République algérienne démocratique et populaire

Ministère de L'Enseignement Supérieur et de la Recherche Scientifique
Université L'arbi ben m'hidi Oum El Bouaghi
Faculté des Sciences de l'Ingénieur
Département d'Informatique

ECOLE DOCTORALE INFORMATIQUE DE L'EST 2007/2008
PÔLE DE CONSTANTINE

Mémoire Présenté pour l'obtention du diplôme de Magister en Informatique

Option : Intelligence Artificielle (IA)

Titre du Mémoire

Une approche de simulation participative à base d'agents pour la négociation dans le e-commerce

par

Redha MIROUD

Composition du Jury

Dr. KHOLLADI Mohamed kheireddine	**M.C**	**Université de Constantine**	**Président**
Dr. KAZAR Okba	**M.C**	**Université de Biskra**	**Rapporteur**
Dr. MAAMRI Ramdane	**M.C**	**Université de Constantine**	**Examinateur**
Dr. CHAOUI Allaoua	**M.C**	**Université de Constantine**	**Examinateur**
Dr. MOKHATI Farid	**M.C**	**Université d'Oum El Bouaghi**	**Examinateur**

2009/ 2010

ملخص — التجارة الإلكترونية والناشئة باعتبارها بديلا حيويا و شكلا جديدا من المعاملات التجارية، هذا من ناحية و من ناحية أخرى، فإن النموذج الوكيل أصبح هو الآخر نموذجا موثوقا لتصميم نظام يقوم على التفاعل بين كيانات مستقلة، والتي في بعض الأحيان تتطور لتصبح أكثر تطورا وتعقيدا. فالآلية الأساسية للتجارة الإلكترونية هي بروتوكول التفاوض الذي يضمن في النهاية اتفاقا بين البائع والمشتري. حاليا المحاكاة بالمشاركة والتي تمكننا من إشراك عدد من المستخدمين فيها أثناء تنفيذ هذا النموذج صارت نهجا موثوقا به لمراقبة حسن سلوك النظام المعقد. في هذا النهج، سوف يكون لدى المستخدم السيطرة على عنصر مكون لنموذج المحاكاة وبإمكانه التحكم فيه كيفما شاء أثناء التنفيذ، عن طريق تعديل معايير معينة للعناصر. الهدف من هذا العمل يتمثل في اقتراح مقاربة تشاركية من المحاكاة تحتوي على عناصر عاملة لأجل بروتوكول التفاوض في مجال التجارة الإلكترونية. لأجل ذلك استخدمنا في نموذجنا آلية تفاوض تقوم على أساس بروتوكول مزاد إنكليزي معكوس، حيث اقترحنا نموذجا جديدا لوكيل مساومة مبني على أساس مشاركة المستخدمين أثناء تنفيذ النموذج. ستكون العناصر المفاوضة المزودة بقدرة التعلم وبشكل معين من الاستقلالية في اتخاذ القرارات ضرورية للقيام بعملية استخراج المعرفة من ناحية، وإلى أتمتة عملية التفاوض من ناحية أخرى. يلعب المستخدم، من خلال واجهة مخصصة، الدور الذي أوكل إليه في إطار المحاكاة للسيطرة على وكلاءنا وتطوير قاعدتهم السلوكية، فالوكيل يلاحظ، ثم يقترح السلوكيات التي يمكن للمستخدم تحسينها. بالتالي وجب وضع نظام تعلم تشاركي بفضل هذه التقنية يمكننا زيادة نطاق نشاط وكلائنا تكييف الوكلاء مع أي مكونات جديدة في النظام) مما يجعل المحاكاة أكثر واقعية. مع مرور الوقت ستتطور معارف الوكلاء، ويتوجب عليهم أن يكونوا قادرين على التعلم وإعادة النظر في قاعدتهم المعرفية.

كلمات البحث: محاكاة بالمشاركة، التفاوض في مجال التجارة الإلكترونية، الأنظمة المتعددة الوكلاء.

RESUME— Le e-commerce s'impose comme une variante incontournable et une nouvelle forme de transaction commerciale. D'autre part, le paradigme agent est devenu lui aussi un modèle fiable pour la conception de système basé sur l'interaction d'entités autonomes et qui parfois le développent vers une forme plus sophistiquée et compliquée. Le mécanisme de base du e-commerce est le protocole de négociation qui assure en fin un accord entre l'acheteur et le vendeur. Actuellement la simulation participative qui consiste à faire impliquer les utilisateurs dans la simulation, durant l'exécution du modèle est devenue une approche fiable pour l'observation du bon comportement de système complexe. Dans cette approche, un utilisateur aura le contrôle d'un composant du modèle simulé et il peut le manipuler à son gré durant l'exécution, en modifiant certains paramètres du composant. L'objectif du travail consiste à proposer une approche de simulation participative à base d'agents pour le protocole de négociation dans le e-commerce. Pour cela nous avons utilisé dans notre modèle un mécanisme de négociation qui est basé sur un protocole d'enchères anglaises inversées, où on a proposé un nouveau modèle d'agent négociateur basé sur la participation des utilisateurs durant l'exécution de modèle. Des agents négociateurs dotés de capacités d'apprentissage et d'une certaine forme d'autonomie décisionnelle seront utiles pour automatiser ce processus d'extraction de connaissances d'une part, et pour automatiser le processus de négociation d'autre part. L'utilisateur joue, par le biais d'une interface dédiée, le rôle qui lui a été confié au sein de la simulation pour contrôler nos agents et développer leur base comportementale, l'agent observe, puis propose des comportements qui peuvent être améliorés par l'utilisateur. Cela revient à mettre en oeuvre un système d'apprentissage participatif. Grâce à cette technique nous pouvons accroître les champs d'action de nos agents (adaptation des agents à des nouvelles configurations de système) rendant plus réalistes les simulations. Leurs connaissances évolueront dans le temps, les agents devront être capables d'apprendre et de remettre en question leur base de connaissances.

Mots clés : simulation participative, négociation dans le e-commerce, système multi-agent.

ABSTRACT— The e-commerce is emerging as a vital alternative new form of business transaction. On the other hand, the agent paradigm has also become a reliable model for system design based on the interaction of autonomous entities, which sometimes develop into a more sophisticated and complicated form. The basic mechanism of e-commerce is the negotiation protocol which provides at the end an agreement between buyer and seller. Currently the participatory simulation which consists in the participation of the users in simulation, during the execution of the model became a reliable approach for the observation of the good behavior of complex system. In this approach, the user will have the control of a component of the simulated model and can manipulate it during the execution, by modifying some component's parameters. The objective of the work consists in proposing an approach of participative simulation containing agents for the protocol of negotiation in the e-commerce. For this purpose we used in our model a negotiation mechanism which is based on a protocol of reversed English auction, where we proposed a new model of bargaining agent based on the participation of the users during the execution of model. Some negotiation Agents equipped with learning capabilities and a certain form of autonomous decision will be useful to automate this process of knowledge extraction in one hand, and to automate the negotiation process on the other hand. The user plays, by the means of a dedicated interface, the role to which he is assigned within simulation to control our agents and develop their behavioral base, the agent observes, then proposes behaviors which can be improved by the user. That requires implementing a system of participatory learning. With this technique we can increase the sphere of activities of our agents (adaptation of the agents to new system configurations) making simulations more realistic. Their knowledge will evolve in throughout time, the agents will have to be able to learn and always question their base of knowledge.

Keywords: participatory simulation, negotiation in e-commerce, multi-agent system.

Remerciements

Je commence, avant toute chose, par remercier ALLAH le tout puissant pour son aide et de m'avoir doté de beaucoup de force physique et morale et du courage nécessaire pour l'élaboration de ce modeste travail. Toute thèse est le résultat de très nombreuses collaborations, tant au niveau scientifique, qu'au niveau humain . Il est très difficile de toutes les mentionner. Je remercie en particulier:

Mon directeur de thèse Monsieur Okba KAZAR, Maître de conférence à l'Université Mohamed Khider Biskra, pour m'avoir dirigé tout au long de cette thèse. J'ai beaucoup bénéficié de leurs conseils, qu'ils m'ont fournie et de leurs suggestions pertinentes pour la mise au point de ce travail, sans oublier sa qualité humaine qu'il m'a manifestés durant la réalisation de ce travail.

Monsieur Mohamed kheireddine KHOLLADI, Maître de conférence à l'Université Mentouri de Constantine, qui m'a fait l'honneur de présider le jury de cette thèse malgré ses nombreuses obligations.

Monsieur Ramdane MAAMRI, Maître de conférence à l'Université Mentouri de Constantine, qui m'a fait l'honneur d'accepter de faire participer au jury pour évaluer cette thèse.

Monsieur Allaoua CHAOUI, Maitre de Conférence à l'Université Mentouri de Constantine, qui m'a fait l'honneur d'accepter de faire participer au jury pour évaluer mon travail.

Monsieur Farid MOKHATI, Maître de conférence à l'Université L'arbi ben m'hidi d'Oum El Bouaghi, qui m'a fait l'honneur d'accepter de faire participer au jury pour évaluer cette thèse.

Enfin nous remercions tous ceux qui nous ont aidés de prés ou de loin à la réalisation de ce travail.

A mes parents,
À mes soeurs et mon frère Abdelhamid,
A toute ma grande famille,
A tous mes amis et mes collègues,
Aux personnes qui me sont chère Car ils
sont tous dans mon coeur pour m'encourager dans les moments difficiles.

Sommaire

Table des matieres

Table des illustrations

Liste des Tables

Liste des abréviations

ACC	*Agent Communication Channel*
ACL	*Agent Communication Language*
AEIOU	Agents, Environnements, Interactions, Organisations, Utilisateur
AFCET	Association Française pour la Cybernétique Économique et Technique
AGR	*Agent/Groupe/Rôle*
AID	*Agent Identifier*
AMRM	*Agent Management Reference Model*
AMS	*Agent Management Service*
AP	*Agent Platform*
API	*Application Programming Interface*
BDI	*Beliefs, Desires and Intentions*
BtoB	*Business to Business*
BtoC	*Business to Consumer*
CFP	*Call For Proposal*
CNP	*Contract-Net Protocol*
COMORE	Contrôle et modélisation de ressources renouvelables
CtoB	*Consumer to Business*
CtoC	*Consumer to Consumer*
DF	*Directory Facilitator*
EDI	*Electronic Data Interchange*
EtoE	*Employe to Employe*
FIPA	*Foundation for Intelligent Physical Agents*
FIPA-OS	*Foundation for Intelligent Physical Agents-Open Source*
FTP	*File Transfer Protocol*
GNP	plateforme générique de négociation
GtoC	*Government to Consumer*
GUI	*Graphical User Interface*
HAP	*Home Agent Platform*
HTML	*Hyper Text Markup Language*
HTTP	*Hypertext Transmission Protocol*
IA	Intelligence Artificielle
IAD	Intelligence Artificielle Distribuée
IAP	Intelligence Artificielle Parallèle
IBM	*International Business Machines*
IIOP	*Internet Inter-ORB Protocol*
IP	*Internet Protocol*
ISO	*International Organization for Standardization*
J2EE	*Java 2 Enterprise Edition*
J2ME	*Java 2 Micro Edition*
J2SE	*Java 2 Standard Edition*
JADE	*Java Agent DEvelopment Framework*
JAL	*Jack Agent Language*
JAT	*Java Agent Template*
JAT-Lite	*Java Agent Template-Lite*
JDK	*Java Development Kit*
JESS	*Java Expert System Shell*
JRE	*Java Runtime Environment*

JVM	*Java Virtual Machine*
KIF	*Knowledge Interchange Format*
KQML	*Knowledge Query Modeling Langage*
KS	*Knowledge Source*
LAN	*Local Area Network*
MacOS	*Macintosh Operating System*
MAGNET	*Multi AGent NEgotiation Testbed*
MAS	*Multi-Agent Systems*
MIB	*Management Information Base*
MIRIAD	Modélisation des Interactions et Recherches en Intelligence Artificielle Distribuée
MTP	*Message Transport Protocol*
MTS	*Message Transport Service*
PDA	*Personal Digital Assistant*
PDU	*Protocol Data Unit*
PMV	*places de marché virtuel*
RADL	*Reticular Agent Definition Language*
RDF	*Resource Description Framework*
RDP	Résolution Distribuée des Problèmes
RMA	*Remote Monitoring Agent*
RMI	*Remote Method Invocation*
RPG	*Role Playing Games*
SAD	Systèmes d'Aide à la Décision
SL	*Semantic Language*
SMA	Systèmes Multi-Agents
TCP	*Transmission Control Protocol*
TINA	*Telecommunication Information Networking Architecture*
TMN	*Telecommunication Management network*
UDP	*User Datagram Protocol*
UML	*Unified Modeling Language*
URL	*Uniform Resource Locator*
VM	*Virtual Machine*
WAP	*Wireless Application Protocol*
XML	*eXtensible Markup Language*

Introduction Générale

1. Présentation du domaine de recherche

L'intelligence artificielle, comme domaine de l'informatique concerné entre autres par le développement de simulations de l'intelligence humaine avait seulement été, jusqu'à récemment, utilisée dans la modélisation de la cognition individuelle. Mais, dans les années 1980, un intérêt croissant pour l'intelligence artificielle distribuée et la technologie *agent* a conduit au développement de modèles qu'ils prenaient en compte des agents autonomes en interactions, pouvaient être appliqués à la simulation de sociétés humaines. De plus, les chercheurs en intelligence artificielle ont prêté une grande attention durant les dix dernières années aux techniques d'apprentissage, lançant ainsi la recherche sur la piste de modèles capables d'apprendre, utiles à la fois pour simuler les processus cognitifs des individus et pour modéliser des sociétés entières qui s'adaptent au fil du temps à de nouvelles circonstances.

Les Systèmes Multi-Agents *(SMA)* représente une branche de l'intelligence artificielle distribuée qui s'intéresse plus particulièrement à la conception d'entités artificielles (homogènes ou hétérogènes) capables de s'organiser efficacement pour accomplir collectivement les tâches qui leur sont demandées. Ils permettent, de manière idéale mais dynamique, de modéliser et de représenter des problèmes complexes (exemples pilotage et gestion d'entreprises virtuelles, marchés électroniques,...etc.). Les principales qualités des modélisations multi-agents sont leur capacité d'intégration, adaptation et leur flexibilité. Ces systèmes ont ainsi la possibilité de solutionner des difficultés complexes tout en ayant les avantages de la résolution distribuée et de l'intelligence artificielle. Ils permettent aussi de faire intervenir des types d'interaction assez complexes tels que la coopération et la coordination d'actions. Notre recherche se place dans le domaine des protocoles de négociation, Ce domaine est très important pour deux raisons. La première est que la négociation s'est imposée comme étant la meilleure solution pour résoudre des conflits au sein d'un *SMA*. D'autre part, de son point de vue économique, est à la base de nombreuses applications commerciales, notamment dans le commerce électronique avec les systèmes de vente aux enchères. Notre objectif est de fournir un protocole de négociation permettant de réaliser un système d'enchères. Nous souhaitons fournir un modèle de négociation, et son implémentation, ce qui permet de l'utiliser plus facilement pour créer une application de négociation.

La simulation multi-agent est utilisée pour mieux comprendre la dynamique des systèmes complexes, en particulier les phénomènes de négociation entre entités, elle est fondée sur l'idée qu'il est possible de représenter sous forme informatique le comportement des entités qui agissent dans le monde et qu'il est ainsi possible de représenter un phénomène comme le fruit des interactions d'un ensemble d'entités disposant de leur propre autonomie opératoire. Les modèles qui en résultent sont composés d'un ensemble d'entités dynamiques, ou agents, dont les comportements dépendent de leurs interactions et de l'environnement au sein duquel ils évoluent. Ce paradigme garanti l'autonomie décisionnelle et la distribution des interactions entre les entités du système, et facilite leur implémentation dans un processus de simulation. Les *SMA* apparaissent donc comme des outils de modélisation et de simulation, ils sont totalement adaptés à la complexité des systèmes tels que les sociétés humaines et les écosystèmes. Ils peuvent être à ce titre appliqués à différents domaines (santé, transport, écologie, télécommunication, …etc).

En ce qui concerne la simulation participative, elle consiste à faire impliquer les utilisateurs dans la simulation, durant l'exécution du modèle. Autrement dit, un utilisateur aura le contrôle d'un composant du modèle simulé et il peut le manipuler à son gré durant l'exécution, en modifiant certains paramètres du composant.

Pour lier les deux idées, il suffit de procéder en une modélisation multi-agent du système à étudier, ce qui va donner une simulation multi-agent, puis faire participer les utilisateurs par la suite durant l'exécution du modèle pour avoir une simulation participative. Le composant manipulé par l'utilisateur sera donc un agent. Cette nouvelle approche de mener une simulation sert à l'*enseignement* et l'*apprentissage* de certains phénomènes complexes. L'utilisateur qui n'est donc dans ce cas qu'un apprenant, sera bien convaincu de la façon dont le système réel se comporte...etc. On peut imaginer un autre usage, comme la *modélisation de l'expertise humaine*. Sachant que celle ci est souvent difficile à modéliser, il suffit de concevoir un SMA et de donner le contrôle de chaque agent à un expert, l'agent ayant la capacité d'apprendre et d'interagir ...au fur et à mesure.

2. Objectif de notre travail

Aujourd'hui le commerce électronique s'impose comme une variante incontournable comme nouvelle forme de transaction commerciale, il apparaît comme une des applications majeures du Web et prend une place croissante dans la conduite des affaires au sein de l'économie traditionnelle. D'autre part, le paradigme agent est devenu lui aussi un modèle fiable pour la conception de système basé sur l'interaction d'entités autonome et qui parfois le développent vers une forme plus sophistiquée et compliquée. Le mécanisme de base du e-commerce est le protocole de négociation qui assure en fin un accord en le demandeur et le vendeur. Donc le besoin d'agents automatiques capables de négocier avec les autres pour le compte d'un utilisateur devient de plus en plus fort. De plus, l'utilité d'utiliser un agent pendant les négociations est parfaitement justifié par l'explosion du nombre de messages échangés entre les agents.

Actuellement la simulation participative et qui consiste à faire impliquer les utilisateurs dans la simulation, durant l'exécution du modèle est devenue une approche fiable pour l'observation du bon comportement de système complexe. Dans cette approche, un utilisateur aura le contrôle d'un composant du modèle simulé et il peut le manipuler à son gré durant l'exécution, en modifiant certains paramètres du composant.

L'objectif du travail consiste à proposer une approche de simulation participative à base d'agents pour le protocole de négociation dans le e-commerce. Pour cela on a développer un système de négociation pour les enchères multi-attributs anglaise inversées, et de reproduire à l'aide de ce système multi-agents les comportements des utilisateurs et de simuler l'adaptation de nos agents à des nouvelles configurations, l'approche employée est donc celle de la simulation participative.

3. Que sont les simulations multi-agents participatives ?

Ce concept est né de travaux sur la formalisation des connaissances acquises par les êtres Humains [Drogoul et al 02]. "Ce sont *des expériences menées en laboratoires ou à travers le réseau Internet, avec des participants humains et qui s'inscrivent dans une démarche multi-agents"* [Guyot 06].Ces simulations sont caractérisées par deux propriétés :

– les agents dans ces simulations participent à la simulation comme les humains et ne sont pas des entités qui fournissent des services.

– les utilisateurs accèdent à la simulation exactement comme le font ou le feraient des agents. En d'autres termes, chaque utilisateur est assis à un poste de travail, et toutes les interactions, conçues comme des interactions entre agents, se font par le biais de l'ordinateur.

Donc pour [Guyot 06] les simulations multi-agents participatives sont des expériences, Les trois composantes de l'expression "*simulations multi-agents participatives*" sont essentielles et décrivent la spécificité de ce type d'expériences. Le terme de "*simulations*" décrit le rapport de ces expériences avec la réalité. Les simulations multi-agents participatives sont conçues pour explorer, modéliser et reproduire des phénomènes réels en les simulant [Guyot 06]. Le terme "*multi-agents*" est central dans cette expression. Les simulations multiagents participatives ce sont des expériences informatiques, constituent un type très spécifique et particulier de simulations participatives. Les simulations multi-agents participatives entretiennent un lien fort avec le domaine des agents : elles s'inscrivent dans la tradition des simulations multi-agents. Chaque participant associé à l'agent qu'il contrôle peut constituer un agent au sens informatique, avec toutes les propriétés désirées des agents dans le domaine de l'intelligence artificielle distribuée : un tel agent est intelligent, autonome, pro-actif, guidé par des buts, etc. Les simulations multi-agents participatives constituent un cadre idéal pour explorer, modéliser et reproduire des comportements collectifs sous la forme de simulations et de systèmes multi-agents.

En réalité, les simulations multi-agents participatives permettent de construire des systèmes multi-agents capables d'effectuer des tâches et de résoudre collectivement des problèmes aussi complexes que les tâches effectuées et les problèmes résolus par les communautés humaines. Ces systèmes sont, comme les groupes sociaux, composés d'agents qui sont autonomes et qui ont leurs propres buts.

4. Pourquoi des simulations multi-agents participatives ?

Comme nous savons tous que l'intelligence artificielle est la simulation de comportement humain, donc elle est fondée sur le mimétisme. Elle vise à créer des systèmes artificiels qui reproduisent des comportements intelligents ou qui ont des compétences rattachées à l'intelligence. Les systèmes artificiels intelligents peuvent par exemple résoudre des problèmes, apprendre et s'adapter. En revanche, les groupes sociaux humains traitent de manière collective des problèmes qui sont autrement plus compliqués, et s'en inspirer nécessite de pouvoir les observer, les comprendre et les modéliser.

L'approche que nous avons proposée pour construire des systèmes artificiels inspirés des comportements sociaux, il s'agit de faire participer des utilisateurs humains à des simulations inspirées du réel, c'est le principe des simulations multi-agents participatives [Guyot 06]. Cette approche a deux avantages essentiels. D'une part, tous les détails du contrôle exercé par les humains sont enregistrés et peuvent être analysés afin de révéler leurs stratégies et leurs comportements. D'autre part, la médiation de l'outil informatique force les acteurs à avoir un comportements délibéré et donc plus conscient, ce qui favorise la discussion lors des séances de verbalisation qui suivent les expériences de simulation.

Parvenir à modéliser les comportements sociaux d'acteurs dans des situations dans lesquelles ils utilisent une expérience fortement liée au domaine de commerce électronique, et plus précisément les protocoles de négociation reste toujours une activité difficile. Une solution possible à ce problème d'extraction de connaissances situées réside dans les méthodologies participatives

de conception, où les acteurs sont appelés à mener un rôle actif dans la définition des comportements des agents de la simulation par l'intermédiaire de jeux de rôles. Donc les simulations multi-agents participatives permettent à la fois d'explorer et de modéliser des comportements collectifs (objectif de compréhension) et de les reproduire (objectif d'ingénierie). L'objectif de compréhension des phénomènes collectifs relève de la tradition des recherches en intelligence artificielle distribuée [Guyot 06].

5. Organisation du mémoire

Après cette introduction, et dans le but d'aborder les différents aspects, et afin d'aboutir à l'objectif projeté, nous avons choisi de structurer la thèse de la manière suivante:

Le premier chapitre introduit les concepts de base de l'intelligence artificielle distribuée et des systèmes multi-agents.

Le second chapitre est consacré à la présentation de la norme *FIPA* et la plateforme *JADE* pour les systèmes multi-agents ainsi que la technologie des environnements de développement orientée agent.

Dans le troisième chapitre, qui a pour objectif de présenter la problématique générale de la simulation des systèmes complexes. On a expliqué la notion de la simulation multi-agents participative. Et les domaines qui utilisent cette approche.

Le quatrième chapitre fait intervenir les éléments de base constituant le modèle de négociation adopté. Il présente le déroulement de la négociation comme processus itératif, qui inclut des protocoles et des stratégies variant selon les contextes. Ainsi que les principes et mécanismes utilisés dans les systèmes multi-agents, en particulier les différentes techniques de négociation utilisées dans le commerce électronique. Ce chapitre se termine par l'exposition de quelques systèmes de commerce électronique basé agents.

Le cinquième chapitre sera consacré, à l'approche proposée. Il présente les motivations pour le choix de l'approche agent, la spécification du système proposée et tous les concepts et agents déployés pour le développement de ce système. Par ailleurs, nous avons détaillé le processus de négociation utilisé ainsi que la communication entre agents.

Le sixième est le dernier chapitre, illustre les choix techniques et la mise en oeuvre du modèle de la simulation multi-agents participative proposés pour la négociation dans le domaine de commerce électronique, et il permet la validation de cette approche, et de donner des explications supplémentaires concernant certains aspects qui n'ont pas pu être bien éclairés dans le chapitre précédent. Cette validation est consolidée par une simulation faite dans l'environnement *JADE*.

En conclusion, nous dressons un bilan du travail effectué, ainsi l'outil utilisé, en mettant en évidence les différents points abordés par ce travail, et les perspectives de recherche ouvertes et envisagées qui en découlent.

Chapitre 1
Intelligence artificielle distribuée et systèmes multi-agents

1. Introduction

Depuis quelques années, les systèmes multi-agents ont pris une place de plus en plus importante en informatique, que ce soit dans le domaine de l'intelligence artificielle, dans ceux des systèmes distribués, de la robotique, ou même dans ce champ disciplinaire nouveau qu'est *la vie artificielle*, Les recherches dans le domaine de systèmes multi-agents poursuivent deux objectifs majeurs : le premier concerne l'analyse théorique et expérimentable des mécanismes d'auto-organisation qui ont lieu lorsque plusieurs entités autonomes interagissent : le second s'intéresse à la réalisation d'artefacts distribués capables d'accomplir des tâches complexes par coopération et interaction. Leur position est donc double :d'un côté elles se placent au sein des sciences cognitives et sociables (psychologie, éthologie, sociologie, philosophie…)et naturelles (écologie, biologie…) pour à la fois modéliser, expliquer et simuler des phénomènes naturels, et susciter des modèles d'auto-organisation ;de l'autre, elles se présentent comme une pratique, une technique tendue vers la réalisation de systèmes informatiques complexes à partir des concepts d'agents, de communication, de coopération et de coordination d'actions.

Ce chapitre a pour objectif d'introduire les systèmes multi agents qui constituent un des piliers de notre travail. La recherche dans ce domaine suscite un certain nombre d'interrogations : Qu'est-ce qu'un agent qui interagit avec d'autre agents ? Comment peuvent-ils coopérer ? Quels sont les modes de communication nécessaire pour qu'ils se répartissent des tâches et coordonnent leurs actions ? Quelle architecture peut-on leur donner pour qu'ils satisfassent leurs buts ? Nous essayons à travers ce chapitre d'analyser les aspects de base des systèmes multi agents, ainsi que de faire un état de l'art sur leurs applications, notamment, dans les domaines de la simulation et du commerce électronique. Enfin, nous concluons par les raisons qui nous ont fait choisir cette approche pour répondre à notre problématique. Avant de commencer, il serait intéressant de comprendre quelques concepts comme l'intelligence artificielle distribuée *(IAD)* et ses axes fondamentaux dans la recherche.

2. L'Intelligence Artificielle Distribuée

2.1. Historique

L'expression intelligence artificielle *(IA)* est employée pour la première fois *(1955-1970)* par *John McCarthy*. Il fonde l'*IA* sur le postulat mécaniste qui veut que toute activité intelligente soit modélisable et reproductible par une machine. L'*IA* a pour but de faire exécuter par l'ordinateur des tâches pour lesquelles l'homme, dans un contexte donné, est aujourd'hui meilleur que la machine. Mais l'évolution des domaines d'application de l'intelligence artificielle pour recouvrir des domaines complexes et hétérogènes a montré les limites de l'approche classique de l'*IA* qui s'appuie sur une centralisation de l'expertise au sein d'un système unique. Les travaux menés au début des années *70* pour palier aux insuffisances de l'*IA* ont contribué à la naissance d'une nouvelle discipline : l'intelligence artificielle distribuée *(IAD).*

A la différence de l'*IA* classique qui modélise le comportement intelligent d'un seul agent, l'*IAD* s'intéresse à des comportements intelligents qui sont le produit de l'activité coopérative de plusieurs agents. Le passage du comportement individuel au comportement collectif est considéré non seulement comme une extension mais aussi comme un enrichissement de l'*IA*. Donc l'*IAD* a pour but de remédier aux insuffisances de l'approche classique de l'*IA* en proposant la distribution de l'expertise sur un groupe d'agents devant être capable de travailler et d'agir dans un

environnement commun et résoudre des problèmes complexes. D'où la naissance de notions nouvelles en *IA*, telles que la coopération et la coordination [Labidi 94].

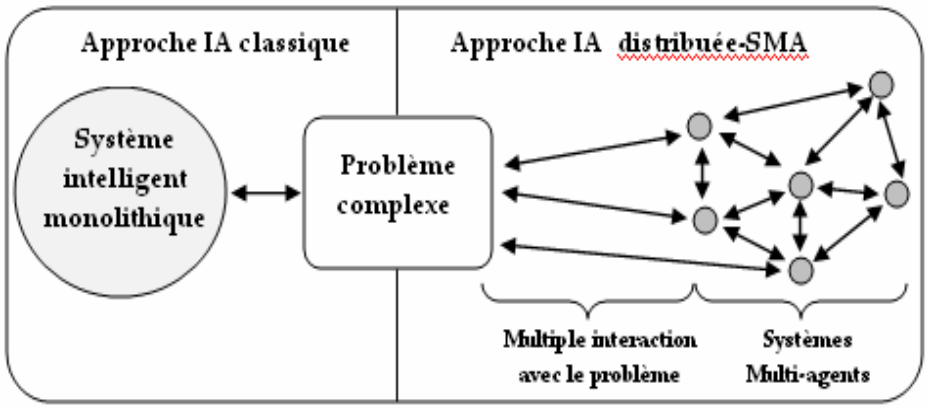


Figure 1.1. Comparaison entre les approches *IA* et *IAD*/*SMA* [Romaric 03].

En 1973, le premier système d'*IAD* a pu voir le jour. Il s'agit du système *HEARSAY* pour la reconnaissance de la parole, basé sur l'architecture de blackboard[1]. Dans les recherches qui ont suivi, diverses approches liées à la distribution se sont succédées, notamment : les acteurs[2] de Hewitt, la société de l'esprit de Minsky, le système *DVMT* de Lesser, le Contract Net[3] de Smith, le système Mace de Gasser et les micro-robots de Brooks [Agha 86].

2.2. Axes fondamentaux dans la recherche en IAD

L'*IAD* propose la distribution de l'expertise sur un groupe d'agents devant être capables de travailler, d'agir et de s'organiser pour arriver à résoudre un problème posé, dans un environnement commun et résoudre les conflits éventuels. D'où la naissance de notions nouvelles en *IA*, telles que la coopération, la coordination d'actions, la négociation et l'émergence. L'*IAD* peut alors être définie comme étant la branche de l'*IA* qui s'intéresse à la modélisation de comportement *intelligent* par la coopération entre un ensemble d'agents. Après avoir défini l'*IAD*, nous présentons les trois axes fondamentaux dans la recherche en *IAD* [Labidi 94] :

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

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

---

[1] Le modèle de *blackboard [Agha 86]* définit une architecture qui organise la résolution de problèmes par coopération de plusieurs modules, appelés sources de connaissances (*Knowledge Source 'KS'*) autour d'une base de données partagée appelée blackboard. le rôle d'une KS est de résoudre un sous-problème particulier en fonction de l'état du blackboard. Ce modèle vise la mise en oeuvre d'une résolution opportuniste, où le choix de la *KS* à activer est déterminé en fonction des critères de contrôle actifs. Ce modèle est repris dans beaucoup d'autres travaux et d'autres domaines.

[2] Le modèle acteur est développé au MIT par l'équipe de C. Hewitt. On lui doit un bon nombre d'aspect repris en *IAD*. Ce sont les travaux sur ce modèle qui ont abouti à une nouvelle conception des systèmes d'*IAD*.

[3] Le Contract Net est un système de résolution de problème distribué, conçu par R. Davis et R. Smith. L'objectif principal de ce système étant la distribution, il procède par allocation des tâches à un ensemble de résolveurs de problèmes et utilise le concept de négociation pour adjuger des contrats.

• ***L'intelligence Artificielle Parallèle (IAP) :*** Elle concerne le développement de langages et d'algorithmes parallèles pour l'*IAP* vise l'amélioration des performances des systèmes d'intelligence artificielle sans tout fois s'intéresser à la nature du raisonnement ou au comportement intelligent d'un groupe d'agent.

2.3. L'apport de l'Intelligence Artificielle Distribuée

L'*IAD* permet de *distribuer l'intelligence* entre plusieurs agents. Comme le souligne Bond et Gasser [Bond et Gasser 88] l'intelligence Artificielle Distribuée présente les avantages suivants :

- L'*IAD* est bien adaptée à la distribution de problèmes spatiaux, logiques…
- L'héritage de modules permet aux différentes parties du système de développer de façon indépendante un système continu et extensible,
- Les processus distribués entre différents ordinateurs augmentent la vitesse de calcul et de raisonnement,
- Le contrôle du processus local peut être isolé ou séparé du système,
- Dans certain cas, les systèmes confèrent aux agents individuels des ressources limitées pour résoudre les problèmes et la coopération et la coordination sont essentielles à la résolution de ces problèmes. Bien que les agents soient relativement frustes, les problèmes auxquels ils sont confrontés peuvent être complexes.

2.4. Problématique de l'*IAD*

Les problèmes que l'*IAD* s'attache à résoudre, sont les problèmes classiques de l'*IA* qui ont pris une nouvelle dimension dans le contexte multi-agents et les nouveaux problèmes proprement liés au thème de l'*IAD*, on peut citer [Agha 86] :

• La modélisation de la connaissance et le problème de sa répartition sur les différents agents regroupés en des sociétés : comment formuler, décomposer, allouer des problèmes et synthétiser les résultats d'un groupe d'agents ?
• Les problèmes de génération de plans d'actions où il faut prendre en considération la présence d'autres agents. Ces problèmes sont liés au comportement d'un agent au sein d'un groupe. On s'intéresse alors à ses capacités sociales : la répartition des tâches, le partage des ressources, le raisonnement sur les autres agents (pouvoir modéliser leurs connaissances et être en mesure de connaître leurs plans d'actions et de raisonner en fonction de ces plans) ;
• La gestion des conflits entre les agents et le maintien de la cohérence des décisions et des plans d'actions ;
• Le problème de la communication : comment permettre la communication et l'interaction entre les agents ? Quel langage et quel protocole faut-il employer ? Une communication dans les univers multi-agents n'est plus une simple tâche d'entrée-sortie, mais doit être modélisée comme un acte pouvant influer sur l'état des autres agents.
• Les problèmes spécifiques au groupe d'agents, qui portent sur l'organisation, l'architecture de l'ensemble des agents et les paradigmes de coopération et d'action ;
• D'autres thèmes de recherche sont présents dans le contexte multi-agents, à savoir, le raisonnement temporel, le raisonnement hypothétique, la représentation de la connaissance imprécise, etc.

3. Agents et Systèmes Multi-agents

Cette section a pour objectif de donner la définition des agents logiciels, des systèmes multi-agents, et de montrer l'importance du sujet. Il va aussi rapidement situer les concepts introduits vis-à-vis d'autres disciplines et présenter les domaines importants de recherche liés aux agents et systèmes multi-agents. Il ne rentrera pas dans le détail des concepts et techniques qui seront introduits dans les différents chapitres.

3.1. Les agents

3.1.1. Définition d'agent

Il n'existe pas, actuellement, une définition de l'agent qui fasse foi dans le monde de l'intelligence artificielle distribuée. Plusieurs définitions ont été attribuées au concept d'agent qui a fait l'objet de plusieurs études, l'une des premières définitions on trouve celle de [Ferber 95] : *Un agent est une entité physique ou virtuelle :*

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

La *figure1.2* donne l'aperçu externe d'un agent, dans lequel l'agent est vu en tant que *boite noire,* dont seules les entrées et sorties sont représentées. L'origine (*pour une entrée*) ou le résultat (*pour une sortie*) peut, indépendamment de l'agent lui-même, être lié à des dispositifs physiques tels que des capteurs et actionneurs.

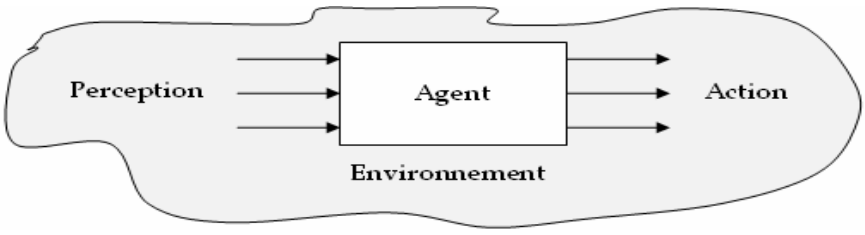


Figure 1.2. Aperçu externe d'un agent.

