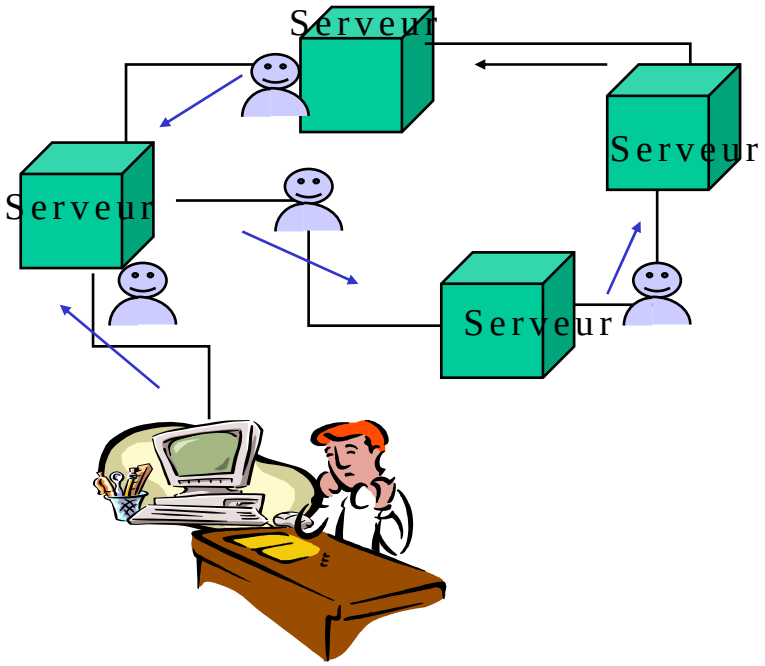


Fig.III.1. Agent mobile

Dans un modèle à agents mobiles, le client n'exécute plus toutes les taches, il délègue le travail. On considère ici l'agent mobile comme un ensemble code/données autonome et capable de se déplacer entre les différents environnements d'exécution des machines hôtes d'un réseau (Fig.III.2).

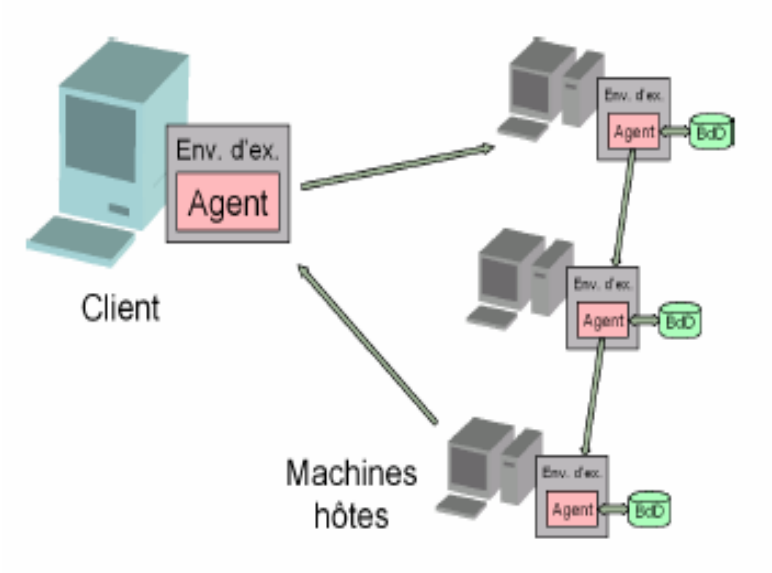


Fig.III.2. Modèle à agent mobile

III.3 Évaluation des performances des agents mobiles

Pour illustrer l'utilité du paradigme des agents mobiles, plusieurs comparaisons ont été faites dans la littérature avec le modèle Client/Serveur, mais ce n'est pas évident de montrer quel est le paradigme le plus efficace et peut-il remplacer l'autre car ça dépend du domaine d'application mis en œuvre. L'utilité de cette section est de montrer dans quelles conditions le modèle des agents mobiles est plus efficace que celui du Client/Serveur sur une application de recherche d'information.

III.3.1 le modèle Client/Serveur

Grâce à la généralisation d'Internet, tout le monde connaît le modèle client/serveur. Vous demandez une page HTML, un serveur vous l'envoie (voir figure 1-a). Le serveur peut aussi *calculer* votre page. Par exemple, il peut exécuter un script CGI ou un script PHP, il consulte une base de données; puis, retourne la liste des valeurs compatibles avec la question.

Cette solution présente encore d'importantes contraintes. La machine cliente doit être connectée pendant toute la recherche, elle se charge de tous les calculs et le trafic sur le réseau est maximal : la requête est envoyée à chaque fois et les serveurs renvoient tous leur page complète de résultats.

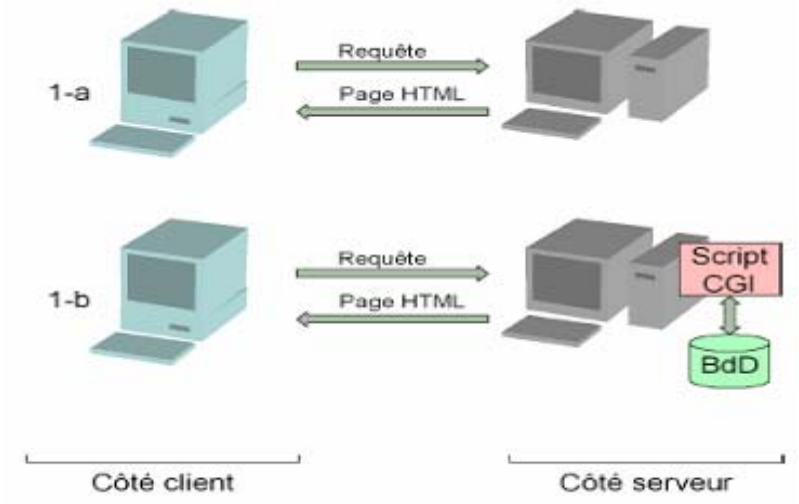


Fig.III.3. Modèle Client/Serveur.

III.3.2 L'application [IsHa00]

Une des applications les plus importantes dans le domaine des agents mobiles est la recherche d'information. Dans ces applications, des agents se déplacent sur différents sites pour chercher des informations pour leurs clients. L'exemple [IsHa00] est une application répartie sur Internet dont le but est de chercher une liste d'hôtel dans une ville. Dans notre scénario, deux bases de données doivent être consultées. La première recense des hôtels et permet d'obtenir la liste des hôtels de la ville où le client désire se rendre. La deuxième base de données est un annuaire permettant d'obtenir les numéros de téléphone de ces hôtels. Ces deux bases de données sont gérées sur des sites différents par des administrations.

Ces deux serveurs fournissent chacun une interface correspondant au service qu'ils offrent. Pour le premier serveur l'interface permet au client de donner le nom de la ville, le serveur lui retournant une table des hôtels dans cette ville. Pour le deuxième serveur, l'interface permet au client de donner un nom, le serveur retournant le numéro de téléphone associé à ce nom.

En utilisant le modèle client/serveur classique, le client va devoir faire un appel à distance pour interroger le premier serveur et récupérer la table des hôtels qui l'intéressent. A partir de cette table, le client va ensuite, pour chaque hôtel dans la table, réaliser un appel à distance au second serveur afin de récupérer le numéro de téléphone de l'hôtel.
Ainsi, dans le mode client/serveur, le client doit appeler le second serveur (B) autant de fois qu'il y a d'éléments dans le tableau d'hôtels retourné par le premier serveur (A). Le temps d'obtention de la réponse finale est égal au coût de communication avec le serveur A (rpc(A)), plus le coût des communications avec le serveur B qui est égal au coût de l'appel au serveur B multiplié par la taille du tableau obtenu comme réponse du serveur A (n*rpc(B)).
D'où un coût total : *coût total = rpc(A) + n*rpc(B)*

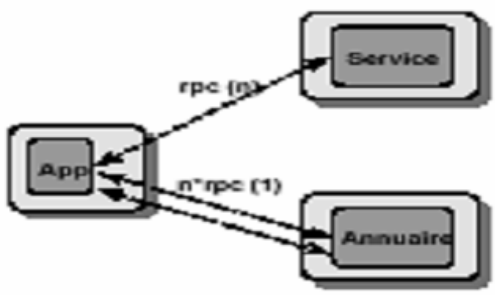


Fig.III.4. L'Application en Mode Client/Serveur.

Une seconde solution qui devient plus efficace lorsque le nombre d'hôtels est grand consiste à transmettre directement la table des hôtels du serveur A au serveur B afin que celui-ci réalise une jointure avec sa table des numéros de téléphone. Pour ce faire, le client crée un agent Ag contenant la requête globale à réaliser. Cet agent se déplace (1) tout d'abord sur le site du serveur A et réalise localement l'appel au serveur A pour obtenir la table des hôtels. La table des hôtels est stockée dans le contexte de l'agent, puis l'agent se déplace (2) vers le serveur B.

Sur le serveur B, l'agent peut itérer sur la table des hôtels et demander au serveur B le numéro de téléphone de chaque hôtel (cette étape correspond à la jointure). Ces numéros de téléphones sont stockés dans le contexte de l'agent qui se déplace (3) finalement vers son site d'origine où il délivre le résultat au client. Dans cette mise en œuvre avec des agents mobiles, un agent avec peu de données dans son contexte doit être envoyé sur le serveur A (coût A), puis l'agent se déplace avec la table des hôtels vers le serveur B (coût A(Thotel)), puis cet agent retourne à son site d'origine avec la table enrichie avec les numéros de téléphone (coût A(Ttel)). D'où un coût total de :
coût total = A + A(Thotel) + A(Ttel)

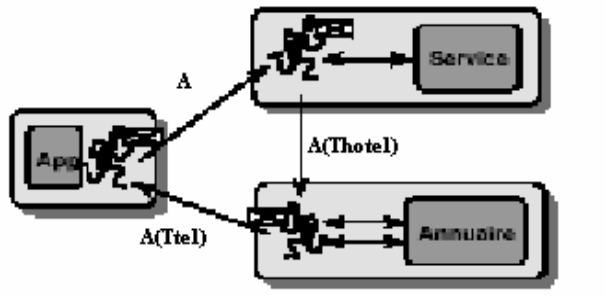


Fig.III.5. L'Application en Mode Agent mobile

En conséquence, la solution à base d'agents mobiles sera plus efficace si :

$$A + A(Thotel) + A(Ttel) < rpc(A) + n*rpc(B)$$

Pour évaluer cette application, voici les mesures prises dans [IsHa00] :

connexion	Latence(ms)	Débit(ko/s)
France - Angleterre	20	129
Angleterre - Suisse	16	280
France - suisse	26	123

Table.III.1. Capacité de l'environnement utilisé

La taille d'un enregistrement est de 80 octets, et les mesures sont effectuées avec le système Aglets. Les résultats sont reproduits sur la figure III.6.
Nous observons que pour un nombre d'enregistrements retournés petit (moins de 20 enregistrements), l'implantation basée sur RMI (qui implante sur Java le modèle Client/Serveur) est plus efficace que celles basée sur des agents mobiles. Ceci s'explique par le fait que pour un nombre d'enregistrements petit, le nombre d'appel à distance économisé n'est pas suffisant pour amortir le coût de la migration de l'agent sur le réseau. Cependant, pour un nombre d'enregistrements suffisant, la solution à base d'agent est bien plus efficace.

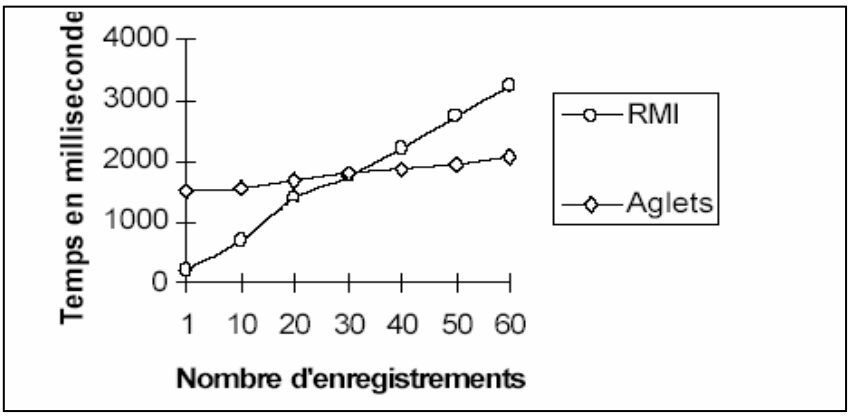


Fig.III.6. Comparaison entre RMI et les agents mobiles pour l'application

En conclusion, Les résultats de l'évaluation montrent que les agents mobiles peuvent amener des gains d'efficacité significatifs lorsqu'ils permettent de transformer les communications à distance en communications locales ou de réduire le volume d'information transféré sur le réseau. Ces gains sont d'autant plus importants que le réseau est lent, car cela permet alors d'amortir plus rapidement le coût du déplacement de l'agent.

III.4 Infrastructure système

La mise en œuvre du code mobile suppose, sur chaque site susceptible d'accueillir des agents mobiles, un support système pour l'exécution, la communication, la migration, l'accès au ressources externes et la prise en compte de la sécurité [Rod&01]. Il existe beaucoup de système à agents mobiles (S.A.M.) mais le problème de l'interopérabilité entre ces différents S.A.M. se pose, c'est pour cela que différentes recherches se poursuivent dans ce domaine. A.S. Rodrigues et al. [Rod&01] ont proposé une architecture générique et globale d'un S.A.M (Fig.III.7), qui sert comme un modèle de référence pour la comparaison et pour la standardisation des plates formes existantes.

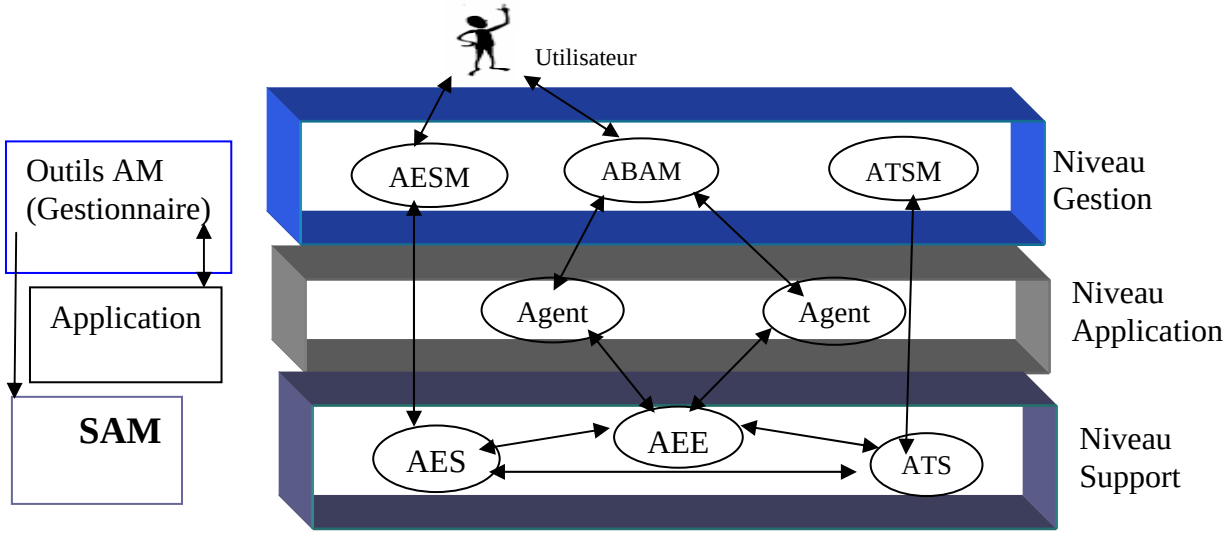


Fig.III.7. Architecture Globale d'un S.A.M. [Rod&01]

Cette architecture a trois niveaux et se compose des éléments suivants :

AES (Agent Execution System): fournit un support pour la communication la migration, la sécurité et la persistance. Il a une BD pour garder les traces de ses ressources internes (agents, places…)
ATS (Agent Type System): garde dans sa BD les classe d'agent, et fournit l'accès aux interfaces.
AEE (Agent Execution Environment): fournit l'environnement pour l'exécution des agents, c'est un (Interpréteur, ou VM) du langage utilisé.
AESM (AES Manager): à travers une interface utilisateur, il configure et gère, surveille et donne les info à AES.
ATSM (ATS Manager): fournit toute les info à l'ATS.
ABAM (Agent Based Application Manager): à travers une interface utilisateur, il surveille tous les agents, notamment les capacités de créer, suspendre, d'envoie de message et de déplacer sur les places.

III.4.1 Support d'exécution

Un environnement d'agents mobiles fournit une interface de programmation, et un environnement d'exécution offrant des primitives pour créer, lancer et obtenir des résultats des calculs effectués pas les agents. Un environnement d'exécution d'agents mobiles est constitué d'un ensemble de programmes statiques, appelés places, s'exécutant sur les sites des systèmes susceptibles d'accueillir des agents. Les places sont des programmes qui fournissent aux agents l'infrastructure de base pour leurs exécution et leurs permet de se déplacer.

Les environnements d'exécutions offrent plusieurs ressources permettant l'exécution de chaque agent sur un site, ces ressources sont par exemple : mémoire, temps CPU, espace disque, une voie de communication…etc. L'AEE est le responsable de l'exécution ou de l'interprétation du code de l'agent, et que les langages interprétés (directement tel que TCL, Perl ou indirectement tel que Java) sont préférés aux langage compilés (tel que C) pour des raisons de portabilité sur des machines hétérogènes et de meilleur contrôle de l'utilisation des ressources locales.

III.4.2 Gestion des types d'agent

Un agent est une instance de son type, il est nécessaire de comprendre comment l'agent a été *instancié* de sa classe. Une instance d'agent mobile comporte trois parties : des données, du code et un contexte d'exécution. Les données sont des valeurs des paramètres définit par le type d'agent, le code met en œuvre les primitives d'accès aux valeurs des paramètres, le contexte d'exécution reflète l'état d'exécution courant de l'agent mobile (valeurs des registres, pile d'exécution).

La gestion de la notion type/classe d'agent est relative au type de la programmation, nous remarquons que cette notion n'existe pas dans les langages de programmation non orientée objet (TCL), par contre dans les langages orienté objet tel que Java, les classe sont stockés dans un fichier système local ou distant et gérer par l'ATS.

III.4.3 Gestion des identités

L'interaction entre les agents soulève la question suivante:" comment l'agent peut identifier et référencer lui-même, les ressources ainsi que la place où il se trouve ?".

1. type d'identificateur : une ressource peut être identifié par différentes voies : par adresse électronique ou par un nom, l'adresse correspond à l'accès à la ressource (contient l'adresse électronique du nœud où l'objet vie). Le nom est un identificateur alphanumérique utilisé pour caractériser la ressource, ce nom est converti ensuite en adresse par un service spécialisé.

2. modèle de gestion : il y a deux modèles de gestion de l'espace adresse/nom : linéaire ou hiérarchique. Dans le modèle de gestion linéaire, chaque ressource est associé par un identificateur unique globale indépendant de la place où la ressource existe, il y a un seul et unique responsable service/serveur pour l'attribution d'une adresse unique, cette approche est adéquate pour les applications qui a un petit nombre ressources ou un niveau faible de distribution. Dans le modèle hiérarchique il y a une hiérarchie de domaine et de ressources, chaque un est géré par son serveur, une adresse est composée d'une séquence d'éléments, chaque élément représente une

sous adresse dans l'hiérarchie, cette approche est bonne pour les systèmes à agents mobiles complexes contenant beaucoup de ressources ou bien un niveau fort de distribution.

III.4.4 Migration

Tous langage de programmation de code mobile fournit une primitive de migration volontaire de l'agent, C'est l'agent qui détermine quand, et où, se déplacer (suivant une liste statique ou un résultat d'exécution) [IsBe02] [Fug&98].

La migration est un mécanisme qui vise à transférer un agent d'un site à un autre pour qu'il y poursuive son exécution. Deux type de mécanisme de migration ont été proposés dans les environnement d'agents mobiles : la migration faible et la migration forte.