Selon Y. Demazeau [Demazeau 96], un agent est une entité réelle ou virtuelle dont le comportement est autonome, évoluant dans un environnement qu'il est capable de percevoir et sur lequel il est capable d'agir, et d'interagir avec les autres agents. Cette définition introduit l'interaction qui, comme nous le verrons par la suite, est le moteur des systèmes multi agents. En effet, l'interaction suppose la présence d'agents capables de se rencontrer, de communiquer, de collaborer et d'agir.

Une définition plus récente du concept d'agent proposée par N. Jennings [Jennings 00] consiste en : Un *agent* est un système informatique, *situé* dans un *environnement*, et qui *agit* d'une façon

autonome et *flexible* pour atteindre les *objectifs* pour lesquels il a été conçu. Les notions '*situé*', ' *autonome*' et ' *flexible*' sont définies comme suit :

- *Situé :* l'agent est capable d'agir sur son environnement à partir des entrées sensorielles qu'il reçoit de ce même environnement. Exemple : systèmes de contrôle de processus, systèmes embarqués, etc.
- *Autonome :* l'agent est capable d'agir sans l'intervention d'un tiers (*humain ou agent*) et contrôle ses propres actions ainsi que son état interne.
- *flexible :* l'agent dans ce cas est :
  - *capable de répondre à temps :* l'agent doit être capable de percevoir son environnement et élaborer une réponse dans les temps requis ;
  - *proactif :* l'agent doit exhiber un comportement proactif et opportuniste, tout en étant capable de prendre l'initiative au bon moment ;
  - *social :* l'agent doit être capable d'interagir avec les autres agents (*logiciels et humains*) quand la situation l'exige afin de compléter ses tâches ou aider ces agents à accomplir les leurs.

Cette définition est conforme à la vision avec laquelle on aperçoit l'agent dans le contexte de notre travail expliqué dans ce chapitre et par conséquent on est convenue de l'adapter tout au long de ce mémoire. Malgré qu'il n'existe pas une définition universelle du terme agent, il y a un accord que l'autonomie est le point clé de cette notion. Cette autonomie signifie que l'agent n'est pas dirigé par des commandes venant de l'utilisateur (ou d'un autre agent), mais par un ensemble de tendances qu'il cherche à optimiser. C'est lui qui est actif. A la différence des objets semblables : objets, processus ou modules logiciels, l'agent a la possibilité de répondre par l'affirmative ou le refus à des requêtes provenant des autres agents. En plus de l'autonomie, un agent est caractérisé par un degré d'interactivité et de réactivité lui permettant d'exercer des actions sur son environnement et de s'adapter aux changements. A ces caractéristiques de base peuvent s'ajouter d'autres caractéristiques telle que *la capacité sociale, la coopération, l'apprentissage, la mobilité, …etc.* permettant à l'agent d'atteindre ces objectifs.

3.1.2. Caractéristiques d'un agent

Nous pouvons distinguer les caractéristiques suivantes d'un agent [Labidi 94] :

- **Intentionnalité :** un agent intentionnel est un agent guidé par ses buts. Une intension est la déclaration explicite des buts et des moyens d'y parvenir. Elle exprime la volonté d'un agent d'atteindre un but ou d'effectuer une action ;
- **Rationalité :** un agent rationnel est un agent qui effectue les bonnes actions au bon moment. Les agents rationnels disposent de critères d'évaluation de leurs actions, et sélectionnent selon ces critères les meilleures actions qui leur permettent d'atteindre le but. De tels agents sont capables de justifier leurs décisions ;
- **Adaptabilité :** un agent adaptatif est un agent capable de contrôler ses aptitudes (Communicationnelles, comportementales, etc.) Selon l'agent avec qui il interagit. Un agent adaptatif est un agent d'un haut niveau de flexibilité ;
- **Engagement :** la notion d'engagement est l'une des qualités essentielles des agents coopératifs. Un agent coopératif planifie ses actions par coordination et négociation avec les autres agents. En construisant un plan pour atteindre un but, l'agent se donne les moyens d'y parvenir et donc s'engage à accomplir les actions qui satisfont ce but : l'agent croit qu'il est en mesure d'exécuter tout le plan qu'il a élaboré, ce qui le conduit à agir en conséquence ;
- **Intelligence :** en conclusion, on appelle un agent intelligent un agent cognitif rationnel, intentionnel et adaptatif.

3.1.3. Les différentes structures d'agents

Les agents peuvent être classés en deux catégories principales selon leur comportement et leur granularité. Cette notion de granularité [Labidi 94] est bien sûr très subjective, elle exprime la complexité de *raisonnement* d'un agent afin de séparer les agents dits *intelligents* des agents moins *intelligents*. On parle d'agents cognitifs et d'agents réactifs, mais d'un point de vue conceptuel, on définit plusieurs types d'agents :

3.1.3.1. Agents réactifs

Les agents réactifs sont fondés sur des modèles d'organisations biologiques comme les sociétés de fourmis. Dans telles sociétés, il n'est pas nécessaire que chaque individu soit intelligent pour parvenir à un comportement global intelligent au niveau de la société. Cette approche se base donc sur la coopération d'agents de faible granularité mais beaucoup plus nombreux, ne disposant que d'un protocole et d'un langage de communication réduits et dont les capacités répondent uniquement à la loi de *stimulus/action* [Labidi 94]. Par des mécanismes simples de réaction aux événements, ne prenant en compte ni une explication des buts, ni des mécanismes de planifications, les systèmes d'agents réactifs peuvent ainsi résoudre des problèmes complexes *(cf. figure 1.3).* Les agents réactifs ne font aucune référence au passé au moment de prendre une décision car ils ne disposent d'aucun historique. Leurs comportements sont cycliques sous la forme d'un ensemble *{perception, action}*. Le concepteur d'un agent réactif définit des règles *perception action* dans chaque cas de perception sur l'environnement [Wooldridge 97]. On peut dire aussi que les agents réactifs ne disposent pas une représentation explicite de leurs environnements et ils prennent des décisions en se basant sur des données locales [Chaib-Draa et Jarras 01].

L'approche réactive connaît certaines limites lorsqu'il est question de réaliser des tâches complexes. Le grand nombre d'agents utilisés introduit des problèmes de chutes de performances à partir d'un seuil de surpopulation et d'un coût financier lorsqu'il s'agit d'applications réelles [Drogoul 93].

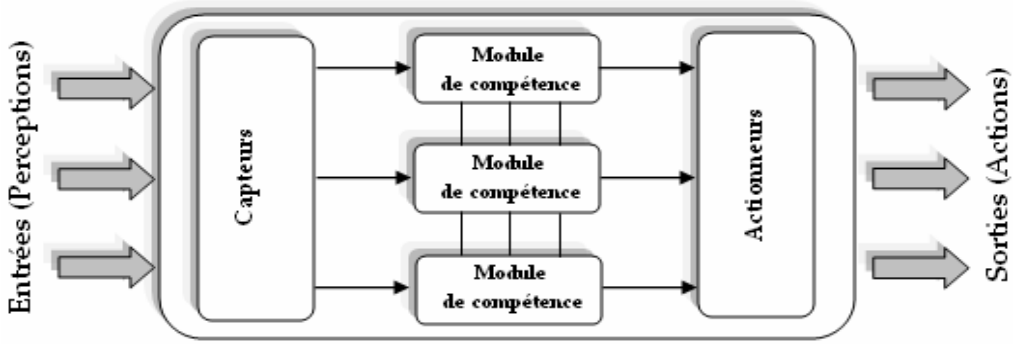


Figure 1.3. Structure d'un agent réactif.

3.1.3.2. Agents cognitifs ou délibératifs

Les agent cognitifs héritent du concept *Belief Desire Intention (cf. figure 1.4)* qui permet à l'agent d'avoir une représentation explicite de son environnement, de prendre ses décisions en se basant sur un historique et de détenir des objectifs bien définis. L'aspect *Belief* correspond à l'état de l'agent mis à jour en fonction de l'évolution de l'environnement. Le *Desire* constitue l'ensemble des options disponibles pour l'agent en fonction de l'état de l'environnement. Alors que l'*Intention* correspond aux objectifs en cours de réalisation [Wooldridge 02].

Les agents cognitifs sont fondés sur des modèles d'organisations sociales humaines telles que les groupes et les marchés. Ainsi, l'approche cognitive repose sur la coopération d'un petit nombre d'agents capables à eux seuls d'effectuer des opérations complexes. Les agents sont donc dotés d'une grande capacité de *perception*, de *raisonnement* et *d'action*. Ils sont assez proches des systèmes experts, dans leur aptitude à manipuler la connaissance *(cf. figure 1.4)*. Ainsi, chaque agent dispose d'une base de connaissance comprend l'ensemble des informations nécessaires à la réalisation de sa tâche et à la gestion des interactions avec les autres agents et avec l'environnement. Les agents cognitifs sont aussi dits intentionnels, c'est-à-dire possèdent des buts et des plans explicites leur permettant d'accomplir leurs tâches [Ferber 95].

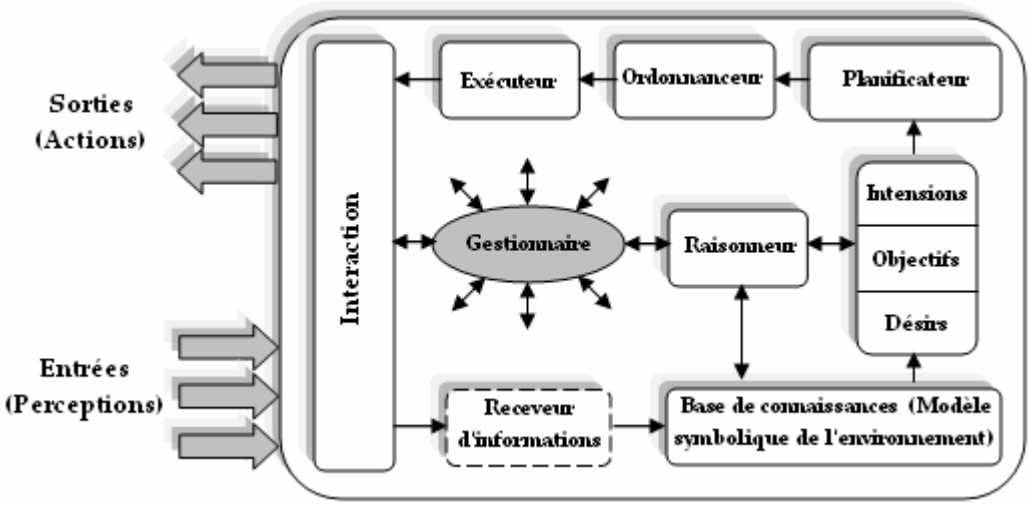


Figure 1.4. Structure d'un agent cognitif.

L'approche cognitive pose deux problèmes importants [Guessoum 96], [Guessoum 03] :

- Le problème de la traduction de *l'univers de l'agent* en *une description symbolique*, en tenant compte du temps ;
- Le problème de *représentation/raisonnement* qui consiste à trouver une représentation symbolique des informations de l'univers complexe des entités et des processus, et à trouver comment raisonner sur ces informations.

Plusieurs types d'agents cognitifs peuvent être distingués, on trouve en particulier :

• ***Les agents intelligents****:* combinent les trois caractéristiques *(autonomie, coopération, adaptation)* à leur plus haut niveau. Ils sont donc en principe capables de planifier leurs actions, de négocier avec d'autres agents et d'acquérir ou de modifier leurs connaissances. Ils sont en général dotés de *la capacité d'apprentissage* [Kuflik et Shoval 00].

• ***Les agents collaborants****:* ce sont des agents cognitifs non apprenants ; on considère qu'ils sont à la fois fortement autonomes et coopérants, mais peu adaptatifs. Ils sont surtout utilisés dans les domaines qui nécessitent une décentralisation comme par exemple la maintenance de réseau, ou encore pour simuler le comportement d'organisations humaines ou animales [Lant 94].

• ***Les agents d'interface***: [Lashkari et Metral 94] (*appelés aussi agents assistants*) ils ont pour mission de capturer les actions de l'utilisateur *(clavier, souris, à terme : voix, gestes, expression du visage).* Dans l'état actuel des recherches, ces agents possèdent en général une *capacité de coopération limitée*, ils sont principalement utilisés pour l'assistance à l'utilisateur dans le cas d'interfaces aux fonctionnalités nombreuses et complexes *(par exemple, dans certaines suites bureautiques),* mais également dans le domaine des systèmes tuteurs intelligents afin de faciliter *l'apprentissage humain.*

• ***Les agents d'information****:* ces agents sont dédiés à la recherche d'information, principalement sur l'Internet. ils sont dédiés à *la recherche d'information*, principalement sur

internet. Ces agents possèdent une grande autonomie ; ils agissent seuls, soit en fonction d'un calendrier, soit en fonction d'un manque d'information, soit en fonction d'une nouvelle disponibilité d'information. Ils sont capables d'adapter leurs fonctionnements en fonction du besoin de l'utilisateur ou de la quantité ou pertinence de l'information *(par exemple, si un nouveau site propose des informations plus pertinentes, ce site sera alors privilégié pour les recherches futures)*. Toutefois, ces agents agissent le plus souvent de manière isolée, ce qui peut entraîner un problème de redondance d'informations[Rus et Gray 97].

Notons que l'approche réactive et cognitive se complètent, chacune d'elles répond à des besoins précis, mais manque de ce que l'autre peut apporter. La solution de ce problème vient souvent de la conception de systèmes à agents hybrides capables de coopérer en introduisant une couche délibérative au-dessus d'une couche réactive. Toutefois, les problèmes de tels systèmes restent les mêmes que ceux des agents cognitifs.

3.1.3.3. Agents hybrides

Un agent hybride est une *architecture d'agent* qui combine *un système réactif* à *un système cognitif.* Le système réactif assure la réalisation des activités de type réflexes en réponse aux stimuli de l'environnement. Ces activités ne nécessitent pas la mise en oeuvre de raisonnements complexes. Le système cognitif assure la réalisation d'activités basées sur la planification et la délibération, nécessitant la mise en oeuvre de raisonnements complexes. Donc les agents hybrides possèdent des capacités de raisonnement individuel sur le problème global et sur autres. Cependant, ces agents cognitifs doivent également intégrer des capacités réactives. Le but est d'une part d'essayer de construire des agents cognitifs à partir d'organisations réactives et d'autre part de réaliser des agents qui disposent à la fois des capacités cognitives et réavties. Il s'agit d'agents hybrides qu'ils conjuguent en effet la rapidité de réponse des agents réactifs ainsi que les capacités de raisonnement des agents cognitifs. Cette famille regroupe donc des agents dont le modèle est un compromis *autonomie/coopération* et *efficacité/complexité* (*Les architectures hybrides s'avèrent plus efficaces pour appréhender les capacités de décisions et d'actions des agents dans le temps lors de la phase de conception du SMA*).

3.1.3.4. Agents a états

Ces agents maintiennent un ensemble d'états internes qui mémorisent leurs perceptions sur l'environnement [Wooldridge 97]. Soient :

- ✓ $S = \{s0,s1,s2....sn\}$ l'ensemble des états internes de l'agent. Le choix d'une action à entreprendre par l'agent est basé, en partie, sur les informations en provenance de cet ensemble.
- ✓ $See : E \rightarrow Per$ la fonction qui retourne les perceptions *Per* de l'environnement $E = \{e_0,e_1,e_2....\}$
- ✓ $Ac = \{ac1,ac2,.....,acm\}$ l'ensemble des actions que l'agent peut entreprendre.

La fonction $action : S \rightarrow Ac$ est définie pour retourner les actions à entreprendre pour chaque état. La fonction $next : S \times Per \rightarrow S$ est définie pour retourner l'état futur de l'agent en fonction de la perception et de l'état interne courant. Le comportement d'un agent à états commence par un état initial *s0*. Ensuite, l'agent observe l'état de l'environnement *e0* et génère la perception *see(e0).* L'état interne de l'agent est alors mis à jour via la fonction *next(s0,see(e0)).* L'action sera sélectionnée avec la fonction *action (next(sQ,see(e0))).* Cette action est exécutée et l'agent reprend de nouveau son cycle.

3.1.3.5. Agents avec buts

L'agent est parfois incapable de sélectionner la meilleure action à entreprendre en se basant uniquement sur l'état de l'environnement et la séquence des perceptions mémorisés [Wooldridge

02]. L'agent doit essayer d'obtenir le maximum de performance et il doit alors sélectionner les actions appropriées. Il existe plusieurs façons de définir la mesure de performance d'une action. Une première façon est d'indiquer à l'agent ce qu'il doit faire en toute circonstance pour atteindre ses objectifs. C'est le cas des agents réactifs. Toutefois, puisque les agents intelligents sont autonomes et proactifs, il serait intéressant de dire à l'agent les actions à entreprendre, sans avoir à lui dire exactement comment le faire dans chacune des situations. Dans ce cas, il est possible de définir la mesure de performance de l'agent en fixant des « états buts » ou états désirables : but : E→{0,1} est une fonction qui a la valeur 1 pour les états buts atteints et 0 pour tout état non but atteint. Une deuxième façon consiste à modéliser les performances à atteindre par l'agent en accordant à chaque état E une valeur réelle, appelée *utilité*[1].

3.1.4. Architectures d'agents

Différents types d'architecture d'agents ont été définis. Les principales architectures développées sont basées sur *les comportements réactifs*, sur *la logique* ou sur *les croyances et intentions (BDI)* des agents.

3.1.4.1. Architectures réactives

L'idée principale des architectures basées sur des comportements réactifs est que le comportement intelligent de l'agent émerge des interactions entre ses comportements plus simples. Ce type d'architecture est fondé sur le concept de comportement. Chaque comportement est considéré comme la réalisation d'une action basique. Un comportement est régi par un cycle de Perception/Action dans lequel l'agent effectue directement une action en fonction de sa perception, sans réaliser de réflexion. Comme plusieurs comportements peuvent être exécutables en même temps, l'agent doit disposer d'un moyen de choisir parmi les comportements entrant en conflit. Pour cela, l'architecture à subsomption, définie par Brooks propose d'organiser les comportements de manière hiérarchique dans laquelle les comportements de niveaux inférieurs peuvent inhiber les comportements de niveaux supérieurs. Ce type d'architecture purement réactive a l'avantage d'être simple et tolérant aux pannes. Cependant, il présente certains difficultés: la prise en compte d'informations non locales à l'agent, la mise en place de mécanismes d'apprentissage, et surtout la conception de l'émergence du comportement global en fonction des comportements locaux.

3.1.4.2. Les architectures logiques

Dans les architectures fondées sur la logique, l'agent doit disposer de représentations symboliques de l'environnement. L'agent dispose donc un ensemble d'informations qui sont exprimées sous forme de prédicats de la logique du premier ordre (base de faits). Cette dernière est vue comme la base des croyances chez l'humain. L'agent est également doté de règles d'inférence et la réalisation d'une action revient à appliquer une règle. La réalisation d'une action modifie l'environnement et donc la base de faits de l'agent. Un problème survient quand plusieurs règles (ou aucune) peuvent être tirées en même temps, l'agent doit-il faire un choix. Les exécuter toutes en parallèle ou les fusionner ? La manière dont l'agent résout ce problème peut être considérée comme son comportement.

[1] L'utilité est une représentation numérique de l'état désiré par l'agent. Plus l'utilité est élevée, plus elle est meilleure (Wooldridge99). L'agent cherche alors, à choisir les états à utilité maximale. L'utilité est souvent définie par la fonction suivante : *Utilité* : *E*→[0,1]. La fonction d'utilité est plus adaptée que la fonction *but* dans certains cas.

3.1.4.3. Les architectures de type BDI

Les architectures de type BDI (*Believe, Desire, intention*) sont basées sur un PRS (*Procedural Reasoning System*) où l'idée principale est de construire un *plan d'actions* pour atteindre le but fixé. Un tel système repose sur quatre modules :

- ***Les croyances de l'agent (Believe)*** ce sont les connaissances de l'agent sur l'état du monde (obtenues via ses capteurs et les communications),
- ***Les désirs (ou buts) de l'agent (Desire)*** c'est l'état du monde que l'agent souhaite atteindre,
- ***L'ensemble des plans*** c'est la connaissance procédurale de l'agent,
- ***Les intentions de l'agent (Intention)*** c'est la liste des plans que l'agent a sélectionnée en fonction de ses croyances et de ses désirs.

3.1.4.4. Les architectures multi-niveaux (les architectures en couches)

Partant du principe qu'un agent est capable de comportements réactifs et de comportements pro-actifs, les architectures multi-niveaux décomposent le comportement de l'agent en couches de complexité croissante. Un tel système est composé d'au moins deux couches : une couche réactive et une couche pro-active. Parmi ces architectures, on peut distinguer trois types de système :

- ***Système en couche horizontale :*** chaque couche perçoit l'environnement et propose l'exécution d'une action. Il faut alors définir la manière dont l'agent choisit la couche prioritaire dans le cas de conflits.

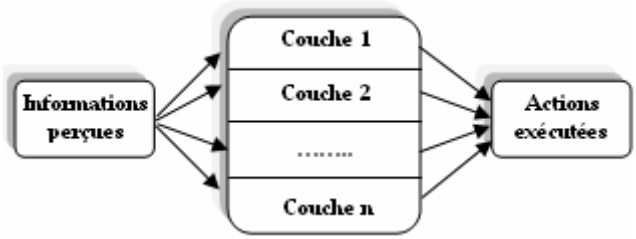


Figure 1.5. Architecture en couches horizontales.

- ***Système en couche verticale à une passe*** : les informations perçues passent de couche en couche en commençant par celle de niveau inférieur. Chaque couche propose l'exécution d'une action à la couche de niveau supérieur et c'est la dernière couche qui désigne l'action à exécuter.

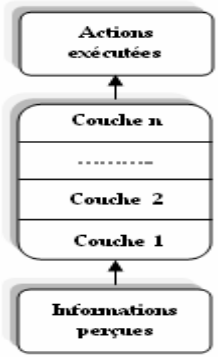
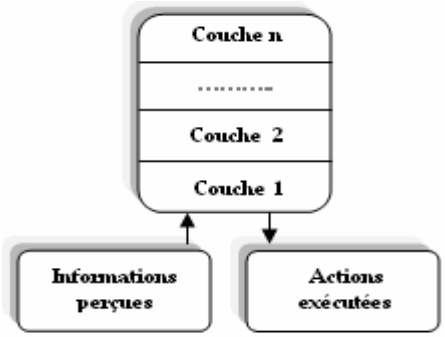


Figure 1.6. Architecture en couches verticales (contrôle en une passe).



- ***Système en couche verticale à deux passes*** : dans cette architecture, la première passe est d'abord exécutée, le choix de l'action redescend de couche en couche et c'est la couche inférieure qui effectue l'action.

Figure 1.7. Architecture en couches verticales (contrôle en deux passes).

L'avantage principal de l'architecture en couches horizontales est qu'il suffit d'ajouter des couches à l'agent pour qu'il exhibe un nouveau comportement. Comme toutes les couches proposent des actions, il est indispensable d'introduire une fonction de médiateur qui définira à chaque moment quelle est la couche prioritaire. Ce besoin de contrôle centralisé nécessite que le concepteur appréhende l'ensemble des interactions entre les couches. Ce problème est partiellement résolu par l'utilisation d'une architecture en couches verticales. En effet, l'information traverse les couches séquentiellement, réduisant ainsi considérablement les

interactions entre les différentes couches *(cf. figure 1.7).* Cependant, la limite majeure des ces architectures reste la non tolérance aux fautes : l'échec dans n'importe quelle couche aurait des conséquences sérieuses sur les performances de l'agent.

3.2. Les systèmes multi-agents

3.2.1. L'approche multi-agents

L'approche multi-agents considère que le concept d'agent ne prend tout son intérêt qu'au sein d'un univers d'agents dans lequel les agents se communiquent pour résoudre un problème complexe difficile à résoudre par un seul agent. On parle donc *des systèmes multi-agents (SMA).* Dans un tel système, l'agent a un comportement à la fois individuel et collectif. Le caractère individuel d'un agent se manifeste à travers des objectifs qui lui sont propres et qu'il essaye d'atteindre avec les ressources et les compétences qu'il dispose. L'agent est donc une entité intentionnelle [Ferber 95]. Toutefois, un *SMA* ne se résume pas à un ensemble d'agents agissants de façon individuelle. L'intérêt des *SMA* provient des interactions entre ses agents. Le caractère collectif des agents se traduit par leurs facultés à gérer leurs interactions : *coopération, négociation et coordination.*

3.2.2. Définitions

Selon Chaib-draa, « *un SMA est un système distribué composé d'un ensemble d'agents. Contrairement aux systèmes d'IA, qui simulent dans une certaine mesure les capacités du raisonnement humain, les SMA sont conçus et implantés idéalement comme un ensemble d'agents interagissants, le plus souvent, selon des modes de coopération, de concurrence ou de coexistence* » [Chaib-Draa et Jarras 02].

Plus précisément, « *un SMA est un système composé des éléments suivants* [Ferber 95]*:*

- un environnement E i.e., un espace disposant généralement d'une métrique;
- un ensemble d'objets O. Ces objets sont situés i.e., pour tout objet il est possible, à un moment donné, d'associer une position dans E. Ces objets sont passifs i.e., qu'ils peuvent être perçus, créés, détruits et modifiés par les agents;
- un ensemble A d'agents, qui sont des objets particuliers ($A \subseteq O$), lesquels représentent les entités actives du système;
- un ensemble de relations R qui unissent des objets et des agents entre eux;
- un ensemble d'opérations Op permettant aux agents de A de percevoir, produire, consommer, transformer et manipuler des objets de O;
- des opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette technique de modification, que l'on appellera les lois de l'univers ».

Notons, selon cette définition, que tous les systèmes n'auront pas nécessairement besoin d'être décrits avec tous ces éléments *(cf. figure 1.8).* D'ailleurs, l'auteur définit les agents purement communicants et les agents purement situés. Dans le premier cas, *E* est l'ensemble vide, *A* correspond à *O*, et *R* représente un réseau. Dans le deuxième cas, les agents ne possèdent aucune capacité de représentation et qui communiquent indirectement à travers l'environnement *(agents réactifs).* Dans un *système de gestion de trafic aérien* par exemple, les agents seraient *les avions*, l'environnement étant *l'espace aérien* dans lequel la position de chaque avion est définie par ses coordonnées. Ces agents suivent des trajectoires et doivent mettre en oeuvre *des stratégies* pour éviter les collisions. Ils peuvent ainsi *communiquer* directement entre eux ou à travers *une tour de contrôle.*

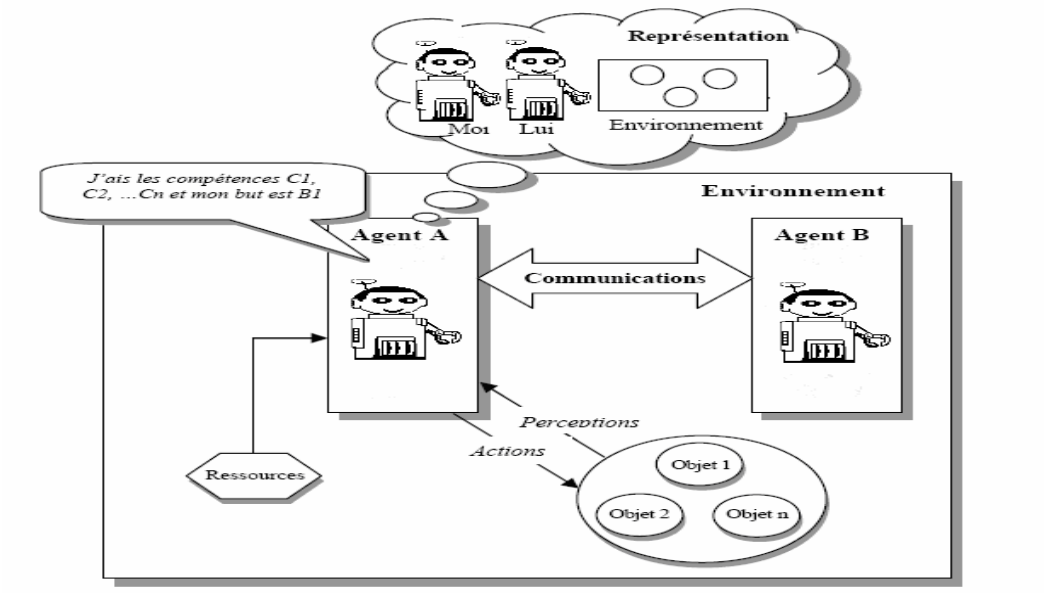


Figure 1.8. Les éléments d'un système multi-agents**.**

3.2.3. Caractéristiques des SMA

Par leur nature répartie, les SMA offrent les propriétés intéressantes suivantes :

• ***l'efficacité des traitements :*** les agents travaillent en parallèle et communiquent de façon asynchrone ;
• ***la robustesse et la sûreté de fonctionnement :*** la mise hors fonctionnement de quelques agents ne modifie pas sensiblement le comportement global du système;
• ***la flexibilité et le traitement des systèmes à grande échelle :*** on peut toujours augmenter le nombre d'agents pour traiter des systèmes de plus en plus gros, sans pour autant perturber le travail des agents existants;
• ***un coût de fonctionnement faible :*** la répartition des traitements, en utilisant des agents réalisant des tâches simples, conduit à des coûts faibles;
• ***un coût de développement et de réutilisation intéressant :*** il est plus simple de développer par des spécialistes des agents indépendamment les uns des autres, pour les réutiliser dans diverses applications.

3.2.4. L'environnement

Dans un système multi agents, on appelle *environnement l'espace commun aux agents du système.* L'environnement peut être considéré comme la représentation du monde dans lequel les agents se situent. Il est modifiable par les agents, soit de façon globale, soit en faisant la distinction entre objets passifs *(soumis aux actions des agents)* et entités actives *(les agents)* [Ferber 95].

3.2.5. Organisation d'un SMA

En général, un *SMA* est constitué d'un grand nombre d'agents. Ces agents forment *une organisation* ou *une société.* L'organisation sociale est un aspect très important dans la conception d'un *SMA* capable de réaliser une tâche collective. D'une manière générale, l'organisation est un modèle permettant aux agents de coordonner leurs actions au cours de la réalisation d'une ou plusieurs tâches. Elle décrit d'une part une structure comprenant un ensemble de rôles qui doivent

être attribués aux agents et un ensemble de chemins de communication entre ces rôles. D'autre part, elle décrit un régime de contrôle qui dicte le comportement social des agents. Enfin, elle définit des processus de coordination qui déterminent la décomposition des tâches en sous-tâches, l'allocation des tâches aux agents et la réalisation des tâches dépendantes de façon cohérente [Chadès 03].

3.2.5.1. Types d'organisation

Il existe deux façons d'organiser une société d'agents :

- ***Organisation statique :*** chaque agent peut prendre la responsabilité d'effectuer une tâche s'il en a la capacité. Il prend alors un rôle dans l'organisation;
- ***Organisation dynamique :*** l'organisation s'adapte à chaque fois en fonction de la tâche à accomplir. On parle donc d'auto-organisation qui permet à la société de changer sa structure afin de s'adapter de manière autonome à la dynamique des situations imprévues.

3.2.5.2. Niveaux d'organisation

Il existe plusieurs niveaux d'organisation :

- ***micro-social :*** on s'intéresse aux différentes formes de liaisons et d'interactions entre un petit nombre d'agents (exemple : des robots sur la table ou des robots chercheurs de minerai),
- ***groupe :*** l'organisation est plus complète. On étudie les différenciation des rôles et des activités des agents, l'émergence de structures organisatrices entre agents...(ex de l'écosystème et des robots chercheurs de minerai),
- ***sociétés globales (population) :*** on étudie la dynamique d'un grand nombre d'agents, la structure générale du système et son évolution (exemple des fourmis).

3.2.6. Interaction dans un SMA

3.2.6.1. Définition

J. Ferber donne la définition suivante de l'interaction : " *Une interaction est la mise en relation dynamique de deux ou plusieurs agents par le biais d'un ensemble d'actions réciproques… Les interactions sont non seulement la conséquence d'actions effectuées par plusieurs agents en même temps, mais aussi l'élément nécessaire à la constitution d'organisations sociales*."[Ferber 95]. L'interaction entre agents s'effectue par la communication, les actes de langages et les protocoles d'interaction. Les agents interagissent entre eux. Pour atteindre son objectif ou pour améliorer la coordination des actions, un agent peut demander des services à un autre agent.

3.2.6.2. Les modes d'interaction

Les interactions sont au centre de la problématique des SMA, car elles permettent aux agents de produire ensemble le comportement attendu du système. Une interaction entre agents est une mise en relation dynamique de deux ou plusieurs agents. Dans [Ferber 95], Ferber classe les types d'interactions entre agents dans un SMA selon la compatibilité de leurs buts, la disponibilité des ressources et leurs aptitudes propres. On aura donc les types d'interaction suivants:

- ***Indifférence des agents :*** le cas d'indépendance entre agents ;

• ***Coopération des agents :*** situation de collaboration (simple ou coordonnée) ou d'encombrement;
• ***Antagonisme entre les agents :*** situation de compétition (individuelle ou collective) ou de conflits (individuels ou collectifs) pour des ressources.

La figure 1.9 représente une classification des différentes méthodes d'interactions selon le type et la situation [Chadès 03]. Cette classification n'est pas la seule possible, elle peut varier selon que l'on se place au niveau des méthodes ou au niveau de situations.

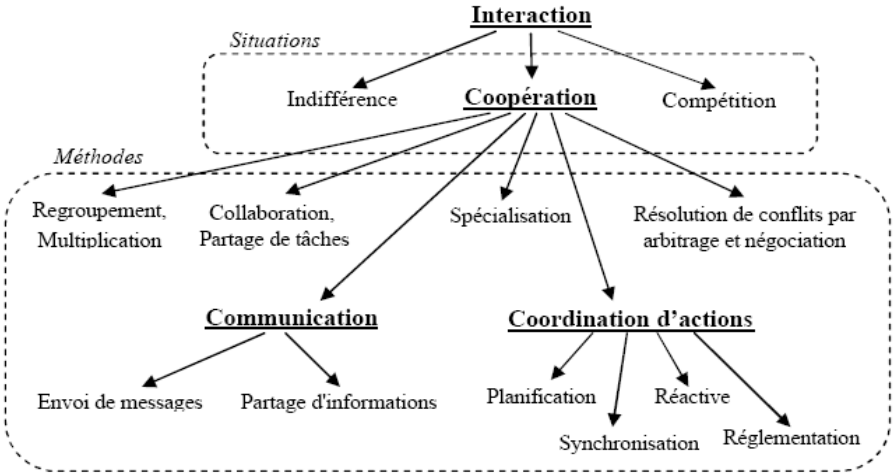


Figure 1.9. Situations et méthodes d'interactions dans un SMA.

3.2.6.3. Les protocoles d'interaction

Les protocoles d'interaction permettent de décrire explicitement les enchaînements conversationnels lors des communications entre les agents. Ils représentent un schéma commun de conversation utilisé pour exécuter une tâche, une stratégie de haut niveau. Un protocole précise qui peut dire quoi à qui et les réactions possibles à ce qui est dit. Les protocoles d'interaction permettent de définir une séquence causale des messages communiqués entre les agents et décrivent comment les agents doivent réagir aux messages reçus durant les interactions. Les aspects de ces protocoles d'interaction diffèrent selon le type des agents (Agents concurrents ou agents à des buts communs). Il existe différents types de protocoles d'interaction, on cite les protocoles : de coordination, de coopération, de négociation, ainsi que les mécanismes du commerce électronique [Mazouzi 01]. L'organisation (FIPA)[1] a été créée en 1996. Parmi ses préoccupations, une place importante concerne l'élaboration des spécifications standards des protocoles d'interactions. On présente ci-dessous un certain nombre de protocoles de FIPA, ces derniers peuvent être appliqués à n'importe quel domaine d'application. Chaque protocole est désigné par un nom prédéfini.

3.2.6.3.1. Le protocole FIPA-request[2]

Il permet à un agent d'inviter d'autres agents à exécuter une certaine action. L'agent récepteur peut refuser ou accepter la tâche. S'il l'accepte, il doit exécuter l'action et informer l'agent qui l'a demandée.

[1]Foundation for Intelligent Physical Agent. FIPA Interaction Protocol Library Specification. http://www.fipa.org/specs/fipa00025/.

[2] FIPA Request Interaction Protocol Library Specification. http://www.fipa.org/specs/fipa00026/.

3.2.6.3.2. Le protocole FIPA-request-when[1]

Ce protocole est utilisé par les agents voulant demander à d'autres agents d'exécuter une certaine action à l'avenir une fois la condition préalable donnée devient vraie. Si l'agent destinataire accepte la demande, il attend l'état vrai de la condition préalable pour exécuter l'action puis il informera l'agent expéditeur de demande que l'action a été exécuté. L'agent destinataire peut refuser l'exécution de la demande dans le cas où il serait occupé par des tâches plus prioritaires une fois que la condition préalable soit vraie.

3.2.6.3.3. Le protocole FIPA-contract-net[2]

Ce paragraphe présente une version du protocole Contract-Net, initialement développé par Smith en 1980 [Smith 80]. *FIPA-Contract-Net* est une modification mineure du protocole initial car il ajoute des actes communicatifs de rejet et de confirmation. Dans ce protocole, un agent prend le rôle du manager. Le manager souhaite faire accomplir une certaine tâche par un ou plusieurs autres agents, et souhaites plus loin optimaliser une fonction qui caractérise la tâche. Cette caractéristique est généralement exprimée comme coût, spécifique au domaine, et pourrait être par exemple le temps nécessaire à l'accomplissement d'une tâche (on va voir le détail de ce protocole dans le quatrième chapitre).

3.2.6.3.4. Le protocole FIPA-Iterated-contract-net[3]

C'est une extension du protocole précédent. Il diffère de la version de base en permettant plusieurs itérations. Une fois que l'agent *manager* reçoit les propositions des *contractants*, il peut accepter une ou plusieurs offres, rejeter d'autres ou peut réitérer le processus en émettant une nouvelle offre *révisée*. L'objectif est de permettre au manager d'obtenir de meilleures offres des contractants en modifiant l'appel et en demandant de nouvelles offres (d'une manière équivalente mais révisée). Le processus termine quand le manager refuse toutes les propositions, n'émet pas de nouveaux appels en acceptant la meilleure offre de l'itération antérieure ou tout simplement lorsque les contractants refusent de faire des offres.

3.2.6.4. La coopération

La coopération est une des caractéristiques les plus intéressantes des *SMA* qui permet d'améliorer les performances individuelles ou globales. Pour mettre en oeuvre la coopération, Ferber [Ferber 95] distingue en plus de la négociation, la coordination d'actions et la communication, qui seront détaillées par la suite, les méthodes suivantes :

- ***Regroupement et multiplication :*** la simple agrégation des agents est une forme de coopération. Le regroupement simplifie la navigation de nombreux agents. La multiplication des agents assure une grande fiabilité (robustesse) au groupe ;
- ***Spécialisation :*** c'est un processus qui conduit un agent à progressivement se spécialiser dans certaines de ses tâches ;

[1] FIPA Request When Interaction Protocol Library Specification. http://www.fipa.org/specs/fipa00028/.

[2] FIPA Contract Net Interaction Protocol Library Specification. http://www.fipa.org/specs/fipa00029/.

[3] FIPA Iterated Contract Net Interaction Protocol Library Specification. http://www.fipa.org/specs/fipa00030/.

• ***Répartition des tâches, des connaissances et des ressources :*** il s'agit d'un processus collaboratif permettant aux agents de se répartir les tâches, les connaissances et les ressources pour réaliser un objectif commun. Cette répartition peut ce faire au sein d'un système à agents cognitifs par des mécanismes d'offre et de demande. Dans un système à agents réactifs, cette répartition se fait par le bais de l'environnement, et conduit à la spécialisation des agents et à leur répartition géographique.

3.2.6.5. La négociation

Les activités des agents dans un système distribué sont souvent interdépendantes et entraînent des conflits. La négociation est un moyen pour résoudre ces conflits, elle permet de résoudre des conflits lorsque les agents interagissent pour prendre des décisions communes, alors qu'ils poursuivent des buts différents. Le processus de négociation ne consiste pas forcément à trouver un compromis mais peut s'étendre à la modification des croyances des autres agents pour faire prévaloir un point de vue. Pour cela, un système de communication de haut niveau est nécessaire afin de parvenir à une solution ou à un compromis. De nombreuses techniques de négociations pour la coordination dans les systèmes multi-agents sont proposées dans la littérature, [Jennings et al 01]. Ces techniques de résolution de conflits sont issues de différents domaines de recherche (théorie de la décision, économie, etc.). La vente aux enchères offre des protocoles de négociation dans lesquels un vendeur souhaite maximiser son profit et les acheteurs souhaitent minimiser leurs coûts. Plusieurs processus d'enchères sont proposés dans la littérature (Anglaise, Hollandaise, etc.). De nombreuses autres approches s'appuient sur la négociation pour coordonner les agents (le vote, la formation de coalition, etc.), [Sandholm 99].

3.2.6.6. La coordination d'actions

La coordination est une caractéristique essentielle dans les *SMA*. Elle permet par exemple aux agents d'éviter le conflit d'accès aux ressources, les attentes indéfinies et l'inter-blocage. La coopération est la forme de coordination entre agents non antagonistes ; à l'inverse, la négociation correspond à la coordination en univers compétitif ou entre agents égoïstes ou concurrentiels [Mazouzi 01]. Dans un environnement dynamique, les situations où les agents doivent résoudre les conflits sont coûteuses et impliquent un protocole de négociation très complexe. Le but de la coordination est de permettre aux agents de s'adapter à la dynamicité de l'environnement tout en évitant cette phase de conflits.

Selon Jennings [Jennings 96], la coordination d'actions est le processus par lequel un agent raisonne sur ses actions locales et les actions (anticipées) des autres agents pour assurer que la communauté agit de manière cohérente. La coordination, qui peut se dérouler avec ou sans communication, a quatres formes :

• ***Coordination par synchronisation :*** il s'agit de décrire la façon dont s'enchaînent les actions. Elle est très utilisée dans les systèmes automatiques industriels ;
• ***Coordination par planification :*** elle peut être centralisée ou distribuée. Généralement, elle est associée aux techniques traditionnelles de planification mono-agent;
• ***Coordination réactive :*** il n'est question d'aucune planification, les agents réactifs s'auto-organisent au travers de leurs interactions avec l'environnement. Ce type de coordination est celui qu'utilisent les oiseaux migrateurs ;
• ***Coordination par réglementation :*** cela consiste à suivre des règles de comportement pour éviter les conflits. Le meilleur exemple est le code de la route.

3.2.6.7. La communication entre agents

La communication entre agents est à la base de la résolution coopérative de problèmes. Elle constitue l'ossature sur laquelle des modes d'interaction plus évolués s'appuient. Les agents communiquent entre eux en émettant et en recevant des messages selon deux modes de communication:

3.2.6.7.1. Communication par partage d'informations

Les agents ne sont pas en liaison directe mais communiquent via une structure de donnés partagée. Cette structure contient les connaissances relatives à la résolution (état courant du problème) qui évoluent durant le processus d'exécution. L'architecture de *Blackboard* (tableau noir) constitue un exemple parfait de l'utilisation de ce mode de communication. Dans un système à *Blackbord*, on trouve les trois éléments suivants:

- Un ensemble d'agents indépendants (sources de connaissances) ;
- Une structure de données partagées *(Blackbord)*, divisée souvent en plusieurs niveaux d'abstractions ;
- Un mécanisme de contrôle assurant l'intervention des différentes sources de connaissances pour la résolution du problème.

3.2.6.7.2. Communication par envoie de messages

Les échanges entre agents se font par envoi de messages selon un protocole bien défini. Deux types de transmissions sont possibles : la transmission directe (point à point) et la transmission par diffusion *(broadcasting)*. Dans le premier, l'émetteur du message connaît et précise l'adresse de (ou des) destinataires. Dans le second, le message est envoyé à tous les agents du système. Les deux modes sont souvent combinés *(multicasting)* : le premier permet d'établir une liste de correspondants privilégiés, utilisés ensuite par le second.

3.2.6.8. Les langages de communication

Plusieurs langages ont été proposés pour facilité les communications entre les agents. Les plus connus sont *KQML (Knowledge Query and Manipulation Language)* et *ACL (Agent Communication Language)* qui ont en commun de considérer les énoncés comme des actes, qui visent à accomplir ou à faire accomplir quelque chose. Ils décrivent les états des agents en termes d'attitudes mentales: *savoir, croire, désirer, intention.*

3.2.6.8.1. Le langage KQML

Pour définir des protocoles de communication entre agents, on peut s'appuyer sur la théorie des actes de langage. C'est l'objet de *KQML* défini aux *USA* par la *DARPA-KSE3*[1] qui propose de définir ainsi un standard de communication pour les systèmes multi-agents. Un message asynchrone peut être décrit sous la forme suivante [Ferber 95]: ***(<id>) <émetteur> : <destinataire> << <énoncé>***

id est un identificateur de message (codage direct ou indirect du temps), ***émetteur*** est un agent qui prend la parole, ***destinataire*** est un agent, une liste d'agents, une description des destinataires ou

[1] Knowledge Sharing Effort : http://www.cs.umbc.edu/kse/

tous pour les messages publics, ***énoncé*** est le corps du message, il doit comporter un contenu propositionnel et une force illocutoire.

Pour communiquer par messages, il faut définir des conventions et implanter les supports nécessaires à la transmission des messages. On définit trois couches :

- La couche communication qui met en place l'infrastructure physique et sociale pour que les messages soient transmis : support physique, système d'adressage, gestion du temps.
- La couche contenant qui définit le langage du message, l'ontologie de référence (qui correspond aux connaissances supposées partagées entre les agents), le type d'acte de langage utilisé, un identificateur de message.
- Le contenu du message qui contient une expression dans le langage requis.

Ces trois couches se retrouvent dans les messages *KQML* mais pas dans cet ordre puisque le performatif introduit le message. *Exemple:*

(ask-if
:sender A
:receiver B // :to, :from en cas de renvoi
:langage Prolog
:ontology ÊêtresHumains
:reply-with id1 // :in-reply-to
:content ami(Ned, Marco))

Les types de messages de *KQML* sont des instructions de routage *forward* et *broadcast*, des instructions liées à l'implantation de la recherche de partenaires : *advertise*, *recrut*, *broker*… donc des instructions hétéroclites. Les actes performatifs de *KQML* relèvent de 3 catégories : 18 performatifs de discours, assez proches de la théorie des actes de langage, comme des requêtes : *ask-if*, *ask-one*, *stream-all* et des assertions *tell*, *deny*. 11 performatifs d'interconnexion: *register*, *unregister*, *broadcast*, *broker*... 7 performatifs d'exception : *Error* : le message est mal formaté, *Sorry* : on ne peut pas donner d'autre réponse, *Standby* : indiquer quand on est prêt à répondre, *Ready*, *Next* : envoyer la réponse suivante, *Discard* : plus besoin des réponses au message indiqué.

3.2.6.8.2. Le langage FIPA-ACL

Il est plus récent et il est aussi fondé sur la théorie des actes de langage. La syntaxe est assez proche de celle de *KQML*, par *exemple* :

(request
:sender (agent-identifier :name i)
:receiver (set (agent-identifier :name j)
:content (…)
:protocol fipa-request
:langage fipa-SL
:reply-with order567
)

Il définit précisément la sémantique du contenu des messages *(fipa-SL)* et un protocole de communication. *ACL* définit 4 actes primitifs : *inform, request, confirm et disconfirm*. 18 actes

composés, qui prennent des actes comme arguments : *request-when, refuse, propose, reject-proposal….* A chaque acte est associée une finalité : *transmettre une information, demander une information, négocier, accomplir une action, gérer un problème*. La sémantique formelle du langage associe à chaque acte des *pré-conditions*, des *post-conditions* et *des conditions d'arrêt.*

3.2.7. Les plateformes de développement d'un SMA

Il existe actuellement de nombreuses plates-formes multi-agents qu'on peut classer en plusieurs catégories.

- Les plates-formes pour agents mobiles *(Voyager, Odissey, Aglet, …etc.)* qui fournissent la mobilité à des agents,
- Les plates-formes pour agents cognitifs *(AgentBuilder, etc.)* dans lesquels on trouve les plates-formes *FIPA (Jade, FIPA-OS, etc.)* ou *KQML (JAT, JAT-Lite, …etc.),*
- Les plates-formes pour agents collaboratifs *(Zeus, JAFMAS, KAoS, JAFIMA, …etc.),*
- Les plates-formes pour la simulation multi-agents *(Cormas, Swarm, …etc.)*.

Toutes ces plates-formes sont dédiées soit à un certain type d'agents (agents à base de règles pour *AgentBuilder* ou agents collaboratifs pour *Zeus* par exemple) soit à un certain type d'applications (*simulation, mobilité*). Il n'existe pas à notre connaissance de plate-forme générique qui permette de réaliser des applications dans tous ces domaines. Dans le chapitre2, nous allons présenter en détail quelques plates-formes multi-agents, et plus particulièrement la plate forme Jade[1].

3.2.8. Les domaines d'applications des systèmes multi-agents

Les systèmes multi agents étant issus du domaine de l'*IAD*, ils permettent de modéliser des applications où une modélisation classique est inadéquate, et où le système est souvent naturellement distribué. Les domaines d'application des *SMA* sont particulièrement riches. Nous en citerons seulement les principales directions.

3.2.8.1. La Simulation Multi-Agents

Les systèmes complexes sont des systèmes où les techniques de modélisation classique sont difficilement utilisables. En, effet, dans ces systèmes, les paramètres sont beaucoup trop nombreux ou contradictoire pour pouvoir être pris en compte. Parfois même, il n'est pas possible de connaître l'ensemble des paramètres qui interviennent pour la modélisation. L'approche multi agent permet alors d'avoir recours à une modélisation local. Cette modélisation permet grâce aux principes d'émergence présents dans les SMA d'obtenir un système ayant les propriétés attendues. Il s'agit notamment de l'ensemble des systèmes permettant la simulation d'écosystèmes. Un bon exemple concerne la simulation d'une fourmilière [Drogoul 93]. En effet, lors de ses travaux de thèses, Alexis Drogoul a modélisé une fourmilière en utilisant des agents pour modéliser les fourmis. Il a alors réussi à démontrer que l'on pouvait obtenir un but global qui était la survie de la fourmilière sans jamais avoir programmé cet élément dans le système mais uniquement à partir de l'interaction des agents par émergence organisationnelle.

[1] Jade : http://jade.cselt.it

3.2.8.1.1. L'intérêt des SMA dans la simulation

On utilise les SMA pour simuler des interactions existantes entre agents autonomes. Le but est de déterminer l'évolution de ce système afin de prévoir l'organisation finale. Ce qui importe c'est le comportement d'ensemble et non pas le comportement individuel. L'autonomie permet ici de simuler le comportement exact d'une entité. Les intérêts des SMA comparés à d'autres techniques de modélisation peuvent être résumés en trois points principaux :

- Un *SMA* est plus à même de générer des phénomènes émergents,
- Un *SMA* fournit une *description naturelle* du système,
- Un système multi-agents est *flexible*.

3.2.8.1.2. Les applications des SMA dans la simulation

Les champs d'application sont nombreux, en particulier dans *les sciences sociales, politiques ou économiques*, on citera par exemple *la modélisation des flux (évacuation, trafic, gestion de flux de consommateurs, migration intra-urbaines, …etc)*, celle *des marchés (la bourse, simulation stratégique)*, ou bien encore *des entreprises (risque, organisation de la société)*. Les applications existent aussi dans des domaines plus différents comme en *météorologie, l'écologie (Cycle du Carbone), Contrôle et modélisation de ressources renouvelables (COMORE à l'INRIA))*, ou même *le sport (STAPS Marseille)*, on prend comme exemple les thèmes suivants :

- ***Sociologie :*** étude de la modélisation de communautés de développeurs open source,
- ***Socio-ethnologie :*** modélisation de *la coopération entre êtres humains* au sein d'un groupe, des dynamiques d'opinion, et de déplacements de populations en réponse à des modifications de l'environnement ou des règles sociales,
- ***Biologie* :** Simulation de différents aspects du *système immunitaire, cancérologie, croissance de bactéries,*
- ***Economie :*** Micro/Macro économie, flux monétaires,
- ***Physique :*** modélisation d'écoulement,
- ***Ethologie :*** animaux sociaux, relations entre espèces *(compétition, sélection parallèle, migrations, etc.…).*

L'intérêt de ces simulations et de pouvoir considérer aussi bien des paramètres quantitatifs (c'est-à-dire des paramètres numériques) que qualitatifs (des comportements individuels faisant éventuellement appel à des raisonnements stratégiques). De ce fait, l'utilisateur d'un tel simulateur a un rôle actif. Il emploie un système multi-agent comme s'il s'agissait d'un laboratoire miniature, déplaçant des individus, changeant leur comportement et modifiant les conditions environnementales. Ainsi, à la différence des approches classiques, la simulation multi-agent ne se réduit pas à l'implémentation d'un modèle puis à l'analyse d'une réponse de ce modèle en fonction des paramètres d'entrées, mais participe au processus de recherche de modèles [Ferber 95]. Enfin, les systèmes multi-agent permettent la modélisation complexes dont les structures globales émergent des interactions entre individus [Ferber 95]. Il existe bien évidement un grand nombre d'autres applications de simulation comportementale, donnent un aperçu des domaines variés dans lesquelles les *SMA* peuvent être employés.

3.2.8.2. Ecosystème et modèle individus centrés

Les modèles individus centrés, c'est-à-dire orientés individus sont des simulations basées sur les conséquences globales d'interactions locales entre membres d'une application. Ces individus peuvent représenter des plantes et des animaux dans un *éco-système*, *des véhicules dans la circulation, des personnes dans une foule, ou des personnages autonomes dans une animation ou un jeu.* Ces modèles consistent typiquement en un environnement dans le quel les interactions se passent et un certain nombre d'individus définit par leur comportement *(règle comportementale)* et des paramètres caractéristiques. Dans un modèle *orienté-individus*, les caractéristiques de chaque individu peuvent être suivies de façon continue. Les modèles *orientés individus* sont aussi appelés *orientés entités* ou *orientés-agents*, et on parle de *simulation multi agent.* Dans ces systèmes, un agent *(individu)* correspond toujours à une entité du monde réel *(un animale, une personne, un objet, une bactérie, etc.).* De nombreuses applications de simulation utilisent des *SMA* basés sur un modèle individu centré.

3.2.8.3. La robotique distribuée

La robotique distribuée porte sur la réalisation non pas d'un seul robot, mais d'un ensemble de robots qui coopèrent pour accomplir une mission. A la différence du domaine d'application précédent la robotique distribuée utilise des agents concrets qui se déplaçant dans un environnement réel. Le domaine de la robotique distribué recouvre en fait deux types de robotique bien distincte, la robotique cellulaire et la robotique mobile [Ferber 95].

3.2.8.4. Résolution de Problèmes et Systèmes d'aide à la décision

L'*IAD* est permet de résoudre les problèmes de complexité des gros programmes de l'*IA*, l'exécution est alors distribuée, mais le contrôle reste centralisé. Contrairement aux *SMA*, où chaque agent possède un contrôle total sur son comportement. Pour résoudre un problème complexe, il est plus simple de concevoir des programmes relativement petits *(les agents)* en interaction, qu'un seul gros programme homogène. L'autonomie permet au système de s'adapter dynamiquement aux changements imprévus qui interviennent dans l'environnement. Les systèmes d'aide à la décision *(SAD)* sont présents dans de nombreux domaines et ont pour objectif d'aider le décideur dans sa tâche en lui fournissant tous les éléments pertinents pour la prise de décision. Cela consiste très souvent à extraire de l'information depuis de multiples sources et à la traiter. Des traitements de l'information pour une prise de décision multi-criteres sont alors nécessaires. Les *SMA* apparaissent comme étant bien adaptés pour traiter de l'information qui peut revêtir diverses formes et provenir de diverses sources. En effet, il faut pouvoir faire la corrélation entre l'ensemble des éléments obtenus pour les présenter à l'utilisateur. L'approche à base d'agents permet d'effectuer cette corrélation en utilisant la négociation et la coopération entre agents. Il s'agit donc d'un domaine d'application pour les *SMA* [Duvallet 01].

3.2.8.5. Le domaine du commerce électronique et les agents du WEB

Le domaine du commerce électronique est un domaine en plein essor qui permet de favoriser les transactions commerciales, Ce domaine utilise l'outil informatique et plus particulièrement les ressources mises à disposition par Internet pour rapprocher les acteurs commerciaux dans certains domaines. Ce domaine se rapproche de celui de l'aide à la décision et peut même être confondu avec celui-ci dans certaines circonstances car il est caractérisé par la mise en place de moyens permettant d'extraire de l'information sur les produits, les marchés, Les acteurs du marché. L'utilisation de systèmes multi agents est une bonne solution ici aussi car elle permet comme pour

le domaine de l'aide à la décision de faire de la fusion d'informations. La dénomination *agent du WEB* désigne les agents qui circulent sur le réseau Internet pour extraire de l'information. Ces agents qui sont en général mobiles se déplacent au travers des sites *WEB*. Certains systèmes ont pour objectif de fournir au consommateur une valeur ajoutée en lui fournissant des informations supplémentaire afin qu'il fasse un choix approprié selon les critères qu'il s'est fixé. Il peut s'agir d'informations comparatives entre produits ou encore d'études qualitatives.

Prenons un exemple concret: vous souhaitez acheter une nouvelle voiture mais vous n'avez pas encore décidé du modèle exacte. Jusqu'à maintenant, vous deviez visiter un certain nombre de concessionnaires, étudier des documentations techniques, acheter des magazines spécialisés qui effectuent eux-mêmes des tests et études comparatives. Tout ce travail est fastidieux et coûteux en temps. Le but des applications dans le domaine du commerce électronique va être d'extraire toute cette information qui est disponible à divers endroits du *WEB* et de n'en retenir que celle qui vous intéresse. Il va s'agir d'un travail d'extraction, de filtrage et d'analyse de l'information disponible. La multiplicité des sources d'informations disponibles sur le WEB et la complexité des traitements à effectuer rend particulièrement appropriée l'utilisation du paradigme agent pour le développement des applications de commerce électronique [Duvallet 01].

4. Conclusion

Ce chapitre nous a permet de faire un état de l'art sur l'intelligence artificielle distribuée, et plus particulier le domaine des systèmes multi-agents. Nous avons présenté les notions de base qui définissent ce domaine en considérant le concept d'agent comme entité individuelle, puis comme individu au sein d'un univers d'agents. Enfin, nous avons montré les domaines d'applications et l'intérêt des SMA, notamment celui de la simulation et le commerce électronique.

D'après cet aperçu, on constate que les systèmes multi-agents ont l'avantage d'être proches des problèmes et de champs d'applications réels, qu'ils soient purement informatiques, biologiques, sociaux ou physiques. Les approches multi-agents permettent, non seulement une conception modulaire du système, mais également une distribution du contrôle et des données au sein des sous systèmes. L'utilisation des systèmes multi-agents comme outils de simulation présente plusieurs avantages, particulièrement l'aide à la compréhension du fonctionnement d'organisations à des fins d'aide à la décision. Toutefois, dans le cadre d'une simulation social, la mise au point de modèles *SMA* pose le problème d'acquisition de connaissances d'un collectif.

Enfin, les systèmes multi-agents constituent aujourd'hui un domaine de recherche à part entière. Cependant, ce domaine, qui est très récent et très ouvert, reste non formalisé de façon consensuelle et exhaustive. Au chapitre 2, nous présentons certains efforts qui visent à normaliser ce domaine et surmonter ses difficultés. Il s'agit de la norme *FIPA* et les environnements de développement orienté-agent.

Chapitre 2
La norme FIPA et la plateforme multi-agents JADE

1. Introduction

Actuellement les réflexions dans le domaine des *SMA* s'orientent vers la mise en application de ce paradigme. Ainsi, plusieurs travaux ont tenté de réutiliser les architectures théoriques et applicatives existantes pour construire des environnements de développement de ces systèmes. Ces environnements de développement ou plateformes multi-agents sont nécessaires pour renforcer le succès de la technologie multi-agents. En effet, elles permettent aux développeurs de concevoir et réaliser leurs applications sans effort et éliminent, dans la plupart des cas, la nécessité d'être familier avec les différents concepts théoriques des *SMA*. Donc le meilleur moyen pour construire un système multi-agents est d'utiliser une plate-forme multi-agent. *Une plate-forme multi-agent est un ensemble d'outils nécessaire à la construction et à la mise en service d'agents au sein d'un environnement spécifique. Ces outils peuvent servir également à l'analyse et au test du SMA ainsi créé. Ces outils peuvent être sous la forme d'environnement de programmation (API) et d'applications permettant d'aider le développeur.*

Dans ce chapitre nous allons dans un premier temps présenter la norme *FIPA* pour les *SMA* avant de donner un aperçus de quelques plateformes existants. La plateforme *JADE (Java Agent DEvelopment framework)*, compatible *FIPA*, est celle que nous avons retenue pour implémenter notre architecture multi-agents, elle sera davantage détaillée et sera présenté ensuite.

2. La norme *FIPA* et quelques plates-formes *SMA*

2.1. La norme *FIPA*

La *FIPA*[1] est une organisation fondée en 1996 dont l'objectif est de produire des standards pour l'interopération d'agents logiciels hétérogènes. Par la combinaison *d'actes de langages, de logique des prédicats et d'ontologies publiques*, la *FIPA* cherche à offrir des moyens standardisés permettant d'interpréter les communications entre agents de manière à respecter leur sens initial. Afin d'atteindre ce but, le *FIPA* émet des standards couvrant:

- Les applications *(applications nomades, agent de voyage personnel, applications de diffusion audiovisuelles, gestion de réseaux, assistant personnel…) ;*
- Les architectures abstraites, définissant d'une manière générale les architectures d'agents ;
- Les langages d'interaction *(ACL)*, les langages de contenu *(comme SL, CCL, KIF ou RDF)* et les protocoles d'interaction ;
- La gestion des agents *(nommage, cycle de vie, description, mobilité, configuration);*
- Le transport des messages : représentation *(textuelle, binaire ou XML)* des messages *ACL*, transport *(par IIOP, WAP ou HTTP)* de ces messages.

Ces standards évoluent, et sont régulièrement mis à jour, ainsi que de nouveaux standards qui sont nouvellement proposés. Les standards qu'édicte la *FIPA* ne constituent pas vraiment une plate-forme de construction multi-agents. Ce n'est pas non plus l'objectif que s'est fixé la *FIPA.* Tout au plus, la *FIPA* normalise une plate-forme d'exécution standardisée dans un but d'interopérabilité. Ces normes s'appliquent donc pour la plupart en phase de déploiement. Elles n'abordent pas les phases d'analyse ni de conception. Elles peuvent cependant guider certains choix d'implémentation.

[1] *Foundation for Intelligent Physical Agent.* Fipa : http://www.fipa.org

La *FIPA* a publié un ensemble de documents de spécifications qui recouvrent de nombreux aspects des systèmes multi-agents. Ces spécifications concernent deux axes : un aspect technologique et un autre applicatif. Du point de vue technologique, la *FIPA* définit la structure générale d'une plateforme agent [Fipa 02a], le langage de communication entre agents *ACL (Agent Communication Language)* [Fipa 02b, Fipa 02c], le langage de contenu *(sémantique)* [Fipa 02d, Fipa02e], ou encore les protocoles d'interactions *(aspects conversationnels)* [Fipa 03]. Du point de vue applicatif, la *FIPA* définit les applications agents *: assistant personnel, assistant de voyage, gestion du réseau et service audio-visuel.*

Les spécifications fondamentales de la *FIPA* ont été promues au rang de standard *(décembre 2002).* C'est un pas important qui marque la stabilité et le développement de la communauté *FIPA.* En effet, il existe aujourd'hui plus d'une dizaine de plateformes conformes aux spécifications *(cf. tableau 2.1).* Certaines d'elles sont libres et ont été utilisées dans plusieurs projets à travers le monde. Parallèlement à ces efforts, un réseau de telles plateformes a été mis en place dans le cadre du projet européen *Agentcities*[1]. Ce projet vise à fournir un outil d'évaluation et un environnement de démonstration. Ainsi, les spécifications *FIPA* sont en train de devenir un standard pour l'utilisation de technologies orientées agents dans les milieux industriels et commerciaux.

Plateforme	**Société**	**Site Web**
Zeus	Brittish Telecom	http://www.labs.bt.com/projects/
Fipa-Os	Emorphia	http://fipa-os.sourceforge.net/
April Agent Platform	Fujitsu	http://sf.us.agentcities.net/aap/
Grasshopper IV++	Technologies Av	http://www.grasshopper.de/
Jade	Tilab (Cselt)	http://jade.cselt.it/
Leap	Motorola	http://leap.crm-paris.com/
Agent Development Kit	Tryllian BV	http://www.tryllian.com/
JACK Intelligent Agents	Agent Oriented Software	http://www.agent-software.com/

Tableau 2.1. Quelques unes des plateformes répondant aux spécifications de la FIPA.

2.2. Quelques plateformes multi-agents

La notion de plateforme est liée à l'implémentation des systèmes multi-agents. Une plateforme est un environnement permettant de gérer le cycle de vie des agents et dans lequel les agents ont accès à certains services. Dans [Garneau et Delisle 02], une étude comparative des principales plateformes multi-agents existantes a été effectuée, selon des caractéristiques jugées importantes pour les environnements de développement des *SMA.* Nous avons choisis de décrire, ci-après, une liste des plateformes caractérisés par leur popularité et de leur pertinence. Pour chacun de ces plateformes, nous mettons l'accent sur ses principes de base, ses avantages et ses limites.

2.2.1. MadKit

La plateforme *MadKit*[2], développée par le laboratoire *Lirmm* de l'université de MontpellierII, est un environnement basé sur la méthodologie *Alaadin* ou *AGR (agent/groupe/rôle). MadKit* fournit une *API* permettant la construction d'agent en étendant une classe d'agent abstraite. Chaque agent peut tenir différents rôles au sein de différents groupes. Les agents sont lancés par le noyau de *MadKit,* qui assure aussi les services de gestion des groupes et de communication. Il est ainsi possible d'échanger des messages directement à un agent ou à l'ensemble d'un groupe. Cette plateforme est

[1] Agentcities : http://www.agentcities.org

2 MadKit : http://www.madkit.org

surtout intéressante pour l'approche organisationnelle qu'elle met en avant lors de l'analyse et de la conception d'un *SMA.*

2.2.2. Zeus

La plateforme *Zeus*[1] est développée en Java et selon la norme *FIPA,* par *British Telecom. Zeus* est un environnement complet qui utilise une méthodologie fondée sur la notion de rôle appelée *role modeling* pour le développent de systèmes collaboratifs. Un agent dans *Zeus* est constitué en trois couches : *la couche de définition* qui contient les capacités de raisonnement et des algorithmes d'apprentissage, *la couche organisationnelle* qui décrit les relations entre les agents et *la couche de coordination* qui définit les interactions entre les agents. L'utilisation de *Zeus* pour le développement de *SMA* est cependant conditionnelle à l'utilisation de la méthodologie *role modeling*. De plus, L'outil est assez complexe et sa maîtrise nécessite beaucoup de temps.

2.2.3. AgentBuilder

La plateforme *AgentBuilder*[2], développé par *Reticular Systems,* est une implémentation proche du langage *Agent-0* proposé par [Shoham 93]. Les agents sont décrits avec le langage *Radl* (*Reticular Agent Definition Language*), qui permet de définir les règles du comportement de l'agent. Les règles se déclenchent en fonction de certaines conditions et sont associées à des actions. Les conditions portent sur les messages reçus par l'agent tandis que les actions correspondent à l'invocation de méthodes *Java.* Il est aussi possible de décrire des protocoles définissant les messages acceptés et émis par l'agent. Le langage *KQML* est utilisé comme langage de communication entre les agents. Toutefois, *AgentBuilder* est un outil complexe qui demande des efforts d'apprentissage importants et de bonnes connaissances dans le domaine des *SMA* pour être utilisé de façon performante.

2.2.4. Jack

Jack[3] *Intelligent Agents* a été proposé par *Agent Oriented Software* pour le développement d'agent de type *BDI,* tout en fournissant une *API* orientée objets. Pour cela, Jack offre un langage proche de la programmation logique *JAL (Jack Agent Language*) qui peut ensuite être compilé en classes *Java.* Le noyau de *Jack* gère la concurrence des tâches entre les agents, la réaction aux événements et l'infrastructure de communication. Cependant, *Jack* est très long à maîtriser et le manque de support graphique complique l'implémentation et le déploiement des systèmes.

2.2.5. Aglet

L'*Aglet* est un environnement conçu initialement par *IBM* pour le développement d'agents mobiles. La plateforme prend en charge les services nécessaires à la migration d'agents et à la gestion des communications. Une *Aglet* est une classe *Java* étendant une classe fournie par le noyau et redéfinissant les fonctions à activer lors des *sérialisations*/ *désérialisations* des agents.

2.2.6. Swarm

Swarm[4] de *Development Group* est un environnement destiné à la simulation des sociétés d'agents réactifs. L'inspiration du modèle d'agent utilisé vient de la vie artificielle. *Swarm* est l'outil

1 Zeus : http://www.labs.bt.com/projects/agents/zeus/

2 AgentBuilder : http://www.agentbuilder.com

3 Jack Intelligent Agents : http://www.agent-software.com.au

4 Swarm : http://www.swarm.org

privilégié de la communauté américaine et des chercheurs en vie artificielle. Cet environnement offre un ensemble de bibliothèques qui permettent l'implémentation des *SMA* avec un grand nombre d'agents simples qui interagissent dans le même environnement.

3. La plateforme *JADE*

3.1. Bref description de *JADE*

JADE[1] est une plate-forme multi-agent créé par le laboratoire *TILAB*[2] et décrite par Bellifemine et al. Dans [Bellifemine et al 99] [Bellifemine et al 00]. *JADE* permet le développement de systèmes multi-agents et d'applications conformes aux normes *FIPA* [Jade 00] [Jade 02]. Elle est implémentée en *JAVA* et fourni des classes qui implémentent *JESS* pour la définition du comportement des agents. *JADE* possède trois modules principaux (nécessaire aux normes *FIPA*) qui sont activés à chaque démarrage de la plate-forme:

- ***DF (Director Facilitor)*** fournit un service de *pages jaunes* à la plate-forme ;
- ***ACC (Agent Communication Channel)*** gère la communication entre les agents ;
- ***AMS (Agent Management System)*** supervise l'enregistrement des agents, leur authentification, leur accès et l'utilisation du système.

La plateforme *JADE* peut être distribuée sur plusieurs machines hôtes. Une seule application *Java,* et donc une seule machine virtuelle *VM (Virtual Machine) Java*, est exécutée sur chaque hôte. Les agents sont implémentés comme des *threads Java* et les événements *Java* sont utilisés pour la communication efficace et légère entre agents sur un même hôte [Bellifemine et al 04].