III.4.4.1 Migration forte

Pour une migration forte, le contexte d'exécution d'un agent est déplacé en même temps que son code et ses données, vers la place de destination. L'environnement D'agent, Utilise TCL et fournit la primitive de migration *agent-Jump*. Celle ci capture l'état interne de l'agent et envoie une image de cet état au site de destination, ce dernier lorsqu'il reçoit l'état exécute un interprète TCL, qui restaure l'agent à partir des données reçues et relance l'exécution de l'agent[Sah99]

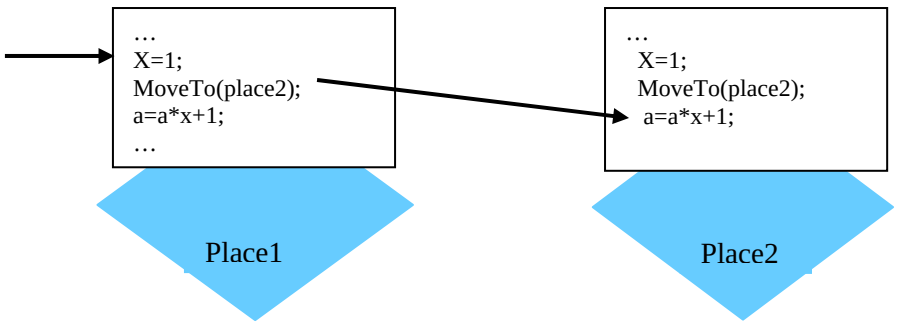


Fig.III.8. Migration forte

III.4.4.2 Migration faible

La migration faible d'un agent, consiste à faire migrer uniquement le code et ses données. C'est le cas de la majorité des S.A.M développés dans le langage Java comme Mole ou Aglets. Ils se servent du mécanisme de sérialisation fourni par Java (Java Object Serialization). Les programmes Java sont compilés dans un langage intermédiaire appelé bytecode. Quand un agent migre c'est sont bytecode associé qui est transféré entre les places.

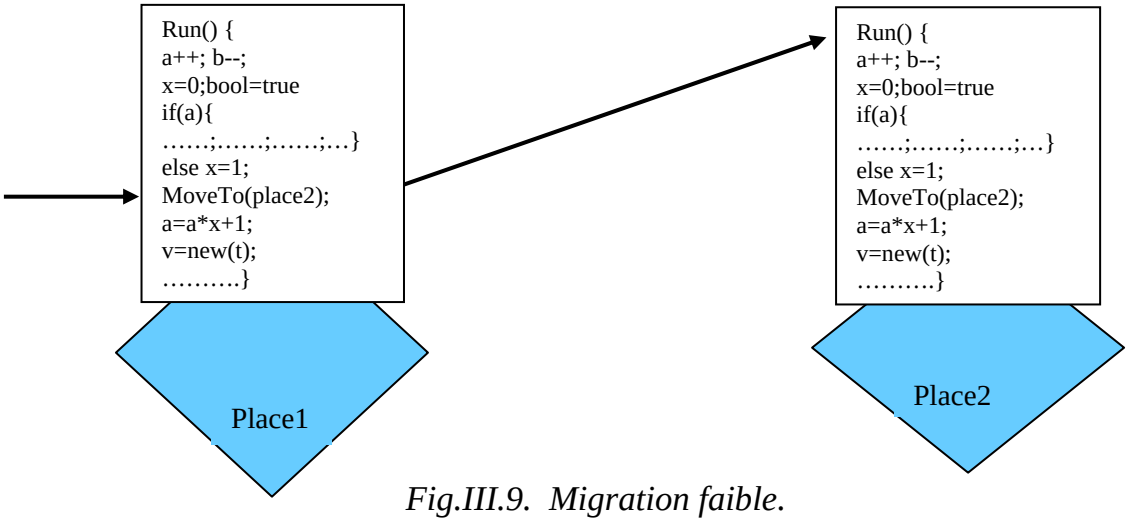


Fig.III.9. Migration faible.

III.4.5 Communication

Un agent a besoin de communiquer soit avec son propriétaire, avec d'autre agents ou avec des services.
La communication est synchrone lorsque les deux parties invoquent eux mêmes le synchronisme avant de transfert des données. Ce genre de communication est adéquat quand les données transférées requièrent une confirmation urgente ou bien un dialogue, à part ces situations, la communication est asynchrone, et elle peut être locale lorsque les agents se trouvent dans la même place ou bien des places distantes.

L'interaction entre les agents peut être directe ou indirecte. Dans l'interaction directe, l'agent appelle les méthodes des autres explicitement. Si elle n'est pas limitée par des mécanismes de sécurité additionnels, quand l'interaction est indirecte, les agents ne se communiquent pas directement. Ils emploient plutôt, un service d'intermédiation qui offre généralement des services ajoutés (tel que Object Request Brokers (ORB) ou bien un service partagé d'information) [Rod&01]. En général quand un agent veut interagir avec d'autres agents, il suit un schéma de communication point à point. Cependant il existe des situations où un agent communique avec groupe d'agents.
Le problème qui se pose essentiellement lors de l'interaction entre agents est : Comment maintenir la communication lorsque un agent migre? C'est à dire que si un agent Ag2 communique avec l'agent Ag1, que se passe t'il si lorsque Ag2 se déplace de la place 1 à la place 2 (Fig.III.10) ?

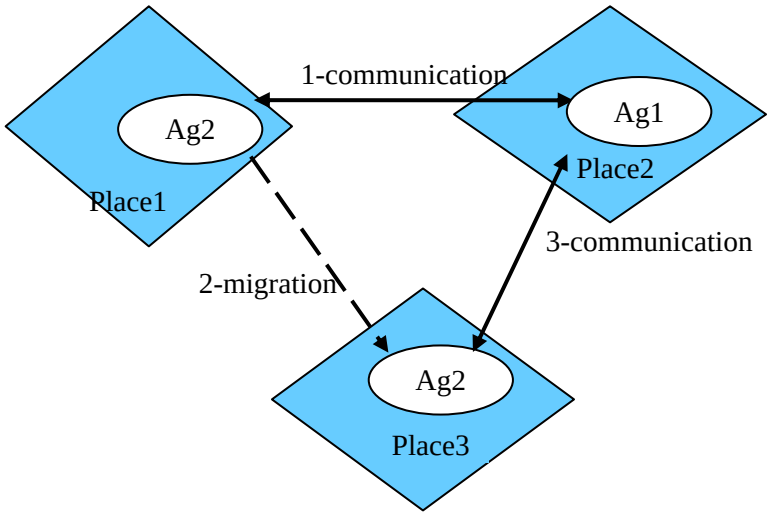


Fig.III.10. Migration lors d'une communication.

Deux approches ont été proposées pour résoudre ce problème [Rod&01] :

- La première consiste à fermer le canal de communication par le système lors de la migration de l'agent, en gardant des informations suffisantes sur son interlocuteur. Dès l'arrivé de l'agent migrant à la place désirée, il peut rétablir la communication.
- La deuxième approche consiste à tenir le canal de communication ouvert, cette approche est implémentée par l'utilisation d'un agent spécialisé appelé *proxy-agent* responsable de la communication, c'est à dire que toutes les communication du systèmes sont indirecte et via cet agent proxy. Lorsque l'agent migre sur un autre nœud, il communique à l'agent proxy un message lui donnant sa nouvelle adresse, ce dernier rétabli la communication entre eux d'une manière transparente vis à vis du programmeur.

III.4.6 Accès aux ressources externes

Les agents mobiles peuvent être amenés à effectuer des entrées/sorties, accéder à l'interface utilisateur, au système de gestion de fichier, base de données. C'est la responsabilité du S.A.M. de fournir aux agents l'accès aux ressources existantes. Pour cet accès on trouve deux approches dans les S.A.M.[Rod&01] :

- Les agents mobiles ont un rôle limité, ils ne peuvent pas interagir directement avec les ressources, mais via des agents stationnaires, ces derniers sont responsables d'explorer les capacités des différentes ressources, ils deviennent des médiateurs entre les agents mobiles et les ressources.
- Les systèmes d'agents mobiles fournissent des API's nécessaires aux agents mobiles pour accéder aux ressources, ceci rend les agents plus libres et versatiles, et peut engendrer des problèmes de sécurité, c'est pour cela que cette approche n'est jamais utilisée dans les environnements hétérogènes et ouverts.

III.4.7 Sécurité

La sécurité était toujours une question critique dans l'exécution des agents mobiles. Le site d'accueil doit se protéger contre la sur-utilisation des ressources et les actions malveillantes des agents accueillis. Les aspects suivants doivent être considérés [IsBe02] [Rod&01]:

- Authentification et contrôle d'accès : le site d'accueil doit vérifier l'identification de l'agent mobile ainsi son propriétaire.
- Intégrité : le site d'accueil peut changer les buts et les comportements superflu et non souhaité des agents.
- Contrôle des ressources à consommer : le site d'accueil limite la consommation des ressources (temps CPU, mémoire, système de fichier), et peut donner des permissions différentes aux agents dont le code provient d'un fichier local et aux agents dont le code est reçu via le réseau.

III.5 Analyse de quelques systèmes d'agents mobiles

De nombreux systèmes d'agents mobiles sont aujourd'hui accessibles, une liste complète de ces systèmes est disponible sur http://www.cetus-links.org/.
Dans cette section, nous analysons et nous comparons quelques systèmes à agents mobiles selon les différents aspects que nous avons abordés auparavant (Tab.III.1 et Tab.III.2), ces systèmes sont choisis en raison de leurs maturités.

III.5.1 Telescript

Telescript [Whi97] est développé par la société General Magic, c'est un produit commerciale inspiré du paradigme du marché électronique, composé par deux types de participants, clients et fournisseurs, les clients envois leurs agents mobiles pour visiter un ensemble de places pour trouver des produits.
Telescript est composé principalement de :

- Un langage de programmation (appelé aussi Telescript) orienté objet similaire à Smalltalk, conçu pour le développement d'un grand nombre d'applications distribuées.
- Un système d'exécution appelé ''engine'', exécute le langage Telescript, supporte et exécute les ressources (agent, place), et gère ces ressources via trois API :gestion de persistance, transport d'agent et accès au ressources externes.

Telescript est une plate forme avancée dans différents critères tel que la sécurité, mobilité, persistance et la communication, cependant elle n'est pas construite pour des environnements ouverts, par exemple la gestion des adresses n'est pas basés sur des standards Internet, et que c'est impossible d'ajouter des nouveaux agents avec de nouvelles interfaces [Rod&01].

III.5.2 Aglets

Aglets[LaOs98] est une contraction du mot ''Agent'' et ''Applet '', développé par IBM Tokyo, à un environnement de programmation et d'exécution d'agents mobile écrits en langage Java, l'environnement comporte un ensemble de classe Java pour le support de la mobilité et de la communication. Un Aglet est un objet java disposant des méthodes de ces classes. Il s'exécute dans une machine virtuelle d'exécution Java (JVM). La plate forme se compose de : [Rod&01]

- Java Aglets API : contient un ensemble de classes et interfaces qui permettent la construction d'agents et d'applications basés sur ces agents.
- Serveur Aglets : est une application Java qui comporte :
    1. serveur ATP(Agent Transport Protocol) qui supporte les opérations de migration et de communication des agents.
    2. un contexte d'exécution d'Aglets.
- Fiji : permet la création des Applets pour les Aglets existants, ce composant permet d'envoyé les Aglets à n'importe quel navigateur.

Aglets est l'un des systèmes le plus utilisé parmi ceux qui sont écrit en Java, il supporte la migration avec mobilité faible, ceci provient de l'architecture actuelle de la machine virtuelle Java. Le défaut d'Aglets est la lourdeur de programmation, due au langage lui-même et à l'impossibilité d'accéder à l'état d'exécution.

III.5.3 D'Agent

Appelé aussi Agent-Tcl [Gra95] est un système développé à l'université de Dartmouth au USA. D'Agent veux être complètement indépendante aux machines virtuelles et leurs langages de programmations, actuellement D'agent supporte en plus des agents écris en Tcl, ceux écris en Scheme, Java et C/C++.
Sur chaque machine susceptible d'accueillir ou de créer des agents, trois processus doivent être lancés. L'un exécutant l'interpréteur Tcl, le deuxième étant chargé de l'accueil des agents externes et le dernier du contrôle de l'utilisation des ressources.
La limitation de la consommation des ressources s'effectue par l'intermédiaire d'un fichier de configuration utilisé par les sites serveurs à l'initialisation du système. L'administrateur peut spécifier au niveau du serveur, le nombre maximum d'agents en exécution en parallèle, le nombre maximum de temps d'exécution, et le nombre maximum de migration [IsBe02].

Les deux tableaux suivants résument les caractéristiques de chaque système selon le modèle de référence défini par [Rod&01]:

Référence	Telescript	Aglets	D'Agent
Terminologie			
Agent	Mobile agent	Aglet	Agent
Place	Place	/	/
contexte	Place	contexte	serveur
Niveau Support			
AES	Engine	Agletsd+Tahiti	D'agent server
AEE	Engine	Java VM	TCL,Java,Sheme
Niveau Gestion			
AESM	/	Tahiti	AESM
AEEM	/	/	/
Langage de Programmation	Telescript	Java	N'importe
Primitive de migration	Send()	Run()	Agent-jump()

Tab.III.1.comparaison de concepts et de terminologies.

Référence	Telescript	Aglets	D'Agent
Gestion d'identité	Linéaire	hiérarchique	linéaire
Mobilité	Forte	faible	Forte
Communication			
Type	Sync.	Sync\Async	Async.
Locale	Locale\Distante	Locale\Distante	Locale\Distante
Broker	Directe	Directe\Indirecte	Indirecte
Utilisation d'un Proxy	Non	Non	Oui
Accès aux ressources externes	Via places (agents Stationnaires)	Via Java API	Via API
Sécurité			
Authentification	Oui	Oui	Oui
Intégrité	Oui	Non	Oui
Contrôle des ressources à consommer	Oui	Non	Oui
Intégration Web	Oui (Tabriz)	Oui.	Non

Tab.III.2.comparaison des caractéristiques techniques.

III.6. Conclusion

Les agents mobiles joueront un rôle important dans la conception des applications distribuées et dynamiques sur Internet, surtout celles dédiés à l'acquisition et à la recherche de l'information, car les agents mobiles visent principalement des applications réparties sur des réseaux à grande distance dans des environnements ouverts. Ils permettent aussi de déplacer l'exécution vers d'autres serveurs et de réduire ainsi le nombre d'interactions distantes d'une part, et de minimiser le volume de données transportées sur le réseau d'autre part.

Ces points nous sont apparus essentiels pour développer des applications en utilisant les agents mobiles dans le futur. Enfin, -selon notre point de vue - le concept des agents mobiles est important car il nous permet de lier, d'une manière naturelle, le niveau conceptuel d'un SMA doté d'une organisation dynamique à son implémentation distribuée.

Dans le chapitre suivant, nous utiliserons cet avantage des agents mobiles pour le développement de notre architecture multi agent pour l'Acquisition Coopérative de l'Information (ACI).

Chapitre IV

Un Modèle Organisationnel pour l'Acquisition Coopérative de l'Information

L'Internet est, après l'invention de l'écriture et celle de l'imprimerie, le troisième grand bouleversement de l'humanité dans le domaine de la communication."

Michel Serre

Chapitre IV

Un Modèle Organisationnel pour l'Acquisition Coopérative de l'Information

IV.1. Introduction.

Avec la généralisation d'Internet ces dix dernières années, une explosion en terme de quantité d'information sous forme électronique est aujourd'hui disponible sur le web. Le défi qui doit être relevé par la communauté des scientifiques est de développer des logiciels complexes basés sur des architectures de recherche d'informations qui prennent en considération la complexité de l'information elle même, car ces informations sont géographiquement dispersés, dynamiques et hétérogènes. En plus sur le Web, il n'y a pas d'entité centrale en charge de la gestion des informations, puisque n'importe quel de nous peut publier ces pages personnels (environnement dynamique).

C'est dans ce contexte que le thème de l'Acquisition Coopérative de l'Information (ACI) a vu le jour. L'ACI est une extension des systèmes de recherche d'informations classique, puisque la nature des sources d'informations (SI's) exploitées sont *distribuées* sur un ou plusieurs sites et ne sont pas *fixées ou connues a priori* par celui qui les interroge, alors que dans la Recherche d'Informations classique, l'utilisateur traite une collection d'information connu et bien défini. De plus, elles (les SI's) sont *Autonomes*, car conçues les unes indépendamment des autres, et sans contrôle global commun.

L'autonomie engendre plusieurs problèmes pour l'acquisition et la combinaison des informations provenant de ces sources et notamment : la dynamique, l'hétérogénéité et la redondance. Les sources d'informations sont *dynamiques* dans la mesure où elles peuvent être créées, modifiées ou supprimées à tout moment. Sur le plan conceptuel, les sources d'informations étant le plus souvent conçues de manière non concertée, leur langage et leur système de représentation et de manipulation peuvent être *hétérogènes*. On rencontre également le problème de *l'hétérogénéité sémantique* qui caractérise une situation où la même information est modélisée ou interprétée différemment selon les sources d'informations considérées. Enfin, l'accès à ces sources d'informations peut se faire selon *des protocoles hétérogènes*. Quant à *la redondance*, elle est fortement probable du fait de l'indépendance des sources, et l'information dupliquée n'a pas la même qualité (fréquence de rafraîchissement, précision, rapidité d'accès) selon les sources qui les diffusent [Tou02].

La figure suivante (Fig.IV.1) nous montre cet espace distribué de l'information où chaque serveur d'information détient q'une partie de donnée, alors que les autres peuvent contenir la même information (redondance) ou bien d'autres informations supplémentaire (complémentaire) dans différent domaines.

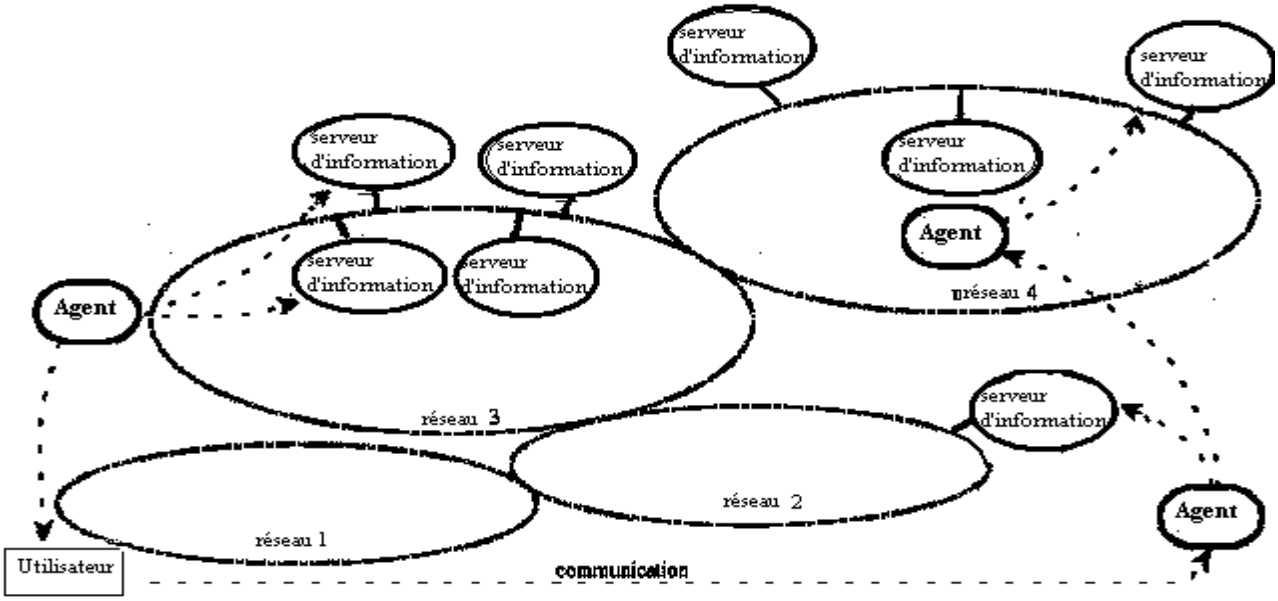


Fig.IV.1. Exemple d'un environnement distribué de recherche d'informations

Dans ce chapitre nous préciserons pourquoi les systèmes multi agents sont considérés comme une approche de résolution du problème de l'ACI et son apport dans le cas de notre travail, nous analysons ensuite dans la section suivante quelques systèmes qui nous sont apparus comme les plus représentatifs, enfin nous proposons une architecture d'un système d'ACI basés sur un modèle organisationnel que nous détaillons.

IV.2. Approche Systèmes Multi-Agents pour l'ACI

Les méthodes de recherche d'informations basées sur les bases de données (BD) sont réduites à une recherche par ''mots clés'' sur des environnements statiques de BD, qu'elles soient centralisées ou distribuées. En effet l'intégrité et l'autonomie de chaque source d'informations doivent être maintenues.

L'approche des systèmes multi-agents (SMA) s'avère intéressante à utiliser pour le problème d'Acquisition Coopérative de l'Information, puisque la notion de l'agent offre la flexibilité et l'autonomie exigée, en plus des raisons suivantes :

- L'environnement de recherche est un environnement distribué et hétérogène, ce qui correspond aux capacités des agents autonomes de différents types à se déplacer dans divers zones géographiquement éloignées.
- Plusieurs agents sont concurrents, ce qui offre un gain considérable en matière de temps lorsque l'espace de recherche est grand.
- L'acquisition d'information nécessite une coopération et une coordination lorsque l'environnement est distribué, d'où la technologie des agents est la mieux adaptée à ce problème, surtout lorsque la coopération engendre des conflits.

- lorsque le système se compose d'une grande quantité de données, l'accès au serveur où les données résident est plus intéressant que la migration d'une grande quantité de données, ce qui surcharge le trafic du réseau. Les agents mobiles peuvent migrés aux serveurs d'informations, faire une recherche locale et transmettent des sous ensembles d'informations au serveur source.

- le processus de résolution d'une requête nécessite la coopération entre plusieurs systèmes de recherche d'informations, et doit interagir avec l'utilisateur. Dans ce cadre, la recherche d'informations ainsi que le plan d'exécution sont vues comme une Résolution Distribuée de Problèmes (RDP), où un ensemble de sources d'informations se fédèrent dynamiquement pour répondre, et en coopération avec l'utilisateur, à un problème donné. L'exemple le plus cité dans les travaux est l'organisation d'un voyage qui nécessite la coopération de nombreuses sources d'informations pour connaître les vols, hôtels, prévisions météo, informations culturelles,…etc. L'une de l'approche la mieux adaptée pour la RDP est les SMA [Oat&97].

- Les architectures basées sur les SMA offrent une modularité, robustesse et autre avantage pour les systèmes distribués. un agent d'information peut créer un autre agent. En plus la source de données passive comme une base de données peut être transformées à des agents informationnels. [Dec&95]

IV.3. Systèmes d'Acquisition Coopérative d'Information

IV.3.1. RETSINA

RETSINA (**RE**usable **T**ask **S**tructure-based **I**ntelligent **N**etwork **A**gents) est une architecture multi-agents développée au sein de l'institut de robotique de l'université de Carnegie-Mellon en 1998 [Syc&98]. RETSINA est qualifiée d'infrastructure car elle constitue un système général qui peut être instancié par des applications concrètes et différentes. Ainsi, RETSINA a notamment permis le développement d'un ensemble d'agents logiciels coopérants de manière asynchrone pour la quête et pour l'intégration de prises de décisions variées, telles que l'aide à la décision dans les organisations, la gestion de porte-feuille d'actions, etc...[Dec&95]. L'architecture RETSINA est basée sur trois types d'agents [Syc&98] :

- **Des Agents interface** qui interagissent avec les utilisateurs en recueillant l'information minimum nécessaire pour initier la résolution du problème, ils fournissent et présentent les résultats, modélisent les préférences des utilisateurs et les utilisent pour guider le système à remplir ses tâches. Les principales fonctions de ces agents sont :
  – Collecter les spécifications de l'utilisateur ;
  – présenter à l'utilisateur les résultats et les explications sur ces résultats ;
  – demander à l'utilisateur, s'il y a lieu, des informations supplémentaires durant la procédure de résolution de problèmes ;
  – demander d'éventuelles confirmations à l'utilisateur.
- **Des agents de tâches** les agents de ce type s'occupent de l'aide à la décision en formulant et en exécutant des plans pour la résolution du problème visé. Ils ont une

bonne connaissance d'un domaine particulier et peuvent aider les autres agents sur ce domaine. Plus précisément, un Agent de tâches réalise les fonctions suivantes :
– il reçoit de l'agent interface les spécifications de l'utilisateur ;
– il interprète ces spécifications et en déduit des buts ;
– il construit des plans pour satisfaire ces buts ;
– il identifie les sous-buts de recherche d'informations présents dans ces plans ;
– il décompose les plans et se coordonne avec d'autres agents (de tâches, informationnels) pour les exécuter. Enfin, il compose les différents résultats produits par l'exécution de ces plans.

- **Des agents informationnels** qui fournissent un accès intelligent aux données hétérogènes et réparties. Un agent informationnel est capable d'extraire l'information pertinente, résoudre les conflits et faire la fusion de certaines données. Ces agents peuvent assurer trois types de services :
  – répondre à des questions de type « réponses immédiates » ;
  – répondre à des questions périodiques ;,
  – contrôler une source d'information pour informer l'utilisateur de changements intéressants.

Par ailleurs, chaque agent dispose de quatre modules: un module de communication et de coordination, un module de planification, un module d'ordonnancement, et un module d'exécution et de contrôle.

Le module de communication et de coordination permet à un agent de communiquer avec d'autres agents en utilisant le langage KQML. La communication peut avoir plusieurs objets :
– leur demander ou leur fournir des informations ;
– leur demander de rechercher une source d'information ;
– filtrer ou intégrer des informations ;
– négocier pour résoudre des conflits.

Le module de planification construit un plan satisfaisant les buts de l'utilisateur. Pour cela, il s'adresse à une librairie qui contient des structures de tâches indexées par les buts (que peut réaliser l'agent) et à une librairie qui contient des schémas de réduction des tâches. Dans ce schéma, une tâche est décomposée en sous-tâches jusqu'à un niveau de tâches élémentaires appelées actions.
Le module d'ordonnancement est basé sur des méthodes heuristiques et permet à l'agent d'ordonner les actions des plans fournis par son module de planification. Le module d'exécution et de contrôle exécute les actions. Enfin, il faut indiquer que des capacités d'apprentissage peuvent être ajoutées à chaque type d'agents.

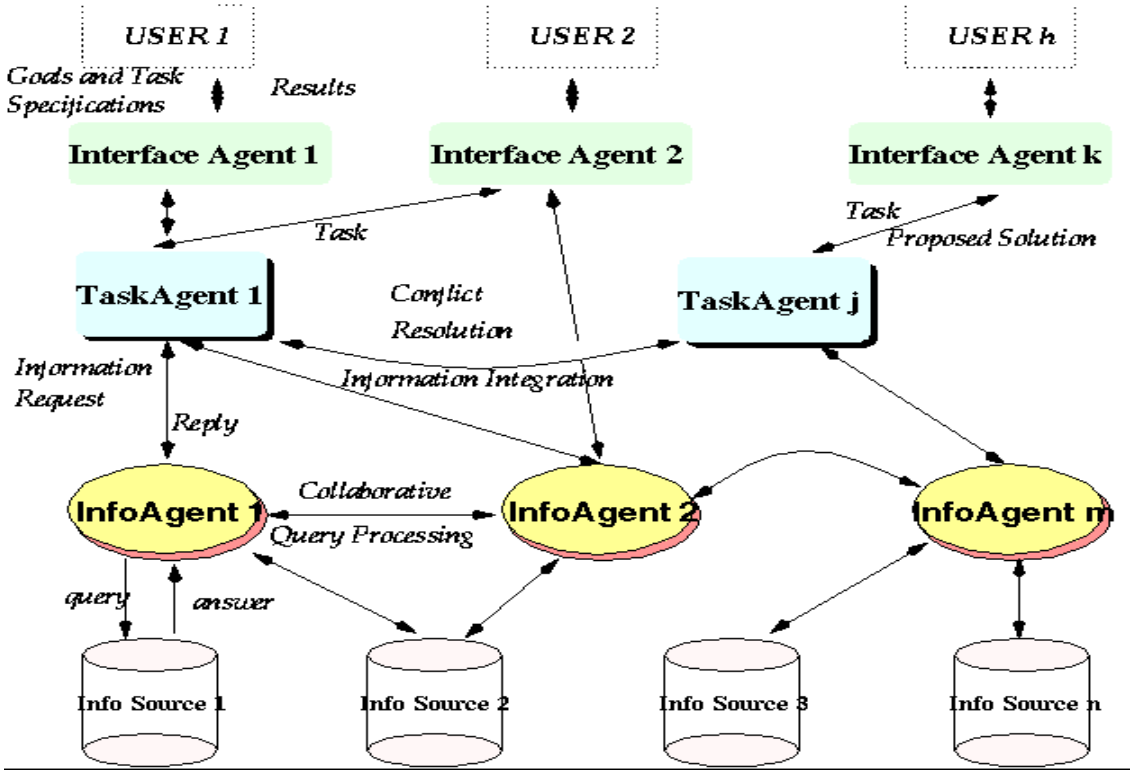


Fig.IV.2. Architecture RETSINA.

Lorsqu'un utilisateur soumet son problème, l'agent interface le transmet à un agent de tâches. Ce dernier se base sur la connaissance qu'il a du domaine pour décomposer le problème en sous-tâches ; il délègue ensuite ces sous-tâches à d'autres agents (de tâches ou informationnels). Pour localiser un agent capable de remplir une tâche, RETSINA demande les services d'un *matchmaker*.

IV.3.2. Le système MACRON

MACRON (**M**ulti-agent **A**rchitecture for **C**ooperative **R**etrieval **ON**line) est une architecture multi-agent pour l'Acquisition Coopérative d'Information. Il a été développé dans le département d'informatique de l'université du Massachusetts aux Etats-Unis en 1995 [Dec&95]. Les agents dans MACRON sont organisés en [Dec&95] :

1. unités fonctionnelles (Functional Units). Une unité fonctionnelle est une organisation à long terme, spécialisée dans un domaine, gérée par un Agent de Gestion Fonctionnelle, et composée d'un ensemble d'agents (appelés agents de raisonnement) fournissant des services relatifs à ce domaine.

2. unités de Questions-Réponses (Query-Answering Units). Chaque unité de Questions-Réponses est une organisation à court terme créée dynamiquement pour répondre à une question. Elle est gérée par un Agent de Gestion de Requêtes, et comprend un ensemble d'agents appartenant à différentes unités fonctionnelles. Chacun de ces agents a été recruté par l'Agent de Gestion Fonctionnelle de son unité fonctionnelle. Il faut préciser que les agents dans MACRON interagissent en utilisant le langage d'interaction KQML.

MACRON se base sur trois types d'agents (fig.IV.3) : des agents de gestion de requêtes (ou agents d'interface), des agents de recherche d'informations et des agents de raisonnement [Dec&95].

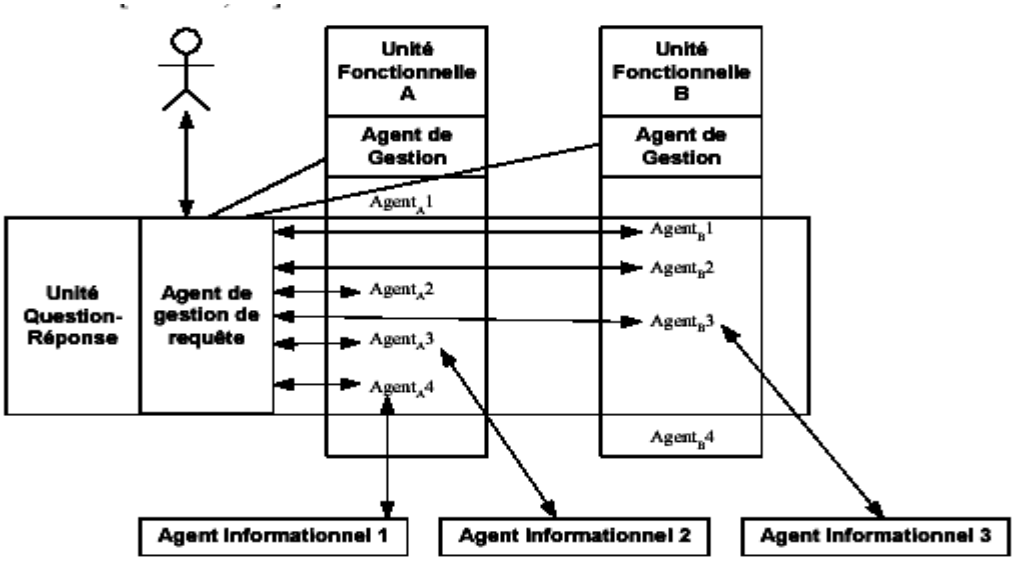


Fig.IV.3. Architecture du système MACRON [Dec&95]

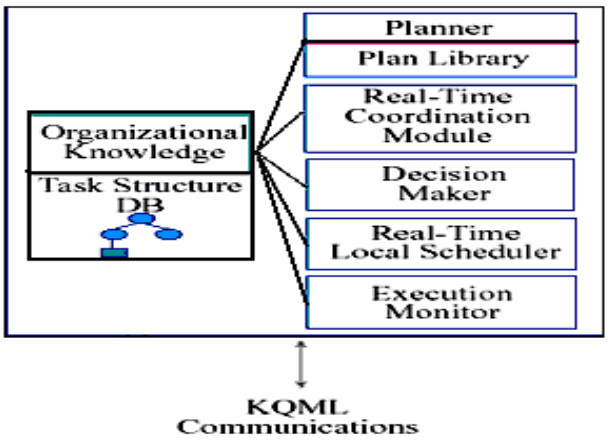


Fig.IV.4. Agent de raisonnement [Dec&95]

Un agent de raisonnement, qui détient le rôle central dans le processus de traitement d'une question, est constitué de plusieurs composants :

– une « base de données » qui contient la vue que l'agent a sur ses tâches et leurs liens avec les tâches des autres agents,

– un module de planification qui, basé sur « une base de données» de plans, instancie une structure de tâches pour répondre à une tâche donnée.

– un module de coordination qui, basé sur les liaisons entres les sous-tâches, peut proposer de nouvelles tâches, par exemple des coordinations avec d'autres agents,

– un module d'ordonnancement heuristique local qui prend en entrée une tâche et produit différentes possibilités d'ordonnancement pour ses sous-tâches,

– un module de prise de décision qui choisit en temps réel un ordre d'exécution de tâches parmi ceux fournis par le module d'ordonnancement,

– un module de contrôle d'exécution qui surveille l'exécution des tâches de l'agent.

Le traitement d'une question se déroule comme suit : Un utilisateur soumet son problème à un Agent de Gestion de Requêtes qui joue le rôle d'interface entre l'utilisateur et les autres agents du système. Ce dernier développe un plan (de tâches) pour l'acquisition d'informations, et choisit des unités fonctionnelles pour exécuter son plan. Il contacte, ensuite, l'agent de gestion de chacune de ces unités pour lui demander de recruter des agents de raisonnement. L'agent de gestion recrute les meilleurs agents fonctionnels de son unité capables de répondre aux exigences de l'Agent de Gestion de Requêtes. L'ensemble des agents fonctionnels recrutés compose ainsi une Unité de Question-Réponse. A son tour, chaque agent fonctionnel délègue la tâche de recherche d'informations à des agents informationnels de bas-niveau. Chacun de ces agents dispose de connaissances sur un domaine. Il connaît notamment quelques sites Web, ce qui lui permet de trouver et d'interpréter les URLs appropriées et d'interagir avec des moteurs de recherche d'informations et des bases de données. Ces connaissances permettent en plus, à l'ensemble des agents de recherche d'informations de fournir une abstraction du Web aux autres agents du système. En cours de résolution, l'Agent de Gestion de Requêtes garde le contrôle de l'exécution de ses plans, tient l'utilisateur au courant de l'état d'avancement du processus d'Acquisition et lui permet éventuellement de modifier les plans.

Pour que l'Agent de Gestion de Requêtes puisse sélectionner les unités fonctionnelles qui peuvent exécuter les tâches de son plan, il consulte le gestionnaire graphique d'organisation OCM (*Organizational Chart Manager*) qui est une mémoire organisationnelle contenant des informations sur ces unités.

Durant la procédure d'acquisition d'informations, l'agent interface affiche à l'utilisateur les tâches et sous-tâches en cours d'exécution. Ainsi, l'utilisateur peut sélectionner une des tâches (ou sous-tâches) pour obtenir plus de détails sur celle-ci ou pour éventuellement modifier ses paramètres.

En ce qui concerne les fondements théoriques, l'architecture MACRON n'en possède pas réellement. En revanche, elle se base sur des modèles et algorithmes reconnus pour traiter le problème essentiel de la planification distribuée et de la modélisation des tâches :

– La planification distribuée basée sur l'algorithme du GPGP [Dec92] (*Generalized Partial Global Planning, Planification Globale Partielle Généralisée*) permet de construire un plan global pour l'exécution d'une tâche en coordonnant les plans partiels fournis par différents agents pour cette exécution.

– La modélisation des tâches est réalisée par le modèle TAEMS qui permet de décrire sous forme de graphe la structure des tâches et leurs liens avec d'autres tâches.

IV.3.3. Le système InfoSleuth

InfoSleuth [Bay&97] a pris la relève du projet Carnot en 1995, développé par la compagnie Microelectronics and Computer Technology Corporation (MCC) au Etats Unis. Pour développer InfoSleuth, les chercheurs de la MCC ont utilisé des technologies standardisées comme: (a) KQML (Knowledge Query and Manipulation Language); (b) KIF (Knowledge Interface Format); (c) HTTP (Hyper Text Transfert Protocol); (d) Java. Ceci procure à InfoSleuth un haut niveau d'ouverture du système, de flexibilité et d'extensibilité.

L'architecture générale d'InfoSleuth est divisée en trois couches principales : la couche agent, la couche sémantique (ontologies) et la couche application : [Nod&98][Nod&00]

- ***La couche agent* :** La couche inférieure de l'architecture est la couche agent. Cette couche est composée d'agents coopératifs où les règles d'induction sont implantées en LISP, CLIPS, LDL++ et en Java. Ces agents annoncent d'abord leurs services et font par la suite des requêtes afin de trouver un agent pouvant répondre à un besoin spécifique. Un besoin peut être divisé en plusieurs sous-requêtes qui sont ensuite distribuées aux agents appropriés. Des réponses provenant de ces derniers sont finalement réunies par l'agent demandeur pour l'obtention de la réponse à la requête originale.
- ***La couche sémantique* :** Cette couche indique les agents coopératifs ayant le même vocabulaire et le même modèle sémantique qui sont capables d'interagir dans un domaine particulier. Par exemple, dans le cas où le travail se fait sur le domaine de matériels et de logiciel informatique, alors les agents aptes à communiquer entre eux sont seulement ceux utilisant des termes propres à l'informatique, ces termes constituent une ontologie pour le domaine de l'informatique.
- ***La couche application* :** Cette couche couvre trois régions principales soit : (1) la recherche de ressources (par exemple trouver les agents et les bases de données relatives au domaine de l'ontologie), (2) la recherche et l'analyse de gabarits (par exemple par data-mining), (3) la collaboration, la recherche et l'exécution.

Ces régions s'inter relient et se chevauchent. D'un point de vue architecture, le cœur d'InfoSleuth peut être vu comme un nuage d'agents (Voir Fig.IV.5). L'usager envoie des requêtes et reçoit des réponses de ce nuage par l'intermédiaire d'un agent utilisateur. Ce nuage puise ces informations des agents ressource qui communiquent directement avec des bases de données.
La communication entre agents se fait à l'aide du protocole KQML en utilisant le format KIF dans son champ contenu.

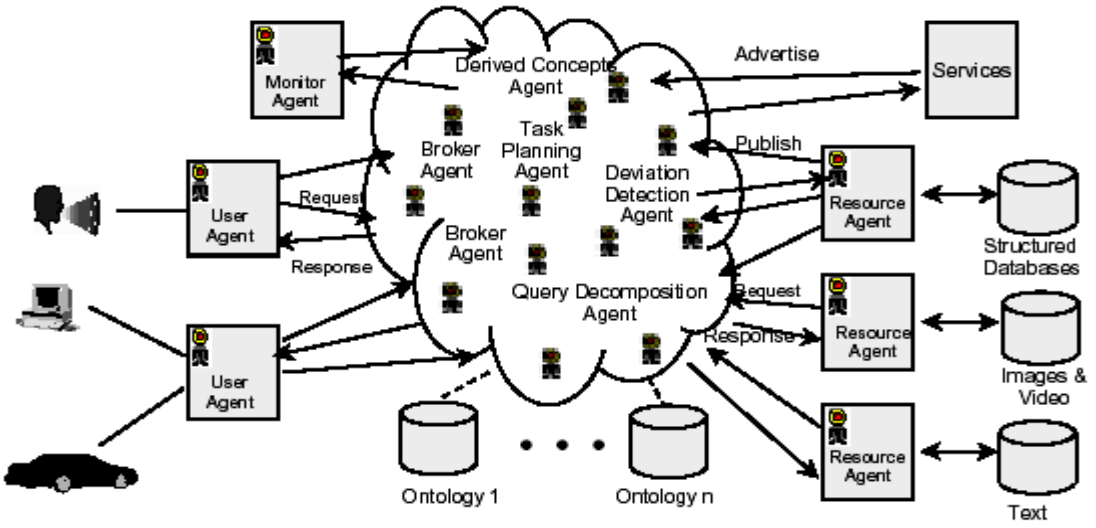


Fig.IV.5. Le système InfoSleuth [Nod&98]

IV.4. Notre Modèle Organisationnel pour l'ACI

Dans cette section, on présentera une architecture fonctionnelle pour l'ACI qui se caractérise par un modèle organisationnel, ce dernier est composé d'un modèle informationnel (basé sur les ontologies) et un modèle de rôle qui est celui proposé dans GAIA qu'on détaillera.

IV.4.1. Modèle Informationnel

IV.4.1.1 Utilités du modèle

Le Modèle Informationnel décrit la structure des informations multi-domaines de l'univers de discours. Il doit permettre un accès homogène aux multiples sources d'informations susceptibles de contenir des informations relatives à ces domaines, la construction d'un tel modèle est utile pour les raisons suivantes :

1. il fournit aux utilisateurs le vocabulaire utile pour formuler leurs questions ;
2. il permet au système de décomposer la question initiale de l'utilisateur en sous question à traiter parallèlement ;
3. il fixe le vocabulaire des messages que s'échangent les agents composant le système ;
4. il prend en compte le problème de l'hétérogénéité sémantique des informations. Cette hétérogénéité s'exprime par le fait qu'un même terme peut être représenté par différentes définitions dans des sources d'informations différentes.