3.2. Architecture logiciel de la plate-forme JADE

JADE reprend donc l'architecture de l'*Agent Management Reference Model* proposé par *FIPA.* Les différents modules présentés dans la figure2.1 sont présentés sous forme de services. Les services de base proposés sont le *Directory Facilitator (DF)* et l'*Agent Management System (AMS).* Il est possible de lui demander de tenir en plus le service de *Message Transport Service (MTS)* pour communiquer entre plusieurs plates-formes. Mais ce service sera chargé à la demande pour ne conserver par défaut que les fonctionnalités utiles à tout type d'utilisation. L'agent est l'acteur fondamental de la plate-forme, un *Agent Identifier (AID)* identifie un agent de manière unique. Le *DF* est un composant qui fait office d'annuaire. C'est un *service de pages jaunes* qui permet de mettre en relation les agents avec leurs compétences. Un agent peut enregistrer ses compétences dans le *DF* ou interroger le *DF* pour connaître les compétences proposées par les autres agents. L'*AMS* est un autre composant important car il contrôle l'accès et l'utilisation de la plate-forme et maintient un répertoire contenant les adresses de transport des agents de la plate forme. Ce service est plus un service de type *pages blanches* qui effectue la correspondance entre l'agent et l'*AID.* Chaque agent doit s'enregistrer à un *AMS* pour avoir un *AID.* Il n'y a qu'un *AMS* par plate-forme. Le *MTS* est une méthode par défaut de communication entre agents de différentes plates-formes. Cela permet l'interconnexion entre systèmes hétérogènes ou tout au moins de système ne communicant pas de la même façon. L'*Agent Platform (AP)* constitue l'infrastructure physique sur laquelle se déploient les agents. Il contient le *DF,* l'*AMS* et le *MTS.* Enfin, l'*Agent Identifier (AID)* est un

1 JADE (Java Agent DEvelopement framework): http://jade.cselt.it

2 TILab (anciennement CSELT) groupe de recherche de Telecom Italia Lab S.p.A.

identifiant précis d'un agent. On lui donne plusieurs paramètres tels que *l'adresse de transport, l'adresse de service de résolution de nom, …*exemple: *name@HAP (Home Agent Platform).*

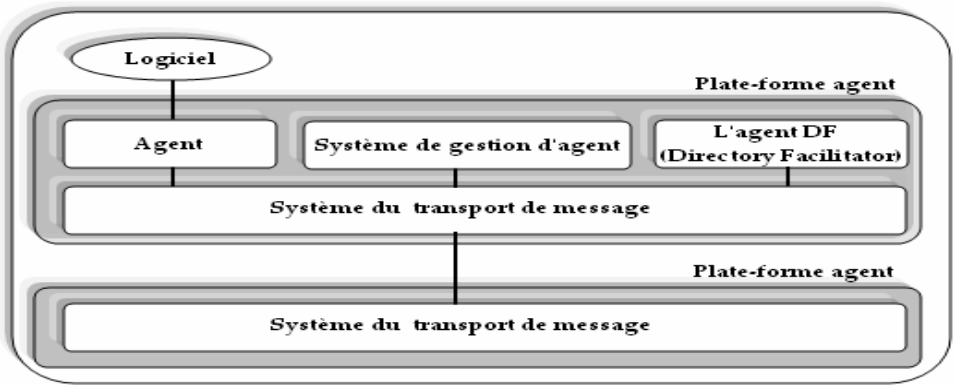


Figure 2.1. Architecture logiciel de La plate-forme JADE

Dans la plate-forme *JADE,* deux méthodes sont fournies par la classe *Agent* afin d'obtenir l'identifiant de l'agent *DF* par défaut et celui de l'agent *AMS: getDefaultDF()* et respectivement *getAMS()*. Ces deux agents permettent de maintenir une liste des services et des adresses de tous les autres agents de la plate-forme. Le service *DF* propose quatre méthodes afin de pouvoir : *Enregistrer* un agent dans les pages jaunes *(register); Supprimer* un agent des pages jaunes *(deregister); Modifier* le nom d'un service fourni par un agent *(modify);* et enfin *Rechercher* un service *(search).* Le service AMS s'utilise généralement de manière transparente (chaque agent créé est automatiquement enregistré auprès de l'*AMS* et se voit attribué une adresse unique). Ces deux services fournissent donc les *annuaires* qui permettent à n'importe quel agent de trouver un service ou un autre agent de la plate-forme.

3.3. Langage de communication de la plate-forme JADE

Le langage de Communication de la plate-forme *JADE* est *FIPA-ACL (Agent Communication language).* La classe *ACLMessage* représente les messages qui peuvent être échangés par les agents. La communication de messages se fait en mode asynchrone. Lorsqu'un agent souhaite envoyer un message, il doit créer un nouvel objet *ACLMessage*, compléter ces champs avec des valeurs appropriées et enfin appeler la méthode *send()*. Lorsqu'un agent souhaite recevoir un message, il doit employer la méthode *receive()* ou la méthode *blockingReceive()*.

3.3.1. Structure d'un message

Un message *ACL* dispose obligatoirement des champs suivants [Fipa 02b] :

- ***Performative*****:** type de l'acte communicatif ;
- ***sender*****:** l'émetteur du message ;
- ***receiver*****:** le destinataire du message ;
- ***reply-to*****:** participant à l'acte de communication ;
- ***content*****:** le contenu du message (l'information transportée par la performative) ;
- ***language*****:** le langage dans lequel le contenu est représenté;
- ***encoding*****:** décrit le mode d'encodage du contenu du message ;
- ***ontology*****:** le nom de l'ontologie utilisé pour donner un sens aux termes utilisés dans le *content* [Fipa 02d] ;
- ***protocol*****:** contrôle la conversation ;

• ***conversation-id*:** identificateur de la conversation ;
• ***reply-with*:** identificateur unique du message, en vue d'une référence ultérieure ;
• ***in-reply-to*:** référence à un message auquel l'agent est entrain de répondre (précisé par l'attribut *reply-with* de l'émetteur) ;
• ***reply-by*:** impose un délai pour la réponse.

Tous les attributs de la classe *ACLMessage* peuvent être obtenus et modifiés par les méthodes *set/get()*. Le contenu des messages peut être aussi bien du texte que des objets car la sérialisation *Java* est supportée.

3.3.2. Actes de communication

FIPA-ACL possède *21 actes communicatifs,* exprimés par des performatives, qui peuvent être groupés selon leur fonctionnalité de la façon suivante [Fipa 02c] :

• ***passage d'information :*** *inform, inform-if* (macro act), *inform-ref* (macro act), *confirm* et *disconfirm;*
• ***réquisition d'information :*** *query-if, query-ref* et *subscribe;*
• ***négociation :*** *accept-proposal, cfp, propose et reject-proposal*
• ***distribution de tâches (ou exécution d'une action) :****request, request-when, requestwhenever, agree, cancel et refuse;*
• ***manipulation des erreurs :*** *failure* et *not-understood*.

3.3.3. Le transport des messages

Les plateformes pour agent *(AP)* fournissent un service de transport de message *(MTS)* aux agents qu'elles contrôlent. Ce service comprend le transport des messages entre des agents d'une même plateforme et des agents de plateformes distantes. Pour cela, l'*AP* utilise une entité qui lui propose ce service, le canal de communication pour agent *(ACC)*. Ces canaux utilisent un protocole de transport de messages *(MTP)* pour transférer physiquement les messages.

3.3.3.1. Les différents modes d'envoie de messages

Un agent à la possibilité d'envoyer un message de trois façons différentes:

• L'agent *A* envoie à sont *ACC* local le message. L'*ACC* prend alors toutes les précautions pour envoyer le message à l'*ACC* distant correct en utilisant le *MTP* approprié. L'envoie du message à l'agent *B* sera alors géré par l'*ACC* distant.
• L'agent *A* envoie directement le message à l'*ACC* de l'*AP* distant où l'agent *B* réside. L'*ACC* distant délivre ensuite le message à l'agent *B.* Pour utiliser cette méthode, l'agent *A* doit supporter l'accès à une interface *MTP* de l'*ACC* distant.
• L'agent *A* envoie directement son message à l'agent *B* en utilisant un mécanisme de transfert direct. Le transfert du message, l'adressage et les messages d'erreurs doivent alors être gérés par les agents. Ce mode de communication n'est pas couvert par la *FIPA.*

3.3.3.2. Structure du message de transport

Les messages de transport sont composés de deux parties: une enveloppe contenant les informations de transport et le message *ACL* à transporter. La structure des messages de l'*ACL* a déjà été vue précédemment, détaillons maintenant la structure de l'enveloppe. Chaque protocole

de transfert peut utiliser une représentation interne différente pour décrire les enveloppes de message. Mais ces représentations doivent exprimer les mêmes termes, représenter la même sémantique et engendrer les actions correspondantes. Voici les déclarations générales sur la forme d'une enveloppe de message:
- Une enveloppe de message comprend une collection de paramètre qui peuvent être optionnels, cette enveloppe doit contenir au moins les paramètres suivant: *to, form, date* et *acl-representation.*
- Un paramètre est une paire formée d'un *nom* et d'une *valeur.*

3.4. Protocoles d'interaction (FIPA-Query et FIPA-Request)

Dans le protocole *FIPA-request,* un agent sollicite un autre agent pour exécuter des actions et l'agent récepteur retourne soit une réponse favorable à l'exécution d'actions, soit une réponse défavorable expliquée par telle ou telle raison. Supposons que l'agent ***i*** a besoin de l'agent ***j*** pour exécuter l'action ***action.***

a. L'agent ***i*** envoie *request* à l'agent ***j***. Si l'agent ***j*** accepte la requête, il retourne *agree*. Ensuite, quand ***j*** a fini d'exécuter *"action"*, il en informe ***i*** en utilisant *inform*.
b. Si l'agent ***j*** accepte mais rencontre un problème durant le traitement de *"action"*, il retourne *failure* et les raisons de l'échec.
c. Si l'agent ***j*** n'accepte pas la requête de l'agent ***i***, ***j*** retourne *refuse* et les raisons de ce refus.

Le protocole *FIPA-Query* signifie que l'agent émetteur sollicite l'agent récepteur pour exécuter un des types d'un performatif *inform*, c'est-à-dire pour répondre à la demande. Supposons que l'agent ***i*** fasse une demande à l'agent ***j***.

a. L'agent ***i*** envoie un performatif *query* à l'agent ***j***. Si l'agent ***j*** peut répondre à la demande, il l'informe en utilisant le performatif *inform*.
b. Si l'agent ***j*** a essayé de répondre à la demande mais qu'il ne le peut pas, il retourne *failure* et les raisons de cette impossibilité.
c. Si l'agent ***j*** refuse de répondre à la demande, il retourne *refuse* et les raisons de ce refus.

3.5. Le protocole *Contract-Net* de *JADE*

Le Protocole *Contract-Net (CNP)* a été défini par Smith [Smith 80]. Il est un mécanisme de négociation par *appel d'offre (ou Contract)* entre deux types d'agents : *l'agent gestionnaire* et *les agents contractants. L'agent gestionnaire,* souhaitant sous-traiter une tâche qu'il doit accomplir, est l'initiateur du contrat. Chaque *agent contractant* est un agent auquel on propose ce contrat. Ce protocole très souvent utilisé, a été normalisé par l'organisation *FIPA* et implémenté dans la plate-forme *JADE.*

3.5.1. Initiateur *Contract-Net*

La classe *ContractNetInitiator* (abstraite) implémente le protocole *FIPA ContractNet* du point de vue de l'agent qui initialise le protocole. L'agent qui envoie un message *CFP (Call For Proposal).* Le comportement d'initiateur doit également prendre en compte les délais d'attente pour les réponses. Le temps limite est spécifié dans le champ *reply-by* du message *ACL* passé au constructeur de cette classe. Si rien n'est spécifié alors une attente infinie est utilisée par défaut. Toutes réponses arrivées hors délais ne sont pas consommées et reste dans la file d'attente des messages de l'agent.

Cette classe propose également un jeu de méthodes destinées à superviser chaque étape du protocole. Elles sont appelées lorsque des messages particuliers sont reçus ou sont à envoyer et doivent être surchargées afin d'adapter le protocole à son contexte d'utilisation :

- La méthode *prepareCfps()* est la première invoquée.C'est l'*appel d'offre* ou *proposition (message CFP).*
- La méthode *handlePropose()* est appelée dés que le message *PROPOSE* a été reçu (ie. l'offrant fait une proposition). Elle prépare les messages d'acceptation ou de refus correspondant (message *ACCEPT-PROPOSAL* ou *REJECT-PROPOSAL).*
- La méthode *handleRefuse()* est appelée dés que le message *REFUSE* a été reçu (ie. l'offrant refuse toute proposition).
- La méthode *handleNotUnderstood()* est appelée dés que le message *NOT-UNDERSTOOD* a été reçu (ie. l'offrant ne comprend pas la proposition).
- La méthode *handleInform()* est appelée dés que le message *INFORM* a été reçu. (ie. l'offrant confirme que sa proposition a bien été acceptée par l'initiateur).
- La méthode *handleFailure()* est appelée dés que le message *FAILURE* a été reçu (ie. l'offrant ne peut satisfaire la demande de l'initiateur pour une raison quelconque).
- La méthode *handleOutOfSequence()* est appelée si un message hors séquence (ie. avec un mauvais conversation-id) et respectant le protocole *ContractNet* est reçu.

Afin de pouvoir être en interaction avec plusieurs offrants, deux méthodes supplémentaires sont disponibles afin de collecter les messages par volé : *handleAllResponse()* et *handleAllResultNotifications*() qui permettent respectivement de récupérer la première série de réponses (ie. *NOT-UNDERSTOOD, REFUSE, PROPOSE*) et la seconde (ie. *FAILURE, INFORM*).

Enfin, la classe *ContractNetInitiator* dispose d'une alternative aux méthodes décrites ci-dessus : il est possible de les remplacer par des comportements génériques correspondants. Les méthodes *registerPrepareCfps(), registerHandlePropose(), registerHandleAllResponses(), registerHandleAllResultNotifications()*, permettent de spécifier ces comportements et d'écraser les méthodes qu'ils remplacent.

3.5.2. Offrant *Contract-Net*

La classe *ContractNetResponder* (abstraite) implémente le protocole *FIPA ContractNet* du point de vue d'un agent qui répond au message *CFP* (ie. l'offrant). Il est important de passer le bon modèle de message au constructeur de cette classe car il est utilisé afin de séparer les messages *ACL* du protocole des autres messages reçus par l'agent. La méthode *createMessageTemplate* peut être utilisée à ces fins. Tout comme la classe *ContractNetInitiator*, cette classe peut être facilement étendue en surchargeant une ou plusieures de ses méthodes *prepare()* afin de l'adapter à son contexte d'utilisation. Ces méthodes permettent de superviser les étapes du protocole et en particulier de préparer les messages à renvoyer à l'agent initiateur :

- La méthode *prepareResponse* est invoquée dès que l'appel d'offre de l'initiateur a été reçu. Cette méthode prépare le message de proposition *PROPOSE* (pour répondre au message *CFP*). Elle permet aussi de préparer un message *REFUSE* (refus de toute négociation) ou un message *NOT-UNDERSTOOD* en cas d'incompréhension de la demande de l'initiateur.
- La méthode *handleRejectProposal()* est appelée si un message de rejet d'offre est reçu.

• La méthode *prepareResultNotification()* est appelée dés que le message *ACCEPT_PROPOSAL* a été reçu (l'initiateur a accepté la proposition). Elle prépare le message de confirmation correspondant (message *INFORM*) ou en cas de problème un message *FAILURE* signalant l'échec de la négociation.
• La méthode *handleOutOfSequence()* est appelée si un message hors séquence (ie. avec un mauvais *conversation-id*) et respectant le protocole *Contract-Net* est reçu.

Les attributs des messages *ACL* retournés par ces différentes méthodes doivent être correctement sélectionnés. L'utilisation de la méthode *createReply()* est donc incontournable pour préparer les réponses. Elle permet entre autre de mettre la valeur appropriée dans le champ *in-reply-to* du message *ACL*. Ce champ est important car il contient le *conversation-id* qui est en fait un identificateur commun pour une série de messages échangés entre deux agents. Ceci permet à tout agent de maintenir plusieurs conversations sans se tromper dans les différents messages à envoyer. Tout comme la clase *ContractNetInitiator* il est possible de remplacer les méthodes *prepare()* de la classe par des comportements spécifiques. Les méthodes suivantes nommées *registerHandleRejectProposal(), registerPrepareResponse(), registerPrepareResultNotification()* et *registerHandleOutOfSequnece()* permettent cette opération.

L'*API* fournie par la plate-forme pour le protocole *ContractNet* est donc très complète et générique. Les opérations qui sont à la charge du programmeur reste la surcharge des différentes méthodes afin de les adapter au contexte d'utilisation du protocole et éventuellement la définition de *comportement(s)* visant à remplacer certaines méthodes.

3.6. L'API de JADE

JADE est composé des principaux *packages* suivants:

• *Jade.core* implante le noyau du système. Il possède la classe *agent* qui doit être étendue par les applications des programmeurs. Une classe *behaviour* (comportement) est contenue dans *jade.core.behaviours,* une sous classe de *jade.core.* Les comportements implémentent les tâches ou intentions des agents;
• Le package *jade.lang* contient un *sous-package* pour chaque langage de communication utilisé par *JADE,* en particulier *jade.lang.acl*;
• *Jade.content* contient un ensemble de classes qui permettent de définir des ontologies;
• *Jade.domain* contient toutes les classes *Java* qui représentent les entités de service définies par *FIPA,* en particulier *AMS* et *DF*;
• *Jade.gui* contient un ensemble générique de classes utiles pour la création d'interfaces graphique pour l'affichage, l'édition des messages *ACL,* et de la description des agents;
• *Jade.mtp* contient une interface *Java* que chaque *MTP* doit implémenter;
• *Jade.proto* contient des classes qui modélisent les protocoles standard d'interaction *FIPA (FIPARequest, FIPAQuery, FIPAContractNet,* etc.*)* et permettent aux programmeurs de définir leurs propres protocoles ;
• Enfin, le package *Jade.tools* implémente certains outils graphiques qui facilitent l'administration de la plate forme et le développement d'applications.

3.7. Comportements des agents dans la plate-forme JADE

Un agent doit être capable de gérer plusieurs tâches de manière concurrente en réponse à différents évènements extérieurs. Afin de rendre efficace cette gestion chaque agent de *JADE* est

composé d'un seul *thread* et chaque comportement qui le compose est en fait un objet de type *Behaviour*. Des agents *multi-thread* peuvent être créés mais il n'existe pour l'heure actuelle aucun support fournis par la plate-forme (excepté la synchronisation de la file des messages *ACL*). Afin d'implémenter un comportement, le développeur doit définir un ou plusieurs objets de la classe *Behaviour*, les instancier et les ajouter à la file des tâches *ready* de l'agent. Il est à noter qu'il est possible d'ajouter des comportements et sous-comportements à un agent ailleurs que dans la méthode *setup().* Tout objet de type *Behaviour* dispose d'une méthode *action()* (qui constitue le traitement à effectuer par celui-ci) ainsi que d'une méthode *done()* (qui vérifie si le traitement est terminé). Dans les détails, l'ordonnanceur exécute la méthode *action()* de chaque objet *Behaviour* présent dans la file des tâches de l'agent. Une fois cette méthode terminée, la méthode *done()* est invoquée. Si la tâche a été complétée alors l'objet *Behaviour* est retiré de la file. L'ordonnanceur est non-préemptif et n'exécute qu'un seul comportement à la fois, on peut donc considérer la méthode *action()* comme étant atomique. La façon la plus classique de programmer un comportement consiste à le décrire comme une machine à états finis. L'état courant de l'agent étant conservé dans des variables locales. Enfin, il existe également quelques méthodes supplémentaires afin de gérer les objets *Behaviour*:

- *reset()* qui permet de réinitialiser le comportement;
- *onStart()* qui défini des opérations à effectuer avant d'exécuter la méthode *action()*;
- *onEnd()* qui finalise l'exécution de l'objet *Behaviour* avant qu'il ne soit retiré de la liste des comportements de l'agent;

La plate-forme *JADE* fournit sous forme de classes un ensemble de comportements ainsi que des sous-comportements prêt à l'emploi. Elle peut les exécuter selon un schéma prédéfini, par exemple la classe *SequentialBehaviour* est supportée et exécute des sous-comportements de manière séquentielle. Toutes les classes prédéfinies dans *JADE* hérite de la classe Abstraite *Behaviour* :

• Classe *SimpleBehaviour* (abstraite): modélise un comportement simple. Sa méthode *reset()* n'effectue aucune opération.
• Classe *CompositeBehaviour* (abstraite) : modélise un comportement composé. Les actions effectuées par cette classe sont définies dans les comportements fils.
• Classe *FSMBehaviour* : Cette classe hérite de *CompositeBehaviour* et exécute des comportements fils suivant un automate à états finis défini par l'utilisateur. Lorsqu'un comportement enfant termine, sa valeur de fin retournée par la fonction *onEnd()* indique le prochain état à atteindre. Le comportement correspondant à cet état sera exécuté à la prochaine exécution de la classe. Elle se termine lorsque qu'un comportement associé à un état final à été exécuté.
• Classe *SenderBehaviour* : elle étend la classe *OneShotBehaviour* et encapsule une unité atomique qui effectue une opération d'envoie de message.
• Classe *ReceiverBehaviour* : Elle encapsule une unité atomique qui effectue une opération de réception de message. Ce comportement s'arrête dés qu'un message a été reçu. S'il n'y a pas de message dans la file d'attente de l'agent ou que le message ne correspond pas au *MessageTemplate* du constructeur de cette classe, alors il se met en attente.
• On peut aussi citer d'autres classes (abstraites) par exemples : *WakerBehaviour*, *ParrallelBehaviour*, *SequentialBehaviour*, *CyclicBehaviour*, *OneShotBehaviour*.

3.8. Outils de débogage de JADE

Pour supporter la tâche difficile du débogage des applications multi-agents, des outils ont été développés dans la plate-forme *JADE.* Chaque outil est empaqueté comme un agent, obéissant aux mêmes règles, aux mêmes possibilités de communication et aux mêmes cycles de vie d'un agent générique *(agentification de service).*

3.8.1. Agent *RMA* (Remote Management Agent)

Le *RMA* permet de contrôler le cycle de vie de la plate-forme et tous les agents la composant. L'architecture répartie de *JADE* permet le contrôle à distance d'une autre plate-forme. Plusieurs *RMA* peuvent être lancés sur la même plate-forme du moment qu'ils ont des noms distincts.

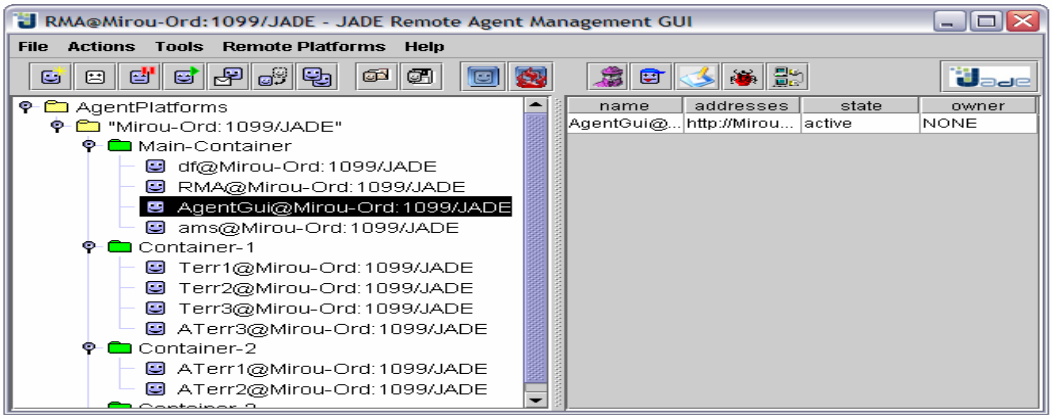


Figure 2.2. Interface graphique de l'agent *RMA.*

3.8.2. Agent Dammy

L'outil *DummyAgent* permet aux utilisateurs d'interagir avec les agents *JADE* d'une façon particulière. L'interface permet la composition et l'envoi de messages *ACL* et maintient une liste de messages *ACL* envoyés et reçus. Cette liste peut être examinée par l'utilisateur et chaque message peut être vu en détail ou même édité. Plus encore, le message peut être sauvegardé sur le disque et renvoyé plus tard.

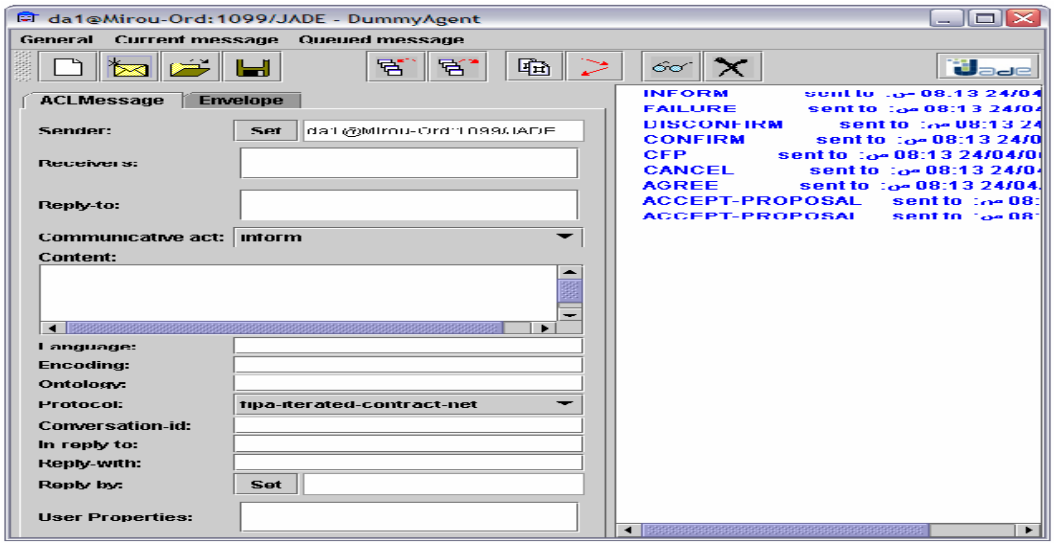


Figure 2.3. Interface graphique de l'agent *Dummy.*

3.8.3. Agent Direcory Facilitator

L'interface du *DF* (*Directory Facilitator*) peut être lancée à partir du menu du *RMA* .Cette action est en fait implantée par l'envoi d'un message *ACL* au *DF* lui demandant de charger son interface graphique. L'interface peut être juste vue sur l'hôte où la plate-forme est exécutée. En utilisant cette interface, l'utilisateur peut interagir avec le *DF*.

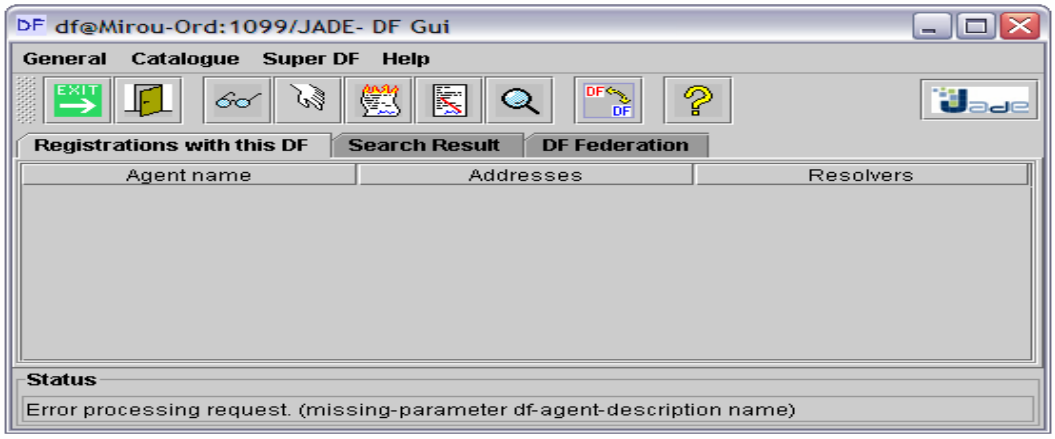


Figure 2.4. Interface graphique de l'agent *Directory Facilitator*.

3.8.4. Agent Sniffer

Quand un utilisateur décide d'épier un agent ou un groupe d'agents, il utilise un agent *sniffer*. Chaque message partant ou allant vers ce groupe est capté et affiché sur l'interface du *sniffer*. L'utilisateur peut voir et enregistrer tous les messages, pour éventuellement les analyser plus tard. L'agent peut être lancé du menu du *RMA* ou de la ligne de commande suivante : *Java jade.Boot sniffer:jade.tools.sniffer.sniffer* .

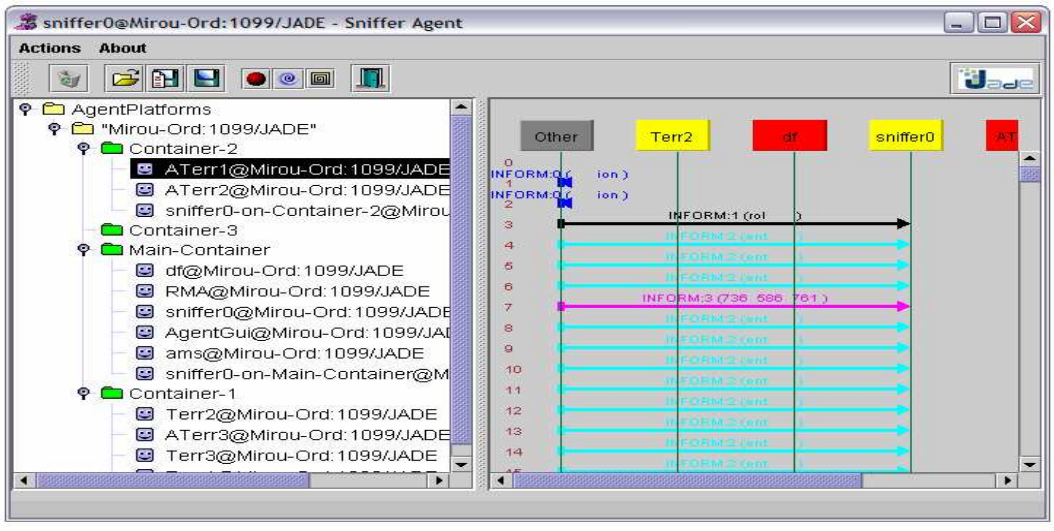


Figure 2.5. Interface de l'agent *Sniffer*.

3.8.5. Agent Inspector

Cet agent permet de gérer et de contrôler le cycle de vie d'un agent s'exécutant et la file de ses messages envoyés et reçus.

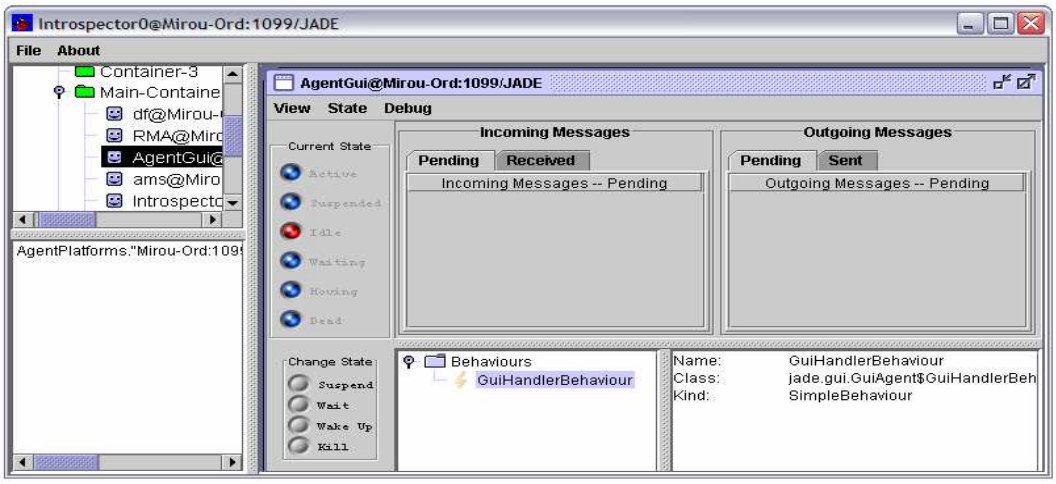


Figure 2.6. Interface de l'agent *Introspector*.

3.9. Le modèle d'agents

On a vu qu'une propriété importante d'un agent est son autonomie : un agent ne doit pas se limiter à réagir aux événements externes, mais il doit être aussi capable de prendre l'initiative de nouveaux actes communicatifs d'une façon autonome. Cette propriété d'autonomie exige que chaque agent ait un thread interne de contrôle. Cependant, un agent peut engager des conversations simultanées multiples, tout en poursuivant d'autres activités qui n'impliquent pas d'échanges de messages. *JADE* est totalement neutre vis à vis de la définition d'un agent et ne traite que des capacités bas niveau permettant la mise en oeuvre de communication et d'interaction entre les agents. Comme le modèle de comportement d'un agent *JADE* permet l'intégration simple de code dans les tâches des agents.

Avec la solution de multi-threading offerte directement par *Java, JADE* utilise l'abstraction comportement (*Behaviour*) pour modéliser les tâches qu'un agent peut exécuter. Les agents instancient donc leurs comportements selon leurs besoins et leurs capacités (cf. figure 2 . 7.). Les comportements travaillent tous comme des threads d'exécution coopératifs, mais il n'y a pas de pile qui ait besoin d'être sauvée. Son ordonnancement est assuré par un ordonnanceur (*scheduler*), exécuté par la classe de base *Agent* et caché au programmeur, selon une politique de round-robin de non-préemption [Bellifemine et al 04].

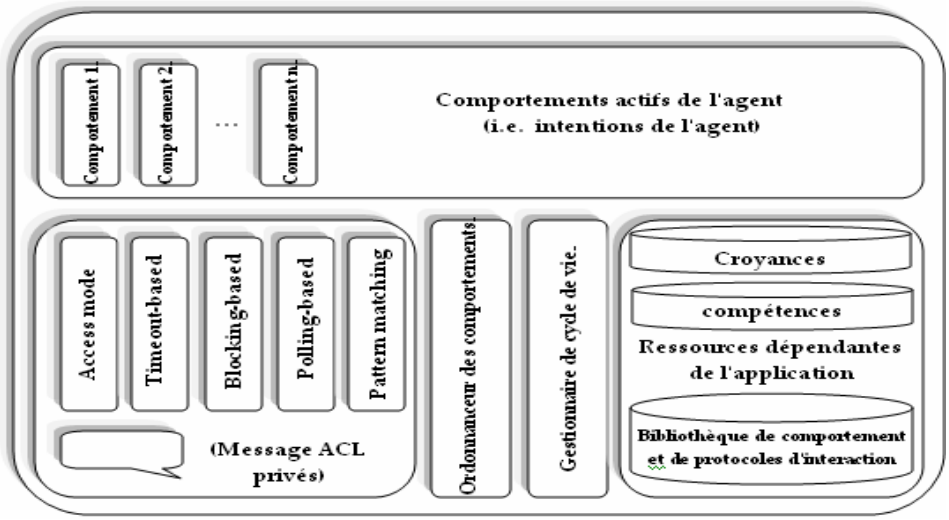


Figure 2.7. Modèle d'agent dans *JADE.*

Le développeur d'agents doit étendre la classe abstraite *Agent* et implémenter les tâches spécifiques de l'agent par une ou plusieurs classes *Behaviour*, les instancier et les ajouter à l'agent. La super-classe *Agent* permet à l'agent d'hériter un comportement fondamental caché qui traite toutes les tâches liées à la plateforme *(l'enregistrement, la configuration, la gestion à distance, etc.)*, un ensemble de méthodes qui peuvent être appelées pour implémenter les tâches spécifiques à l'agent *(envoi des messages, utilisation des protocoles d'interaction standard, enregistrement sur plusieurs domaines, etc.)* ainsi que deux méthodes pour gérer la file de comportements d'agents (*addBehaviour* et *removeBehaviour*).

JADE inclut quelques comportements prêts à être utilisés pour les tâches les plus communes dans la programmation des agents, tels que l'envoi et la réception des messages et la décomposition des tâches complexes en des agrégations de tâches plus simples [Bellifemine et al 04]. Entre autres, *JADE* offre aussi une classe *JessBehaviour* qui permet l'intégration avec le moteur d'inférence *JESS*[1] *(Java Expert System Shell)* qui est écrit entièrement en *Java* et, à l'origine, inspiré du système expert *CLIPS*[2]. Ainsi, *JADE* fournit le noyau de l'agent et garantit la conformité avec les normes *FIPA*, alors que *JESS* est le moteur d'inférence de l'agent qui exécute le raisonnement nécessaire pour la résolution du problème en utilisant une base de connaissance interne.

4. Conclusion

L'objectif de ce chapitre était de faire un état de l'art sur la technologie des environnements orienté agents. Nous avons présenté l'initiative de la *FIPA* pour la normalisation des *SMA* et des plateformes multi-agents, en particulier. Un certain nombre de plateformes existants ont été décrites avant de présenter davantage l'environnement *JADE*. Les spécifications *FIPA* représentent un effort énorme et un grand pas vers la standardisation de la technologie agent. Ce succès peut être justifié par l'accroissance du nombre des plateformes répondants à cette norme ainsi que par le nombre des applications basées sur ces plateformes.

L'interopérabilité des systèmes est une raison puissante pour lesquelles les développeurs préfèrent employer une plateforme agent standard, pour mettre en oeuvre leurs solutions d'*IAD*. *JADE* est l'une des plateformes multi-agents conformes à la norme *FIPA*. Sa neutralité vis-à-vis la définition d'un agent représente, à la fois, un atout et une difficulté (aucune méthodologie de développement). Pour ce faire, le choix de cette plateforme pour mettre en oeuvre notre solution nous amène, dans le chapitre5, à la définition des éléments suivants : un modèle de négociation, un modèle d'agent et un modèle d'interaction.

[1] JESS: http://herzberg.ca.sandia.gov/jess/
[2] CLIPS: http://www.ghg.net/clips/CLIPS.html

Chapitre 3 :
Modélisation et simulation multi-agents participative

1. Introduction

Lorsque l'on souhaite passer du problème identifié à la simulation, cela suppose de franchir un certain nombre d'étapes : l'*analyse,* la *modélisation,* l'*implémentation informatique* et la *simulation.* Soulignons toutefois que ces étapes ne se parcourent généralement pas de façon séquentielle mais plutôt itérative et incrémentale. Pour [Ricordel et Demazeau 00], celles-ci sont au nombre de quatre[1].