IV.4.1.2 Solution adoptée

Parmi les solutions existantes pour faire face à ce problème, nous choisissons une solution inspirée des Bases de Données fédérées. Elle consiste à définir un modèle sémantique, commun à toutes les sources d'informations et indépendant de leur contenu, comme celui du système InfoSleuth [Bay&97]. Ce modèle représente l'univers de discours du système, c'est à dire un ensemble partagé de vocabulaires à communiquer pour l'ensemble des domaines traités par le système d'ACI. Cette définition du modèle sémantique commun est basée sur la notion d'ontologie.

IV.4.1.3 Définition

La plupart des travaux citant les ontologies, prennent la définition de T.R Gruber [Gru93] : « Une ontologie est une spécification formelle et explicite d'une conceptualisation partagée » cette conceptualisation est : « une vue abstraite et simplifiée du monde à représenté ».
M.Stula et al.[Stu&01] précisent cette définition par : « la Conceptualisation définit la description des entités, attributs et relations existantes dans une place, c'est une vue simplifiée d'une partie du monde que nous allons représenté, cette partie du monde est généralement appelé domaine ».

Les ontologies sont donc des représentations formelles de connaissances d'un domaine sous forme d'un réseau conceptuel. Elles sont considérées aujourd'hui un des moyens d'assurer la communication entre des agents logiciels. Tous les agents emploient des ontologies pour communiquer entre eux.

Nous représentons plusieurs types d'information dans une ontologie :

- entités, attributs et relations ;
- relations entre classes et sous-classe ;
- références standard aux lexiques ;
- contraintes sur des types, des cardinalités et des valeurs d'attribut ;
- entités fondamentales calculables.

Des travaux plus récents utilisent plusieurs ontologies -une par domaine- au lieu d'une seul globale car [Tou03]:

- Ces ontologies sont plus faciles à construire et à maintenir :
  - elles sont de taille réduite,
  - une ontologie modélise un seul domaine, et peut être construite par un expert de ce domaine.
- Elles sont plus faciles à utiliser par un utilisateur, car décomposées par domaine.
- Enfin, cette approche facilite l'utilisation d'ontologies préexistantes.

Notre approche consiste à utiliser plusieurs ontologies, dont chacune représente un modèle sémantique d'un domaine spécifique, en ajoutant un rôle qui a la responsabilité de faire partagé les ontologies aux différents rôles qui l'ont besoin, en leur fournissant les adresses nécessaire.

IV.4.1.4 Exemple

Nous rappelons le destin de la navette spatiale Mars Climate Orbiter de la NASA qui s'est brisé dans l'atmosphère de Mars le 23 Septembre 1999 [Nasa]. La navette spatiale devait être insérée dans l'orbite autour de Mars, le 23 septembre 1999 après un voyage de 9 mois et demi. Malheureusement, un manque d'identifier une erreur dans un échange d'information entre l'équipe de vaisseau spatial dans le Colorado et l'une des équipes de navigation de mission en Californie a mené à la perte du vaisseau spatial. Les résultats préliminaires d'examen indiquent qu'une équipe a employé les unités anglaises (par exemple, pouces, pieds et livres) tandis que les autres ont utilisées des unités métriques pour une opération principale de vaisseau spatial. Cette information était critique aux manœuvres exigées pour placer le vaisseau spatial dans l'orbite appropriée de Mars, donc au lieu d'établir un orbite à une altitude de 140Km, il a fait ainsi à 60Km, l'entraînant à se brûler dans l'atmosphère martien.

Cet exemple illustre l'importance des ontologies dans la communication pour un système donné. Les systèmes qui communiquent et travaillent ensemble doivent partager une même ontologie, qui représente le même modèle fondamentale en terme d'objets, d'attributs et de relations, d'assigner des symboles, des limites, des règles et des contraintes pour ce rapporter à ces objets.

IV.4.2. Modèle de rôle

Dans les systèmes multi-agents, les spécifications les plus complètes de la notion de rôle sont celles proposées dans [Woo&00]. Ainsi nous allons nous inspirer de la notion de rôle de GAIA, où chaque rôle est défini par un ensemble de responsabilité (devoir) et un ensemble de ressource (droit) accessible aux rôles pour exercer leurs responsabilités (voir

Chapitre II). Cependant on raffinera la méthodologie présentée dans (§4.4 du chapitre II), par des diagrammes de séquence UML pour chaque rôle pour décrire le fonctionnement d'un rôle (Liveness proprieties). Ces derniers sont représentés par des enchaînements d'actions internes (activités) et d'actions externes (protocoles d'interactions entre ces rôles).

IV.4.2.1 Identification des rôles pour l'ACI

Étant donné qu'un rôle représente une fonction abstraite d'un système d'acquisition coopérative de l'information, nous allons identifier les rôles qu'on peut trouver dans un tel système. Leurs spécifications formelles à l'aide des diagrammes de séquence UML seront données au paragraphe 4.3

IV.4.2.1.1 Le rôle Interface

Dans un système d'acquisition d'information, il est nécessaire que l'utilisateur qui compose une requête de coopérer avec le système, le rôle interface doit permettre à l'utilisateur de soumettre un problème au système et de lui fournir les résultats de l'ACI.

IV.4.2.1.2 Le rôle Broker

Ce rôle à une particularité complexe dans le système, il assure plusieurs fonctions :

- Décomposer la requête du rôle *interface* en plusieurs tâches élémentaires,
- Analyser, filtrer et combiner les résultats retournés,
- Composer le résultat final.

IV.4.2.1.3 Le rôle Gestionnaire

Il permet de :

- créer les rôles *migrateurs* selon les tâches disponibles retournées pas le rôle *broker*,
- gérer l'échange d'information des rôles *migrateurs*, en tend qu'un Proxy,
- et de gérer l'exécution des tâches, car ces taches sont interdépendantes.

IV.4.2.1.4 Le rôle Migrateur

Ce rôle est utilisé dans le processus d'acquisition coopérative d'information (ACI), il migre d'un serveur d'information à un autre pour rechercher de l'information. Il reçoit les informations nécessaires à son exécution de la part du gestionnaire. Ainsi il communique avec d'autre rôle *migrateur* par le biais du rôle *gestionnaire* et avec les rôles *traducteurs* de chaque serveur visité.

IV.4.2.1.5 Le rôle Matchmaker

Chaque système d'ACI qui a la particularité d'offrir un service sémantique utilise la notion d'un *Matchmaker*. Un *Matchmaker* est un courtier qui joue le rôle des "pages jaunes ou d'annuaire" : il fournit à ses clients les adresses des fournisseurs capables de répondre aux services demandés. Dans notre cas il fournit au rôle broker les adresses d'ontologies correspondantes au domaine demander, et il fournit aussi aux rôles *migrateurs* les adresses des sources d'informations où l'information à rechercher est susceptible d'être.

IV.4.2.1.6 Le rôle Traducteur

Le rôle *traducteur* est utilisé dans le système pour faire face à l'hétérogénéité, car les sources d'informations utilisées dans l'ACI sont autonomes, ce qui nous produit

l'hétérogénéité sémantique ainsi que celle de leur langage de requête. Un *traducteur* est associé à une source d'information, il joue le rôle d'interface entre l'information recherchée et le rôle *migrateur*, le *traducteur* traduit la tâche du *migrateur* dans le langage de la source d'information, et il convertit les résultats dans le langage du *migrateur* en utilisant les termes ontologiques.

IV.4.2.2 Langage de Communication pour l'ACI

Pour accomplir une tâche, les agents s'envoient des messages en utilisant un langage de communication de haut niveau. Nous choisissons le standard KQML (**K**nowledge **Q**uery and **M**anipulation **L**anguage) [Fin94] qui est un langage standard composé d'un ensemble de performatifs.
Le principe de KQML est de séparer la sémantique liée au protocole de communication (indépendante du domaine d'application), de celle liée au contenu des messages (dépendante du domaine d'application). Le protocole de communication doit donc être universel (pour tous les types d'agents), concis et constitué d'un nombre restreint de primitives de communication. Dans KQML un message contient toutes les informations nécessaires à sa compréhension :

(KQML-performatif
:*émetteur* <*texte*> // adresse de l'émetteur du message.
:*récepteur* <*texte*> // la ou les adresses des récepteurs.
:*langage* <*texte*> // langage de représentation du contenu.
:*ontologie* <*texte*> // l'ontologie du domaine relative au contenu
:*contenu* <*expression*> // l'information communiquée
...)

La syntaxe utilisée est celle du Lisp, les arguments peuvent être donnés dans n'importe quel ordre. Les performatifs utilisés dans KQML sont indépendants du domaine d'application dans lequel évoluent les agents. En effet la sémantique des messages se trouve définie dans les champs ***:contenu*** (le message lui-même), ***:langage*** (le langage de programmation avec lequel le message a été écrit) et ***:ontologie*** (le vocabulaire utilisé dans le message).
Dans KQML les communications peuvent être asynchrones grâce aux champs ***:reply-with*** côté émetteur et ***:in-reply-to*** côté récepteur. Les performatives de KQML sont organisées dans sept catégories :

1. questions de base (*evaluate, ask-one, ask-all* ...)
2. questions à réponses multiples (*stream-all* ...)
3. réactions (*reply, sorry* ...)
4. informations (*tell, achieve, untell,* ...)
5. états (*standby, ready, next, rest* ...)
6. définition de compétences (*advertise, subscribe, monitor* ...)
7. réseau (*register, unregister, forward, broadcast* ...)

Nous utilisons six performatifs dans notre cadre d'étude *Ask, Tell, Recomand, Reply, Inform* et *Advertise* : [Fin94].

ASK	Utilisé par un agent pour poser une question à un autre agent.
TELL	Utilisé par un agent pour répondre à une question de type Ask posée par un autre agent.
ADVERTISE	Utilisé par un agent pour annoncer ces capacités à un Matchmaker.
RECOMMAND	Utilisé par un client pour demander à un Matchmaker l'identité d'un fournisseur capable de répondre à sa question.
REPLY	Utilisé par un Matchmaker pour répondre à un message de type RECOMMAND soumis par un client.
INFORM	Utilisé pour définir des échanges d'informations qui, contrairement au cas d'un TELL, ne surviennent pas en réponse à une question.

TAB.IV.1. Description du langage utilisé dans notre approche

Exemple :

On prend l'exemple du dernier performatif du tableau ci-dessus, un agent A veux informer B sur une information du prix d'un livre, alors A envoi à B le message suivant :

Inform **(**
Sender : A
Receiver : B
ontology : O1
language :KQML
content : ***"price(ISBN3294,24.95)"***
)

Remarquons que même si l'interaction ne soit pas directe entre A et B, c'est à dire via un autre agent intermédiaire, ce dernier en lisant le champ *Receiver* comprend que le message est adressé à B, en conséquence il a la tâche de lui transmettre le même message.

IV.4.2.3 Protocoles d'interactions

Nous allons identifier les différents protocoles d'interactions qui peuvent être estimer lors de la résolution d'une requête d'un rôle interface.
Le diagramme de collaboration UML (fig.IV.6) représente l'ensemble des rôles, leurs interactions et l'ordonnancement de ces interactions, nous pouvons identifier deux protocoles d'interaction :

- le traitement d'une question de l'interface (de 1a à 12a)
- l'ajout d'une nouvelle source d'information (1b)

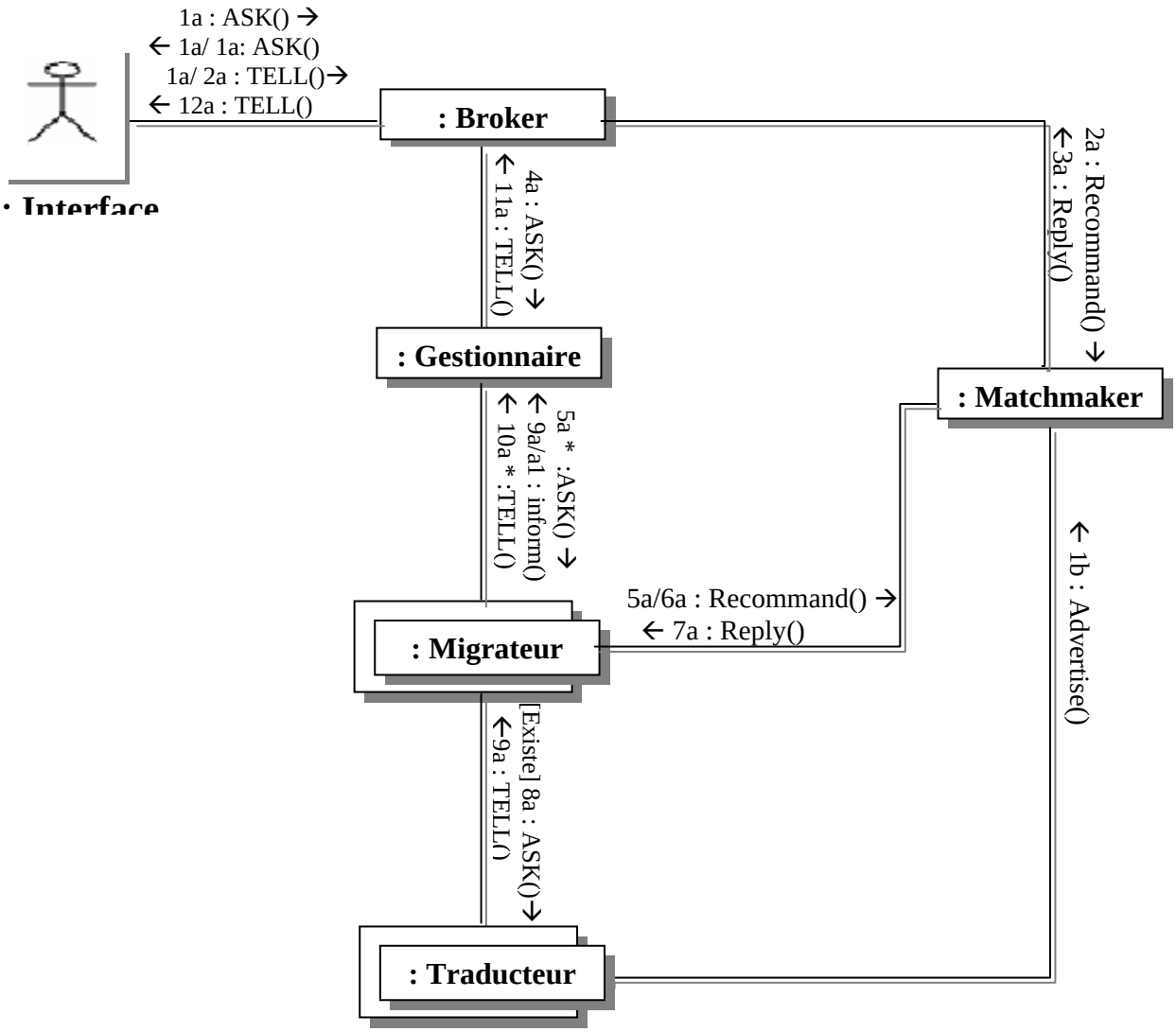


Fig.IV.6.Diagramme de collaboration entre les rôles impliqués dans notre ACI.

IV.4.2.4 Spécification des rôles

Nous allons décrire les différents rôles identifiés dans le paragraphe 4.2. Chaque rôle est spécifié par un diagramme de séquence UML. Les diagrammes de séquences permettent de représenter les collaborations entre les rôles selon un point de vue temporel, on y met l'accent sur la chronologie des envois de messages et les actions internes propres aux rôles.

Le diagramme de séquence UML [OMG] représente la chronologie des interactions entre les rôles dans l'accomplissement d'une tâche.
Ce diagramme représente chaque rôle par une ligne verticale, chaque interaction par une flèche horizontale reliant le rôle émetteur au rôle récepteur (en indiquant le performatif utilisé), et chaque action interne par une flèche verticale sur la propre ligne du rôle (en indiquant le nom de la procédure). L'ordre d'envoi d'un message est déterminé par sa position sur l'axe vertical du diagramme; le temps s'écoule "de haut en bas" de cet axe. La disposition des rôles sur l'axe horizontal n'a pas de conséquence pour la sémantique du diagramme.

IV.4.2.4.1 Le rôle Interface

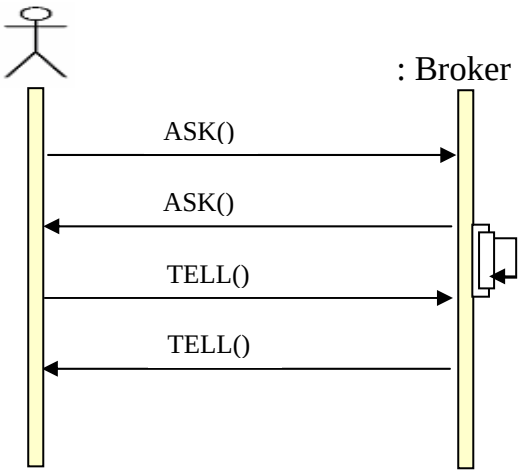


Fig.IV.7 Le rôle Interface.

Le rôle *interface* n'interagit qu'avec le rôle *Broker*, il lui livre la requête de l'utilisateur. À un moment donné le *Broker* peut donné des sous question au rôle *interface* pour lui guider à travers ses préférences, ou bien autre information utiles (ce processus de coopération peut avoir lieu plusieurs fois), enfin le *broker* lui transmis le résultat final de la requête pour l'affichage.

IV.4.2.4.2 Le rôle Broker

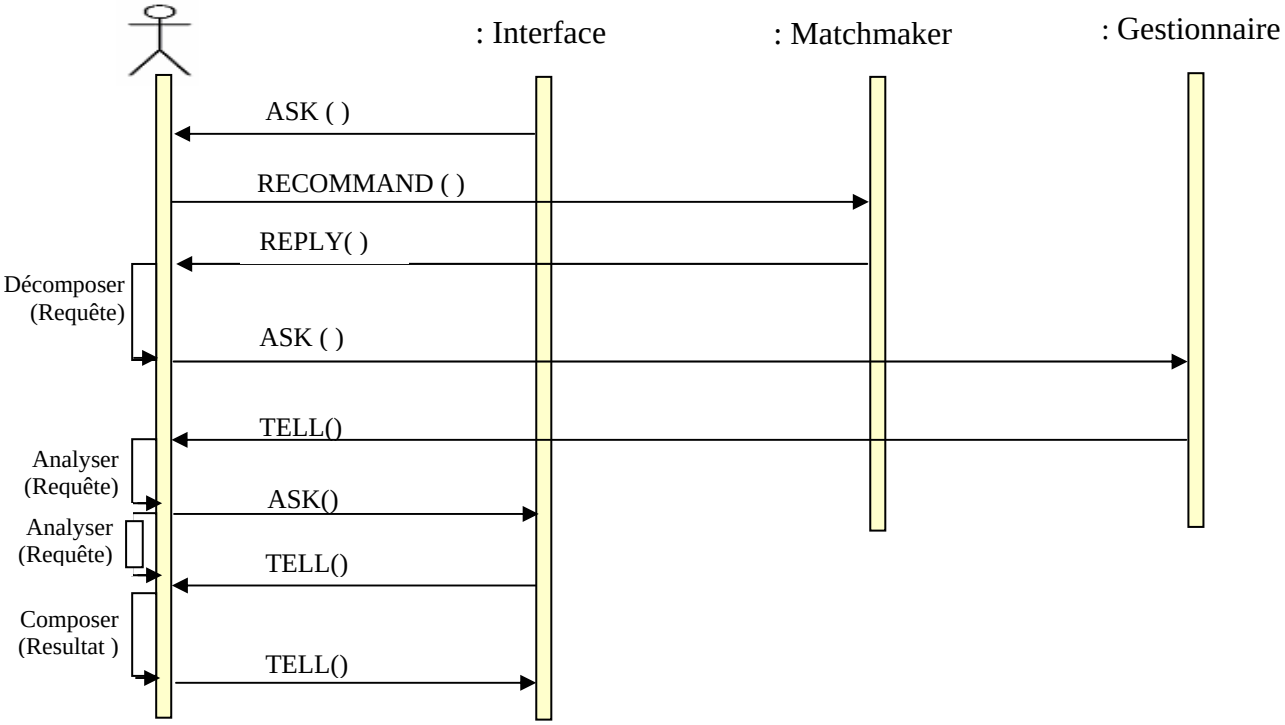


Fig.IV.8 Le rôle Broker.

Le rôle *Broker* assure plusieurs fonctions dans le système, il interagit avec les trois rôles *Interface*, *Matchmaker* et le *Gestionnaire*, il utilise le modèle informationnel comme ressource. Le comportement du *Médiateur* peut être décomposé en deux étapes :

- La première partie permet au *Broker* d'accepter la question du rôle *Interface*, de la reformuler, de la décomposer en sous-questions en tenant compte du contexte dans la quel été posée, c'est-à-dire que le *broker* transforme la requête en une collection de tâches élémentaires, en se basant sur les attributs et les relations qui existent dans l'ontologie du domaine, cette planification aura lieu en interagissant avec le rôle *Matchmaker*, ensuite il soumit les sous questions ainsi l'adresse d'ontologie au rôle *Gestionnaire*.
- La deuxième partie commence dés que les résultats sont arrivés du *Gestionnaire*, le *Broker* analyse en combinant et filtrant les résultats partiaux pour composer le résultat final, cette fonction est réalisable avec la coopération du rôle *interface* pour lui guider sur ces préférences.

IV.4.2.4.3 Le rôle Gestionnaire

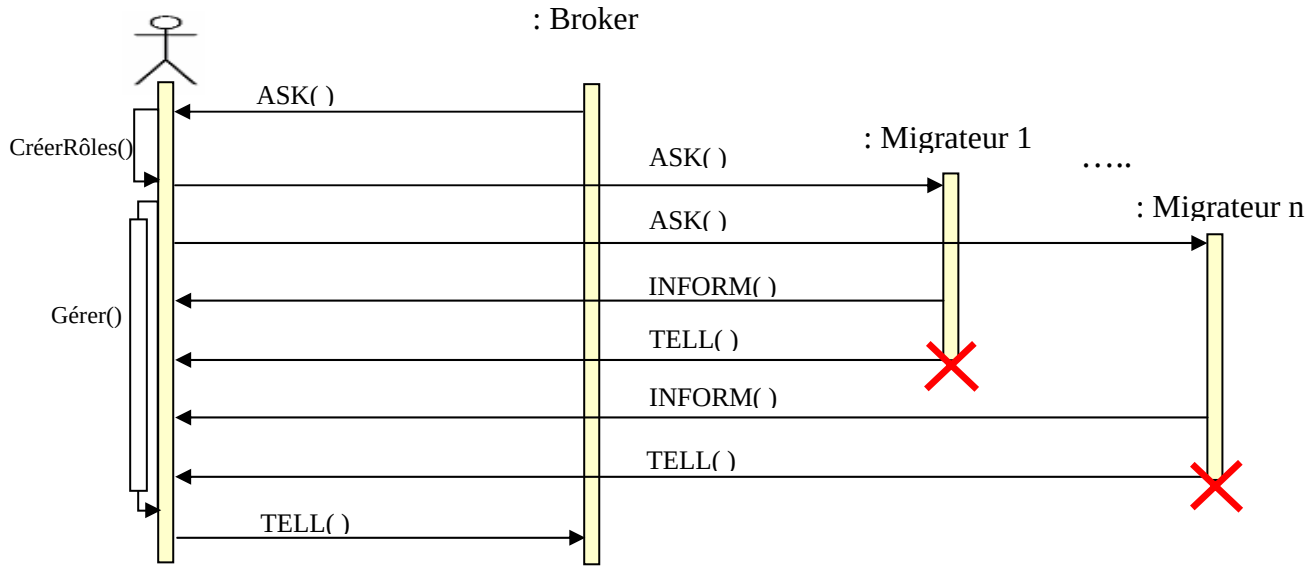


Fig.IV.9 Le rôle Gestionnaire.

Comme son nom l'indique, ce rôle a la particularité de gérer toute communications entre les rôles *migrateurs* (comme un *proxy*, voir chapitre III.5.5). En recevant les tâches du rôle *Broker*, le *Gestionnaire* crée dynamiquement les rôles *migrateurs* selon les tâches disponibles (nombre de migrateur créé est égale au nombre des tâches élémentaires retournées par le *Broker*), il attribue à chaque *migrateur* une tâche et les intègrent dans une partie de son ontologie du domaine selon le besoin de la tâche, c'est pour fournir aux rôles *migrateurs* une capabilité de résoudre et de communiquer . Ensuite il gère l'exécution des tâches, car ces tâches sont interdépendantes et une coopération doit se réaliser entre les différents rôles, pour cela, il utilise un tableau noir comme ressource. Enfin il remet tous les résultats parvenus par les *migrateurs* au rôle *Broker*.

IV.4.2.4.4 Le rôle Migrateur

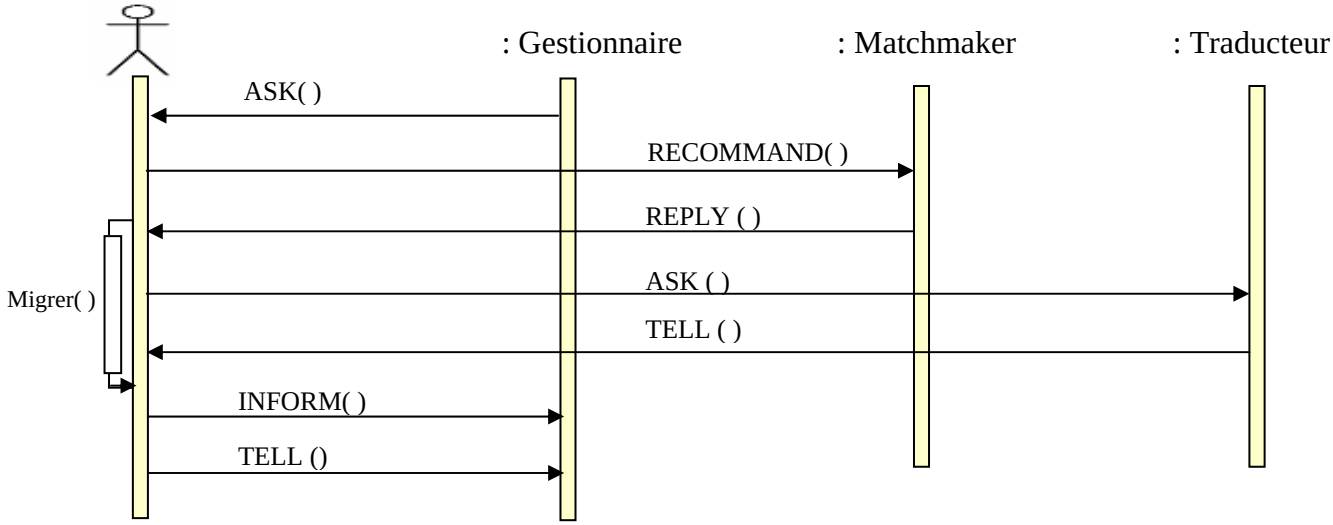


Fig.IV.10 Le rôle Migrateur.

Le rôle *migrateur* est le rôle responsable de collecter les informations dispersées sur plusieurs sources informationnelles. On recevant une tâche élémentaire du rôle G*estionnaire*, le *migrateur* interagit avec le *Matchmaker* pour lui fournir toutes les sources d'informations capable de contenir les informations désirées, ensuite il migre sur chaque nœud retourné par le *Matchmaker*, et à chaque nœud, il interagit avec le rôle *traducteur* pour accomplir sa tâche (processus itératif), et il peut communiqué avec le gestionnaire (par le performatif INFORM()). En revenant à sa place d'origine, il fournit au gestionnaire tous le résultat de ce voyage (TELL( )). Le migrateur doit communiquer que sur la partie de sa vue locale pour aider le gestionnaire de trouver une vue cohérente globale.

IV.4.2.4.5 Le rôle Matchmaker

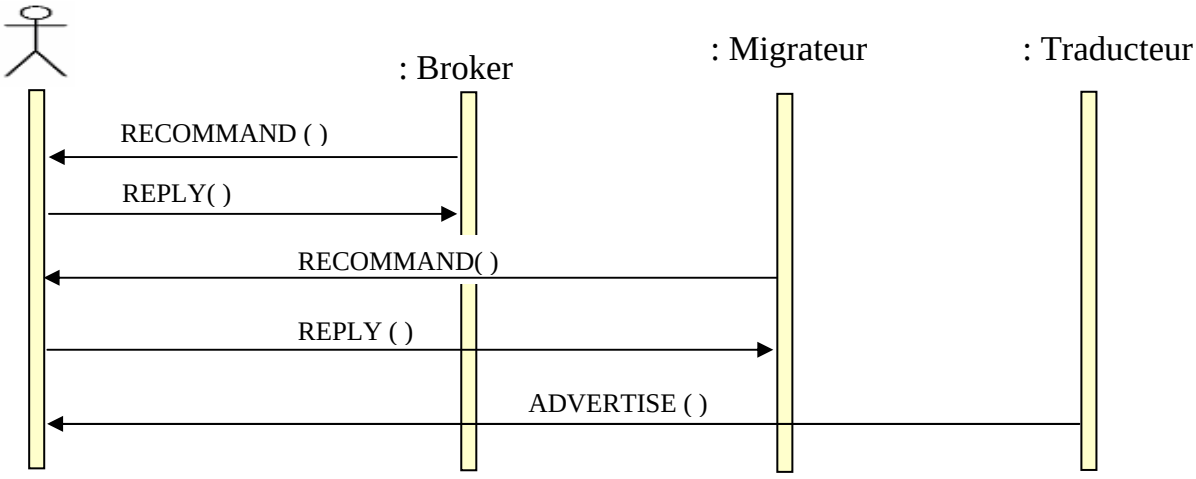


Fig.IV.11 Le rôle Matchmaker.

Ce rôle répond aux demandes du *broker* et du *Migrateur*, il fournit au *broker* l'adresse de l'ontologie du domaine et aux *migrateurs* les adresses des sources informationnelles où

l'information rechercher est probablement située. Il joue donc un rôle des 'pages jaunes' qui connaît au préalable toute les informations concernant les sources d'informations, ces informations ne seront connus par le *Matchmaker* que si chaque rôle *traducteur* inscrivent toutes les information de leurs sources chez le *matchmaker* (par le biais du performatif ADVERTISE ())

IV.4.2.4.6 Le rôle Traducteur

Le rôle *Traducteur* (Fig.IV.12) interagit avec le *Migrateur* et le *Matchmaker*. Il utilise deux Ressources *:*
- *une Source d'Information* (SI) dont il extrait des informations pour répondre à des questions du *Migrateur.*
- *Un dictionnaire* définissant les liens sémantiques entre les termes de sa source et les termes ontologiques du domaine. Ce dictionnaire permet au *Traducteur* de remplacer les termes ontologiques utilisés dans la question du *Migrateur* par des termes de sa SI, et inversement pour les termes du résultat.

Le *traducteur* est impliqué dans deux protocoles : i) la publication de ses capacités, qui se réduit au performatif ADVERTISE ( ) ii) la réponse à la question du Migrateur en utilisant les performatifs ASK ( ) et TELL ( ).

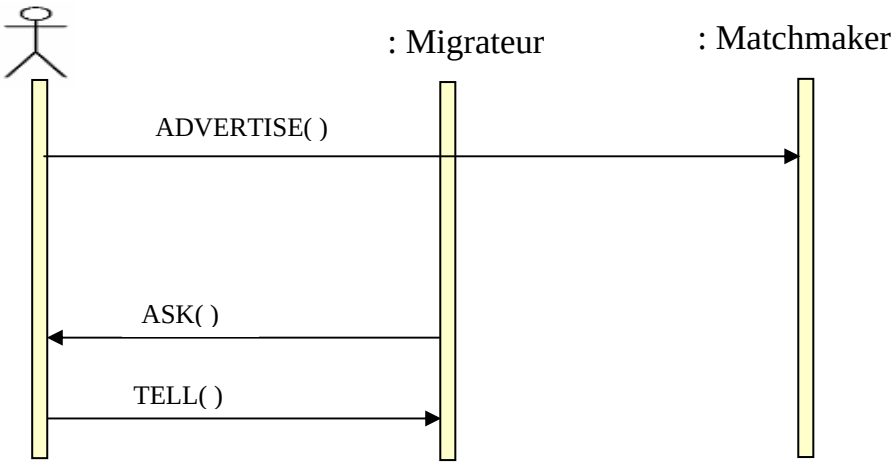


Fig.IV.12 Le rôle Traducteur.

IV.4.2.5 Structure organisationnelle

Une structure organisationnelle est une spécialisation du modèle de rôle pour une tâche particulière, un rôle dans une structure organisationnelle est capable de réaliser des actions spécifiques à une tache donnée.
Une structure organisationnelle d'une tâche contient les rôles suivants :

- Un rôle Interface, qui reçoit la requête et le pose au système ;
- Un rôle Broker, qui compose la requête à des taches élémentaires ;
- Un Matchmaker, qui je joue un rôle de pages jaunes ;
- Un Exécuteur, qui sélectionne et gère des Migrateurs;
- Autant de Migrateur que de taches élémentaires ;
- Autant de Traducteur que de Sources d'Informations.

Pour illustrer cette définition, la figure (Fig.IV.13) suivante présente un diagramme de collaboration UML [OMG], qui décrit la structure d'une organisation multi-agent d'ACI pour l'exemple le célèbre dans ce domaine, qui est celui de l'organisation d'un voyage. Dans cette figure, un Voyageur pose une question au système par le biais de l'interface voyageur, l'organisateur de voyages charge l'ontologie du domaine de voyage en demandant le service du matchmaker. Selon cette ontologie, il décompose le problème en cinq sous-problèmes: un organisateur de visites historiques, un organisateur de visites naturelles, un organisateur de météo, un organisateur d'hébergement, et un organisateur de transport, et les soustraite respectivement au gestionnaire, ce dernier crée dynamiquement cinq rôles de migrateurs, en lui attribuant les cinq tâches respectivement.

Chaque Migrateur envoie au Matchmaker une question lui demandant de lui fournir l'adresse d'un Traducteur pouvant répondre à la question qu'il prend en charge. Le Matchmaker répond à chaque Migrateur en fournissant les adresses des Traducteurs sélectionnés. Chaque Migrateur peut ensuite migré aux sources d'informations sélectionnées et sous-traite auprès du traducteur correspondant la question qu'il prend en charge. Lorsque tous les Migrateurs fournissent la réponse à leur question au gestionnaire, celui-ci peut les envoyés au Broker et fournir le résultat à l'utilisateur.

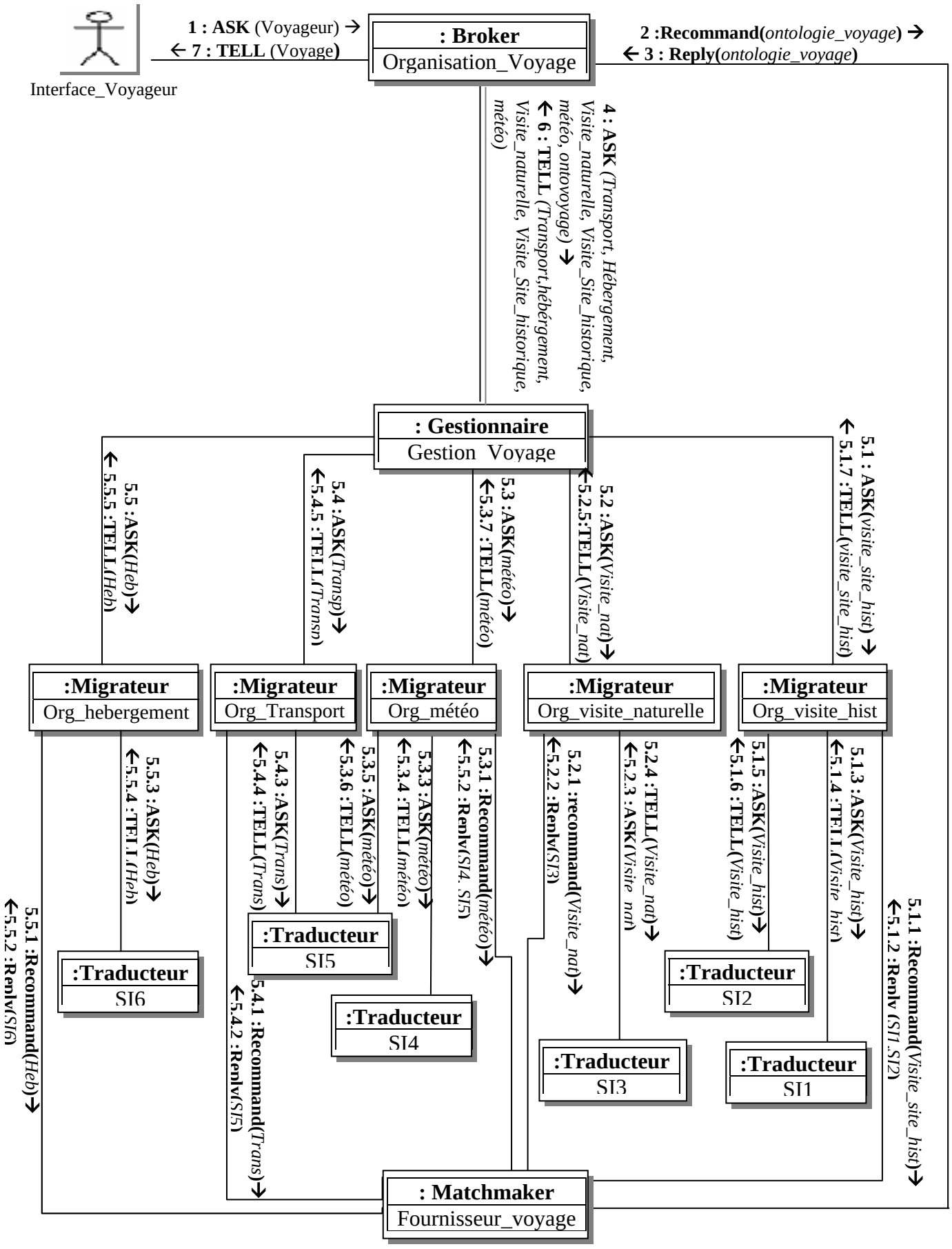


Fig.IV.13 Diagramme de collaboration de l'exemple d'organisation de voyage.

IV.4.3 Modèle d'agents et d'accointances

Ce modèle est important pour la phase de conception d'un système multi-agents, c'est la passerelle entre le modèle de rôle et une réelle conception du système d'ACI, il documente sur les types d'agent qui seront utilisés dans le système, et les instances d'agents qui les réaliseront dans l'exécution.

La figure VI.14 nous montre ce modèle d'agent pour le processus d'ACI, où chaque agent est mis en correspondance à un rôle, la figure suivante (VI.15) définit la communication qui existe entre les agents.

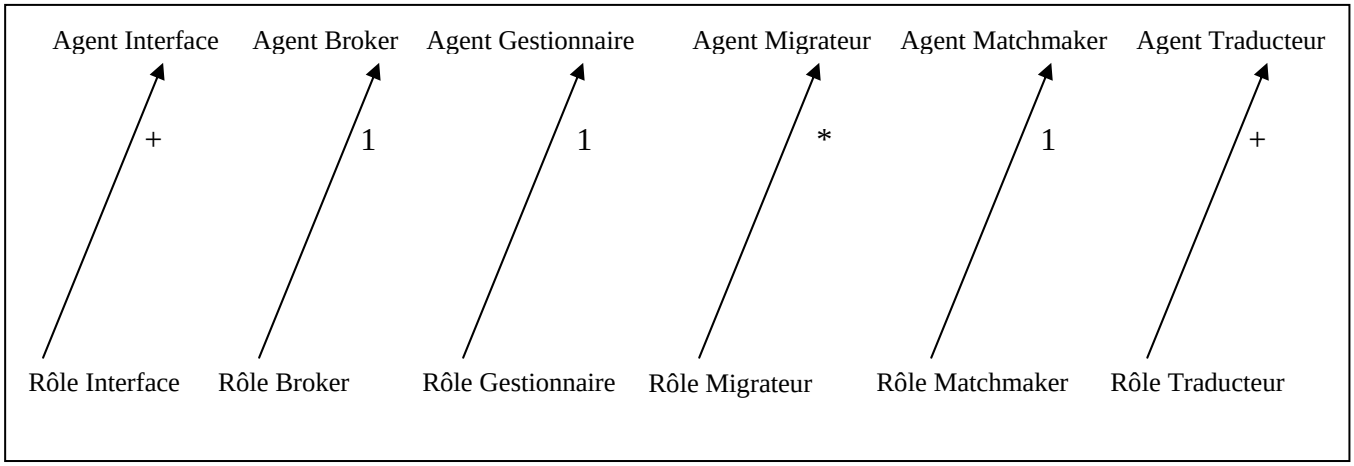


Fig. VI.14. Modèle d'agent pour notre système d'ACI.