Dans notre approche, nous nous intéresserons plus particulièrement à la phase de simulation. Cette dernière est utilisée largement pour étudier des phénomènes physiques et sociaux et pour la prédiction de futures conditions de tels phénomènes, aussi elle est utilisée pour l'apprentissage de certains phénomènes complexes. Cependant, parvenir à modéliser les comportements sociaux d'acteurs dans des situations dans lesquelles ils utilisent une expérience fortement liée à un domaine spécifique reste toujours une activité difficile. Une solution possible à ce problème d'extraction de connaissances situées réside dans les méthodologies participatives de conception (la proposition de *conception participative de comportements d'agents* [Drogoul et al 03]), [Nguyen-Duc 05], [Guyot 06], où les acteurs sont appelés à mener un rôle actif dans la définition des comportements des agents de la simulation par l'intermédiaire de jeux de rôles. Donc la simulation participative consiste à faire impliquer les utilisateurs dans la simulation, durant l'exécution du modèle. Autrement dit, un utilisateur aura le contrôle d'un composant du modèle simulé et il peut le manipuler à son gré durant l'exécution, en modifiant certains paramètres du composant. La problématique de la simulation multi-agents participative dérive de celle des simulations multi-agents et de celle de la modélisation des pratiques collectives. Pour l'héritage multi-agents, il s'agit de concevoir des agents logiciels capables de reproduire un modèle dans une simulation. L'aspect participatif est le mélange d'agents logiciels et d'acteurs humains au sein de simulations afin d'étudier et de modéliser des phénomènes sociaux ou collectifs.

2. La Modélisation Multi-agents

2.1. La Modélisation

2.1.1. Principes généraux

Un modèle est une image simplifiée de la réalité qui nous sert à comprendre le fonctionnement d'un système, et qui est constitué d'une part de la description de la structure du système, qui introduit les spécifications sémantiques intégrées, et d'autre part de la description des fonctionnements réguliers ou non et des dynamiques qui modifient cette structure au cours du temps.

La description de la structure du système peut aller d'un ensemble de variables quantitatives (numériques) ou qualitatives (à valeurs discrètes) dans le cadre de ce qu'on appelle les systèmes dynamiques, jusqu'à un ensemble d'entités munies de propriétés et de relations entre elles dans le

[1] **1 L'analyse :** Elle correspond au processus de découverte, à la description du type de problème et à son domaine d'application. **2 La modélisation :** Elle correspond au processus de définition d'une représentation structurée alors que les problèmes sont souvent à la base non structurés. Cette définition peut être donnée sous un mode déclaratif ou s'appuyer sur des langages spécifiques (UML, MERISE, réseaux de Petri…). **3 L'implémentation :** informatique. Elle correspond au processus de construction d'une solution fonctionnelle du problème. En pratique, cela consiste à coder la solution avec des langages particuliers de programmation. **4 La simulation :** Elle correspond à la phase d'utilisation du modèle implémenté pour des problèmes concrets.

cadre de ce qu'on appelle les systèmes d'information (spatialisés ou non) et, en simulation, les systèmes individus-centrés et multi-agents. *La description de la dynamique* peut aller de l'utilisation d'équations différentielles (dans le cas continu) ou d'équations aux différences (dans le cas discret) dans le domaine des systèmes dynamiques, à n'importe quelle combinaison de ces dernières et de modèles informatiques dans le domaine des systèmes individus centrés et multi-agents. Ces descriptions permettent de simuler les évolutions possibles du système que l'on décrit.

Afin de structurer davantage les différents types de modèles que nous venons d'évoquer, il faut encore distinguer entre *les modèles descriptifs*[1] et *les modèles explicatifs*[2] d'une part et *les modèles basés sur des mesures*[3] ou *sur des entités*[4] d'autre part.

Les *SMA* ont vocation d'être des *modèles explicatifs basés sur les entités.* Pour comprendre leur couverture existentiel il est important de distinguer divers types de systèmes à décrire selon qu'ils sont munis :

- ***D'une simplicité organisée :*** il est facile d'exhiber leur comportement à l'aide de quelques lois sur un ensemble limité de variables, par exemple les systèmes physiques simples pour lesquels des jeux limités d'équations différentielles suffisent;
- ***D'une complexité désorganisée :*** muni d'un grand nombre d'entités en interaction avec des comportements uniformes, par exemple, certaines dynamiques des populations dans lesquelles les différences entre individus ont peu d'importance et pour lesquels des approches par automates cellulaires ou utilisant la mécanique statistique sont possibles;
- ***D'une complexité organisée :*** muni d'un nombre moyen d'entités hétérogènes produisant des comportements localement structurés tels que les *éco-systèmes* et les *sociétés humaines* observées à l'échelle d'entreprises, réseaux sociaux, …etc.

Les *SMA* sont naturellement adaptés à l'étude de cette dernière catégorie de systèmes dès lors que l'on cherche des modèles explicatifs, donc à comprendre les mécanismes sous-jacents aux phénomènes globaux observés, y compris les phénomènes d'auto-organisation et de reconfiguration dans des systèmes ouverts.

2.1.2. Modélisation des systèmes

Afin d'appréhender et de réduire la complexité du système réel, la modélisation est une étape essentielle pour l'étude des systèmes complexes et dynamiques. La modélisation consiste à construire une représentation simplifiée d'un système appelée généralement un modèle. Une définition d'un modèle énoncée par l'*AFCET,* est la suivante : *Un modèle est un schéma, i.e. une description mentale*

[1] Ont pour vocation de saisir un invariant observé d'un système, par exemple l'évolution d'une population. Il n'est alors pas nécessaire de savoir pourquoi il y a cette évolution mais seulement comment.

[2] ont pour vocation de rendre compte d'un mécanisme dont le résultat peut être entre autres, une évolution déterminée si tant est que les conditions initiales et les valeurs des paramètres du modèle produise effectivement un tel invariant. On citera, par exemple, les comportements de reproduction pour l'évolution d'une population.

[3] Dans ces modèles, l'état s'exprime comme un ensemble de variables représentant des grandeurs mesurables du système que l'on modélise (position, nombre d'individus dans une population).

[4] Dans ces modèles, l'état s'exprime comme un ensemble structuré, éventuellement variable, d'objets dont non seulement les états mais aussi les relations et le nombre peuvent changer au cours du temps.

(intériorisée), ou figurée (diagrammes, formules mathématiques, etc. ...) qui, pour un champ de questions est pris comme représentation abstraite d'une classe de phénomènes, plus ou moins habilement dégagés de leur contexte par un observateur pour servir de support à l'investigation, et/ou la communication".

Un modèle est donc une représentation d'un système (réel ou imaginaire) dont le but est d'expliquer et de prédire certains aspects du comportement de ce système. Cette représentation est plus ou moins fidèle car d'une part le modèle devra être assez complet afin de pouvoir répondre aux diverses questions qu'on peut se poser sur le système qu'il représente et d'autre part, il ne doit pas être trop complexe pour pouvoir être facilement manipule. Ceci implique immédiatement qu'il y a intérêt a bien définir les limites ou frontières du modèle qui est censé représenter le système.

Les principaux avantages de manipuler un modèle plutôt que le système qu'il modélise est que ce modèle évite la construction d'un système qui n'existe pas, et il permet aussi d'éviter de faire des expérimentation directes sur un système existant (*problèmes de sécurité, ou économiques, ou impossible : système solaire).*

2.1.3. Processus de modélisation

Le processus de construction d'un modèle de simulation peut être schématise comme suit :

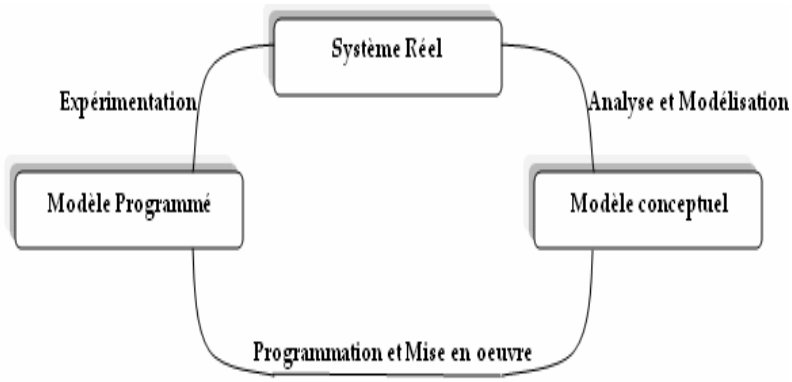


Figure 3.1. Processus de modélisation.

Un autre schéma distinguant entre *modèle conceptuel* et *modèle programmé* sur ordinateur est le suivant :

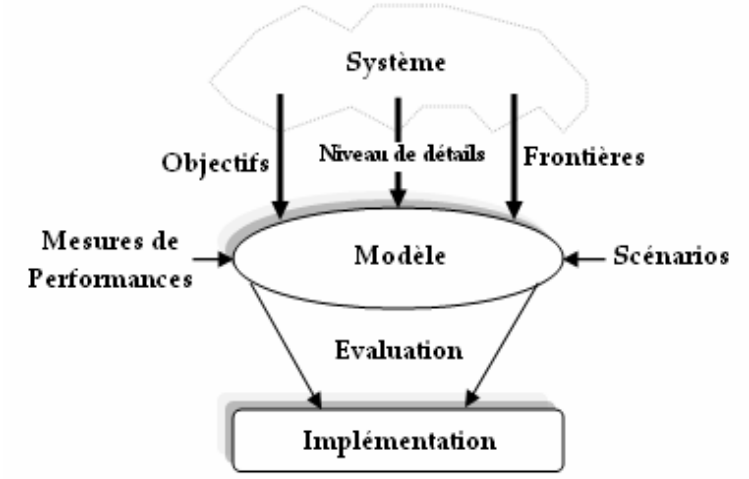


Figure 3.2. Frontières du système et modélisation.

Le *modèle conceptuel* est une représentation *(mathématiques logique, verbale, etc. ...)* du système réel. Il est obtenu dans une phase d'analyse et de modélisation. *Le modèle programmé* est la mise en oeuvre du *modèle conceptuel* sur un calculateur. Il est obtenu dans une phase de programmation et de mise en oeuvre. Les enseignements sur le système réel sont obtenus a la suite d'expérimentations sur *le modèle programmé* dans une phase d'expérimentation.

2.1.3.1. Construction d'un modèle

La construction d'un modèle consiste, a partir d'une représentation mentale du système liée à une certaine connaissance acquise par le modélisateur, a développer un modèle sous forme de programme d'ordinateur. Ainsi, c'est le modélisateur (guidé par les méthodes d'analyse et les outils conceptuels) qui définit les frontières du modèle et donc des variables internes, d'entrées et de sorties nécessaires, il se base sur une appréciation du niveau de détails a incorporé dans le modèle, et le choix des techniques de modélisation. Deux méthodes complémentaires sont couramment employées pour la construction d'un modèle :

- ***L'affinage progressif (descendante):*** Appelée aussi méthode de modélisation par accumulation de degrés de complexité. Elle procède par itérations vers des niveaux de complexité croissante. Ce processus évolutif permet de prendre en compte les insuffisances du modèle à une étape donnée, de les améliorer a l'étape suivante, tout en évitant d'entrer dans un niveau de détails superflu.
- ***L'accumulation de sous modèles (ascendante):*** Utilisée lorsque le système a étudier est de taille importante. Les sous modèles sont issus d'un découpage du système en sous-systèmes distincts. La complexité du système est alors mieux maîtrisée mais l'étape d'intégration conduite a des problèmes d'interfaces souvent délicats.

La modélisation procède a l'aide de deux démarches. La démarche directe consiste à établir les variables décrivant le système, les lois et propriétés caractérisent le système ainsi que les équations régissant le système directement (par expérience et intuition) de l'univers physique. La démarche indirecte consiste à établir ces éléments a partir du traitement de données tirées de mesures (cas déterministes) ou de sondage (cas aléatoire)

2.1.3.2. Classification des modèles

S'agissant de la modélisation, [Bousquet 94] distingue cinq familles de modèles :

1. les modèles statiques et les modèles dynamiques,
2. les modèles prédictifs ou empiriques qui résument un ensemble de relations et les modèles explicatifs qui s'appuient sur la compréhension de ces relations,
3. les modèles déterministes et les modèles stochastiques qui sont fondés principalement sur la notion de hasard dans le tirage de certaines valeurs de paramètres,
4. les modèles à résolution analytique et les modèles nécessitant des simulations,
5. les modèles dynamiques qui évoluent dans le temps.

Donc de nombreux critères peuvent être employés pour classifier les modèles. En simulation, on s'attache généralement à distinguer les types de modèles suivants :

2.2. La Modélisation Multi-agents

La modélisation multi-agents permet de conceptualiser et de simuler un ensemble organisé d'agents en interaction entre eux et avec leur environnement. Ces agents peuvent être dotés de capacités plus ou moins développées, allant des agents réactifs constitutifs de l'intelligence collective aux agents cognitifs dotés de formes de rationalité plus sophistiquées.

Dans *une démarche de modélisation* avec des univers multi-agents. On distingue plusieurs étapes [Bousquet et al 01]:

- La première étape permet de construire un monde artificiel, c'est celle de l'acquisition de connaissances sur le domaine d'étude. Il s'agit d'identifier les différents acteurs, les différentes perceptions et d'utiliser les univers multi-agents pour une modélisation. Face à un monde très complexe les univers multi-agents permettent de rechercher la simplification la plus acceptable en plaçant les questions sur les problèmes de représentations, interactions et contrôles.
- La deuxième étape est une étape de restitution que l'on pourrait aussi appeler validation du modèle cognitif. Il s'agit de tester le modèle proposé pour le processus de prise de décision. C'est la mise à plat des représentations et des processus d'interactions entre les agents.
- Une troisième phase est celle de la simulation. La simulation montre comment la dynamique du système est issue des interactions entre des acteurs qui ont des poids et des représentations différentes. On peut ici distinguer deux sous-phases. Dans un premier temps la simulation peut être effectuée sous forme de jeu de rôle, ce qui permet aux acteurs de valider le fait que c'est bien dans les interactions entre représentations différentes que se trouve le moteur de la dynamique du système. Cette première sous-phase permet aussi de faire émerger les différents scénarios intéressants à tester. Puis, cette phase étant acquise on peut utiliser le modèle multi-agents pour faire des simulations sous différents scénarios.

2.2.1. Intérêts de l'approche agent

Les *SMA* sont adaptés pour la modélisation et la simulation des systèmes distribués et dynamiques surtout les systèmes complexes. Les propriétés qui caractérisent des agents sont adaptées pour la représentation et l'étude comportementale des entités qui constituent un tel système. Un intérêt de la modélisation orientée agents[1] est qu'elle permet en général une simulation et la représentation de la dynamique du système considéré, qu'il soit *physique*, *biologique*, *économique* ou encore *social*, peut s'appuyer sur différentes approches de modélisation.

2.2.2. Modélisation orientée agents

La modélisation orientée agents, ou *Agent Based Modeling*, s'intéresse à la représentation des comportements des entités du système et de leurs interactions. L'utilisation des *SMA* pour modéliser les systèmes dynamiques complexes conduit à des modèles plus réalistes que ceux obtenus par des

[1] L'approche de modélisation orientée agents permet de représenter le système à travers l'identification et la spécification du comportement des individus en interactions qui le composent. Ces individus peuvent évoluer et agir sans contrôle ni intervention extérieure *(autonomie)*, peuvent percevoir leurs environnements et répondre à ces modifications *(réactivité)*, initier des comportements dirigés par des buts internes *(pro-activité)*, et interagir avec d'autres individus *(habilité sociale)*.

approches de modélisation plus conventionnelles [Parunak 96]. L'approche agent offre la possibilité de concevoir plus facilement des modèles proches des systèmes réelles étudiés. En effet, l'association *agent/entité* permet de représenter de façon plus réaliste une série d'entités et de processus décisionnels. Ce mode de représentation permet d'appréhender le système comme une organisation d'entités en interactions, au sein de laquelle la connaissance et la prise de décisions sont décentralisées.

3. La Simulation Multi-agents

3.1. La Simulation

3.1.1. Principes généraux

Pour [Drogoul 93] *la simulation est une démarche scientifique qui consiste à réaliser une reproduction artificielle, appelée modèle, d'un phénomène réel que l'on désire étudier, à observer le comportement de cette reproduction lorsqu'on en fait varier certains paramètres, et à en induire ce qui se passerait dans la réalité sous l'influence de variations analogues. La démarche de simulation passe donc par trois étapes distinctes: l'étape de modélisation, qui consiste à construire le modèle du phénomène à étudier, l'étape d'expérimentation, qui consiste à soumettre ce modèle à un certain type de variations, et l'étape de validation, qui consiste à confronter les données expérimentales obtenues avec le modèle à la réalité.*

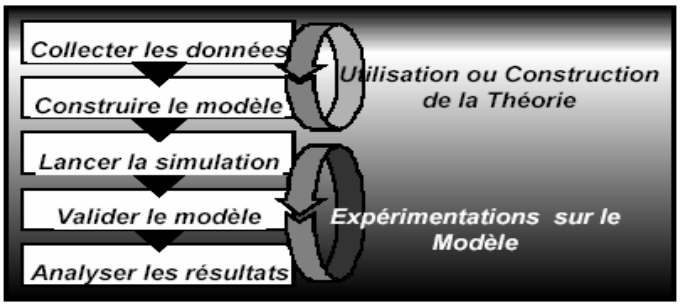


Figure 3.3. Les étapes du processus de simulation[Drogoul 93].

La construction d'un modèle se fonde toujours sur une *théorie* [Drogoul 93], c'est à dire, en simulation classique, une description abstraite de certains aspects du phénomène modélisé en termes de concepts ou de variables, et de relations ou de lois. Construire une nouvelle méthode de simulation consiste donc à utiliser, voire inventer, une théorie de modélisation et c'est cette partie-là que nous développerons au cours de ce chapitre pour montrer l'apport des systèmes de modélisation multi-agents.

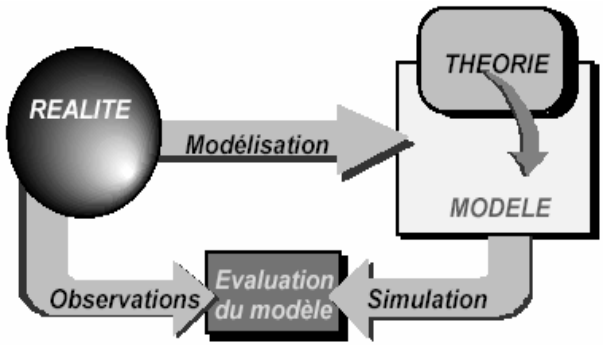


Figure 3.4. Intervention de la théorie dans la construction d'une simulation [Drogoul 93].

On a recours aux techniques de simulation essentiellement dans deux types de situation. Le premier type se définit par l'impossibilité de recourir à l'expérimentation directe, Le second type de situation est celui où l'on ne dispose pas de bases théoriques solides sur un phénomène donné et où l'on cherche à élaborer une théorie, par voie de simulation.

3.1.2. Définitions et classification

Pour Quéau, "*la simulation consiste à mettre en oeuvre des modèles dans des conditions variées, pour tenter d'explorer leurs possibilités, leurs défaillances, et éventuellement découvrir telle ou telle trajectoire comportementale encore inconnue* ". C'est "*un outil expérimental s'attachant à l'exploration non pas du réel mais des modèles qu'on s'en forme*", qui "*permet de modifier à petits risques, de jouer le stratège en chambre* " [Quéau 86], selon [Bratley et al 87] " *la simulation signifie faire évoluer le modèle d'un système avec les entrées appropriées et observer les sorties correspondantes* ". De son côté, [Anderson 89] définit la simulation *comme une manipulation numérique d'un modèle symbolique conçu pour représenter l'évolution d'un système dans le temps. Ainsi qu'il le souligne par ailleurs, les processus mis en oeuvre sont souvent de nature complexe et aléatoire, et les problèmes traités par cette approche sont typiquement peu structurés. Ces définitions nous paraissent bien préciser les contours d'une telle technique, la référence unique à la manipulation numérique est toutefois à réenvisager. Si celle-ci est de mise traditionnellement en RO, l'Intelligence Artificielle a permis d'élargir cette « manipulation » à des concepts non numériques tels que la connaissance,*

De là on peut dire que la simulation est une technique qui tend à reproduire le comportement d'un système complexe qui évolue avec et dans le temps ou qui évolue de manière aléatoire. Opérant à partir d'un modèle, la simulation peut être définie par la manipulation dans le temps et parfois dans l'espace des éléments du modèle afin d'en observer l'évolution, les interactions internes et externes. La simulation informatique est le processus de conception du modèle d'un système réel et la conduite d'expériences au travers de ce modèle à l'aide d'un ordinateur pour un objectif précis d'expérimentation. Lors de l'exécution, la simulation est conditionnée par l'utilisation de paramètres et de données à traiter et qui en constituent le point d'entrée, et ce, afin de produire des résultats de sortie. La simulation s'inscrit naturellement auprès des décisionnaires comme un outil d'aide à la décision.

On peut distinguer deux grandes classes de simulation : *la simulation des systèmes continus* et *la simulation des systèmes à évènements discrets:*

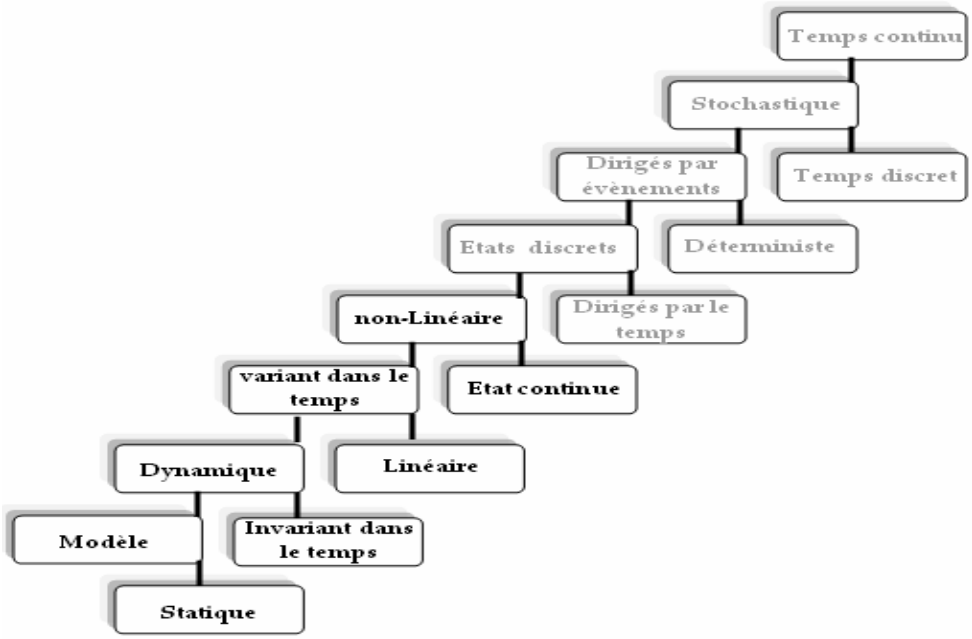


Figure 3.5. Classification des systèmes de simulation [Cassandras et Lafortune 99].

- ***La simulation des systèmes continus*** dans ce type de simulation, un cycle est effectué à chaque incrémentation d'une unité de temps. Au cours de ce cycle, seules les actions concernées par cette date seront exécutées. L'ensemble des actions potentielles peut alors être vide. La simulation continue reproduit le comportement de systèmes dynamiques dans lesquels les variables d'états évoluent sans interruption dans le temps. La simulation continue est une technique qui en théorie ne peut être exécutée que par une machine analogique. Ce type de simulation est particulièrement adapté aux systèmes évoluant de façon systématique et régulière. Il en est ainsi de systèmes biologiques et environnementaux [Barreteau 98] [Drogoul 92] [Feuillette 01] et de gestion de flux [Zeng et Sycara 99]. La détermination du meilleur pas de temps possible n'est pas toujours évidente. Ceci pose le problème de couplage et prendre le pas de temps le plus fin n'est pas forcément la solution la plus pertinente. Par contre, pour des systèmes évoluant par sauts, la simulation à évènements discrets peut s'avérer plus intéressante.
- ***La simulation des systèmes à évènements discrets*** La simulation à événements discrets *(Discrète Event Simulation : DES)* reproduit le comportement de systèmes dynamiques dans lesquels les variables d'états changent à des instants précis dans le temps. Le comportement du système, dans cette approche de simulation, est ainsi décrit par une suite de changements d'état où les événements dont on simule l'évolution sont représentés sous la forme d'ensembles modélisés de façon discrète à des unités de temps régulières ou irrégulières dictées par la nature du système. Dans ce type de simulation le temps utilisé pour dater les événements produits évolue continûment, toutefois l'état du système ne change que par sauts [Cheikhrouhou 02].

3.1.3. Limites des simulations classiques

Les limites de ces simulations numériques (classiques) sont de plusieurs ordres :

- les modèles mathématiques sont des modèles équationnels à grand nombre de variables, différents des modèles utilisés en biologie, sociologie, etc., ce qui ne permet pas de tester les modèles et théories manipulées dans le domaine,
- ces modèles rendent difficile le passage du niveau micro *(individuel)* au niveau macro *(collectif),*
- ces modèles ne s'intéressent pas à la représentation des comportements individuels, mais uniquement à leurs résultats globaux,
- ces modèles ne permettent pas d'expliquer l'émergence de structures spatio-temporelles.

A l'inverse, la simulation multi-agents s'intéresse explicitement aux comportements individuels des entités qui composent un système et vise à reproduire ces comportements à l'intérieur d'un monde artificiel. Ce monde artificiel d'agents en interaction est caractérisé par trois composantes: *les agents, les règles de comportement,* et *l'environnement.*

3.2. La Simulation Multi-Agents

3.2.1. Principes Généraux

Un des domaines d'application des *SMA* est la simulation [Ferber 95]. Contrairement aux simulations classiques, la réalité n'est pas modélisée sous forme de relations mathématiques. On s'attachera par la suite plus particulièrement à la simulation multi-agent. Elle propose de créer un monde artificiel dans lequel interagissent des agents évoluant dans un environnement. Les *SMA* apportent une solution nouvelle au concept même de modèle et de simulation dans *les sciences de l'environnement* ([Barreteau 98] [Bousquet et Le Page 01]…) et *les sciences socioéconomiques* ([Axelrod 97] [Conte 97] [Terna 98]).

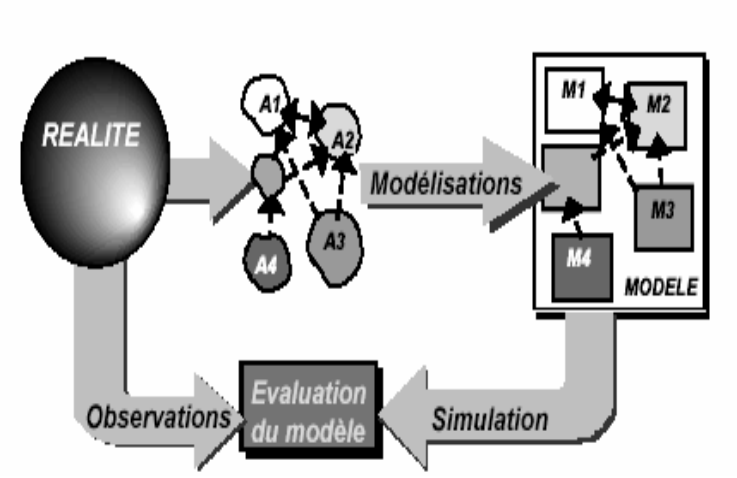


Figure 3.6. Principe de la simulation multi-agents[1][Drogoul 93].

[1] Le phénomène réel est décomposé en un ensemble d'éléments qui agissent ou interagissent. Chacun de ces éléments est modélisé par un agent, et le modèle général est la résultante des interactions entre ces agents.

La conception d'une nouvelle méthodologie de simulation se base sur la définition d'une théorie de modélisation. La modélisation d'un phénomène dans une perspective multi-agents se traduit par [Drogoul 93]:

• ***Une décomposition du phénomène:*** en un ensemble d'éléments discrets autonomes dont les interactions reproduisent le phénomène. Il est à noter que ce préliminaire nécessite une vision déjà distribuée du phénomène à modéliser.

• ***La modélisation de chacun de ces éléments par un agent:*** pour définir les connaissances de l'agent, ses capacités fonctionnelles, ses comportements et les modes d'interaction qu'il adoptera à l'encontre des autres agents.

• ***La définition de l'espace:*** dans lequel évoluent ces agents et des lois qui le gouvernent. On appellera cet espace l'environnement des agents. Sa définition permettant d'affiner la description des actions possibles des agents, ainsi que celle de leurs moyens de communication.

• ***La définition des objets inertes****:* Seront considérés comme objets inertes les agents du système qui ne sont dotés d'aucune capacité d'action ni de communication.

D'après Drogoul [Drogoul 00] les simulations Multi-Agents peuvent être utilisées pour (Figure 3.7):

Tester des hypothèses*: sur* l'émergence de structures sociales selon le comportement et l'interaction de chaque individu.

Construire des théories*:* qui peuvent contribuer au développement général et à la compréhension de l'éthologie, de la sociologie, et de systèmes psycho-sociologiques en représentant le comportement organisationnel et structurel d'un agent.

Intégrer différentes théories*:* provenant de disciplines variées telles que *la sociologie, l'éthologie, l'ethnologie….*

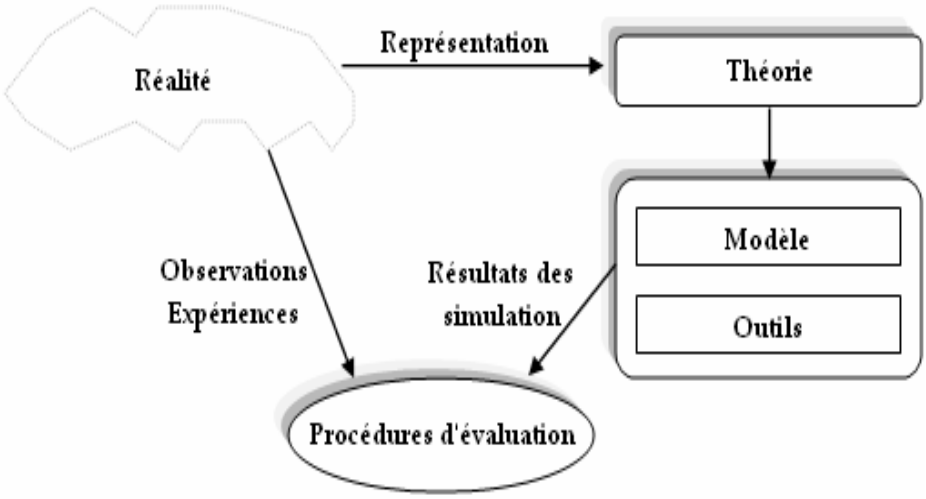


Figure3.7. Processus de simulation [Drogoul 92].

La simulation est donc une branche très active de l'informatique. Elle consiste à analyser les propriétés de modèles théoriques du monde réel. L'usage de la simulation est très fréquent dans des disciplines diverses telles que les sciences sociales pour essayer d'expliquer et de prévoir les phénomènes naturels. Les *SMA* apportent une solution en offrant la possibilité de représenter les individus, leurs comportements et leurs interactions. La simulation multi-agents est fondée sur l'idée qu'il est possible de représenter sous forme informatique des comportements individuels des entités

dont les interactions font apparaître des phénomènes nouveaux. Une caractéristique intéressante des *SMA* généralement conçus à partir *des systèmes à bases de connaissances* réside dans leur capacité à reproduire, et, dans une certaine mesure, à expliquer les lignes de raisonnement suivies pour résoudre un problème.

3.2.2. Simulation par systèmes multi-agents

Les propriétés des *SMA* favorisent la conduite de simulations basées sur des entités autonomes afin d'appréhender le fonctionnement complexe du système réel modélisé. Ce type de simulation s'intéresse à la description et à l'étude comportementale du système réel par l'exécution des agents en interactions dans un contexte dynamique. Les simulations orientées agents sont constitués d'un ensemble d'entités autonomes, dont les comportements dépendent de leurs interactions et de l'environnement au sein duquel ils sont situés. Ces modèles représentent les actions individuelles des agents, les interactions entre les agents ainsi que les conséquences de ces interactions sur l'environnement.

Les simulations par systèmes multi-agents peuvent s'envisager comme *un outil d'aide à la décision, à la gestion des flux, à l'analyse des processus décisionnels,* et *à l'évaluation des performances.* Ces performances peuvent se formaliser en termes d'objectifs divers qui sont directement liés à des facteurs qualitatifs et/ou quantitatif. L'évaluation des performances peut s'obtenir à travers la conduite de simulations selon différents scénarii contextuels. La conduite de simulations et l'observation du comportement des entités qui les composent mettent ainsi en avant des applications orientées sur l'exploitation du paradigme agent.

3.2.3. La Simulation Multi-agents en Sociologie

La simulation sociale est appliquée dans les domaines qui s'intéressent aux groupes sociaux ou aux comportements sociaux des êtres humains. Par exemple, elle est utilisée pour expliquer la croissance d'une économie au niveau *macro-économique* [Hammond et Sun 03], ou encore pour la simulation des modèles économique [Rejeb 05]. Les études en *géographie humaine,* par exemple celle sur les mobilités urbaines, exigent également des simulations fines [Vanbergue 00]. Dans les simulations réalisées en *psychologie cognitive* et *sociale* [Castelfranchi et Conte 92] ou en *sociologie* appliquée à *l'anthropologie* [Doran et al 90], [Doran et al 92]. Les autres travaux existants consistent essentiellement en *simulations prédictives* conçues comme des aides à la décision [Kuhn et Muller 93]. Le domaine qui apparaît cependant comme le plus fécond à l'heure actuelle est celui des agents réactifs, qui prend le contre-pied de cette approche en tentant de modéliser et de simuler des comportements collectifs de haut niveau grâce à des interactions d'agents les plus simples possibles. Le nombre de travaux publiés commence à être relativement important, et particulièrement en *sociologie,* (simulation des mécanismes d'influence sociale chez [Nowak et Latane 92], [Lonborg 92]). En gestion des ressources renouvelables, point actuellement critique dans les pays en voie de développement, peut être étudiée au moyen de simulations [Barreteau et Bousquet 01]. La recherche militaire et celle sur la gestion du trafic aérien [Nguyen-Dun 05] utilisent massivement des simulations parce que les expérimentations réelles dans ces deux domaines sont très coûteuses et risquées.

4. La Simulation Multi-agents Participative

4.1. Principes Généraux

Les simulations multi-agents participatives *"Ce sont des expériences menées en laboratoires ou à travers le réseau Internet, avec des participants humains et qui s'inscrivent dans une démarche multi-agents*" [Guyot 06]. Ces simulations sont caractérisées par deux propriétés :

- les agents dans ces simulations participent à la simulation comme les humains et ne sont pas des entités qui fournissent des services.
- les participants (humains) accèdent à la simulation exactement comme le font ou le feraient des agents. En d'autres termes, chaque participant est assis à un poste de travail, et toutes les interactions, conçues comme des interactions entre agents, se font par le biais de l'ordinateur.

Les simulations participatives[1] donnent un statut particulier à l'homme en l'intégrant dans la boucle, il peut interagir avec elle pendant son déroulement. Une interaction simple consiste à modifier les valeurs des variables manipulées par le modèle pour modifier de façon sensible le comportement observé. La simulation aide alors l'homme à comprendre, à tester et à apprendre. Il peut ensuite proposer des modifications pour améliorer le modèle sur lequel repose la simulation. Il devient alors un cré-acteur (créateur-acteur). La mise en place d'une simulation participative implique l'usage de modèles permettant, l'ouverture[2], la compréhension[3], et la substitution[4] [Pierre 06]. La méthodologie des simulations multi-agent participatives est inspirée à partir des simulations multi-agent et des sciences sociales expérimentales (à partir d'expériences sociales). C'est un processus itératif avec plusieurs boucles (cf. Figure 3.8).