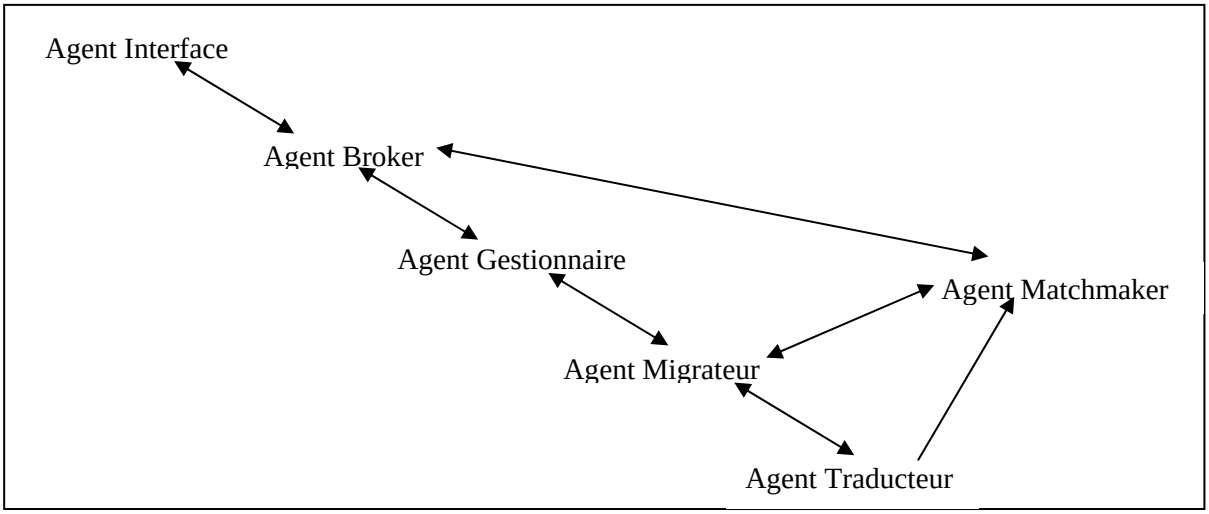


Fig. VI.15. Modèle d'accointances.

IV.4.4 Architecture d'un système d'ACI

Après avoir déterminé le modèle de rôle et d'accointances qui interviennent dans le processus d'ACI, nous pouvons maintenant décrire l'architecture générale de notre système, cette architecture est représentée dans la figure ci-dessous (Fig.IV.16), son fonctionnement est celui décrit lors de la spécification des rôles, car chaque agent du système jouera son propre rôle spécifié auparavant.

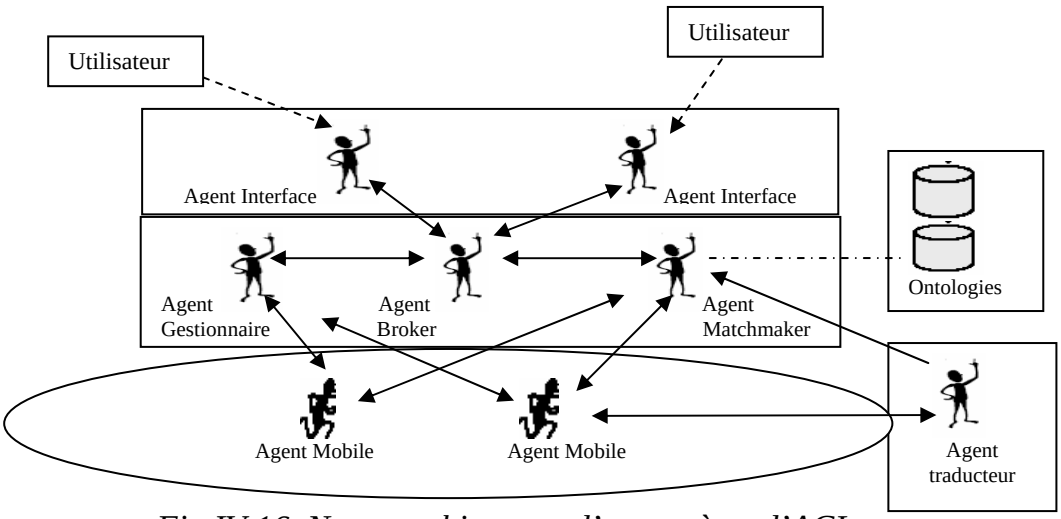


Fig.IV.16 Notre architecture d'un système d'ACI.

Hormis le fonctionnement et les avantages cruciaux - ceux dus à la résolution des hétérogénéités (sémantique, structurelle), et ceux dues à l'environnement (dynamique, distribué) - l'architecture nous offre plusieurs autres avantages, c'est ainsi qu'elle est :

- Extensible : la fonctionnalité du système est extensible par :
  1. l'ajout d'autres agents mobiles avec des nouvelles capabilités, puisqu'ils partagent une partie de l'ontologie.
  2. l'ajout de sources d'informations nouvelles au systèmes, ce qui le rend un système dynamique.
- Modulaire : sur le fait de représenter des agents traducteurs, des agents mobiles et des sources d'informations.
- Flexible : en terme de sélection de la source d'information la plus appropriée pour répondre à une question.
- Adaptable : en terme d'être capable de suivre l'incohérence sémantique dans les sources d'information.
- Réutilisable : sur le fait que les rôles des agents ont un niveau indépendant du problème, et ils sont construit une fois pour toute.

IV.5 Conclusion

Nous nous sommes intéressés à la conception d'une architecture multi agents basée sur un aspect organisationnel, dédiée à l'Acquisition Coopérative de l'Informations dans des sources de données distribuées et hétérogènes, placées dans un environnement ouvert et dynamique.
Pour cela nous nous somme inspirés de la méthodologie GAIA, où la notion de rôle est complète, et nous avons suivi les étapes pour passer d'une description abstraite d'un modèle organisationnel basé sur la notion de rôle à un modèle conceptuel architectural, ainsi nous avons :

i. *Défini le Modèle de Rôles* qui représente, à un niveau indépendant du problème, les fonctions abstraites d'un système d'ACI et les interactions entre ces fonctions.
ii. *Défini les tâches de la Structure Organisationnelle*, une Structure organisationnelle est une spécialisation du modèle de rôle pour répondre à une tâche donnée (une classe de questions).
iii. *Défini le modèle d'agents et d'accointances* qui documente sur les types, les instances, et la communication entre les agents qui seront utilisés dans le système

Cette architecture est en cours de mise à jour et de développement. Dans le prochain chapitre nous essayons de montrer l'implémentation de cette architecture à travers une étude de cas, sur la collecte d'articles de recherche.

Chapitre V

Etude de cas:
La collecte d'articles de recherche

Our problems are man-made; therefore
they may be solved by man

J.F.Kennedy, 1963.

Chapitre V

Etude de cas: La collecte d'articles de recherche

V.1 Introduction

Après avoir donné l'architecture multi agent pour l'acquisition coopérative de l'information, et les différents agents mis en jeu dans cette organisation, nous allons donc présenter tout d'abord l'environnement MADKIT qui nous a servi comme une plate forme de développement, enfin nous présentons l'application proprement dite.

Le choix de MADKIT n'est pas venu par hasard, car en faisant un clin d'œil sur les plates-formes existantes, on remarque qu'il existe quatre grandes familles de ces plates formes :

- Celles d'agents mobiles (comme Aglets, Mole, Odyssey…)
- Celles orientées par rapport à un modèle d'agent particulier (JateLite, COOL, Zeus, Sadabot…)
- Celles de simulations : parfois elles sont construites autour d'un domaine spécifique (Swarm, CORMAS, StarLogo, Manta, Mobydic…)
- Et enfin les plates formes de standardisation : venus de la normalisation FIPA et MASIF (Grasshopper, JADE, FIPA-OS…)

La philosophie de base MADKIT est d'utiliser autant que possible la plate forme pour son propre fonctionnement. On a vu que l'utilisation de structure organisationnelle permet de garder une cohérence globale du système, ainsi, elle supporte la mobilité, et pour la motivation d'avoir une plate forme capable de s'adapter aux différents modèles d'agents et domaine d'application que nous avons choisi ladite plate forme.

V.2 L'environnement MADKIT

V.2.1 Principe

V.2.1.1 Architecture

Pour valider le modèle organisationnel AALAADIN, Ferber et Gutknecht [FeGu98] [Gut&00] ont construit une plate-forme appelé MADKIT (pour ‘‘Multi-Agent Development KIT ’’). Cette plate-forme, développée en parallèle avec le modèle AALAADIN est en java. La structure organisationnelle (AALAADIN) est implémentée au cœur de la plate-forme MADKIT, tant pour fournir un modèle organisationnel aux systèmes multi-agents exécutés que pour le fonctionnement interne du système. En plus de ce modèle d'organisation, MADKIT est basé sur trois principes (FigV.1.) :

– Architecture à micro-noyau,
– Agentification systématique des services,
– Applications hôtes.

Concrètement, MADKIT est un ensemble de packages Java qui implémentent le noyau agent, diverses bibliothèques de base de messages, d'agents et de sondes Le principe essentiel de MADKIT est d'utiliser partout où cela est possible la plate-forme pour son propre fonctionnement. Tous les services hors ceux assurés par le micro-noyau sont implémentés par des agents à part entière. Ceci vient à la fois d'un souci d'unicité de modèle, mais également de mise à l'épreuve des systèmes multi-agents comme modèle de programmation. Cela implique que MADKIT n'est pas une plate-forme agent dans le sens classique, centrée autour d'un modèle particulier d'agent ou d'interaction. La taille réduite et les fonctionnalités du noyau de base, la neutralité des types agents, associée à ce principe de services agentifiés et de découplage par rapport aux applications "hôtes" permet en fait d'obtenir toute une gamme d'environnements d'exécutions aux finalités parfois complètement opposées.

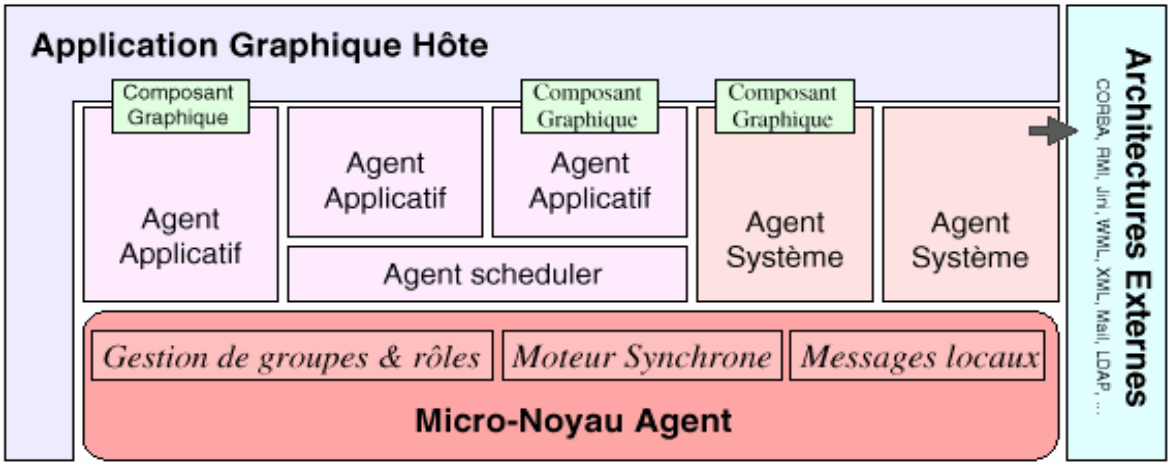


Fig.V.1. Structure générale

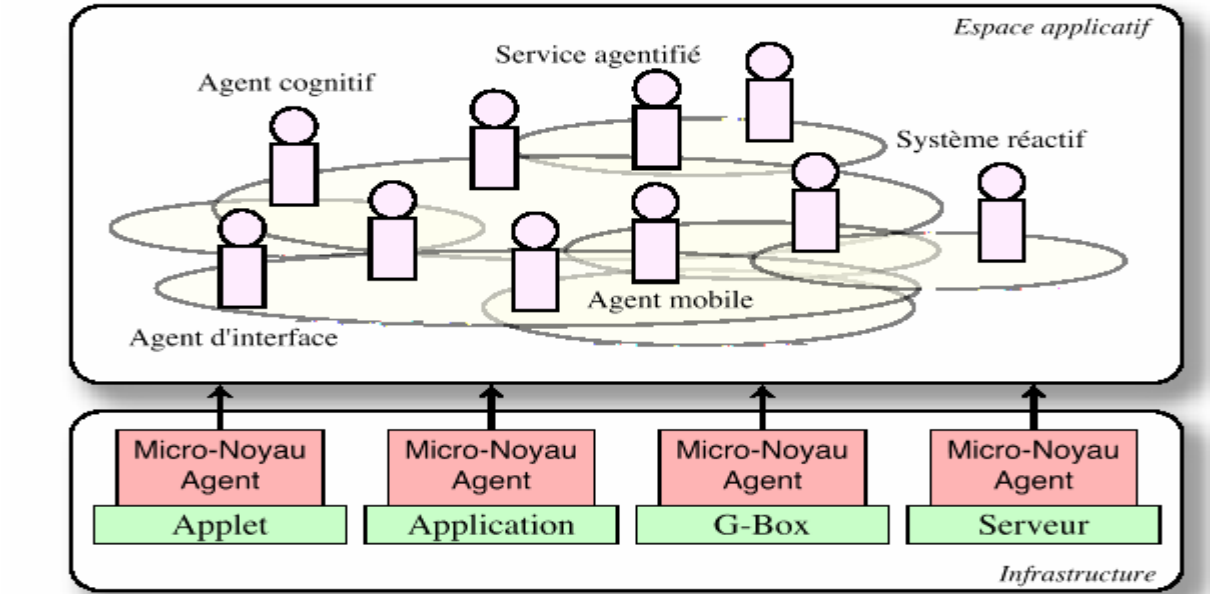


Fig.V.2. Schéma général de fonctionnement

V.2.1.2 Le micro-noyau agent

Le "micro-noyau" de MADKIT est un environnement d'exécution d'agents de taille réduite (moins de 40 Ko). Ce module ne prend en charge que les fonctions suivantes :

- **Gestion des groupes et rôles locaux.** Comme l'interopérabilité et les mécanismes d'extension de MADKIT se basent sur le modèle agent-groupe-rôle, il est essentiel que cette information organisationnelle soit gérée au plus bas niveau afin que tous les agents, quels que soient leurs modèles individuels, y aient accès. 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 (c'est à dire en évaluant ou délégant les fonctions d'acceptation de rôles).

- **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 unique à chaque agent. Cet identifiant, l'AgentAddress, est forgé à partir de l'adresse du noyau et l'identification de l'agent sur le noyau. La forme de cet identifiant peut également être redéfinie pour faciliter l'intégration avec d'autres plate-formes agent.

- **Passage de message local.** Le noyau a la responsabilité de l'aiguillage et de la distribution de messages entre agents uniquement **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.

V.2.2 Fonctionnalités d'un agent

La classe de base d'un agent MADKIT (AbstractAgent), définit quelques fonctionnalités de base pouvant être nécessaires dans les modèles classiques.

A. Fonctionnalités :

Les fonctions associées à tout agent 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 de les désactiver par la suite). Par contre, aucun mécanisme concret d'exécution n'est défini à ce niveau.

- **Communication.** La communication est implémentée sous forme de passage de message asynchrone, soit d'agent à agent identifié 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.

B. Messages

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 Fig.V.3) est néanmoins fournie et permet l'envoi de chaînes, d'objets sérialisés, de documents XML, ou bien de messages conformes aux spécifications
KQML [FiFr94] et FIPA-ACL [FIPA97]. Ils restent extensibles pour s'adapter à tout protocole d'interaction.

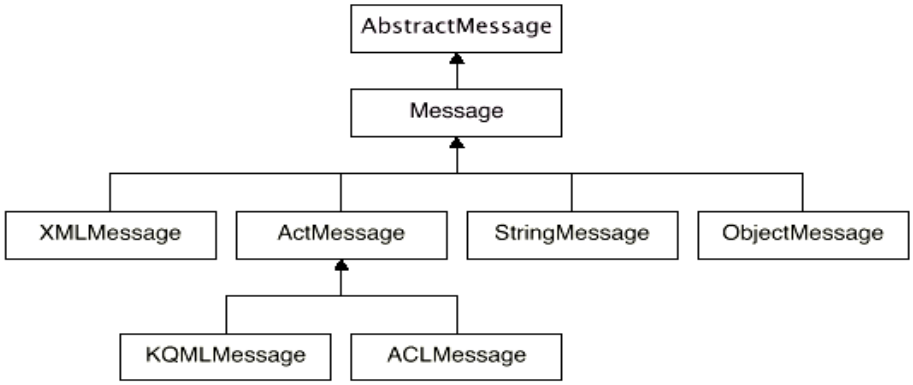


Fig.V.3. Hiérarchie des messages standards

V.2.3 Déclinaisons

V.2.3.1 Modèle d'agent

La définition d'un agent été minimale, différente bibliothèque implémentant des architectures d'agents, on peut citer :
- Une bibliothèque qui permet d'implémenter les agents en Scheme [Bot99]
- Une implémentation d'agents faisant une liaison avec le moteur de règle CLIPS de JESS [ErFr00]
- Différentes bibliothèques pour les SMA réactifs
- Des outils de construction graphique d'agents
- L'implémentation d'un modèle d'acteur.

V.2.3.2 Agentification des services

MADKIT utilise donc des agents pour implémenter des services comme le passage de message distribué, la migration d'agent, la sécurité d'organisations d'agents et divers aspects de gestion du système ; et ce, éventuellement à l'aide des mécanismes d'extension du noyau.
Comme ces services sont mis en œuvre par des agents et décrits par la structure organisationnelle, les interactions entre ces agents systèmes, le noyau et les agents de l'application sont décrits dans le modèle agent-groupe-rôle. Ceci implique qu'un agent offrant une fonctionnalité donnée peut être remplacé par un autre de façon transparente (et éventuellement durant le fonctionnement d'une application), à partir du moment où les groupes, rôles et interactions sont respectés.

Un agent tenant plusieurs rôles peut, au fur et à mesure que les besoins de l'application grandissent, déléguer dynamiquement à de nouveaux agents certains de ses rôles dans le but de réduire sa charge. [Gut&00]
De plus, la structuration en groupe agit souvent en "masquage" et délégation d'un système multi-agents complexe, un rôle ne correspond pas forcément à une mise en œuvre par un et un seul agent. Par exemple, un agent fournissant un rôle de communication système peut être un simple représentant d'un groupe d'agents spécialisés prenant en charge chacun un protocole.

V.2.3.3 Application hôte

Le noyau de la plate-forme ne fournit aucune interface graphique pour l'utilisateur. Il est par contre possible de lui associer une application hôte qui la mettra en œuvre. Chaque agent peut définir un objet graphique pour sa représentation. L'hôte aura alors la charge, au lancement d'un agent, de décider de la mise en place de l'objet graphique défini par l'agent (dans des fenêtres séparées, combiné avec l'interface d'un autre agent, etc..) et de les gérer au niveau global. Associé au principe d'Agentification des services, cela permet de modulariser complètement la plate-forme, de l'infrastructure d'exécution d'agent à l'outil d'aide au développement. [Gut&01]
Un outil de développement : Le plus souvent, l'utilisation de la plate-forme se fait par la G-BOX [Gut&00], un environnement facilitant le test et le développement de systèmes multi-agents (Fig.V.4). Cet environnement (application graphique hôte et agents spécialisés) permet de contrôler le cycle de vie des agents, charger de nouveaux "packs" d'agents ou d'architectures, de manipuler graphiquement les accointances (via les tables de groupes et de rôle) à fin de prototypage ou de correction d'erreur. L'interface graphique permet de garder trace et d'organiser son espace de travail même avec un nombre d'agent important.