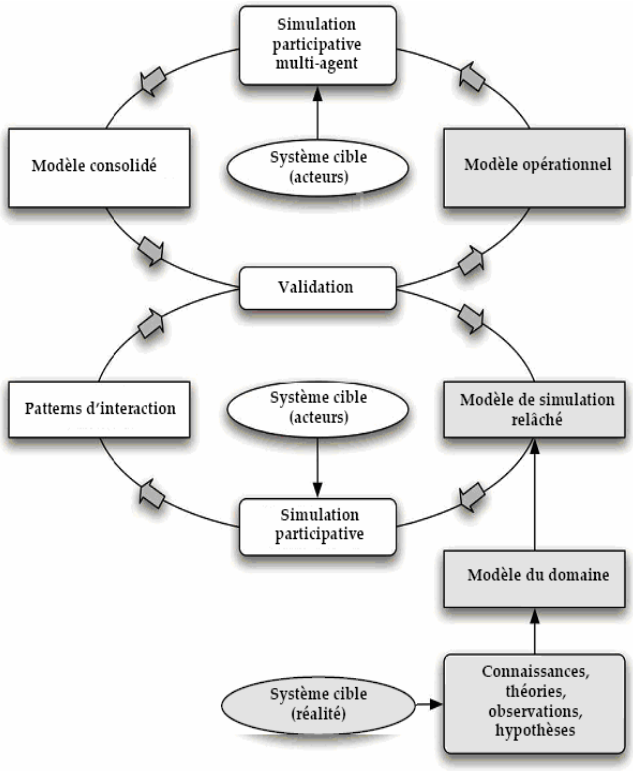


Figure 3.8. La méthodologie de la simulation multi agents participative (d'après [Guyot et Drogoul 04]).

[1] Les simulations multi agents participatives peuvent être décrites comme simulations où les agents et les joueurs ont les rôles semblables. Cependant, il y a quelques différences principales avec les jeux de rôle (RPG Role playing game).

[2] Un modèle ouvert accepte et s'adapte à des modifications "imprévues".

[3] Il faut que les modèles présentent des informations, des comportements et des interactions qui puissent être le support d'une prise de conscience par l'utilisateur. De lui permettre de vivre la simulation.

[4] Elle consiste à pouvoir substituer l'homme au modèle et réciproquement. Le modèle peut ainsi faire à la place de l'homme pour simuler ses comportements et l'homme se mettre à la place du modèle pour lui montrer un comportement ou tester la réaction des autres modèles.

Le processus commence par une connaissance initiale *(théories)* et un modèle initial. Le modèle est alors détendu pour être jouable et modifié afin d'accomplir le but de l'expérience. Par exemple, pour valider quelques hypothèses sur le comportement des acteurs, les actions que les acteurs peuvent exécuter, (c'est ce qu'on appelle l'environnement), doivent être décomposés en petits primitifs. Les règles du jeu, sont implémentées dans le jeu de rôle.

La première boucle consiste a lancer une simulation participative qui est très semblable aux jeux de rôle dans la terminologie de *MAS/RPG.* La simulation participative est négociée par des ordinateurs et les expériences peuvent être notées par des agents de notation situés sur le réseau et observer le jeu. Ces notations peuvent alors être traitées sous forme de modèles d'interaction. Les résultats de la simulation sont employés pour valider les hypothèses originales.

La deuxième boucle consiste à construire des agents proactifs à partir d'interaction du model qui ont été extraits pendant la première boucle et pour lancer des simulations multi agent participatives, où les agents proactifs et les acteurs jouent ensemble. Les agents peuvent apprendre des connaissances en visualisant le comportement des acteurs humains et le résultat peut augmenter la compréhension du modèle.

Figure 3.9

Figure 3.10

Figure 3.11

Figure 3.9 et **Figure 3.10.** Les joueurs dans une simulation multi-agents participative dans un *LAN.*
Figure 3.11. Les joueurs dans une simulation multi-agents participative on utilisant les *PDA.*

4.2. Que ce que c'est la participation?

Il y a trois significations très différentes de la participation : ***Interaction :*** les Joueur peuvent changer les paramètres pendant la simulation. ***Conception participative :*** les joueurs participent pour améliorer le processus de conception **. *Les jeux de rôle :*** dépositaires ou les joueur sont les acteurs principaux de la simulation avec les scénarios prédéfinis.

La simulation participative donne un statut particulier à l'homme en l'intégrant dans la boucle : il peut interagir avec elle pendant son déroulement. Une interaction simple consiste à modifier les valeurs des variables manipulées par le modèle pour modifier de façon sensible le comportement observé. La simulation aide alors l'homme à comprendre, à tester et à apprendre. Il peut ensuite proposer des modifications pour améliorer le modèle sur lequel repose la simulation.

La conception participative est l'un des buts principaux des jeux de rôle et la motivation de peu d'expériences des sciences économiques expérimentales. De même, des expériences multi-agent

participatives peuvent être employées pour la conception participative. Ce but peut être réalisé réellement de deux manières différentes:

La première méthode consiste à organisation les simulations pour tester de nouveaux protocoles. Les avantages de telles simulations participatives avec les agents assistants[1] sont doubles, ils rassemblent la rétroaction immédiate des utilisateurs humains et on peut placer des scénarios où les agents suggèrent des comportements basés sur le nouveau protocole à un joueur humain ou ils peuvent jouer des simulations sans n'importe quel utilisateur humain. L'enregistrement des expériences après ces scénarios peut alors être employés pour faire un compte rendu.

Des simulations Multi-agent participatives peuvent également être employées pour la conception participative des systèmes multi-agent. Comme dans la méthodologie de *MAS/RPG (Multi-Agent Systems/Role-Playing Games),* on peut extraire la connaissance à partir des dépositaires pour établir de meilleures simulations multi-agent. En outre, les expériences participatives peuvent bénéficier de faire participer des humains dans un procédé d'émergence et elles peuvent être employées comme manière de concevoir les solutions socialement inspirées aux problèmes

4.3. Les domaines d'applications de la simulation multi-agents participative

4.3.1. La réalité virtuelle

De nombreuses méthodes de conception des *SMA* ont déjà été proposées, mais aucune ne s'est encore vraiment imposée comme c'est le cas en conception et programmation par objet avec la méthode *UML.* Nous retiendrons ici l'approche Voyelles qui analyse les *SMA* selon quatre points de vue : *Agents, Environnements, Interactions,* et *Organisations* (les voyelles *A,E,I,O*). En réalité virtuelle, nous rajoutons le point de vue *Utilisateur* (le voyelle *U*) pour prendre en compte la participation active de l'opérateur humain a la simulation (l'homme est dans la boucle). Ainsi, les simulations multi agents de la réalité virtuelle deviendront participatives. l'utilisateur à se substituer à un agent en prenant le contrôle de son module de décision. Au cours de cette substitution, l'agent peut éventuellement passer en mode d'apprentissage, par imitation ou par l'exemple. A tout moment, l'utilisateur peut rendre le contrôle à l'agent auquel il s'était substitué. Ce principe de substitution entre agents et utilisateurs peut être évalue par une sorte de test de Turing, un utilisateur interagit avec une entité sans deviner s'il s'agit d'un agent ou d'un autre utilisateur, et les agents réagissent à ses actions comme s'il s'agissait d'un autre agent.

L'approche Voyelle ainsi étendue *(AEIOU)* implique pleinement l'utilisateur dans la simulation multi agents, rejoignant ainsi l'approche de la conception participative *(participory design)*. Une telle simulation multi-agents participative en réalité virtuelle met en oeuvre des modèles de types différents *(multi modèles)* issus de domaines d'expertise différents *(multi-disciplines).* Elle est souvent complexe car son comportement global dépend autant du comportement des modèles eux mêmes que

[1] Les agents d'assistance, ou comme on peut aussi les appeler les agents fantôme (ghosts-agents), ces agents sont utilisés pour prendre le contrôle des agents contrôlés par les utilisateurs quand ces derniers sont absents. Mais les l'utilisateurs ne doivent pas tenir compte de l'existence de ces agents. Pour cela ces agents doivent utiliser un haut niveau d'intelligence afin de suivre le comportement des utilisateurs pour bien continuer la simulation. Ils utilisent aussi leur intelligence pour acquérir d'autres connaissances à partir de la simulation et les interactions des agents qu'ils contrôlent.

des interactions entre modèles. Enfin, elle doit inclure le libre arbitre de l'utilisateur humain qui exploite les modèles en ligne.

4.3.2. Réalité partagée.

Il s'agit de mettre en œuvre et de valider une plate-forme de conception participative autour d'une *table magique.* La méthode de conception concernée est de type *scénario,* c'est-à-dire que les utilisateurs et les concepteurs sont invités, ensemble, à construire des objets ou des systèmes soit à l'aide du *papier-crayon* soit par *assemblage,* à l'aide d'éléments ou d'artéfacts matériels existants. *La table magique* doit permettre de manipuler des éléments réels et virtuels. Les acteurs reçoivent des aides vocales ou des instructions tout en travaillant. Ainsi l'univers de travail se présente sous la forme d'un réseau d'agents, artificiels et humains en interaction semi dirigée : tantôt ce sont les objets et leurs relations spatio-temporelle qui construisent l'interaction, tantôt ce sont les acteurs humains à travers les scénarios qui leur sont demandés d'exécuter. Le travail de recherche consistera à dégager les problèmes fondamentaux que pose la multi modalité d'action dans ce contexte, puis à concevoir le système d'interaction et à le réaliser avec des briques logicielles existantes. Les problèmes se situent au niveau de la reconnaissance des objets, de leur localisation, de la coordination des tâches par scénario, de la traçabilité des actions sur les objets.

Figure 3.12. Réalité partager.

4.3.3. La gestion de ressources renouvelables

La gestion de ressources renouvelables nécessite de s'intéresser aux interactions entre des dynamiques naturelles et des processus sociaux objets d'étude complexe, ce travail implique de prendre en compte l'existence de multiples points dans un processus itératif de compréhension et d'analyse. En appui à cette démarche, la modélisation multi-agents peut apporter une aide appréciable lorsqu'elle s'inscrit dans une démarche participative. Parce qu'un modèle multi-agents décrit le comportement des acteurs, il peut être facilement réfuté ou discuté par ceux-ci et peut alors devenir un objet de médiation facilitant la mise en place d'une régulation des usages collectifs [Pierre 05].

4.3.4. L'Émergence

L'émergence est au coeur de l'approche multi-agents : des phénomènes complexes émergent de comportements simples d'agents proactifs et autonomes, en interactions et situés dans un environnement commun. Pourtant, la notion d'émergence recouvre deux objectifs et deux usages distincts des *SMA.* D'une part, il s'agit d'observer l'évolution de phénomènes et en particulier l'émergence de propriétés au sein de simulations multi-agents. D'autre part, les systèmes multi-agents peuvent être conçus pour résoudre de manière distribuée des problèmes. Ce qui émerge est alors une solution distribuée du problème [Guyot 05]. L'usage d'une démarche participative relève toujours

d'un objectif spécifique. Les différentes expériences peuvent être classées en fonction de cet objectif et la typologie ainsi formée correspond à un rôle particulier de l'émergence. Par ailleurs, la validation des approches participatives peut être analysée comme cohérence du modèle avec les phénomènes émergents.

Les différentes expériences, y compris les expériences de modélisation d'accompagnement peuvent être classées en trois catégories en fonction de l'usage qui est fait de la participation. Le Tableau 3.1 résume cette typologie:

Type	Objectif de la participation	Participants	Émergence	Exemple
Sociologique	Concevoir, valider et consolider des modèles	Acteurs du domaine	Testée(comportements collectifs plus ou moins modélisés)	*SimCafé, PAT*
Pédagogique	Faire comprendre le lien entre individuel et collectif	Étudiants	Prévue (phénomènes collectifs induits)	*ButorStar*
Négociation	Favoriser les négociations entre acteur	Acteurs du domaine (rôles inversés)	Recherchée (évolution des dynamiques d'interactions)	*Sylvopast*

Tableau 3.1. Typologie des approches participatives.

L'approche sociologique, consiste à concevoir, valider ou à consolider des modèles de comportements sociaux. Les participants sont les acteurs du domaine [Guyot 06]. L'approche pédagogique consiste à faire comprendre aux participants le lien entre les comportements individuels et les comportements collectifs. Les phénomènes collectifs émergents sont prévus par les concepteurs du modèle. Enfin, l'approche de négociation vise à aider les acteurs du domaine à dégager des évolutions par la négociation d'une situation bloquée. Les acteurs disposent de beaucoup plus de liberté que dans l'approche pédagogique. Ils doivent expliciter leur comportement à travers la participation aux simulations et expliquer leurs choix aux autres acteurs pendant les séances de débriefing. Ces approches sont en réalité combinées dans les différentes expériences participatives.

4.3.5. Validation

Dans le cas des simulations multi-agents, la validation consiste à relier l'émergence de phénomènes complexes au modèle initial, à vérifier la cohérence entre les phénomènes émergents et le modèle. Dans le cas d'une approche participative, on a également deux types de validations possibles. Pour [Bousquet et al 01], la validation réside d'une part dans les phénomènes qualitatifs[1] qu'on observe à la fois dans l'expérience participative et dans la réalité, et d'autre part, dans les discussions avec les acteurs[2] qui valident le modèle et proposent des amendements.

[1] L'observation de phénomènes qualitatifs similaires correspond à la validation externe des simulations multi-agents. Elle est parfois effectuée, dans une approche couplant jeu de rôles et système multi-agents, en retranscrivant les hypothèses des joueurs dans un système multi-agents et en comparant l'évolution de ce système avec ce qui est observé dans la réalité.

[2] Les discussions avec les acteurs se situent au-delà de ces simulations et sont ce que Bousquet appelle une validation sociale. Il s'agit d'explorer des scénarios en commun avec les acteurs.

4.3.6. La gestion des systèmes complexes

La complexité des éco-sociosystèmes fait ressortir un objet d'étude à caractère transdisciplinaire, à savoir les systèmes complexes et leur compréhension. Dès lors les sciences humaines et sociales doivent être convoquées sur deux plans, en tant que contributeurs de connaissances sur ces systèmes complexes eux-mêmes et en tant que contributeurs de connaissances sur les processus de co-construction d'une compréhension de ces systèmes. L'initiative Mimosa s'inscrit dans ce cadre pour proposer un objet médiateur : une plateforme générique de modélisation et de simulation participative afin de faire dialoguer les disciplines et les acteurs [Jacque 04].

4.4. Quelques exemples de l'approche participative

- ***SimCafé*** *LANIA - Xalapa, Veracruz, mai 2003,* Les expériences SimCafé ont été conçues pour étudier la formation de coalition sur le marché du café.
- ***SimBar I & II*** *LIP6 - Paris, juin 2004,* Les expériences SimBar sont fondées sur le modèle du Bar El Farol imaginé par Brian Arthur. Certains joueurs ont participé via le réseau Internet.
- ***La gestion du trafic aérien,*** consisté à concevoir et développer un simulateur multi-agent pour aider à la conception de nouvelles procédures de collaboration entre acteurs de la gestion du trafic aérien [Nguyen-Duc et al 04]. Les fonctionnalités nécessaires à un environnement de conception participative sont implémentées sur ce simulateur et la validation expérimentale de certaines propriétés du simulateur a été effectuée avec les experts du domaine.
- ***SimCommod I*** *NII - Tokyo, septembre 2005,* SimCommod est un modèle élaboré dans le cadre du projet Ecole Commod et fondé sur la gestion collective de ressources renouvelables.
- ***SimCommod II*** *Université Chulalongkorn - Bangkok, novembre 2005,* La seconde série des expériences SimCommod a été l'occasion d'introduire des agents assistants dotés d'un mécanisme d'apprentissage
- ***HubNET*** est un projet basé sur les systèmes dynamiques complexes, qui est réalisé dans un réseau local *(LAN),* pour donner aux étudients la possibilité de participer aux expériences.

5. Conclusion

Les simulations multi-agents participatives, comme type particulier d'expériences de simulation, se situent dans la tradition des méthodes participatives en sociologie. Les premières expériences de ce type avaient déjà un objectif de modélisation de phénomènes dynamiques. L'appropriation de cet outil pour l'aide au développement ancre définitivement les simulations multi-agents participatives dans le domaine des simulations multi-agents en faisant participer les experts du domaine aux expériences. La mise en place de l'outil de simulation a comme objectif à plus long terme d'élaborer des outils et une méthode de validation et de consolidation de modèles de pratiques collectives à l'aide de simulations multi-agents participatives. Finalement on peut dire que l'approche de simulation participative peut être appliquée facilement à d'autres phénomènes sociaux ou biologiques. Un tel travail trouve son utilité dans beaucoup de domaines comme *l'enseignement / l'apprentissage, la prise de décision* en prévoyant les mesures nécessaires face à certains comportements prévisibles ainsi que dans *la recherche* pour mieux comprendre certains phénomènes, valider certaines théories ou en conclure de nouvelles.

Chapitre 4
La négociation dans les Systèmes Multi-agents et le e-commerce

1. Introduction

Dans les systèmes multi-agents, les agents sont des entités de base intelligentes qui interagissent entre eux dans un environnement commun. Dans cet environnement, les agents ont certains buts et doivent représenter le plus fidèlement possible leurs mandants. Et comme les agents sont dotés de qualités d'autonomie, de proactivité et de perception, ils interagissent entre eux en s'appuyant sur une base de compétences et de connaissances individuelles. Lorsque plusieurs agents interagissent, des conflits peuvent survenir, ce qui nécessite l'utilisation de mécanismes de résolution de conflits. Parmi ces mécanismes, on trouve notamment la coordination, et la négociation. Nous nous proposons d'étudier dans ce chapitre la négociation entre agents qui est abordée par différentes disciplines : l'économie, les sciences des organisations, les sciences de l'aide à la décision, l'Intelligence Artificielle Distribuée *(IAD)…* la négociation est utilisée dans le domaine du commerce électronique[1] notamment en utilisant les enchères, dans le domaine des télécommunications par exemple pour partager une bande passante et dans les systèmes multi-agents pour l'allocation de tâches et de ressources [Durfee et Lesser 89], [Sandholm 93], l'identification de conflits [Gamble et Sen 94], la résolution des disparités entre les buts [Sycara 88] et la détermination de la structure organisationnelle.

2. Le commerce électronique

2.1. Généralités

L'utilisation de moyens électroniques pour des transactions commerciales et des échanges d'informations est un mouvement amorcé dans les années 60, essentiellement par les banques. Ce mouvement s'est développé dès les années 70 avec les standards *EDI (Electronic Data Interchange*), autorisant la transmission de commandes, de factures et d'ordres de livraison sur des réseaux de communication souvent privés. Dès les années 80, ces réseaux transportent également des codes permettant le travail collaboratif d'ingénieurs et de techniciens géographiquement éloignés. Cependant, les coûts liés à l'*EDI* sont prohibitifs pour les petites entreprises. Les bases du commerce électronique reposent sur les fondations suivantes: le développement d'un medium de transmission de données universel *(Internet),* la définition d'un langage de structuration hypertexte de l'information *(HTML),* la possibilité de créer des sites d'information répartis *(la notion de Web),* ainsi que la disponibilité de logiciels d'exploration et d'interprétation de ces sites *(les browsers).* A ces bases viennent s'ajouter les outils accompagnant la transaction marchande, qui peut être découpée en trois phases:

- la phase d'information, lors de laquelle des renseignements sur les produits, les prix, la disponibilité, la livraison, etc. sont échangés;
- la phase d'accord, qui correspond à l'aboutissement des négociations et se conclut par un contrat qui doit être légalement valable;
- la phase d'exécution, qui voit l'échange du produit ou service acheté contre une transaction financière, puis un éventuel suivi, sous forme de service après-vente, de programme de fidélisation, etc.

[1] Les qualités d'autonomie, de proactivité, de perception et de négociation, …etc., rendent des agents particulièrement utiles pour l'environnement du commerce électronique qui est riche en termes d'information et de processus.

En regardant de plus près les capacités de commerce électronique, on constate qu'il offre des outils pour toutes les fonctions d'une entreprise, il devient de plus en plus une réalité. Or, à chaque fois qu'un marchand ou un client souhaite d'échanger des produits via l'Internet, il épuise plusieurs heures pour trouver une bonne solution car l'information est décentralisée et elles varient indépendamment entre eux. D'autres part, le commerce électronique est quasiment fait par nos interactions: c'est nous qui décide quand nous allons acheter des produits ou combien nous souhaitons de payer pour chaque produit, etc. Donc, la négociation joue un rôle important pendant les échanges des produits ou des services. Par exemple, dans une entreprise, le directeur lui-même doit négocier par téléphone ou par e-mail, même s'il s'agit d'un commerce électronique. Dans cet exemple-là, c'est fortement possible de ne pas négocier avec tous les fournisseurs, et le gain de l'entreprise n'est donc pas optimal.

En résumé, l'utilisateur délègue la tâche de rechercher le meilleur prix d'un produit à un agent "acheteur" chargé de négocier avec différents agents "vendeurs". Le processus de négociation a pour objectif de converger vers des solutions optimisant les gains des différents intervenants. Donc il nous faut un système qui cherche les produits auxquelles nous nous intéressons (une classification selon certains critères), et qui négocie avec les différents marchands d'une manière automatique. Aujourd'hui, seules les applications agents correspondent à ce besoin.

2.2. Définitions

Le commerce électronique *(e-commerce)* peut être vu comme une forme particulière de vente à distance. Il recouvre l'ensemble des transactions sur un support électronique, tout simplement le *e-commerce* signifie les échanges qui se passent via l'Internet. D'où, les acheteurs et les vendeurs deviennent des entités électroniques [Alessio et al 01]. Plusieurs définitions ont été attribuées au concept de commerce électronique qui a fait l'objet de plusieurs études. Nous avons retenu les définitions suivantes:

« *Le commerce électronique est un système d'information interorganisationnel qui permet à deux agents (acheteur et vendeur) d'échanger des informations relatives au prix et aux caractéristiques du produit* » [Bakos 97].

« *Le commerce électronique est un système conçu pour fournir une information, livrer un produit ou un service, s'acquitter d'un paiement en utilisant des supports tels que le téléphone, les réseaux d'ordinateurs et autres*» [Kalakota et Whinston 97].

« *Le commerce électronique désigne l'ensemble des transactions marchandes effectuées sur un réseau électronique ouvert par l'intermédiaire d'ordinateurs ou d'autres terminaux interactifs* » [Lorentz 98].

« *Le commerce électronique est une forme de vente assimilable à la VPC (La vente par correspondance) "en ligne" et définie comme tout achat ou vente automatique, conclu sur un terminal interactif de réseau électronique*» [CNIS 01].

2.3. Catégories de commerce électronique

2.3.1. B2B (Business to Business)

Le *BtoB* ou *B2B (Business to Business)* recouvre le champ du commerce interentreprises [Macarez 01],[Tavoillot 00],[Muller 01], c'est à dire les activités dans lesquelles les clients ou prospects sont

des entreprises. Les types de sites les plus représentatifs de la notion de *BtoB*, sont les places de marché virtuel *(PMV)* sur le Web [David 00], [Cohen 01].

2.3.2. B2C (Business to Consumer)

Le *BtoC (Business to Consumer)* fait référence au commerce entre une entreprise et une personne privée [Macarez 01].

2.3.3. C2B (Consumer to Business)

Le *CtoB (Consumer to Business)* fait référence aux relations commerciales qui se déroulent entre un client et une entreprise. Expression désignant les sites qui font circuler l'information des consommateurs aux producteurs/distributeurs *(business).* Ce sont par exemple les sites d'achat groupé (les consommateurs qui proposent des prix aux producteurs). Le cas le plus fréquent est le regroupement de consommateurs pour acheter en gros à moindre prix, via un site d'achat groupé [Macarez 01], [Tavoillot 00].

2.3.4. C2C (Consumer to Consumer)

Le *CtoC (Consumer to Consumer)* s'applique aux échanges commerciaux[1] entre des personnes privées dans lesquels un site spécialisé joue le rôle d'intermédiaire [Macarez 01], [Brousseau 00]. Le modèle le plus courant est celui de la vente aux enchères [Tavoillot 00], [Misse 00], popularisé par le site américain *eBay* [Tavoillot 00].

2.3.5. B2G (Business to Government)

Le *BtoG (Business to Government)* s'applique aux relations touchant les transactions électroniques entre une entreprise et une administration gouvernementale.

2.3.6. G2C (Government to Consumer)

Le *GtoC (Government to Consumer)* s'applique aux relations touchant les transactions électroniques entre une personne privée et une administration gouvernementale, phénomène qui devrait se généraliser avec la mise en ligne de formulaires administratifs ou la possibilité de payer ses impôts par Internet [Lorentz 98].

2.3.7. E2E (Employe to Employe)

Le *EtoE (Employe to Employe)* s'applique aux relations touchant les échanges électroniques entre au moins deux employés d'une même organisation (ex : entre les employées de deux filiales du même groupe) ou d'organisations différentes au travers d'un Intranet ou d'Internet.

2.4. Utilisation des agents pour le commerce électronique

En ce qui concerne le e-commerce actuel sur Internet, on peut citer des agents utilisés dans le commerce *Customer-to-Buisiness, les salles d'enchères* et *les marchés virtuels.* Une description des systèmes de commerce avec des agents négociateurs est donnée dans [Sierra 99]. Nous allons maintenant détailler ces trois grandes familles d'agents :

[1] Les affaires qui se déroulent entre les particuliers ou bien les clients. « Expression désignant les sites constituant des points de rencontre entre les internautes ("chat club", forums, sites de vente aux enchères...) ».

2.4.1. Business-to-Customer

Dans cette catégorie, on trouve des agents comme *BargainFinder* et *BargainBot* d'Anderson *Consulting* ou *Firefly* qui sont spécialisés dans la recherche de produits pour les clients. Typiquement, on leur donne une description des articles recherchés (titres, auteurs et autres caractéristiques...) et ils travaillent sur Internet comme des moteurs de recherche. Comme ils sont, la plupart du temps multi-tâches, ils établissent plusieurs connexions simultanément. Le problème est qu'ils renvoient trop de réponses et manquent encore d'exactitude. Les recherches actuelles visent à améliorer les agents acheteurs pour qu'ils prennent en compte davantage de critères et qu'ils puissent inférer les besoins et les préférences des clients.

2.4.2. Places de marché

Parmi les autres applications du commerce électronique, *les marchés virtuels* se développent de façon importante. Il s'agit de points de réunion entre agents négociateurs. *Kasbah* [Chavez et Maes 96], au *MIT Média Lab,* est un bon exemple de marché virtuel flexible. Les vendeurs proposent des produits aux acheteurs qui contrôlent si la description correspond à leurs buts. S'il y a appariement, ils commencent à négocier les prix en utilisant des règles données par leurs propriétaires respectifs. Des systèmes d'intermédiaires [Collins et al 98] et des agents de médiation [Kuokka et Harada 98] ont également été développés pour amorcer ou arbitrer le processus de négociation. Ces intermédiaires reçoivent les requêtes d'un agent demandeur au moyen de descriptions de produits et des offres, de la part des fournisseurs. Après avoir trouvé la meilleure correspondance, l'agent médiateur peut se contenter de lancer le processus de communication, rester en tant qu'arbitre ou encore servir comme intermédiaire de confiance entre les acteurs.

2.4.3. Salles d'enchères

Les salles d'enchères sont des marchés évolués où les rôles des agents sont plus complexes et plus structurés. Les agents sont gérés comme dans une vraie salle des ventes et l'utilisation de protocoles spécifiques permet de garantir l'équité entre les acheteurs d'un même produit. On peut citer *AuctionBot* de l'Université du *Michigan* [Wurman et al 98] qui est un agent offrant un service centralisé d'enchères sur Internet accessible à la fois aux hommes et aux autres agents.

2.5. Catégories d'agents pour le commerce électronique

Généralement on distingue deux catégories d'agents pour le commerce électronique : *les agents acheteurs* et *les agents vendeurs.* Nous détaillons ci-après leur fonctionnement respectif. Internet permet aux acheteurs de réduire le coût et le temps de recherche de l'information et d'améliorer leur niveau d'information. L'information dont ils peuvent disposer sur Internet est souvent plus précise, elle est en temps réel, elle est diversifiée. Ainsi, ils peuvent identifier facilement les produits qui correspondent le mieux à leurs besoins en particulier grâce à l'utilisation des moteurs de recherche, d'agents intelligents capables de faire des comparaisons de prix, etc. De même, pour les vendeurs, la recherche de l'information sur Internet est moins coûteuse.

2.5.1. Les agents acheteurs

Ils sont contrôlés par les clients et ont pour but de faciliter le processus d'achat. En effet, comme pour tout autre recherche, identifier et vérifier l'intérêt d'une offre commerciale est extrêmement difficile sur le réseau des réseaux. Les outils classiques (moteurs et répertoires de recherche) se montrent vite inefficaces pour trouver, valider et confronter une offre commerciale.

Les agents acheteurs sont capables de se connecter sur divers services de vente à distance et ramener les informations de description et de prix de tous les articles d'un type déterminé, pour en proposer la liste comparative, voire passer automatiquement la commande. Contrairement aux agents de recherche d'informations sur le Web, les agents acheteurs ne travaillent pas à partir de mots-clefs mais à partir de noms de produits ou de marques. Ils renseignent l'utilisateur sur la disponibilité d'un produit en menant une recherche par marque ou par catégorie (produit, et accessoires), aussi ils identifient l'utilisateur sur les distributeurs (localisation d'un distributeur précis, liste intégrale ou sélective de distributeurs (en fonction des services qu'ils offrent: garantie, facilité de paiement...)).

Nous présentons ci-après quelques exemples d'agents acheteurs. *Bargain Finder*[1] est probablement le plus connu. A partir du titre d'un album de musique et du nom de l'interprète, *Bargain Finder* se propose de consulter différents disquaires en ligne et de comparer les prix. *Bargainbot Search*[2] *Agent* est spécialisé dans l'édition. A partir d'un titre de livre ou du nom de l'auteur, il adresse simultanément des recherches à des libraires et présente des brèves fiches descriptives sur les références trouvées. *The Movie Critic*[3] demande à ses utilisateurs de donner leur avis sur 12 films qu'ils ont vus afin d'agglomérer les réponses de chacun. Grâce à cette base de données et un système avancé de comparaison, *The Movie Critic* conseille les cinéphiles sur des films qu'ils n'ont pas encore vus, en se basant sur les réponses de personnes qui ont apparemment les mêmes goûts. *Firefly*[4] travaille de manière similaire à *The Movie Critic* avec en plus des conseils sur les concerts.

2.5.2. Les agents vendeurs

Un agent vendeur ayant un produit à commercialiser va traverser le réseau à la recherche des clients intéressés par ce produit. Lorsque l'agent vendeur rencontre un agent client intéressé par ce type de produits, une transaction est alors négociée entre les deux agents. Les différentes fonctions assurées par les agents vendeurs sont :

- Enregistrement du profil et des préférences de l'acheteur,
- Enregistrement des demandes successives de l'acheteur afin d'enrichir, d'affiner, de faire évoluer son profil,
- calculer des recommandations sur l'évolution de l'offre commerciale grâce à des statistiques sur la demande globale des consommateurs.

Voici quelques exemples d'agents vendeurs. *The BroadVision*[5], *SelectCast*[6], *AgentWare*[7].

[1] **Bargain Finder** : http://www.freebiezone.com/bargain.finder.html

[2] **Bargainbot Search Agent** : http://www.ece.curtin.edu.au/~saounb/bargainbot

[3] **The Movie Critic** : http://www.moviecritic.com

[4] **Firefly** : http://www.techweb.com

[5] **BroadVision** : http://orc.sageoln.com

[6] **SelectCast** : http://www.aptex.com

[7] **AgentWare** : http://www.Zdnet

3. La Négociation

3.1. Définitions

Smith [Smith 88] a définit la négociation comme suit: *Par négociation nous entendons une discussion dans laquelle les différentes parties intéressées échangent des informations et arrivent à un accord. Les trois temps de la négociation se décomposent en (a) un échange bilatéral de l'information, (b) chaque partie dans le processus de négociation évalue les informations en fonction de son point de vue, (c) l'accord final est obtenu par consentement mutuel.* Dans cette définition, la négociation implique deux éléments essentiels, d'une part la communication *(a)* et d'autre part la prise de décision *(b) (c).*

Selon [Durfee et Lesser 89], *la négociation consiste à améliorer les accords sur des points de vue ou des plans d'action grâce à l'échange structuré d'informations pertinentes.*

Pour Wooldridge [Wooldridge et al 98] *la négociation est un composant central pour la résolution de problèmes coopératifs.*

La négociation pour [Bedou 97] *est un processus de communication itératif durant lequel il y a attaque et défense de certains points de vues, de solutions, de croyances, de connaissances, de propositions d'alternatives, de façon à atteindre un des accords, et ce par révision des croyances suite à l'obtention de nouvelles informations, par argumentation, par explication, et où l'agent exprime clairement l'état de ses croyances et connaissances pour un sujet donné.*

Dans [Kraus et Schwartz 97], *La négociation est vue, dans l'Intelligence Artificielle Distribuée (IAD) comme un moyen donné aux agents pour communiquer et faire des compromis, afin d'arriver à un état d'accord mutuel bénéfique.*

[Mathieu et Verrons 05] ont donné la définition suivante : *il y a négociation lorsqu'il y a une discussion, des propositions entre les protagonistes et lorsque l'accord final satisfait au mieux tous les participants.*

La négociation dans les SMA constitue une étape importante dans les processus de coordination et de collaboration, elle vise à établir des accords entre agents, chaque agent du système possède une stratégie de prise de décision qu'il utilise pour maximiser son utilité locale et converger vers des solutions globales acceptables par tous les participants à la négociation. Le but de cette étape est d'assurer un équilibre entre les négociateurs et que les gains soient répartis équitablement [El Fallah-Seghrouni 01].

[Beam et Segev 97] ont défini la négociation dans le domaine de commerce électronique comme suit: *La négociation dans le commerce électronique est le processus par lequel deux ou plus de partis négocient multilatéralement des ressources pour gain projeté mutuel, en utilisant les outils et techniques de commerce électronique.*

Les différentes définitions partagent un point commun qui est l'absence de l'accord sur un problème donné (tous les chercheurs s'accordent sur la finalité de la négociation, à savoir l'aboutissement à un accord commun satisfaisant). La négociation peut donc être vue comme une boîte noire ayant en entrée *un conflit* et en sortie *un accord,* dans le meilleur des cas. La recherche sur la négociation consiste donc à étudier les mécanismes de cette boîte noire, pour la rendre transparente [Mathieu et Verrons 05]. En *IAD,* le terme de négociation est employé aussi bien dans le cadre de résolution de conflits [Lander et Lesser 93] [Sycara 88] que dans celui de la coordination entre agents [Davis et Smith 80] [Sycara 92]. Donc généralement, on négocie pour résoudre un

désaccords, prévenir un conflit, obtenir un bien, en somme pour satisfaire un besoin, et ce à travers un processus d'échange impliquant au moins une autre personne.

3.2. Les deux catégories de la négociation

La négociation se décompose en deux catégories principales : *la négociation compétitive* et *la négociation coopérative.* La négociation compétitive intervient entre des agents compétitifs, qui essaient de maximiser leur utilité locale. Par contre, lors d'une négociation coopérative, les agents essaient d'atteindre l'utilité globale maximale, qui prend en compte la valeur de toutes leurs activités. Cette forme de négociation qui est très différente de la négociation compétitive, peut être vue comme un processus de recherche distribué. La négociation est donc peut aboutir à un échec ou à un accord. Dans ce dernier cas, une négociation qui se déroule en mode coopératif conduit généralement à un accord dans lequel les deux parties s'estiment gagnantes *(gagnant-gagnant).* En revanche, si la négociation se déroule en mode compétitif ou distributif, l'accord risque d'être *gagnant-perdant* et instable, voire *perdant-perdant.* Autrement dit, la qualité d'un accord dépend autant, voire plus, de ses conséquences à terme sur les relations des protagonistes que des gains obtenus.

3.3. Composantes de la négociation

De tous les travaux réalisés[1] dans la négociation, il ressort un certain nombre de points communs sur lesquels il faut aujourd'hui s'appuyer. Trois éléments prédominent: un protocole de négociation, des ressources à négocier et un modèle de raisonnement. [Beer et al 98] ont identifié trois composantes qui font l'objet de la recherche en négociation :

Les protocoles de négociation définissent l'ensemble des règles qui gouvernent l'interaction. Le protocole concerne (I) le type de participant possible (les négociateurs et les candidats possibles), (II) les états de la négociation (par exemple les propositions acceptables, la fin de la négociation), (III) les évènements qui entraînent une transition d'état, (IV) les actions valides que les négociateurs peuvent engager pour chaque état particulier.

Les objets de la négociation un objet abstrait qui comprend les attributs qu'on veut négocier; dans certains cas il s'agit de négocier uniquement le prix (le cas des enchères), mais dans d'autres cas il faut aussi négocier plusieurs attributs comme le temps nécessaire pour satisfaire une commande, la qualité des produits, etc. donc les objets de la négociation sont les ressources à négocier, ceci recouvre tout ce qui va être négocié.