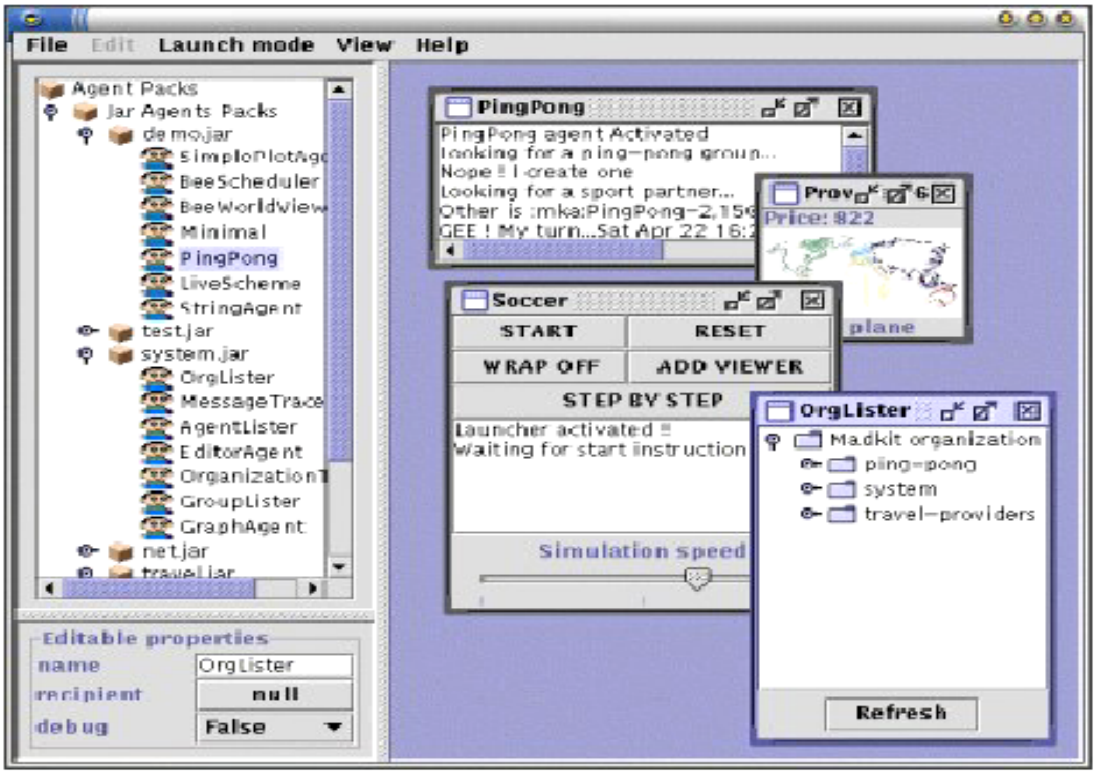


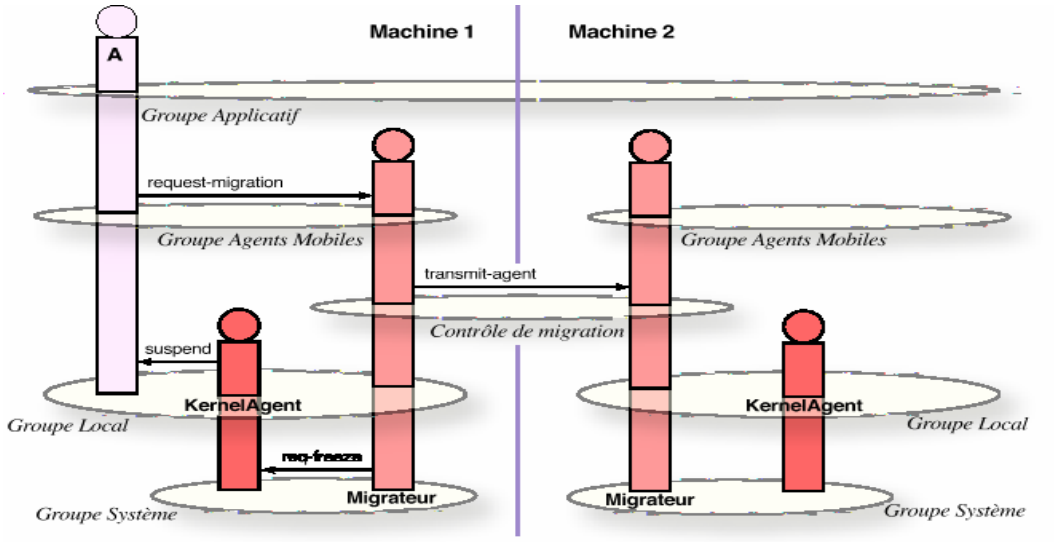
Fig.V.4. La G-Box.

V.2.3.4. Mobilité dans MADKIT

MADKIT n'inclut pas la mobilité dans le noyau, cette tâche est assurée par des agents spécialisés, et exprimés dans le modèle d'agent-groupe-rôle. [Gut&01]
Tous les agents d'un système ne vont pas forcément vouloir obtenir des capacités de migration. L'expression de la mobilité agent sous forme de groupe spécialisé permet d'utiliser le même formalisme standard de groupe et rôle, avec la mobilité gérée en fait via deux groupes. Chaque site d'exécution contient un groupe local mobilité qui contient tous les agents itinérants du site, ainsi qu'un agent outil ayant le rôle particulier de migrateur.
Le seul agent tenant le rôle de migrateur a la charge de faire migrer les agents de ce groupe qui le demandent vers un autre point d'exécution. L'agent migrateur appartient également à un groupe système qui rassemble les agents habilités à manipuler le cycle de vie des autres agents. Un autre groupe (distribué), contrôle de migration, rassemble tous les agents migrateurs sur chacun des sites d'exécution, et permet l'interaction des migrateurs lors d'une demande de migration.

Un scénario typique de migration d'agent se déroule comme suit (Fig.V.5): [FeGu98]

– L'agent A envoie une requête au gestionnaire de groupe mobilité pour demander de le rejoindre avec un rôle d'itinérant. Le gestionnaire de groupe évalue la capacité de l'agent à rejoindre le groupe
– Quand cet agent désire migrer sur un autre noyau d'exécution, il envoie un message demandant sa propre migration vers l'agent ayant le rôle de migrateur dans le groupe mobilité.
– L'agent migrateur demande à l'agent responsable du noyau de suspendre l'activité du candidat, et envoie A vers son pair sur un autre site S2
– L'agent ayant le rôle migrateur sur le site S2, confirme sa bonne réception et demande à son propre agent noyau de reprendre l'exécution de l'agent transmis.
– Le migrateur du site original demande au noyau de détruire l'agent suspendu.



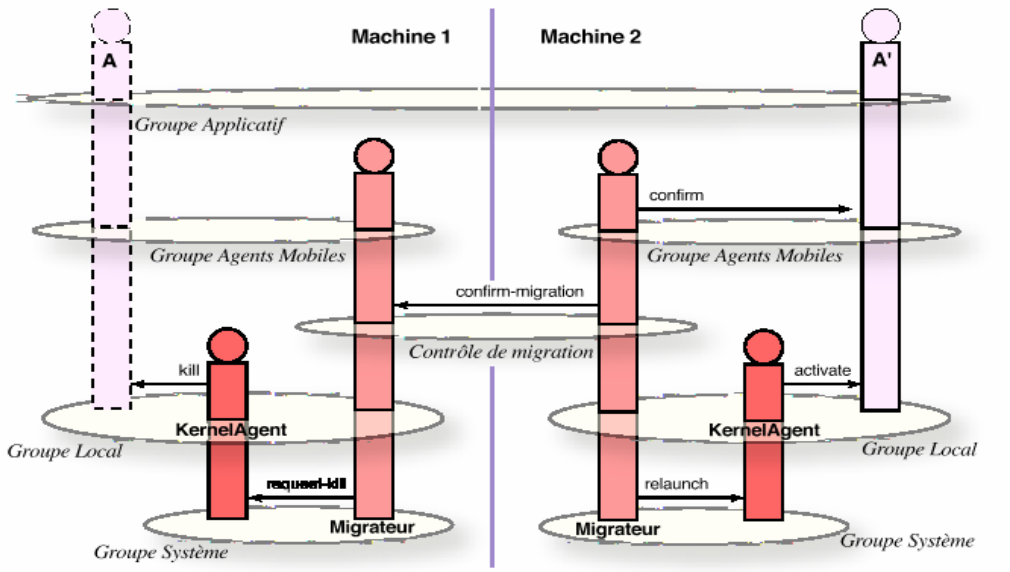


Fig.V.5. Scénario de migration

V.3 L'application

V.3.1 Description

L'objectif de cette application, qui s'est implémentée dans l'environnement MADKIT, est d'acquérir les articles de recherches pour des scientifiques donnés ou par mots clés de leurs travaux de recherche. Les articles sont distribués sur deux serveurs du réseau local du CERIST (Centre de Recherche sur l'Information Scientifique et Technique) qui se nomment respectivement RCNCT et PSTN et tous les deux exécutent MADKIT.
Selon notre architecture, chaque serveur doit avoir un agent traducteur. Ce dernier est le responsable de communiquer les informations avec l'agent demandeur (le Migrateur) via un modèle informationnel commun qui représente ce domaine. Chaque traducteur est implémenté comme une interface qui communique avec le migrateur, et il englobe une base de données qui contient toutes les informations relatives aux articles existant dans son propre serveur.

Le schéma suivant nous montre cette structure de l'application:

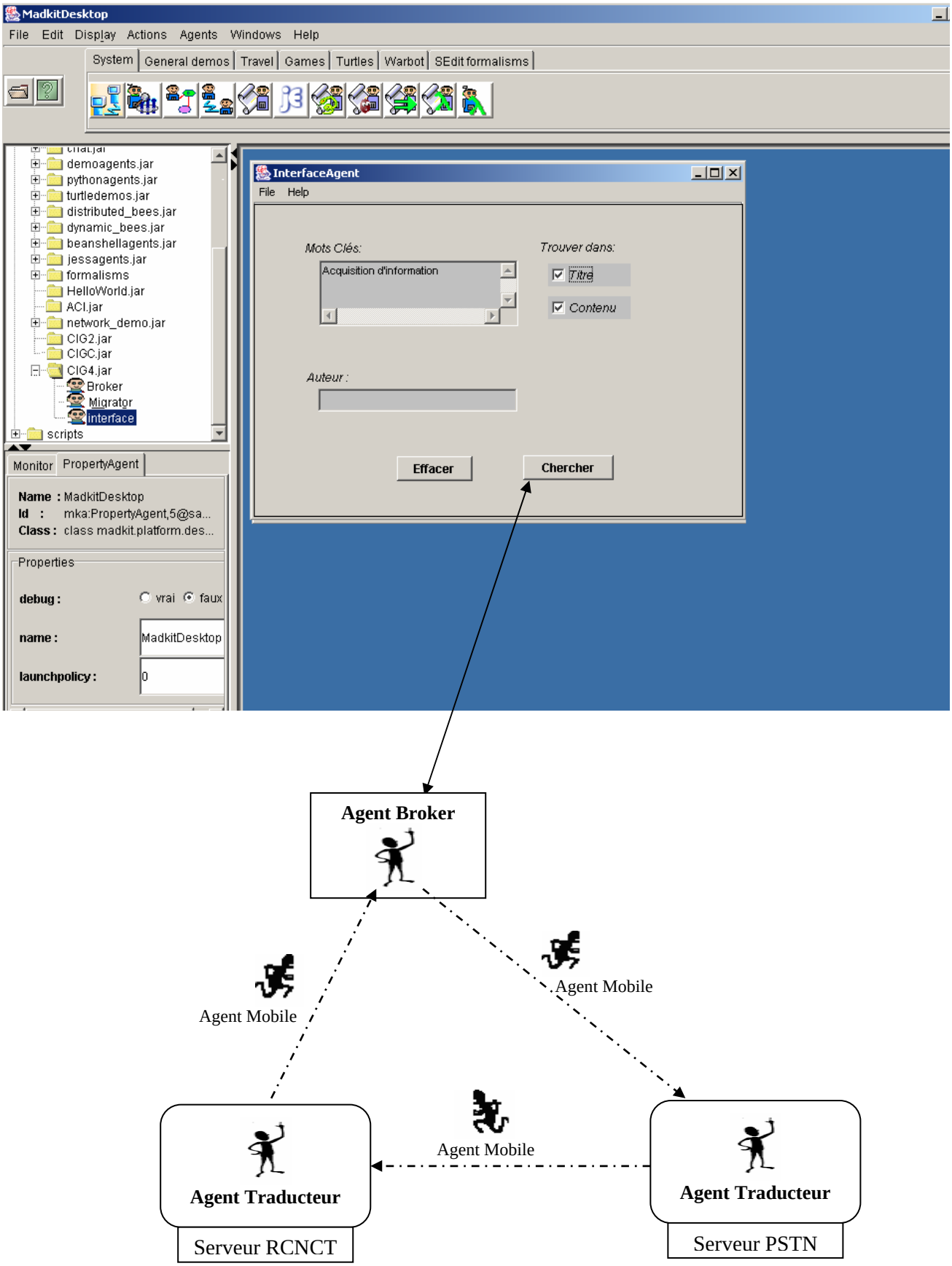


Fig.V.6 Schéma d'application

Tout d'abord, l'agent interface est un formulaire, où l'utilisateur peut introduire ces questions au système, ces questions parviennent à l'agent Broker.
Contrairement à notre architecture, et par la particularité de la simplicité de notre application, le rôle de l'agent gestionnaire sera intégré à l'agent broker, ainsi que l'agent matchmaker ne figurera pas dans cette application, car on n'a que deux sources d'informations et que l'univers de discours entre agent comporte qu'un seul domaine, donc une seule ontologie.

Le Broker qui compose la question de l'agent Interface crée un agent mobile. Ce dernier migre vers la machine PSTN du réseau et communique avec l'agent traducteur du serveur, ensuite il migre à l'autre serveur (RCNCT). En revenant à la machine du départ, le migrateur fournit les résultats au broker, ce dernier filtre et trie les résultats, et à son tour les remis à l'agent interface pour les afficher. La figure ci-dessous (Fig.V.7) nous montre le résultat de cette application pour un exemple de recherche des documents "Acquisition d'information":

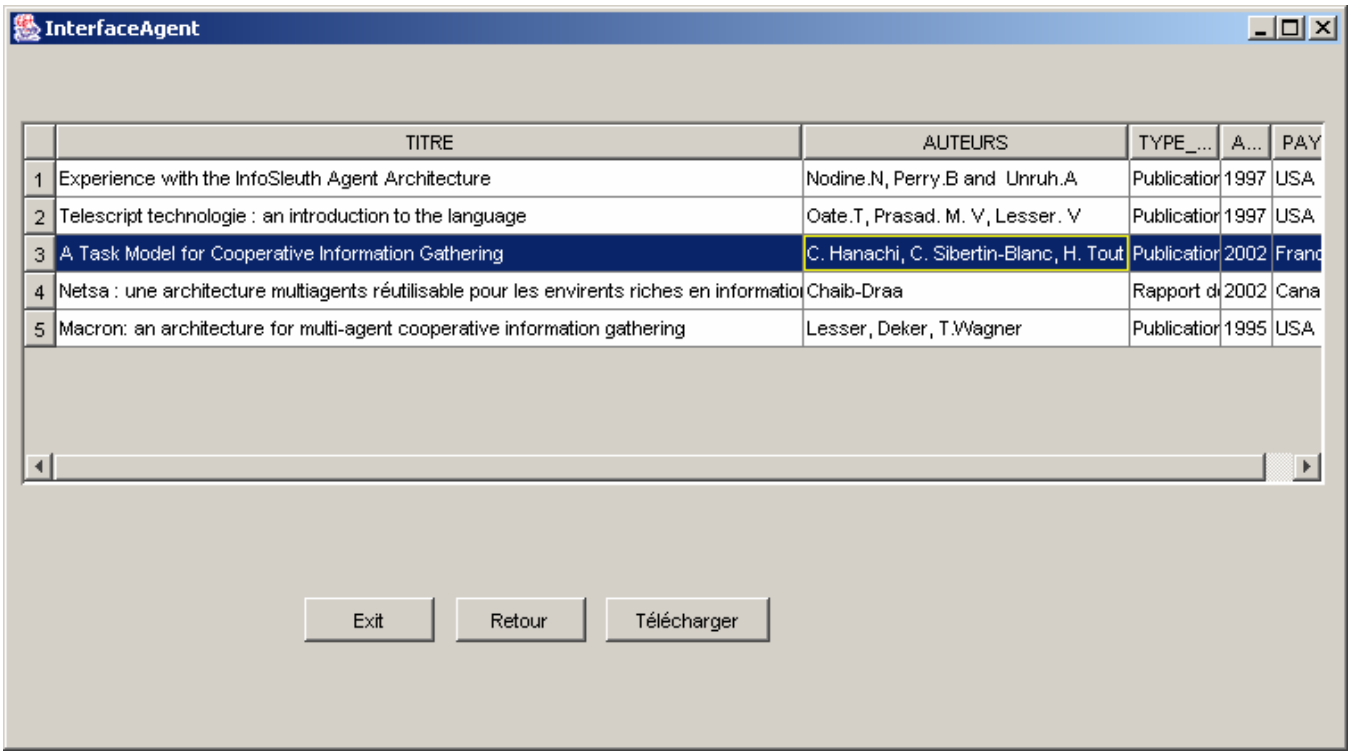


Fig.V.7 le résultat retourné par l'agent interface

V.3.2 Discussions

V.3.2.1 Les agents intervenant dans l'application

1. **Agent Interface** : est une simple interface, qui comporte deux classes, en premier temps une classe formulaire pour saisir les questions, ensuite la classe résultat qui affiche les résultats selon un design.
2. **Agent Broker** : est un programme qui crée des instances d'agents mobiles en lui donnant des tâches, et il peut filtrer et trier les résultats livré par le migrateur.
3. **Agent Traducteur** : est un code qui englobe une base de données contenant toutes les informations qui existe sur les documents de son serveur. Il consulte la base et répond à toutes les questions posées par les migrateurs.
4. **Agent Migrateur** : est agent spécial, son rôle est de migrer vers des destinations connues, une fois migrer, il interagit avec un agent en lui donnant une question, et de recevoir des réponses.
   Voici une partie du code de le fonction Live () de l'agent migrateur:

```
sendMessage(getAgentWithRole("communications","site"),new
NetworkRequest(NetworkRequest.GET_AVAILABLE_DESTINATIONS));

// L'agent concerné doit envoyer un message à l'agent système pour
//demander la liste des noyaux de destination disponible

Message m = waitNextMessage();
KernelAddress[] destinations =(KernelAddress[])((NetworkRequest)
m).getArgument();
if(destinations.length > 0)
         {
         status = 2;
         println("ok let's go to "+destinations[0].toString()+" \n");
         for(int i=5;i>=0;i--)
                    {
                        println(" "+i+" !!!");
                        pause(1000);
                    }
                    println("\n GO !!!!!!!!!!!!!!!!!!!!!!!!!!!");
                    pause(100);

          sendMessage(getAgentWithRole("communications","site"),new
          NetworkRequest(NetworkRequest.REQUEST_MIGRATION,destinations[0]));
          disposeMyGUI();

//  Une demande de migration à un autre agent système qui est chargé de réaliser la migration

 }
                  else
                    println("no destination, waiting...");
         pause(1500);
         break;
```

Une fois reçue par le noyau distant, l'agent est réactivé à partir de zéro (activate -> live ...) c'est-à-dire une migration faible. Il lui faut donc conserver en mémoire le statut dans lequel il se trouvait avant d'être expédié sur le noyau de destination.

V.3.2.2 Modèle Informationnel

Bien que le modèle informationnel n'est pas visible dans l'application, mais tous les agents programmés se communiquent via ce modèle, par exemple lorsque l'agent migrateur interroge le traducteur par un mot clé, le traducteur sait qu'un document scientifique est caractérisé par un titre, résumé et des mots clés. La figure *Fig.V.8* nous montre l'ensemble des attributs utilisés par l'application lors d'une recherche.

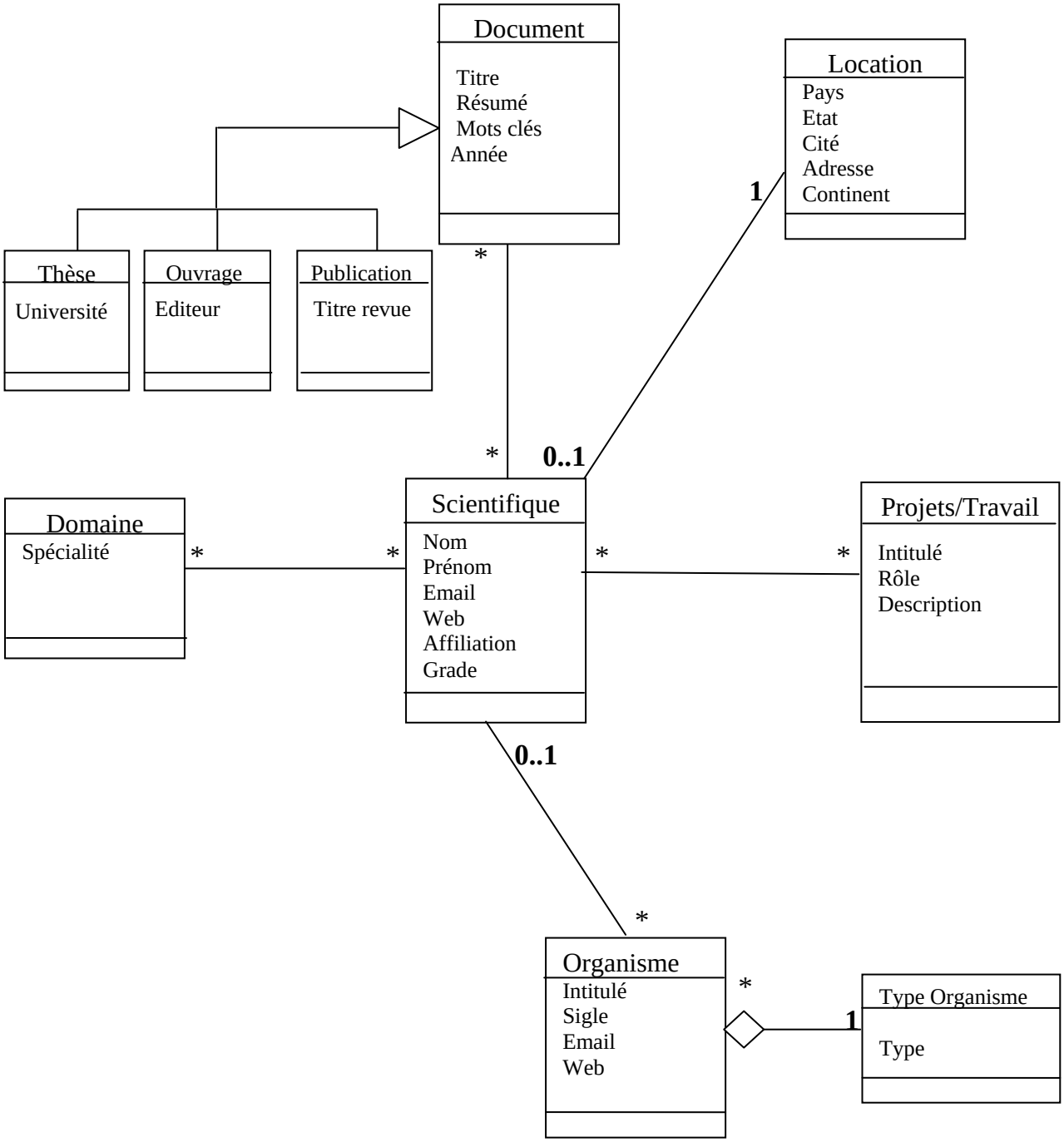


Fig.V.8 Diagramme de classe UML pour l'acquisition des documents scientifiques

V.3.2.3 Limite de MADKIT

1. La fonction de migration n'est pas clairement documentée, En effet, il s'agit d'une fonction qui reste à ce jour expérimentale. Pour changer de noyau, le principe consiste à sérialiser un agent et à le relancer sur le noyau de destination. Il faut par contre que la classe Java se trouve sur les deux noyaux pour que cela fonctionne. L'agent concerné doit donc envoyer un message à l'agent système pour demander la liste des noyaux de destination disponible et ensuite vers une demande de migration à un autre agent système qui est chargé de réaliser la migration. Ce processus est très lourd et demande plus de temps, notamment dans notre architecture où les agents mobiles ont des destinations connues pour migrer.
2. Un agent doit impérativement donner l'adresse MADKIT du noyau de destination. La raison est qu'il peut exister plusieurs noyaux qui ont la même adresse, car il est possible de lancer plusieurs instances de la plate-forme sur une seule machine. Par contre, il serait peut-être en effet intéressant de pouvoir connaître l'adresse IP d'un noyau, pour que des applications à grande distance marchent sur cette plateforme.