Raisonnement, modèle de prise de décision des agents fournit le dispositif de prise de décision dont chaque participant dispose pour atteindre ses objectifs (ce modèle fournit un algorithme de décision aux agents). La partie la plus importante de la prise des décisions dans ce cas est la *stratégie de négociation* qui permet de déterminer quelle primitive de négociation l'agent doit choisir à un certain moment. Le processus de décision revient à répondre à la question: "Que dois-je faire maintenant?", par exemple liciter enchérir, abandonner, etc. Pour prendre une décision adéquate, un agent doit être capable de faire un raisonnement stratégique, notamment raisonner en tenant

[1] La négociation dans les systèmes multi-agents a déjà fait l'objet de très nombreux travaux. Il y a ceux qui ont été basés sur la négociation du point de vue économique. Ils ont ensuite inspiré beaucoup de travaux introduisant la négociation dans l'intersection entre le monde des systèmes multi-agents et celui de commerce électronique. Mais il y a beaucoup d'autre travaux formels réalisés, afin d'essayer de fournir des modèles de négociation.

compte de ce que font/décident les autres agents et, s'il parvient à le savoir ou à le supposer, quel est le modèle de décision des autres agents.

Müller [Müller 96] distingue trois éléments fondamentaux dans la négociation (cf. figure 4.1.) :

1. La négociation qui s'appuie sur des langages comportant des primitives de communication pour la négociation, leurs sémantiques, et leurs usages dans les protocoles. Les primitives concernent les envois et réceptions de messages et intègrent des actes illocutoires. La structure de l'objet de la négociation fait appel à un langage de description de l'objet. Le protocole spécifie les séquences possibles d'actions et les conditions suivant lesquelles une requête peut être effectuée.

2. Les processus de négociation qui peuvent être, soit des procédures pré-établies, soit des comportements. Le processus concerne la proposition de solutions, l'analyse et la révision des solutions préférables.

3. La décision individuelle qui peut être prise en fonction de critères d'utilité, de stratégie, de systèmes de préférences, d'une fonction de comparaison et de mise en correspondance (matching). L'observation des différentes interactions humaines et l'analyse théorique ont permis de mettre en évidence dans les processus de prise de décision d'un acteur, une dimension individuelle et collective et la mobilisation conjointe de l'une et de l'autre en relation avec l'apprentissage [Zeng et Sycara 97].

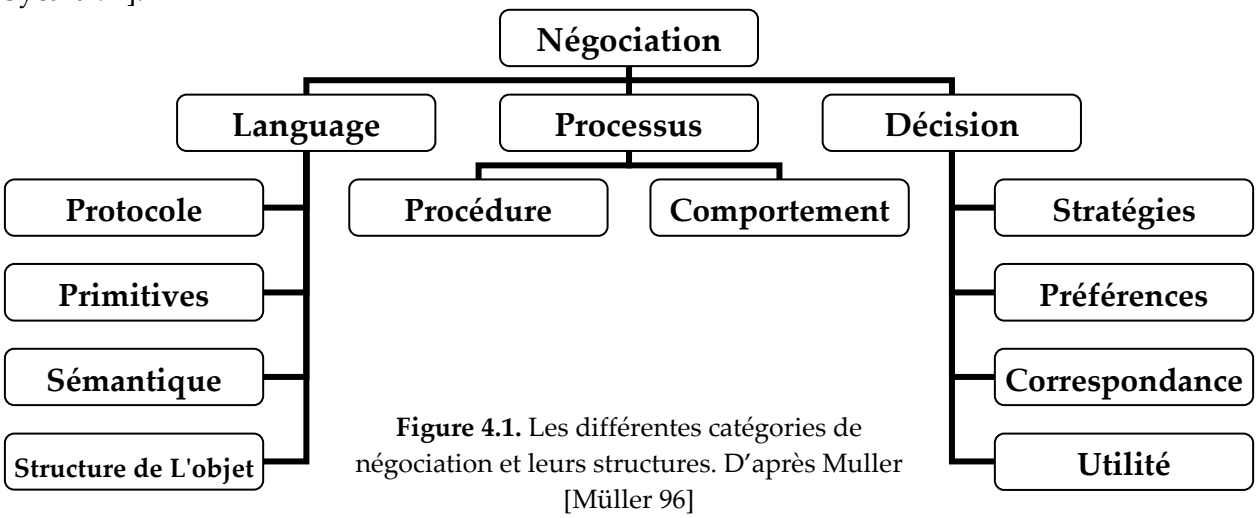


Figure 4.1. Les différentes catégories de négociation et leurs structures. D'après Muller [Müller 96]

3.4. Langages de négociation

Deux langages majeurs de communication ont été mis au point: *KQML (Knowledge Query and Manipulation Language, 1993*) et *ACL (Agent Communication Language, 1999). KQML* a été conçut pour le partage de savoirs issus de bases de connaissances, et consiste à émettre ou recevoir un acte de discours. *ACL* est quant à lui très inspiré de *KQML* mais avec une plus grande finesse dans la description des actes de communication et des protocoles d'échanges de ces actes.

3.5. Les protocoles de négociation

Les protocoles de négociation permettent de définir un cadre pour l'utilisation d'un langage. Ils peuvent être normalisés de plusieurs manières *(graphe de raisonnement, réseau de Pétri, langage formel, graphe état-transition),* [Kreifelts et Von Martial 91] considèrent important qu'un protocole de négociation *(dans des situations de résolution de problèmes distribués)* s'attache *(i)* à prendre en compte la conservation d'un certain niveau de communication entre les différents agents tout en

respectant l'autonomie, *(ii)* à favoriser l'échange de plans d'actions entre des agents, *(iii)* à synchroniser les actions, *(iv)* à laisser ouverte différentes voies possibles pour atteindre un compromis.

3.6. Engagements

Selon [Sen et Durfee 94], " *un engagement est un accord ou une promesse de faire quelque chose dans le futur*". Un engagement émis par un agent permet à l'agent récepteur d'avoir plus d'information pour planifier ses actions futures. Les engagements s'avèrent donc utiles dans un contexte coopératif [Jennings 93], mais aussi dans un contexte de négociation hostile. Un engagement est défini en tant que pro-attitude d'un agent, c'est à dire une attitude qui guide son action (de la même manière qu'une intention, une obligation). Ceci en opposition à une attitude d'information (croyance, connaissance) [Wooldridge et Jennings 95].

3.7. Les différents types de négociation

La négociation peut prendre diverses formes qui peuvent être regroupées en plusieurs catégories:

3.7.1. Les enchères

Avec l'explosion du commerce électronique, les enchères sont devenues un mécanisme d'achat et vente des produits disponibles à une grande échelle; voir *eBay*[1] ou *Yahoo Shopping*[2]. Dans le e-commerce les enchères sont la méthode la plus étudiée (c'est la forme de négociation la plus connue), Parmi les différentes enchères, nous décrivons *les enchères ascendantes, descendantes, offres scellées au meilleur prix* et *offres scellées au second meilleur prix*, qui sont parmi les plus répandues sur Internet. [Verrons 04] a décrit le déroulement de ces quatre types d'enchères comme suit:

3.7.1.1. Les enchères ascendantes

Ces enchères sont dites *ascendantes* car le prix proposé pour le bien mis en vente augmente avec le temps. Elles sont aussi appelées *enchères ouvertes, orales ou anglaises.* Ces enchères sont les plus populaires, qui déroulent sur plusieurs tours. Lors d'enchères ascendantes, le prix est successivement augmenté jusqu'à ce qu'il ne reste qu'un seul acheteur, qui gagne le bien au prix final. Dans le modèle le plus couramment utilisé, le prix est continuellement augmenté par le vendeur et les acheteurs quittent les enchères au fur et à mesure que le prix dépasse leur budget. Les autres acheteurs observent les départs et une fois qu'un acheteur a quitté l'enchère, il ne peut plus y revenir. Ces enchères sont implémentées dans *AuctionBot* [Wurman et al 98], un serveur d'enchères en ligne et sont couramment utilisées pour la vente d'objets d'art ou d'antiquités.

Ces enchères se déroulent sur plusieurs tours, il y a donc discussion entre le vendeur et les acheteurs, ce qui est bien, à notre sens, de la négociation. Le protocole de négociation de ces enchères peut se formaliser ainsi : le vendeur propose son article et le prix qu'il veut en obtenir à un ensemble d'acheteurs. Ceux-ci acceptent ou refusent la proposition. S'ils refusent, ils quittent la négociation. S'ils acceptent, ils restent en course pour l'obtention de l'article. S'il ne reste qu'un

[1] http://www.ebay.com

[2] http://shoping.yahoo.com

seul acheteur, il gagne le contrat. Sinon, le vendeur propose à nouveau son article avec un prix plus élevé. Le processus se termine lorsqu'il ne reste plus qu'un acheteur.

Du point de vue des avantages, il convient de signaler que la simplicité et la popularité de ce type d'enchère en font un excellent moyen de vente. La possibilité de pouvoir annuler une vente qui n'a pas rencontré le prix du marché est un atout pour le vendeur. Du côté des inconvénients, il faut préciser qu'il est généralement facile pour un groupe d'acheteurs de former une coalition qui aurait pour but de faire diminuer le prix des biens.

3.7.1.2. Les enchères descendantes

Ces enchères sont dites *descendantes* car le prix proposé pour le bien mis en vente diminue avec le temps. Elles sont utilisées notamment aux Pays-Bas pour la vente des fleurs, c'est pourquoi les économistes les appellent aussi enchères *hollandaises.* Ces enchères fonctionnent de façon exactement opposée aux précédentes : le vendeur propose un très haut prix pour son bien et le diminue jusqu'à ce qu'un acheteur se manifeste pour acquérir le bien au prix alors mentionné. Ces enchères sont utilisées dans *Fishmarket* [Noriega 98], une place d'enchères électroniques pour la vente de poissons en Espagne développée par Noriega. Ces enchères sont surtout utilisées pour la vente de produits périssables[1].

Ces enchères sont aussi une négociation, puisqu'elles se déroulent sur plusieurs tours. Le protocole se définit comme suit : le vendeur propose son article à un prix très élevé à un ensemble d'acheteurs. Si un acheteur accepte la proposition, il gagne le contrat. Si tous les acheteurs refusent, le vendeur leur propose à nouveau son article à un prix moins élevé. Le processus se termine lorsqu'un acheteur accepte la proposition ou lorsque le prix de réserve est atteint et que personne n'accepte la proposition.

Du côté des avantages, il convient de noter qu'étant donné que le prix débute à un seuil très élevé, il y a moins de risque pour le vendeur d'obtenir un prix inférieur au prix du marché. Si une personne veut vraiment un article, elle ne doit pas attendre longtemps sous peine de le voir attribuer à quelqu'un d'autre. Lorsqu'il y a plusieurs objets du même type à vendre, cette méthode de vente peut s'avérer rapide et efficace. Du côté des inconvénients, il convient de voir qu'il n'y a pas de compétition pouvant faire monter le prix d'un article à un niveau exorbitant comme dans le cas de l'enchère anglaise. Il faudrait mettre le prix de base dès le départ à un prix exorbitant pour observer ce phénomène mais cela ralentirait le processus de vente. Comme pour l'enchère anglaise, il est facile pour un groupe d'acheteurs de former une coalition qui a pour but de faire diminuer le prix des marchandises.

3.7.1.3. Les offres scellées au meilleur prix

Lors de ces enchères *FPSB (First Price Sealed Bid),* qui se déroulent sur un seul tour, chaque acheteur propose un prix unique, sans connaître le prix proposé par les autres (d'où le terme *offres scellées*). Le vainqueur est celui qui a proposé le prix le plus élevé et paie son prix, d'où leur nom. Comme il n'y a pas de seconde proposition possible, ces enchères ne sont pas une négociation à proprement parler. Le protocole de ces enchères est très simple : le vendeur propose un article à un

[1] Les enchères au cadran sont des enchères descendantes automatisées où le prix est affiché sur un cadran, d'où leur nom. Elles sont fréquemment rencontrés sur des marchés au cadran pour la vente de bétail, de poisson ou autres denrées périssables.

ensemble d'acheteurs. Ceux-ci peuvent soit accepter la proposition, auquel cas ils proposent un prix, soit refuser la proposition. Le gagnant de l'enchère est celui qui a proposé le prix le plus élevé (supérieur au prix de réserve). L'une des différences avec les précédentes enchères est qu'il n'y a pas de contre proposition possible.

Les offres scellées au meilleur prix offrent aussi l'avantage de ne pas influencer les enchérisseurs. Il est également plus difficile de former des ententes malhonnêtes, car les enchérisseurs peuvent venir de partout. De plus, les enchérisseurs n'ont pas besoin d'être présents lors de la vente et la mise peut se faire par la poste. Finalement, le plus grand inconvénient de ce type d'enchères réside dans le fait qu'une personne possédant une très grande fortune est fortement avantagée.

3.7.1.4. Les offres scellées au second meilleur prix

Ces enchères sont également appelées *enchères de Vickrey*. Le déroulement de ces enchères est exactement le même que celui des enchères aux offres scellées au meilleur prix, mis à part que le vainqueur paie le second meilleur prix proposé. Ces enchères sont implémentées dans *Magma* [Tsvetovatyy et al 97], un marché virtuel pour le e-commerce utilisant des agents. Ici aussi, il n'y a pas de seconde proposition possible et donc pas de négociation à proprement parler. Le protocole est identique à celui des *offres scellées au meilleur prix*, la seule différence étant le prix à payer par le gagnant du contrat.

Vickrey a montré que ce type d'enchères donne un prix au produit qui se rapproche de celui du marché, évitant ainsi, que des gens trop zélés fassent augmenter le prix d'un produit démesurément. Il ne semble toutefois pas intéressant pour un vendeur d'utiliser ce mode de vente, car les prix semblent suivre le marché. En fait, ce n'est qu'une illusion, dans la mesure où des études ont prouvé que les gens misent toujours un peu plus haut, car ils savent que le prix payé sera moindre que la mise faite.

3.7.1.5. Les enchères doubles

Les enchères doubles se déroulent entre plusieurs vendeurs et plusieurs acheteurs. Les vendeurs publient une offre de vente à un prix donné tandis que les acheteurs publient des offres d'achat à un prix fixé. L'acheteur ayant fait la plus grosse offre remporte la vente avec le vendeur ayant proposé le prix le plus bas, et paie ce prix. Ce processus se déroule tant qu'il y a des vendeurs et des acheteurs dont les offres concordent. Ces enchères sont néanmoins moins utilisées de façon électronique. Bien entendu, une personne aimant le risque sera avantagée par ce genre d'enchères. Les économistes prévoient un avenir prometteur pour ce type d'enchères dans le e-commerce. Du point de vue des inconvénients, nous n'en connaissons aucun.

3.7.1.6. Les autres formes d'enchères

Il existe encore beaucoup d'autres formes d'enchères comme *les enchères inversées (ou dégressives),* ces derniers consistent à proposer un prix de plus en plus bas par les participants pour remporter le marché. Elles sont employées par les grands groupes de l'industrie agro-alimentaire pour faire baisser les prix de leurs fournisseurs, les poussant à vendre en dessous de leur prix. Cette utilisation des enchères par de grands groupes peut ne pas aboutir à un contrat avec le fournisseur ayant proposé le prix le plus bas, mais avec un fournisseur choisi à l'avance qui aura alors baissé son prix au maximum.

D'autres systèmes d'enchères électroniques définissent leur propre protocole d'enchère, c'est le cas de *Kasbah* [Chavez et Maes 96], une place de marché pour l'achat et la vente de biens. *Kasbah* utilise un mécanisme de classification d'annonces et des agents acheteurs ou vendeurs. Chaque agent acheteur (resp. vendeur) consulte l'ensemble des agents vendeurs (resp. acheteurs) potentiels avant de consulter à nouveau l'un d'entre eux. Chaque agent peut faire une proposition à un autre agent, ou lui demander le prix qu'il offre pour l'item, ou encore demander quel est l'item. Lors d'une proposition, l'agent décide alors d'accepter ou non l'offre, mais ne peut formuler de contre-proposition avant le prochain tour de parole. Le vendeur adopte une stratégie de réévaluation du prix demandé qui dépend du temps afin de réussir à vendre son bien.

Les enchères ne sont pas les seules formes de négociation existantes, notamment pour le commerce électronique, comme le montre Kersten dans [Kersten et al 01]. La principale différence entre les enchères et les négociations est que les enchères n'impliquent qu'un seul attribut de négociation, qui est le prix, tandis que les négociations peuvent impliquer plusieurs attributs tels qu'une qualité de service, un délai d'expédition, etc. Nous présentons maintenant d'autres formes de négociation, *plus complexes à mettre en oeuvre.*

3.7.2. Les négociations multi-attributs

Les négociations multi-attributs, sont des négociations qui impliquent différents attributs devant être négociés. Elles sont directement opposes aux enchères qui n'impliquent qu'un seul attribut : un prix. Cette forme de négociation est cependant très répandue et à la base de nombreuses variantes de négociation. Un exemple de négociation multi-attributs est la négociation d'une voiture chez un concessionnaire. Le modèle, la motorisation, la couleur et de nombreuses options comme la climatisation, la direction assistée ou encore la présence d'airbags, en plus du prix, seront négociés. Une fonction d'utilité est utilisée pour comparer les différentes offres. Cette fonction permet de pondérer l'importance de chaque attribut et chaque attribut est associé à une autre fonction d'utilité déterminant la satisfaction pour cet attribut. Dans [Bichler et al 02], Bichler et ses collègues proposent d'utiliser les enchères pour les négociations multi-attributs en utilisant la fonction d'utilité comme seul attribut d'évaluation de l'enchère. [Jonker et Treur 01] une architecture d'agent pour la négociation multi-attributs et décrivent le modèle de négociation qu'ils proposent et l'illustrent avec l'exemple de l'achat d'une voiture.

3.7.3. Les négociations multi-niveaux

Dans ce type de négociation où le contrat se décompose en plusieurs sous-contrats, dépendants les uns des autres. La négociation s'effectue séquentiellement pour chaque sous-contrat, la réussite globale est atteinte lorsque tous les sous-contrats ont été négociés avec succès. Lorsqu'un sous-contrat est en cours de négociation et qu'aucune solution n'est possible, on effectue un *backtrack* pour la négociation du sous-contrat précédent.

3.7.4. Les négociations combinées

Les négociations combinées sont utilisées lorsqu'une personne a besoin d'un ensemble d'objets non disponibles auprès d'un unique vendeur. Il faut alors négocier chaque objet (ou sous-ensemble d'objets) séparément et avoir un mécanisme de liaison entre les négociations, car si l'ensemble des objets ne peut être acquis, aucun objet ne doit l'être. De plus, il ne faut obtenir chaque item qu'en un seul exemplaire. Les différentes négociations sont indépendantes les unes des autres, alors que l'ensemble des objets négociés sont typiquement interdépendants. Les

négociations peuvent être de type différent pour chaque objet ou sous-ensemble d'objets. On peut donc rencontrer des enchères anglaises, des négociations de type multi-niveaux, ou autres.

3.7.5. Les négociations heuristiques

Dans les protocoles de négociation qu'on a déjà discutés, les agents pouvaient accepter ou repousser une proposition. Si les agents peuvent uniquement accepter ou rejeter les propositions d'autres agents, alors la négociation peut être très longue et inefficace puisque l'auteur d'une proposition n'a pas le moyen de vérifier pourquoi la proposition est inacceptable, ni si les agents sont proches d'un accord, ni dans quelle direction une proposition peut être changée pour convenir à l'autre agent. Pour améliorer l'efficacité de la négociation, les agents doivent fournir des réactions plus utiles aux propositions qu'ils reçoivent. Ces réactions peuvent prendre la forme d'une *contre-proposition* (proposition *refusée* ou *modifiée*). Une *contre-proposition* est une proposition alternative engendrée en réponse à une proposition. À partir de telles réactions, l'auteur doit être capable d'engendrer une proposition qui est probablement plus apte à mener à un accord.

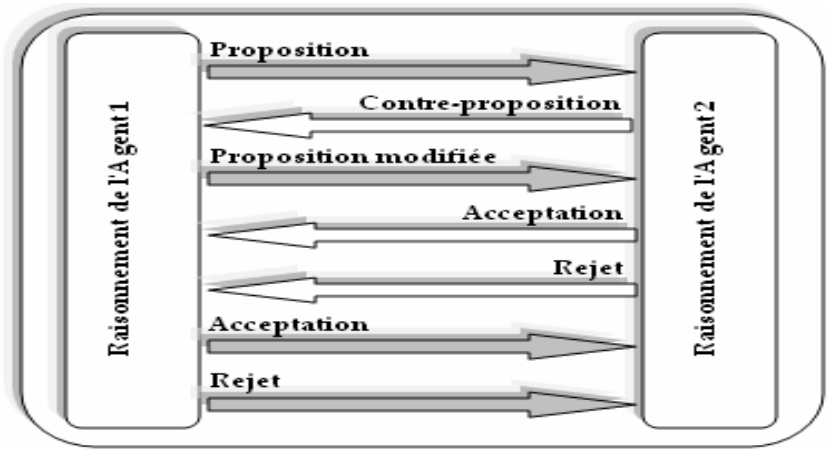


Figure 4.2. Négociation heuristique.

Donc il y a une différence importante entre les mécanismes de négociation que nous avons discutés précédemment et ce type de négociation qui est la négociation heuristique. Cette différence consiste dans le modèle de décision de l'agent. Ce dernier est le dispositif de décision que les participants emploient en conformité avec le protocole de négociation pour atteindre leurs objectifs. Dans certaines circonstances, comme celles que nous avons discutées dans les sections précédentes, le protocole de négociation est dominant (par exemple, dans la plupart des protocoles d'enchères que nous avons présentés, il y a une stratégie optimale (dominante) que l'agent peut utiliser). Le problème central de la négociation heuristique est de modéliser la décision de l'agent au cours de la négociation. Généralement, le protocole choisi ne prescrit pas un cours optimal d'action, notamment une stratégie optimale pour l'agent. Dans ce cas, le succès relatif de deux agents est déterminé par l'efficacité de leur modèle de raisonnement. Les agents doivent découvrir d'une manière heuristique quelle est la meilleure proposition ou contre-proposition qu'ils doivent faire. Ils doivent découvrir cela en partant du modèle de décision qu'ils emploient. Meilleur sera le modèle, et plus grande sera la récompense de l'agent.

Pendant la négociation heuristique, les agents peuvent essayer de changer le rejet ou la modification d'une proposition faite par un autre agent en utilisant des *arguments*, l'argumentation a pour but de modifier les croyances des autres agents afin qu'ils adoptent le même point de vue,

les mêmes croyances et intentions que l'agent argumentant, donc ce type de négociation est utilisé chez les agents logiques qui possèdent une base de connaissances avec des prédicats et des règles d'inférence. Le modèle d'agent dans le système est un modèle *BDI (Belief, Desire, Intention)* et la décision pour choisir le bon argument dépend des propres buts, des rapports entre ces buts, des croyances de l'agent, et de ce que l'agent croit concernant l'autre agent [Verrons 04].

4. Quelques systèmes de négociation utilisés dans le e-commerce

La négociation dans les systèmes multi-agents a déjà fait l'objet de nombreux travaux, nous allons commencer par le *Contract Net Protocol* qui est le plus utilisé par la communauté qui s'intéresse à ce sujet, et qui est la première forme sous laquelle est apparue, puis nous allons examiner différentes *plate-formes de négociation* qui ont été implémentées dans *le e-commerce.*

4.1. Le réseau contractuel

Ce protocole est largement utilisé. Il a été développé par Davis et Smith [Davis et Smith 80] comme modèle de négociation pour la résolution de problèmes distribués. Il est basé sur la coopération entre agents dans le but d'effectuer une tâche donnée. C'est un des principaux protocoles d'interaction, qui a été utilisé dans de nombreuses applications telles que le contrôle du trafic aérien ou le commerce électronique [Sandholm et Lesser 95]. Le protocole Contract-Net est fondé sur la notion d'appels d'offres sur les marchés publics. Les relations entre les clients et les fournisseurs se créent grâce à un mécanisme d'appels d'offres émis par un manager (*gestionnaire*). La décision finale est adoptée par celui-ci à partir des réponses envoyées par les contractants. Ce protocole se décompose en quatre phases [Le Bars 03]:

1. ***Décomposition du problème:*** Appel d'offres pour la réalisation d'une tâche.
2. ***Distribution des sous problèmes:*** le message est envoyé par le manager à tous les agents qu'il estime capable de réaliser la tâche ou à tous les agents du système. A partir de la description de la tâche, les agents contractants construisent une proposition qu'ils envoient au manager.
3. ***Solution des sous problèmes:*** le manager reçoit et évalue les propositions et attribue la tâche au meilleur contractant.
4. ***Réponse:*** le contractant auquel la tâche est confiée envoie un message au manager en lui confirmant son intention de l'accomplir.

Le manager va :	**Le contractant va :**
1. annoncer la tâche dont il veut qu'elle soit accomplie, 2. recevoir et évaluer les propositions des contractants, 3. choisir l'une de ces offres, 4. recevoir et synthétiser les résultats.	1. recevoir la demande de travail, 2. évaluer sa capacité à y répondre, 3. répondre (je ne peux pas, faire une offre), 4. si son offre est retenue, faire le travail, 5. Renvoyer ses résultats.

Tableau 4.1. Les rôles de manager et de contractant.

Le Contract Net est un protocole basé sur l'échange de contrats, qui met en relation un agent *(le manager)* avec plusieurs autres agents *(les contractants).* C'est donc de la négociation de 1 vers n agents. Le Contract Net est un modèle où seul le manager émet des propositions, les contractants ne peuvent que faire une offre et pas de contre-proposition. En revanche, un tel modèle doit inclut un processus de contre-propositions afin de prendre en compte l'avis des contractants, de manière à aboutir plus rapidement à une solution acceptable par tous en comparaison avec un modèle où le

manager est le seul à décider quelle nouvelle proposition envoyer sans savoir si celle-ci va dans le sens des contractants. De plus, il n'y a pas de message indiquant à un contractant qu'il n'a pas été retenu pour la tâche. Un contractant n'a donc pas le moyen de savoir si l'offre qu'il a soumise n'a pas été retenue ou si le manager n'a pas encore pris sa décision, ce qui peut entraîner qu'un contractant ne fasse pas d'offre pour une autre tâche alors qu'il n'a pas été choisi pour la précédente. Bien que le *Contract Net* soit une référence parmi les protocoles de négociation, celui-ci ne convient pas pour de nombreux problèmes, ce qui explique les nombreux travaux sur des extensions de celui-ci.

4.2. Kasbah

Kasbah[1] C'est un système où les utilisateurs créent des agents pour négocier la vente et l'achat de biens pour leur compte sur Internet. Ces biens sont classifiés, reprenant ainsi l'idée des petites annonces classées par type. Lors de la création d'un agent, pour la vente ou l'achat, l'utilisateur spécifie le type de bien à négocier, la date à laquelle il souhaite que la transaction soit effectuée, le prix désiré, le plus petit (resp. grand) prix acceptable et la stratégie de négociation à choisir. L'utilisateur précise également si l'agent doit demander son accord avant de conclure la vente et s'il veut être averti par mail lorsqu'un accord est trouvé. Une fois l'accord atteint, la transaction physique peut avoir lieu, ce qui doit être réalisé par les agents humains.

Du point de vue de l'implémentation, *Kasbah* met en relation les agents ayant des buts communs, les communications entre agents se font de 1 vers 1. Le fonctionnement en parallèle des agents est simulé en accordant un *time-slice* à chacun à tour de rôle. Durant ce *time-slice*, l'agent détermine le prix courant désiré, décide avec quel agent communiquer et enfin communique avec celui-ci. Les agents communiquent via des actes de langages spécifiques à *Kasbah*. Si une évolution vers KQML a été prévue, ce système n'en reste pas moins axé sur l'e-commerce, qui est un aspect très spécifique de la négociation.

4.3. AuctionBot

AuctionBot[2] est un serveur d'enchères expérimental Son but est de permettre à n'importe quel internaute de participer aux enchères sur le net. L'utilisateur a la possibilité de choisir son type d'enchère (anglaise, hollandaise…) et spécifie les paramètres (délai, nombre de vendeurs…). De plus, le site fournit une interface programmable pour les utilisateurs qui veulent créer leur agent logiciel. *AuctionBot* est un *framework* utile à la fois pour le commerce et pour la recherche au sens où il propose une large variété de types d'enchères (English, Vickrey, etc...) et une API pour créer ses propres agents qui participeront à la place de marché d'*AuctionBot*. Son architecture est asynchrone, il stocke les enchères dans une base de données et il peut en gérer plusieurs simultanément. Afin de participer aux enchères, il faut s'enregistrer. Les utilisateurs humains peuvent consulter leurs comptes via une page web ou choisir d'être avertis de l'avancement des enchères par mail. Comme *AuctionBot* répertorie les enchères proposées dans un catalogue organisé de façon hiérarchique, un vendeur peut placer son enchère n'importe où dans le catalogue existant ou étendre celui-ci. Il peut également choisir de mettre son enchère dans le catalogue public ou de la proposer à un groupe privé.

---

1 *Kasbah* a été développé au MIT Media Lab par Pattie Maes [Chavez et Maes 96].

2 *AuctionBo*t [Wurman et al 98] développé et opérationnel au laboratoire d'intelligence artificielle de l'université du Michigan par Michael P. Wellman et Peter R. Wurman. (http://eecs.umich.edu).

Du point de vue de l'implémentation, les agents placent les enchères dans la base de données, tandis qu'*AuctionBot* collecte les enchères, détermine une allocation d'après un ensemble de règles d'enchères bien défini et avertit les participants. En revanche, il n'exécute pas les transactions, il n'impose pas les échanges ni ne vérifie la crédibilité des participants. Ce n'est donc pas l'agent initiateur du contrat qui gère l'enchère mais un programme d'*AuctionBot*. Dans notre modèle, et comme plusieurs d'autres, nous laissons aux agents l'autonomie de la gestion de la négociation, chaque agent initiateur (vendeur dans le cas des enchères) gère le déroulement de sa négociation, il n'y a pas de centralisateur d'offres de vente et d'achat qui effectue l'appariement selon les règles d'enchères définies comme dans *AuctionBot*.

4.4. Fishmarket

Fishmarket[1] est une agence d'enchères électroniques pour vendre du poisson, où les agents peuvent être soit humains, soit virtuels. *Fishmarket* a été conçu pour montrer la complexité de ces interactions tout en gardant aussi fortement que possible une similarité avec l'ontologie des éléments du marché de poisson. Chaque agent logiciel de *Fishmarket* représente soit un intermédiaire du marché, soit un vendeur ou encore un acheteur. *Fishmarket* utilise un cadre dialectique et peut être étendu afin de convenir aux différents types d'enchères (anglaises, hollandaise, Vickrey, etc.). Les enchères y sont traitées séquentiellement, tandis que notre modèle permet d'effectuer plusieurs négociations en parallèle lorsqu'il n'y a pas de conflit sur les ressources.

4.5. Tête-a-tête

Dans *Tete-a-tete*[2] et a l'inverse de la plupart des systèmes de négociation qui négociant exclusivement sur la variable prix, cet agent négocie d'autres items (garantie, temps de livraison, contrat de service, possibilité de retour de la marchandise, options de financement, promotions, …etc.) Basées sur une argumentation bilatérale, les négociations de *tete-a-tete* comprennent des échanges XML, des critiques, des contres proposition. Par exemple, un agent reçoit des propositions de divers agents de vente. Chaque proposition correspond à une offre spécifique. L'agent les compare, les ordonne suivant les préférences de l'utilisateur. Si un utilisateur est insatisfait vis à vis des propositions, son agent renvoie ses préférences et les agents de vente font à leur tour des contres proposition.

4.6. MAGNET

MAGNET[3] est un banc de tests pour la négociation multi-agents, implémenté sous la forme d'une architecture généralisée de marché. Il fournit un support pour un large panel de transactions, du simple achat/vente de biens à la négociation complexe de contrats entre agents. *MAGNET* reprend le protocole du *ContractNet*, il n'y a donc pas de mécanisme de contre-propositions afin de trouver une solution. Dans *MAGNET*, il y a un intermédiaire explicite dans le

---

[1] *Fishmarket* a été développée à l'Institut de Recherche en Intelligence Artificielle en Espagne par Pablo Noriega [Noriega 98].

[2] *Tete-a-tete* est présenté sur le site (http://ecommerce.media.mit.edu/tete-a-tete)

[3] *MAGNET* : Multi AGent NEgotiation Testbed [Collins et al 98] est développé à l'université du Minnesota par John Collins et al [Collins et al 98].

processus de négociation et les agents interagissent par son intermédiaire, tandis que les agents interagissent directement entre eux dans notre processus de négociation. La rétractation est possible moyennant une pénalité à payer.

4.7. GNP

GNP[1] c'est une plateforme générique de négociation d'entreprises à entreprises orientée enchères pour les places de marché. La négociation entre les agents s'effectue via le service de négociation qui s'occupe de l'appariement entre la demande des acheteurs, l'offre en bien et services des vendeurs et la fixation des prix. Cette plateforme reçoit les annonces des acheteurs ou des vendeurs et les mises qui répondent à ces annonces. Les règles de marché sont définies dans des fichiers XML. Un avantage du *GNP* est qu'il propose différents modèles de négociation, ce qui permet d'instancier facilement les différents types de négociation prédéfinis.

5. Conclusion

Avec les progrès des technologies de l'information, des systèmes multi-agents et des places de marché électroniques, le besoin d'agents logiciels capables de négocier avec les autres à la place de leur utilisateur devient de plus en plus important. C'est pourquoi de nombreux travaux ont été accomplis dans ce domaine. La plupart des formes de négociation décrites dans ce chapitre sont dédiées au commerce électronique, qui connaît un essor important de nos jours. Supposent typiquement que les agents ont la possibilité de choisir la meilleure stratégie parmi toutes les stratégies possibles. Il est supposé que la prise de décision est instantanée. Pour implémenter une telle approche, le coût du calcul pour la décision doit être pris en compte et ceci peut mener à des problèmes computationnels complexes. La négociation aux enchères, est un champ important d'étude avec beaucoup d'applications, la plus notable étant celle du commerce électronique.

Basé sur ce que nous avons présenté, il doit être clair qu'il n'y a pas une meilleure approche universelle ou une technique toujours appropriée pour la négociation automatisée, Il y a plutôt un ensemble sélective de méthodes avec des caractéristiques et des propriétés de performances qui varient significativement en fonction du contexte de négociation.

[1] *GNP:* Generic Negotiation Platform est réalisée par [Benyoucef et al 00], [Gerin-Lajoie 00] au département IRO de l'université de Montréal.

Chapitre 5
Approche de Simulation Multi-agents Participative pour la Négociation dans le E-commerce

1. Introduction

Nous avons présenté dans le chapitre précédent les différentes formes de négociation existantes ainsi que différentes plateformes de négociation réalisées parmi les plus connues. Dans ce chapitre, nous présentons un mécanisme de négociation multicritère pour le commerce électronique, basé sur un modèle multicritère utilisant une approche de simulation multi-agents participative. Selon ce modèle, l'utilisateur qui jeu le rôle d'un acheteur/vendeur réel doit spécifier les valeurs désirées sur chaque attribut décrivant le produit à acheter et les valeurs minimales acceptables sur chaque critère. Autrement dit, un utilisateur aura le contrôle d'un composant du modèle simulé et il peut le manipuler à son gré durant l'exécution, en modifiant certains paramètres du composant. Un nombre plus important de ces paramètres rendrait le système plus complexe, et l'approche participative plus efficace. Nous avons utilisé dans notre modèle un mécanisme de négociation qui est basé sur un protocole d'enchères anglaises inversées et conduit la négociation vers le point d'aspiration de l'acheteur.

Notre modèle peut être considéré comme un environnement interactif permettant d'intégrer des participants humains dans le simulateur, au même titre que les agents. Il comporte aussi des interfaces utilisateurs avec assistants qui ont pour but d'aider les professionnels du domaine ou des apprentis à participer, on lui ajoutera les fonctionnalités nécessaires pour l'utiliser dans un processus de conception participative. Dans la suite du ce chapitre, nous inclurons la simulation participative afin que des utilisateurs puissent, grâce à une interface utilisateur, contrôler nos agents et développer leur base comportementale. Cela revient à mettre en oeuvre un système d'*apprentissage participatif* par imitation à l'aide des êtres humains.

2. Participation des humains à des simulations multi-agents

La simulation sert à la *formation* ou à *l'entraînement* [Marsella et Johnson 98] et, dans ce cas, elle exige la participation des apprentis. L'idée est de permettre à ces derniers de pratiquer des opérations professionnelles dans un environnement simulé pour diminuer le nombre des sessions pratiques dans l'environnement réel (difficiles, coûteuses ou risquées à organiser). Il existe une tendance à permettre la participation des experts du domaine à certaines phases de modélisation. Le but de ces recherches est justement de fournir aux experts de nouveaux moyens pour influencer plus activement la modélisation et pour mieux valider le modèle. [Bousquet et al 01] proposent la participation des experts du domaine à des jeux de rôles *(JdR)* dans lesquels le modèle multi-agent est converti en un modèle conceptuel des *JdR*. Cependant, ces *JdR*, bien que distribués et utilisant pour certains un support informatique, ne sont pas encore *multi-agents* parce qu'il n'y a pas d'agents logiciels dedans.