V.4 Conclusion

Nous avons présenté dans ce chapitre une implémentation d'une petite application qui vise l'acquisition des articles de recherche dans un réseau local, pour cela nous avons choisi la plate forme MADKIT pour son aspect organisationnel, où des primitives d'organisation comme JoinGroup() et RequestRole() sont fournit, mais nous avons constaté la lourdeur du processus de la mobilité, ainsi la nature locale des applications visées.

Cette application nous a permis de faire tourner et de manipuler une plate forme multi agents, de créer et d'intégrer des agents, et d'apercevoir les limites de la plate forme MADKIT, que nous nous jugerons utiles pour nos futurs travaux.

Conclusion générale

Conclusion générale

Dans ce travail, nous nous sommes intéressés à la conception d'une architecture d'un système multi agents pouvant être utilisée pour les applications d'Acquisition Coopérative de l'Information (ACI), opérant sur des sources d'informations hétérogènes placées dans un environnement ouvert et dynamique comme par exemple l'Internet.

Pour cela, nous avons opté pour le choix d'une organisation multi agents qui assure le fonctionnement du système, et résout des problèmes reliés aux différentes hétérogénéités. Ladite organisation est dotée d'agents mobiles qui parcourent les nœuds d'un réseau pour satisfaire nos besoins en matière d'informations. Ils permettent d'une part de réduire le nombre d'interactions distantes, et d'autre part de minimiser le volume de données transportées sur le réseau. Enfin un modèle informationnel est intégré à cette organisation, il englobe une ou plusieurs ontologies servant à définir l'univers du discours entre agents. Ce modèle nous résout le problème d'hétérogénéité sémantique.

Notre travail ne s'arrêtera pas là, car il y a des continuations à effectuer sur plusieurs volets, à savoir:

- L'architecture suppose que toutes les sources informationnelles sont dotées d'agents traducteurs pour répondre aux questions exigées, ceci est une limite du système, mais l'évolution du web (web sémantique, services web) et les efforts de standardisation vers XML [www], nous laisse réfléchir à substituer chaque agent traducteur par un fichier XML contenant toutes les informations qui existent dans la source, ainsi l'échange d'informations avec l'agent migrateur deviens en XML.

- Nous avons utilisé dans le modèle informationnel plusieurs ontologies, une par domaine, au lieu d'une seule globale pour des raisons qu'on a détaillées. Cependant, cette notion doit être approfondie, car ce choix doit prendre en compte l'aspect multi domaine des ontologies, donc à définir des liens sémantiques et de règles de passages entre les ontologies. [Tou03]

- Implémenter et valider l'architecture sur plusieurs domaines, et indépendamment de la plate forme MADKIT, pour servir à des applications sur Internet, en la développant par module: Organisationnel, Informationnel, Agent (Rôles).

Références Bibliographiques

Références Bibliographiques

[Agh86] AGHA.G
"Actors: A model of Concurrent Computation for Distributed Systems".
MIT Press, 1986.

[Att97] Attoui.A
"Un système multi-agents a temps réel".
Edition Eyrolles 1197.

[Bau92] Baujard.O
"Conception d'un environnement de développement pour la résolution de problèmes:Apport de l'intelligence artificielle et application à la vision". Thèse de l'université de Joseph Fourier. Grenoble I, 1992

[Bay&97] Bayardo.R, Bohrer.W, Brice.R, Cichocki.A, Fowler.G, Helal.A, Kashyap.V, Ksiezyk.T, Martin.G, Nodine.M, Rashid.M, Rusinkiewicz.M, Shea.R, Unnikrishnan.C, Unruh.A, Woelk.D
"Semantic Integration of Information in Open and Dynamic Environments" In Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 195-206, 1997

[BeIs02] Bernard.G, Ismail.L
"Apport des agents mobiles à l'exécution répartie"
Technique et science informatiques, Hermès, Paris, 2002, Vol. 21, No 6/2002, pp. 771-796.

[Bot99] Bot.P
"Functional scripting languages for the jvm". In 3rd annual European conference on java and object orientation, Arhus, Danmark, 1999.

[Bro86] Brooks R.A.
"A rebust layered control system for a mobile robot". IEEE Journal of Robotics and Automation, 2(1):14-23, 1986.

[Cha94] CHAIB-DRAA.B
"Coordination between agents in routine, familiar and unfamiliar situations". Rapport de recherche DIUL-RR-9401 du Département d'informatique, Université Laval, Ste-Foy, PQ, Canada, 1994.

[Cha96] CHAIB-DRAA.B
"Interaction between agents in routine, familiar and unfamiliar situation". International Journal of Cooperative Information Systems, Vol. 5, N° 1(1996) 1-25.

[Col&94] Coleman.D, Arnold .P, Bodoff.S, Dollin.C, Gilchrist.H and Hayes.F, Jeremaes.P "Object-Oriented Development: The FUSION Method".
Prentice Hall International: Hemel Hempstead, England, 1994.

[Dec&92] Keith S. Decker, Victor R. Lesser,
"Generalizing the partial global planing algorithm", International Journal of Intelligent and cooperative Information Systems, 1(2):319-346, Juin 1992.

[Dec&95] Decker .K, Lesser .V, Nagengra P.M.V, and Wagner .T
"MACRON: An Architecture for Multi-Agent Cooperative Information Gathering", In proceeding of the CIKM workshop on Intelligent Information Agent, December 1995.

[Del00] Della.R
"Une plate-forme de développement de systèmes multi-agents mobiles coopératifs sur Internet/Intranet"
Thèse de Magistère, USTHB 2000.

[Erm&80] Erman L. D., F.Hayes R., Lesser V.R., and Reddy R. D.
"The hearsay-II speech understanding system: Intgrating knowledge to resolve uncertainty". ACM Computing Surveys, 12(2), 1980.

[ErFr00] Ernest. J, Friedman.H
"Jess, the Java Expert System Shell". Distributed Computing Systems, Sundia National Laboratories, Livermore USA. 2000

[Fer89] Ferber.J
"Objet et agent: une étude de structures de représentation et de communication en intelligence artificielle"
Thèse de l'université ParisIV, Juin 1989.

[Fer95] Ferber.J
"Les SMA: vers une intelligence collective",InterEditions, Paris 1995.

[FeCa91] Ferber.J ,Carl.P
"Actors and agents as reflective concurrent object: a MeringIV perspective". IEEE transaction on systems, man and cybernetics,21(6),1991

[FeGh88] Ferber.J, Ghallab.M
"Problématique des univers multi-agents".
Actes des journées nationales du PRC-IA. Mars 1988

[FeGu98] Ferber.J ,Gutknecht. O
"A Meta-Model for the Analysis and Design of Organizations in Multi-Agent Systems".In proceeding of the third international conference on multi-agent systems(ICMAS'98),pp 128-135, Paris 1998.

[FiFr94] Finin.T, Fritzson.R
"KQML—A language and protocol for knowledge and information exchange". In Proceedings of the 13th International Distributed Artificial Intelligence Workshop, pp 127–136, Seattle, USA 1994.

[FIPA97] Fundation of Intelligent Physical Agents 1997.
Agent Communication Language specification.

[Fox81] Fox.M
"An organizational view of distributed Systems"
IEEE Trans.on man systems and cybernetics, 11(1), pp 70-79, 1981.

[Fug&98] Fuggetta.A, Picco. G. P, Vigna.G
« Understanding Code Mobility », IEEE Transactions on Software Engineering 24(5), 1998, pp. 342-361.

[Gal88] GALLIERS J.R.
"A Theoretical Framework for Computer Models of Cooperative Dialogue, Acknowledging Multi-Agent Conflit". PhD thesis, Open University, UK, 1988.

[Gas90] Gasser.L
"Social Conceptions of knowledge and action". Technical report ACT-AI-355-90,MCC, October 1990.

[Gas92] Gasser.L
"Boundaries, aggregation and identity: plurality issues for multi-agent systems". In Werner, E. et Demazeau, Y., editors, Decentralized Artificial Intelli-gence, pages 49–62, Amsterdam. Elsevier 1992.

[Gas95] Gasser.L
"Computational organization reseach". In 1st International conferance on multi-agent systems, San Francisco,V.Lasser Editor, MIT Press,1995.

[Gas&87] Gasser, L., Braganza, C., et Herman, N.
"MACE: a flexible testbed for distributed AI research". In Huns, M. N., Editor, Distributed Artificial Intelligence, pages 119-152. Pitman Publishers,1987.

[Gra95] Gray.R
"Agent Tcl: a transportable agent system"
In proc. CIKM Workshop 17 november 1995.

[Gru93] Gruber.T.R.
"A translation approach to portable ontologies",
Knowledge Acquisition, 5(2) : 199-220, 1993.

[Gut01] Gutknecht.O
"Proposition d'un modèle organisationnel générique de systèmes multi-agents : Examen de ses conséquences formelles, implémentatoires et méthodologiques" Thèse de Doctorat de l'université de Montpelier, France 2001.

[GuFe98] Gutknecht.O, Ferber.J
"A Model for Social Structures in Multi-Agent Systems"
Rapport de Recherche, LIRMM 98040, Montpellier, Avril 98.

[GuFe99] Gutknecht.O, Ferber.J
"Vers une méthodologie organisationnelle pour les systèmes multi-agents". In Marie-Pierre Gleizes Editor, Acte de journées francophones en intelligence Artificielle et systèmes multi-agents (JFIADSMA'99), Hèrmes Novembre 1999.

[Gut&00] Gutknecht.O, Ferber.J,Michel.F
"MadKit:Une architecture de plate-forme multi-agents générique"
Rapport de Recherche, LIRMM 00061, Mai 2000.

[Gut&01] Gutknecht.O, Ferber.J ,Michel.F
"Integrating tools and infrastructure for generic multi-agent systems". Autonomous Agents 2001.

[Har&95] Harrison.C, Chess.D, Kershenbaum.A
"Mobile agents: are they a good idea?"
Technical Report, IBM T.J Watson research centre, march 1995.

[Hat&91] Haton.J.P, Bouzid.N, Charpillet.F, Haton.M.C, Laasri.B, Laasri.H, Marquis.P, Mondot.T and Napoli.A
"Le raisonnement en intelligence artificielle (modèles, techniques et architectures pour les systèmes à base de connaissances".
InterEdition 1991. Paris.

[Hay85] Hayes-Roth.B
"A BlackBoard Architecture for control"
Artificial Intelligence 26(3), pp 321-351,1985.

[Hew91] Hewitt C.
"Open Information Systems Semantics for Distributed Artificial Intelligence", Artificial Intelligence 47(1-3)pp 79-106 1991.

[Hew&73] Hewitt .C,Bishop.P,Steiger.R.
"A universal modular actor formalism for artificial intelligence" In third international joint conferance on artificial intelligence, 1973.

[IsBe02] Ismail.L, Bernard.G
"Apport des agents mobiles à l'exécution répartie"
Technique et science informatiques RSTI série TSI-Agents et codes mobiles. Volume 21-n° 6/2002 pp. 771-796.

[IsHa00] Ismail.L, Hagimont.D
"Agents mobiles et client/serveur:évaluation de performance et comparaison".Technique et science informatiques. Volume 19-n° 9/2000

[Jen&98] Jennings.N, Wooldridge.M, Sycara.K
"Application of intelligent agent". In Agent Technology: Foundations, Applications, and Markets, N. Jennings and M. Wooldridge, eds. Springer Verlag. 1998.

[Kho94] Khoualdi .
"Filtrage d'alarmes pour un système automatisé par une approche multi-agents "
Thèse de l'université Paris VI. LAFORIA TH94/07. 18 Novembre 1994.

[Klu99] Klusch.M
"Intelligent Information Agents",
Springer-Verlag Berlin Heidelberg 1999.

[LaLe93] Labidi.S, Lejouad.W
"Form Distributed Artificial Intelligence to Multi-agent Systems".
INRIA Sophia Antipolis, August 1993.

[LaOs98] Lange.D, Oshima.M
"Programming And Deploying Java Mobile Agent With Aglets"
Addison Wesley 1998.

[Nasa] The NASA Website: http://www.hq.nasa.gov.
MARS CLIMATE ORBITER : http://www.exploringmars.com/missions/mco/

[Nod&98] Nodine.N, Perry.B and Unruh.A
"Experience with the InfoSleuth Agent Architecture". In Proceedings of AAAI-98 Workshop on Software Tools for Developing Agents, 1998.

[Nod&00] Nodine.M, Fowler.J, Ksiezyk.T, Perry.B, Taylor.M and Unruh. A
"Active Information Gathering in InfoSleuth". In International Journal of Cooperative Information Systems 9:1/2, 2000, pp. 3-28.

[Rod&01] Rodriguez Silva.A, Romao.A, Deugo .D, Mira da Silva.M
"Towards a reference model for surveying Mobile Agent Systems"
Autonomous Agents ans Multi-Agent Systems, vol4, number 3,
pp 187-231, Kluwer Academic Publishers.2001.

[Oat&97] Oate.T, Prasad. M. V, Lesser. V
"Cooperative Information Gathering: A Distributed Problem Solving Approach".IEEE procedings on software engineering, special issue on agent based systems, volume 144, Number 1,pp 72-88, January 1997.

[Sah99] Sahai.A
"Conception et réalisation d'un gestionnaire mobile de réseaux fondé sur la technologie d'agent mobile"
Thèse doctorat, Université de Renne 1, 25 janvier 1999, France.

[Sca&96] Scalabrin.E, Vandenberghe.L, Azevedo.H.D, Barthes.J.P
"A Generic Model of Cognitif Agent to Develop Open systems ».
13th Brazilia, Symposium on Artificial Intellgence, Borges D. L. et Kaestner C.A.A (Ed.), Springer Verlag, 1996.

[Sho93] Shoham.Y
"AGENT0: A simple agent language and its interpreter". In Proceeding of the ninth National Conference on Artificial Intelligence (AAAI91), USA 1993.

[SiHa01] Sibertin-Blanc.C, Hanachi.C
"Protocol Moderator as Active Middle-Agents in MAS "
Journal of Autonomous Agents and Multiagent Systems, 2003.

[Stu&01] Maja Štula, Darko Stipaničev, Mirjana Bonković
"Questions and Possible Answers about ontologies Concerning Intelligent Agents". International Conference on Software, Telecommunications and Computer Networks, Bari Italy, October 2001.

[Syc89] Sycara.K
"Multiagent Compromise via Negotiation". In Distributed Artificial Intelligence, Vol. 2, L. Gasser and M. N. Huhns (eds.), Morgan Kaufmann Publishers, Los Altos, CA, 1989, pp. 119-137.

[Syc&98] Sycara.K, Pannu A.S
"The RETSINA multiagent system: Towards integrating planning, execution and information gathering" Proceedings of the 2 nd International Conference on Autonomous Agents (Agents '98). New York, 350-351, 1998.

[Tou03] Tout.H
"ingénierie des systèmes d'acquisition coopérative d'information "
thèse de doctorat de l'université Toulouse I France, septembre 2003.

[OMG] UML Resource Page of Object Management Group
http://www.omg.org/technology/uml/index.htm

[Vin93] VINCENT.C
"Etude et mise en œuvre du paradigme multi-agents de ATOME A GTMAS",
Thèse de Doctorat de l'université de Nancy 1, France 1993.

[Whi97] White.J
"Telescript technologie : an introduction to the language"
In J.Bradshaw(ed.) Software Agent, AAAI/MIT Press, 1997

[WoJe94] Wooldridge.M, Jennings.N
"Towards a theory of cooperative problem solving"
Damazeau.Y, Muller.J-P, Parram.J (Ed).Odense, Danemark,1994.

[WoJe95] Wooldridge.M, Jennings.N
"Intelligent agents: theory and practice"
Knowledge Engineering Review, January 1995.

[Woo&99] Wooldridge.M, Jennings.N, Kenny.D
"A methodology for agent-oriented analysis and design" In proceeding of the third international conference on autonomous agents(Agents99),pp 69-76,seatle, USA May 1999.

[Woo&00] Wooldridge.M, Jennings.N, Kenny.D
"GAIA: a methodology for agent-oriented analysis and design"
3(3), pp 285-312, 2000.

[WWW] The World Wide Web Consortium
http://www.w3.org

[Zam&01] Zambonelli.F, Jennings.N, Wooldridge.M,
"organisational rules as an abstraction for the analysis and design of MAS"
Springer-Verlag 2001.

Résumé

Ce mémoire présente tout d'abord un état de l'art sur les agents et les systèmes multi agents (SMA) en considérant deux niveaux: le niveau microscopique relatif aux agents et le niveau macroscopique relatif aux SMA vus comme une société d'entités coopérantes dans une organisation. Ensuite nous abordons l'acquisition coopérative de l'information (ACI), qui se présente comme une extension intelligente de la recherche d'information classique, notamment lorsque les sources d'informations sont distribuées sur un ou plusieurs serveurs, et dans un réseau à grande échelle comme l'Internet, où le terme *hétérogénéité* (conceptuelle, sémantique et celle des langages) intervient à plusieurs niveaux.

L'objectif de notre travail est de concevoir une architecture basée sur une organisation multi-agents avec des agents mobiles pour l'ACI, qui assure le fonctionnement du système, et elle doit prendre en compte les différentes hétérogénéités rencontrés lors de ce processus, car une organisation est composée d'entités coopérantes et autonomes, une multitude de fonctions (rôles) qui sont assurées. Nous associons à chaque agent un rôle spécifique, des buts, des règles d'interactions et de lui partager des sources d'informations. Ce modèle organisationnel interagit avec un modèle informationnel qui englobe une ou plusieurs ontologies définissant l'univers du discours entre agents.

Un prototype d'exécution a été développé avec la plate forme MADKIT pour valider une partie de notre proposition.

Mots clés

Système Multi Agents, Agent mobile, Organisation multi agents, modèle AALAADIN, modèle GAIA, Acquisition Coopérative de l'Information, Ontologie, MADKIT.

Abstract

This memory presents a state of the art on the agents and the multi agents systems (*MAS*) by considering two levels: the microscopic level relating to the agents and the macroscopic level relating to the MAS seen like a company of cooperating entities in an organization. Then we presents the cooperative information gathering (CIG), which is presented in the form of an intelligent extension of traditional information retrieval, in particular when the sources of information are distributed on one or more server in a network on a large scale like the Internet.

The objective of our work is to conceive architecture based on a multi agents organization with mobile agents for the CIG, which ensures the operation of the system, and it must take into account various heterogeneities during this process. We associate each agent with a specific role, goals and rules of interactions. This organisational model interacts with an informational model which includes one or more ontology defining the universe of the speech between agents.

A prototype of execution was developed with the *MADKIT* platform to validate part of our proposal.

Key words

Multi Agents System, mobile Agent, Organization of agents, *AALAADIN* model, *GAIA* model, Cooperative Information Gathering, Ontology, *MADKIT platform*.