Une topologie de participation appropriée à la simulation multi-agent peut être celle dans laquelle les acteurs *(*dans notre modèle sont *les experts du domaine* ou *des apprentis)* jouent les rôles simulés et interagissent directement avec les acteurs artificiels *(agents)* dans le monde virtuel [Guyot et Drogoul 04], [Guyot et Honiden 06], [Becu et al 08]. Cela exige évidemment de mettre en œuvre *des interfaces homme-machine* qui permettent aux utilisateurs d'effectuer leurs activités dans le simulateur de façon aussi naturelle que possible. Les chercheurs de l'équipe *MIRIAD (Modélisation des Interactions et Recherches en Intelligence Artificielle Distribuée)* dirigé par Alexis Drogoul orientent leurs recherches vers

la conception et l'implémentation des plateformes de simulation et de modélisation multi-agent qui autorisent la participation des acteurs humains. Ils attendent de cette approche un double effet [Nguyen-Duc 05]:

Les utilisateurs d'une telle simulation participative ont la possibilité de contribuer à l'amélioration des comportements d'agents en utilisant leurs connaissances spécifiques au domaine[1]. À un niveau plus élevé, l'utilisation de dialogues du type *humain-agent* comme outils d'amélioration des comportements d'agent est au coeur de *la conception participative de comportements d'agents* proposée par [Drogoul et al 03]. Inspirée du travail de [Bousquet et al 01], cette méthode de conception est également basée sur les *JdR,* ceux-ci étant mis en place directement sur le simulateur lui-même. Les acteurs sociaux vont donc jouer leurs rôles en utilisant *une interface homme-machine dédiée.* Ceci requiert des procédures d'*apprentissage,* qui font l'objet de recherches actives dans de nombreuses sous-disciplines de l'*IA.*

3. Difficulté de la modélisation de comportement Humain

Enseigner le comportement des êtres humains à des agents reste toujours une activité très difficile, le grand problème de la plus parts des simulations provient de la modélisation des connaissances et des comportements des agents. Afin de résoudre ce problème, il est possible d'utiliser la simulation participative qui fait intervenir des utilisateurs humains qui modifieront les décisions des agents selon leurs propres expériences et croyances.

Aussi il est toujours très difficile[2] de modéliser, même de manière simplifiée, le comportement des êtres humains, surtout dans des situations dans lesquelles ils utilisent une expérience fortement liée à un domaine spécifique comme le commerce électronique, et plus précisément les protocoles de négociation. Les difficultés proviennent de la formalisation des comportements des agents d'après les observations ou les théorie disponibles. Trois problématiques doivent être résolues [Nguyen-Duc 05] : (I) Comment modéliser l'environnement des agents?, (II) Quelles perceptions possibles pour les agents?, (III) Quelle granularité pour les actions (elle correspond à un changement de l'état interne de l'agent ou/et de l'environnement)?. La simulation participative comme méthode d'extraction de connaissances aide à résoudre ces problèmes, elle permet d'accroître la fiabilité du modèle en travaillant directement avec des experts du domaine, les modèles sont remis en questions par les experts qui critiquent les différents paramètres ce qui permet une amélioration de ce modèle, d'un autre côté la simulation participative intervient comme méthode d'apprentissage des agents aux actions commandées directement par un être humain[Pascalau 06].

---

1 Dans un premier temps, les concepteurs ont un rôle intermédiaire, en traduisant leurs idées sur certaines opérations concrètes en des connaissances utilisables par les agents. Ensuite, on cherche à atténuer le rôle des concepteurs en introduisant des outils automatiques comme des composants logiciels additionnels. L'outil le plus simple pourrait être un « éditeur de comportement » permettant aux acteurs humains de changer quelques paramètres des comportements implémentés[Nguyen-Duc 05].

2 Le défi majeur de la modélisation multi-agent des phénomènes sociaux est en effet la formalisation des connaissances informelles et acquises en pratique par les acteurs. Il provient principalement de deux sources [Nguyen-Duc 05]: Les différences entre le domaine d'études dans lequel l'outil de simulation multi-agent est utilisé et celui des concepteurs de l'outil. Et la subjectivité des connaissances basées sur l'expérience individuelle ou collective.

Grâce à cette technique des participants humains pourront accroître les champs d'action de nos agents rendant plus réalistes les simulations. Leurs connaissances évolueront dans le temps, les agents devront être capables d'apprendre et de remettre en question leur base de connaissances en fonction des indications apportées de l'extérieur. Une solution existante [Drogoul et al 02] consiste à placer les *enseignants* dans le cadre de la simulation grâce à un jeu de rôle (*JDR*). Cela créera un véritable dialogue *maître-apprenti* que les agents exploiteront pour extraire des connaissances [Pascalau 06].

Nous précisons donc ici les objectifs de l'application de l'approche de simulation participative à la négociation dans le domaine de commerce électronique, nous cherchons :

- À simuler une nouvelle approche de négociation dans le e-commerce en utilisant une approche de simulation multi-agents,
- À développer un outil de simulation qui permet la participation des utilisateurs (experts de domaine ou des apprentis) dans un système d'enchères multi-attributs,
- À lui ajouter des fonctionnalités pour le transformer en un outil de conception participative de comportements d'agents,
- À formaliser une solution méthodologique pour guider les utilisateurs dans ce processus,
- À automatiser le processus de négociation à partir de cette approche de simulation participative, pour reproduire à l'aide d'un système multi-agents les comportement des utilisateurs, et
- À simuler l'adaptation de nos agents à des nouvelles configurations d'environnement.

4. Une approche de simulation participative pour la négociation dans le domaine de commerce électronique

4.1. Représentation de système de négociation proposé

Les protocoles d'enchères offrent les mécanismes les plus connus du grand public et les plus compétitifs pour la négociation [Jennings et al 01], [Lee et al 03]. Parmi ces protocoles d'enchères on trouve également les enchères hollandaises (les enchères descendantes), les enchères à enveloppes scellées (offres scellées aux meilleur prix), les enchères Vickrey (offres scellées aux second meilleur prix) et les enchères anglaises (ascendantes) présentées dans le chapitre précédent. Face aux enchères portant uniquement sur le prix qui sont largement dominantes, les enchères multicritères permettent de négocier sur d'autres critères tels que la qualité et le délai de livraison [Bichler et al 02], [David et al 02]. Les acheteurs définissent leurs préférences sur l'article recherché et les vendeurs sont en concurrence sur tous les attributs spécifiés par l'acheteur. Donc nous prenons en compte dans notre modèle les enchères multi-attributs où leur automatisation s'appuie sur les composants suivants:

- Un modèle de simulation qui permet la participation des utilisateurs à partir d'une interface utilisateurs dédiée pour permettre à ces derniers de contrôler un composant du modèle simulé, les manipuler à leurs grés durant l'exécution, en modifiant certains paramètres du composant, et pour leurs permettre aussi l'expression des préférences de l'acheteur, et de déterminer la contrainte initiale, la durée d'un tour, et la date de clôture de l'enchère,…etc., ce modèle contient des fonctionnalités pour le transformer en un outil de conception participative de comportements d'agents (reproduire à l'aide d'un système multi-agents les comportement des utilisateurs),
- Un modèle pour guider les utilisateurs dans ce processus,

- Une méthode multicritère pour la sélection de la meilleure offre par l'agent acheteur,
- Une méthode pour l'expression des nouvelles propositions par l'agent vendeur,
- Un module de décision de l'agent négociateur pour automatiser le processus de négociation.

4.2. Les enchères multi-attributs anglaises inversée

Selon [Bellosta et al 04] une enchère anglaise inversée est un processus itératif sur plusieurs tours, avec une date de clôture où les agents vendeurs sont en compétition pour vendre un produit à un agent acheteur. Au début un utilisateur doit spécifier à l'agent acheteur, les préférences, la contrainte initiale qui détermine quelles sont les proposition acceptable, la durée d'un tour, et la date de clôture de l'enchère. Puis et au début de l'enchère, l'agent acheteur ouvre les enchères en envoyant à l'ensemble des agents vendeurs, la contrainte initiale, la durée d'un tour et la date de clôture de l'enchère. A chaque tour de l'enchère, l'agent acheteur collecte toutes les propositions (une par agent vendeur), sélectionne la meilleure proposition comme proposition de référence pour l'étape suivante, met en attente le vendeur correspondant et formule la contrainte pour le tour suivant. L'enchère est synchrone. Chaque agent vendeur doit envoyer sa proposition avant la fin du tour, sinon il est éliminé. L'enchère se termine sur la meilleure proposition courante, soit lorsque tous les vendeurs sauf un ont abandonné ou soit lorsque la date de clôture est atteinte. Ainsi, une enchère multicritère se définit à l'aide d'une contrainte initiale, d'une relation de préférence et d'une fonction de relance qui calcule la contrainte que toute proposition devra satisfaire au tour suivant. La plupart des relations de préférence utilisées dans les enchères multicritères sont basés sur un modèle d'agrégation additif [Bichler et al 02], [David et al 02], [Oliveira et al 99] où l'utilisateur(qui jeu dans ce cas le rôle d'un acheteur) définit ses préférences en choisissant un poids pour chaque critère, rendant compte de son importance. Donc nous présentons un cadre permettant de définir des mécanismes d'enchères multicritères anglaises inversées.

La plupart des travaux menés pour automatiser les enchères multi-attributs se basent sur le modèle de la somme pondérée [Bichler et al 02], [Oliveira et al 99]. Où L'évaluation d'une proposition se fait en sommant les valeurs pondérées relatives à chaque critère. La meilleure proposition correspond à l'évaluation la plus élevée. Toutefois, ce modèle présente deux inconvénients majeurs [Vallin et Vanderpooten 02]:

• Les poids associés aux attributs sont difficiles à définir et à interpréter, d'autant plus que de petites variations de ces poids peuvent changer radicalement le choix de la meilleure proposition. Notre modèle nous permette la définition de ces poids par les experts qui sont impliqués dans la simulation.
• Le modèle de la somme pondérée est totalement compensatoire, considérons par exemple un produit évalué suivant les deux critères prix et qualité, il peut arriver qu'une proposition bien équilibrée suivant ces critères, assez bonne sur le prix et la qualité, ne puisse pas gagner une enchère si elle est en compétition avec d'un côté une proposition avec un prix très bas mais de très mauvaise qualité et d'un autre côté une proposition avec un prix élevé mais de très bonne qualité. Notre modèle n'autorise pas la compensation des critères. De plus, il permet la définition d'un mécanisme d'enchères offrant un contrôle direct à l'agent acheteur par les utilisateurs qui sont impliqués à la simulation.

Donc dans notre modèle qui est basé sur le modèle de [Oliveira et al 99], nous avons proposer un protocole d'enchères anglaises inversées multi-attributs, basé sur la somme pondérée. L'agent acheteur négocie directement avec chaque vendeur. Les enchères comportent plusieurs étapes d'une

durée prédéfinie. La négociation est lancée par l'agent acheteur qui envoie aux agents vendeurs concernés ses préférences ainsi que l'évaluation minimale de départ pour une proposition. Les vendeurs évaluent la demande, envoient une proposition acceptable ou se retirent de la négociation. Quand l'acheteur a reçu toutes les offres ou le délai prédéfini a été atteint, les offres sont évaluées et la meilleure sélectionnée. Son évaluation sert de base à l'étape suivante sous la forme d'une contre-proposition envoyée par l'agent acheteur aux vendeurs restant en compétition. Les enchères continuent jusqu'à ce que tous les vendeurs sauf un aient abandonné et se terminent sur un contrat avec le dernier vendeur. Ce protocole a été défini pour prendre en compte l'aspect multicritère des enchères. Nous l'avons étendu dans le cadre du modèle multicritère où l'interventions des utilisateurs peu régler le problème de la somme pondérée.

4.3. Protocole et mécanisme d'enchère

L'objectif du protocole est de définir les messages que les agents pourront s'envoyer avec la dynamique opérationnelle associée. Donc par la suite nous présentons le protocole de négociation utilisé dans notre modèle et les paramètres permettant de le spécialiser, et qui est caractérisé par une suite de messages échangés entre un agent acheteur et un ensemble des agents vendeurs[1]. Ainsi nous présentons par la suite dans le chapitre suivant les messages échangés entre ces derniers et les agents d'interfaces.

4.3.1. Primitives de négociation et leur sémantique

Nous présentons dans ce qui suit les actions valides durant les enchères, les règles qui les régissent ainsi que la sémantique qui leur est associée. Le protocole considéré définit une adaptation à l'aspect multicritère du protocole des enchères anglaises inversées [Oliveira et al 99].

- ✓ ***CallForPropose (a, g, Preferences) :*** c'est la première primitive que l'agent acheteur envoie aux agents vendeurs pour leur proposer ses préférences, donc *a* lance les enchères avec le groupe de vendeurs *g* en donnant ses préférences sur le produit, ici *a* suppose que les vendeurs peuvent fournir le produit désiré.
- ✓ ***propose (v, a, bid):*** *v* envoie une proposition à *a*, en réponse à un message *callForPropose* ou à un *requestForPropose*.
- ✓ ***requestForPropose (a, g, counterproposal):*** *a* demande au groupe de vendeurs *g* restants d'améliorer leurs offres, en réponse aux messages *propose*.
- ✓ ***Accept (a, v):*** *a* accepte la dernière proposition envoyée par *v*, en réponse à un message *propose* et annonçant la fin des enchères.
- ✓ ***Reject (a, v):*** *a* élimine *v* des enchères, en réponse à un message *propose* et annonçant la fin des enchères.
- ✓ ***abort (v):*** *v* abandonne les enchères, en réponse à un message *callForPropose* ou à un message *requestForPropose*.

[1] Il est nécessaire de définir des primitives spécifiques à l'agent acheteur et des primitives spécifiques aux agents vendeurs. Ce qui nécessiterait une plateforme *FIPA* ou plus simplement des agents communiquant via *ACL*.

4.3.2. Etapes de l'enchère

Au début un utilisateur doit spécifier à l'agent acheteur, les *préférences*, la *contrainte initiale*, la *durée d'un tour*, et la *date de clôture de l'enchère*. Puis et au début de l'enchère, l'agent acheteur ouvre les enchères en envoyant à l'ensemble des agents vendeurs un appel d'offre (performative *callForPropose*), la *contrainte initiale*, la *durée d'un tour* et la *date de clôture de l'enchère*. A chaque tour de l'enchère, l'agent acheteur collecte toutes les réponses[1] (une par agent vendeur, i.e.,messages *propose* et/ou *abort*), sélectionne la meilleure proposition comme proposition de référence pour l'étape suivante, et met en attente le vendeur correspondant et formule la contrainte pour le tour suivant, puis il envoie cette nouvelle contrainte aux agents vendeurs (performative *requestForProposal*) sauf le vendeur en attente. Les enchères échouent s'il n'y a pas de proposition, sinon les enchères se terminent avec succès. S'il ne reste qu'une seule proposition, celle-ci est gagnante et l'acheteur envoie un message d'acceptation à l'agent associé (performative *accept*). Dans le cas contraire, la durée de l'enchère est atteinte et plusieurs propositions restent en compétition. L'agent acheteur envoie un message d'acceptation à l'agent associé à la meilleure proposition et un message de rejet autres vendeurs (performative *reject*). L'enchère est synchrone. Chaque agent vendeur doit envoyer sa proposition avant la fin du tour, sinon il est éliminé. L'enchère se termine sur la meilleure proposition courante, soit lorsque tous les vendeurs sauf un ont abandonné ou soit lorsque la date de clôture est atteinte. Ainsi, une enchère multicritère se définit à l'aide d'une contrainte initiale, d'une relation de préférence et d'une fonction de relance qui calcule la contrainte que toute proposition devra satisfaire au tour suivant.

4.3.3. La détermination des contre-propositions

La définition des *contre-propositions* assure la progression des propositions vers le point d'aspiration de l'acheteur et l'efficacité du mécanisme d'enchères proposé. Elle est basée sur la règle du *"beat-the-quote"*[2] introduite dans [Vulkan et Jennings 00], [Wurman et al 02] que nous avons adaptée pour prendre en compte l'aspect multicritère.

4.4. Processus d'enchère

La stratégie de négociation n'est pas la même selon le rôle de l'agent, il y a donc deux sortes de stratégies. Dans notre modèle le processus d'enchère est basé sur les enchères anglaises inversées [Oliveira et al 99], chaque agent négociateur exécute un processus d'enchère suivant la stratégie décrite ci-dessous. Nous proposons néanmoins deux stratégies par défaut (l'une pour l'agent acheteur et l'autre pour l'agent vendeur):

1 Trois situations peuvent alors se présenter : (I) Toutes les réponses sont des refus et la négociation se termine sur un échec.
(II) Au moins deux réponses contiennent une proposition pour le produit recherché. La meilleure proposition est alors sélectionnée, le vendeur correspondant mis en attente et la *contreproposition* calculée et envoyée aux vendeurs restants.
(III) Une seule réponse comporte une proposition, l'agent acheteur l'accepte, envoie un message d'acceptation et attend la validation du vendeur. La négociation se termine sur un succès.

2 Selon le principe des enchères anglaises, la règle du *"beat-the-quote"* impose que toute nouvelle proposition soit meilleure que la meilleure proposition reçue jusqu'alors.

4.4.1. Stratégie de l'agent acheteur

La stratégie d'un tel agent lui permettre de raisonner sur le problème et d'inférer leurs décisions en fonction de la connaissance obtenue des autres agents négociateurs, elle lui permet de décider de confirmer ou d'annuler le contrat et de synthétiser les différentes propositions de modifications des vendeurs afin de proposer un nouveau contrat. élaborer une bonne stratégie nécessite en effet une expertise du domaine que nous pouvons fournir par notre approche de simulation participative.

La stratégie de l'agent acheteur lui permet à chaque tour de l'enchère de sélectionner une action en fonction des message reçus des vendeurs et suivant le protocole des enchères anglaises inversées [Oliveira et al 99], [Bellosta et al 04]. Le processus d'enchère prend en compte trois paramètres (le groupe *G* de *n* vendeurs, la durée d'un tour de l'enchère *Timer*, et le nombre de tours pour l'enchère *nbreTour*s):

```
Début
    CallForPropose(acheteur,n agents vendeurs, Préférences); // envoie une proposition de contrat à n vendeurs
   1: TQ  NotTimer Faire
          Attend les réponses;
     FTQ;
    Si (receive q accords) et (q>=minAccords) Alors
      Accept(acheteur,q); //envoie la confirmation du contrat
     Succès; goto 3;
    FSi;
    Si (receive q accords) et (q<minAccords) Alors
     Si nbreTours>=maxTours Alors
       //envoie l'annulation de contrat;
       échec; goto 3;
     Sinon
       RequestForPropose(acheteur,n agents vendeurs, counterproposal);
       //envoie une demande de  modification aux vendeurs
      2: TQ NotTimer Faire
            Attend les modification;
         FTQ;
       Calcul Les nouvelles possibilités;
       Si nouvellePossibilité Alors  //envoie une nouvelle proposition de contrat aux n vendeurs;
          goto 1;
       Sinon
          Si nbreTours<maxTours Alors  //envoie une demande de  modification aux n vendeurs;
             goto 2;
          Sinon  //envoie l'annulation de contrat;
            goto 3;
          FSi;
       FSi;
    FSi;
    3: Fin de l'enchère;
FIN.
```

Figure 5.1. L'Algorithme qui détermine la stratégie de l'agent Acheteur.

4.4.2. Stratégie de l'agent vendeur

La stratégie de l'agent vendeur lui permet à chaque tour de l'enchère de sélectionner une action en fonction des messages reçus de l'agent acheteur *(envoyer une proposition, abandonner les enchères, …etc.)*, il l'utilise pour décider d'accepter ou de refuser la proposition de contrat et de trouver une modification pour le contrat lorsque l'agent acheteur en demande une.

```
Début
    // reçoit une proposition de contrat;
   1: Evaluer le contrat;
   Si Accepte Alors
      attend la décision de l'acheteur;
      Si reçoit la confirmation du contrat Alors   Succès; goto  2;
      Si reçoit l'annulation du contrat Alors      échec; goto  2;
      Si reçoit une demande de modification  Alors
         4: chercher une modification à envoyer;
            envoie  une modification;
            attend la décision de l'acheteur;
            Si reçoit une demande de modification  Alors      goto 4;
            Si reçoit une nouvelle proposition de contrat  Alors   goto 1;
            Si reçoit l'annulation du contrat  Alors    échec; goto  2;
      FSi;
   Sinon  // refuse
       attend la décision de l'acheteur;
       Si reçoit une demande de modification Alors  goto  4;
       Si reçoit l'annulation du contrat  Alors   échec;  goto 2;
   FSi;
   2: Fin de l'enchère;
FIN.
```

Figure 5.2. L'Algorithme qui détermine la stratégie de l'agent Vendeur.

4.5. Besoin de simulation et de participation

Comme nous l'avons indiqué précédemment, élaborer une bonne stratégie pour un tel agent négociateur nécessite en effet une expertise du domaine que nous pouvons fournir par notre approche de simulation participative.

La conception d'une plateforme multi-agent de modélisation et de simulation de la négociation dans le domaine de commerce électronique qui a mis la difficulté de formalisation des comportements acquis en pratique par les acteurs humains. Donc généralement, cette problématique se pose quand on cherche à simuler le travail compétitif des acteurs impliqués dans un domaine spécialisé comme le commerce électronique et plus précisément les protocoles de négociation (La grande difficulté c'est au niveau de l'automatisation de la négociation).

Dans notre modèle, des agents dotés de capacités d'apprentissage et d'une certaine forme d'autonomie décisionnelle seront utiles pour automatiser ce processus d'extraction de connaissances d'une part, et pour automatiser le processus de négociation d'autre part. L'expert joue, par le biais

d'une interface dédiée, le rôle qui lui a été confié au sein de la simulation pour contrôler nos agents et développer leur base comportementale, l'agent observe, puis propose des comportements qui peuvent être améliorés par l'expert. Cela revient à mettre en oeuvre un système d'*apprentissage participatif*.

Grâce à cette technique des participants humains pourront accroître l'adaptation à des nouvelles configurations d'environnement et les champs d'action, rendant plus réalistes les simulations. Leurs connaissances évolueront dans le temps, les agents devront être capables d'apprendre et de remettre en question leur base de connaissances.

5. Modélisation multi-agent participative et comportements d'agents

Nous modélisons sous forme d'agents artificiels les acteurs humains qui participent au problème de négociation, avec *leurs comportements, leurs procédures et stratégies de décision* et *leurs interactions*. Deux types d'agent principaux sont prévus dans cette modélisation: *l'agent acheteur* et *l'agent vendeur*.

Nous partons du principe qu'un agent modélisant un être humain, il doit déduire les *actions* qu'il doit effectuer de ses *intentions*. Le modèle que nous utilisons pour mettre en application notre simulateur est un modèle cognitif qui établit les protocoles de communication et de négociation principaux entre les agents. Aussi nous avons ajouté quelques possibilités réactives aux agents, c.-à-d. de définir un *modèle comportemental hybride*. Dans ce modèle hybride, les composants clés sont les suivants (cf. figure 5.3):

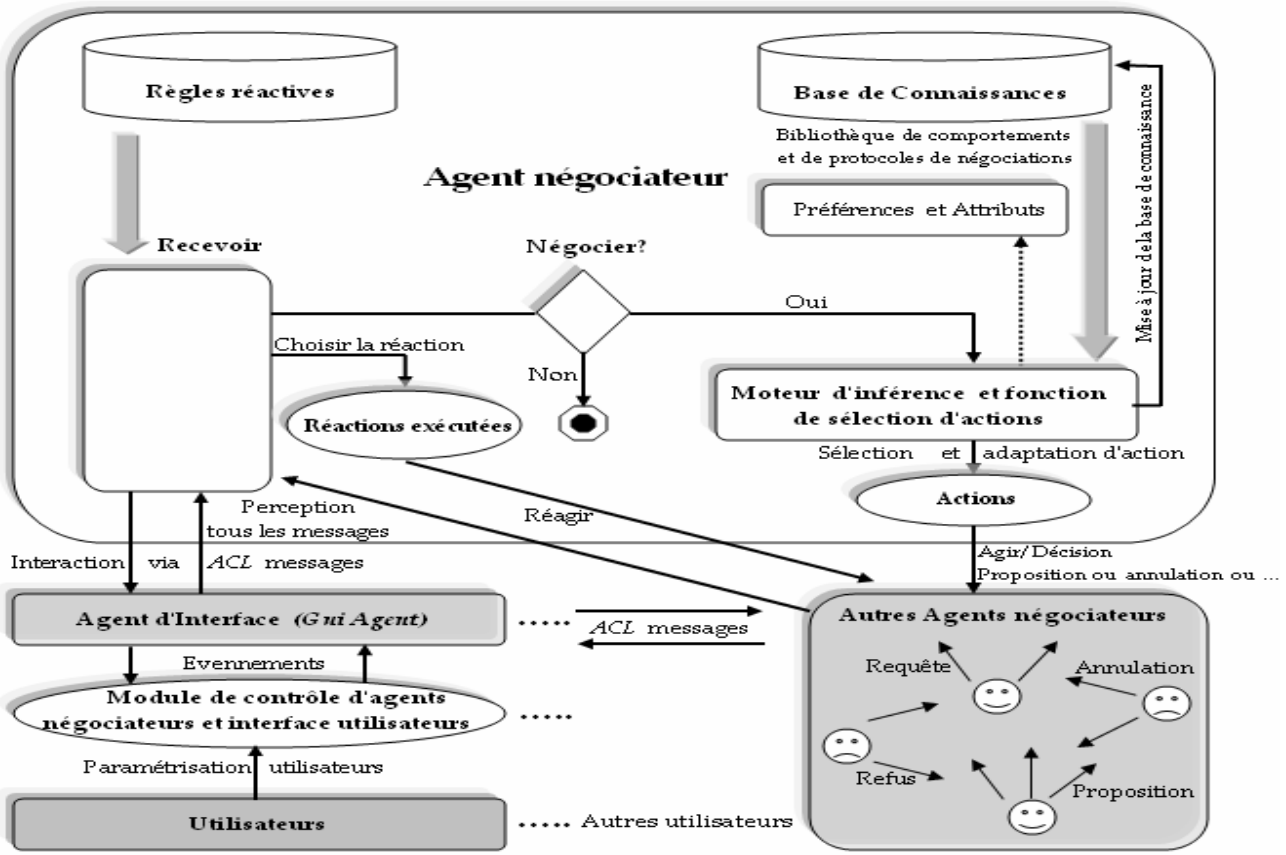


Figure 5.3. Architecture générale du système et boucle de décision d'un agent.

Nous pouvons y trouver, comme présenté par la figure 5.3, le mécanisme de prise de décision d'un agent négociateur, qui produit des *actions* et *réactions* à partir des événements (émis par des utilisateurs à l'aides des agents d'interfaces) et des messages envoyés par les autres agents négociateurs. Parmi les éléments de base, l'*action* ou *réaction* est implémentée comme un changement d'état d'une entité de simulation, par exemple lorsqu'un agent acheteur envoie un message portant une proposition de contrat à un agent vendeur. Les interactions se font à travers des *ACL* messages générés par la plate forme *Jade*. Les *croyances* et *règles de déduction* sont stockées dans les bases de connaissances on utilisant l'outil Jess. Il existe aussi des *règles d'action* réactives codées à la main.

• ***Module de contrôle d'agents négociateurs et interface utilisateurs:*** Ce module nous a permet de définir les entités du système à modéliser, que l'on appelle des agents négociateurs, et leurs interactions, ces interaction s'expriment par des méthodes de communication (envoi de messages), il nous a permet aussi un contrôle de la dynamique globale (ordonnancement des différents évènements d'un pas de temps du modèle).

• ***Agent d'interface :*** le rôle de l'agent d'interface est d'assister un utilisateur dans le processus de modélisation. Il lui offre une interface graphique *(GUI)* composée de vues représentant différents aspects du système à modéliser. Elle permet à l'utilisateur de suivre le comportement du système modélisé par la représentation des différents agents du système, les interactions entre eux, l'historique de simulation, etc. ainsi que la configuration du système par la création, configuration des agents, etc. L'agent d'interface gère les désirs et préférences de l'utilisateur. En effet, il lui permet de modifier les croyances des agents, ainsi leurs décisions, leur demander des informations qui lui semblent significatives et de répondre à leurs requêtes. Les tâches de cet agent sont donc limitées au simple envoie et réception des messages. Par conséquent, l'architecture de cet agent ne sera pas introduite dans les sections suivantes.

• ***Recevoir:*** Un agent acquiert les informations en recevant des messages des autres agents dont la plupart sont des messages pour le processus de négociation, ou des événements des utilisateurs pour mettre en œuvre un contrôle sur l'agent négociateur on utilisant des agents d'interfaces.

• ***Choisir la réaction:*** L'analyse des systèmes de négociation fait apparaître que certaines stratégies n'a nul besoin d'intelligence poussée. Seule importe sa réactivité face à des événements parfaitement connus et, donc, modélisables. Ainsi, l'utilisation des agents réactifs de bas niveau s'avère suffisante. Quand une information quelconque envoyée par d'autres agents, décrite en tant que stimulus, est perçue par un agent, ce dernier peut choisir de réagir pour répondre à ce stimulus selon une liste de *réactions* prédéfinies *(règles réactives).* Ce choix se base sur une correspondance directe entre *l'information perçue* et la *réaction* à effectuer, elles peuvent être considérées comme l'implémentation d'heuristiques spécifiques au enchères anglaises multi-attributs.

• ***Réagir:*** Si une *réaction* est choisie pour répondre à un stimulus, un agent réagit en effectuant les tâches associées, qui peuvent consister à envoyer des messages à d'autres agents dans le processus de négociation.

• ***Base de connaissances:*** La base de connaissances se compose d'une base de "*faits*" et une base de *règles*. Les *règles d'action* sont représentées sous forme de: $\wedge_i$ Fait$_i$ ==> Action$_n$, et les *règles de mise à jour*

de connaissances se représentent comme: $\wedge_i$ Fait$_i$ ==> Fait$_n$, La base de *faits* est constituée par les quatre types suivants [Nguyen-Duc 05]:

- ✓ *Fait perçu:* Un *fait* qui est perçu en provenance d'un autre agent.
- ✓ *Croyance:* Un *fait* qui n'est pas perçu, mais qui est déduit à partir des *faits perçus* ou d'autres *croyances*.
- ✓ *Intention:* Un *fait* qui, en se combinant avec quelques *faits*, permet de déduire d'autres *intentions* et *actions*. Les *intentions* peuvent être individuelles ou collectives. Notons qu'une *action* est toujours associée à une *intention* mais que l'inverse n'est pas vrai.
- ✓ *Rôle:* Un *fait* qui, en se combinant avec une *intention commune*, aide à déduire des *intentions individuelles*.

• ***Mettre à jour la base de connaissances:*** Quand un agent perçoit une information, il met à jour sa base de connaissances en prenant en compte cette *information perçue* qui peut être un *fait* simple ou une *règle d'action*. Dans le cas du *fait* simple, cette mise à jour est un genre spécial de *raisonnement* dont le résultat n'est pas une *action* à effectuer mais de nouveaux *faits* à ajouter dans la base.

• ***Raisonner (sélection d'action):*** Le résultat d'un processus de *raisonnement* est une *action* à effectuer. Un tel processus se déclenche quand un agent découvre de nouveaux *faits* ajoutés dans sa base de connaissances.

• ***Agir:*** Si en raison d'un processus de *raisonnement* une *action* est choisie, l'agent l'effectuera. Exceptées quelques *action*s particulières.

Chaque agent négociateur interagit avec la stratégie qui lui a été attribuée. À chaque arrivée de message concernant la négociation de son contrat, l'agent informe la stratégie qui prend en compte le message. Celle-ci indique par la suite aux agents négociateurs le message qu'elle souhaite envoyer en réponse. Prenons l'exemple d'une proposition de contrat arrivant à un agent négociateur engagement, celui-ci envoie la proposition à la stratégie qui lui demande ensuite d'envoyer une acceptation ou un refus.

5.1. Modèle d'un agent négociateur

Dans un processus de négociation des procédures et stratégies de décision peuvent être employer. Ces procédures et stratégies sont, le plus souvent, des tâches très complexes et font appel à des compétences diverses. A ce niveau, les agents doivent donc avoir des connaissances relatives à leur environnement et une grande capacité de raisonnement et tenir compte de leurs croyances mutuelles. Ainsi, l'agent doit disposer de leur propre modèle (croyances, buts, etc.) et de ses accointances (agents avec lesquels il est sensé communiqué). D'après ces critères, l'agent est suffisamment complexe pour s'approcher du modèle d'agents cognitifs. Notre choix s'est porté donc sur des agents intelligents de haut niveau complexe.

Afin de satisfaire la description des agents, donnée plus haut, nous proposons une architecture caractérisée par des représentations procédurales, un raisonnement à base d'utilité autour des buts multiples et un comportement orienté but (proactif) et orienté événement (réactif). L'architecture d'agent retenue possède cinq composantes : état intellectuel, communication, expertise, apprentissage

participatif, et décision. La figure 5.4 illustre cette architecture. Les flux de données et les flux de contrôle montrent comment ces composantes interagissent pour dynamiser, automatiser, et paramétrer le comportement de l'agent négociateur.

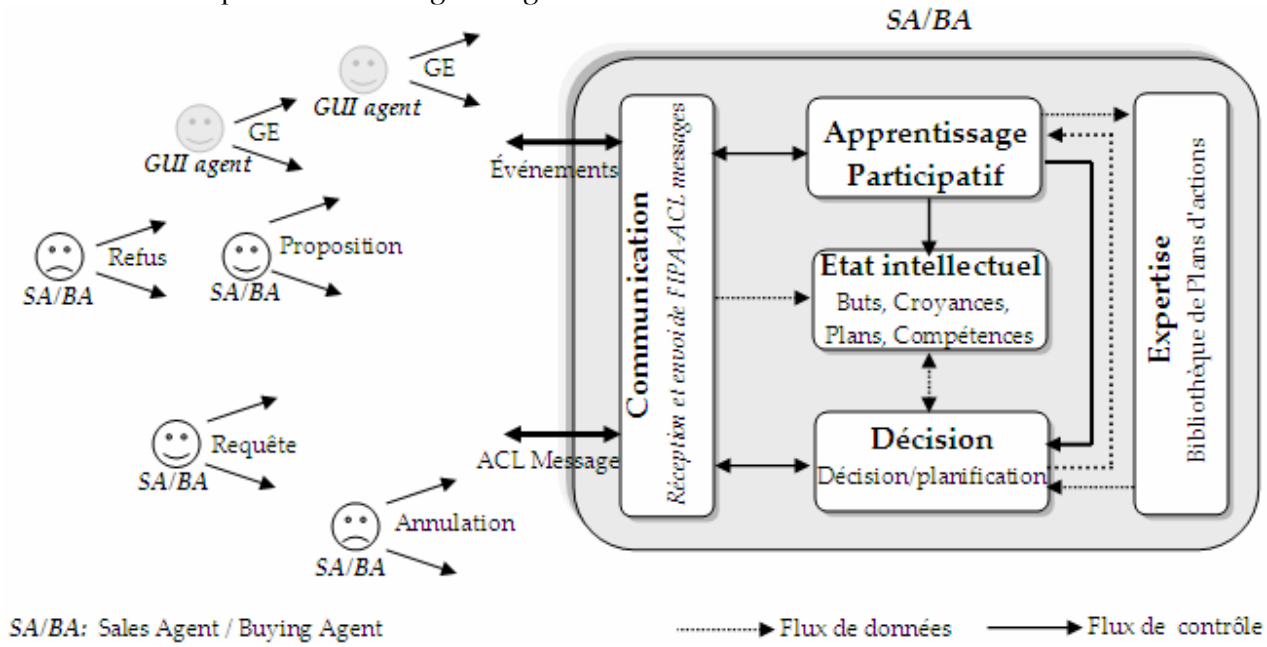


Figure 5.4. Architecture d'agent négociateur proposée.

• **Etat intellectuel :** contient toutes les informations détenues par l'agent et qui interviennent dans son raisonnement. Ces informations désignent toute la partie dynamique des différentes représentations et connaissances de l'agent sur la négociation. Ils sont structurés en *buts*, *croyances*, *compétences*, et *plans*. Les croyances sont les connaissances supposées vrais sur l'agent lui-même ou sur les autres, décrivant ce qu'un agent fait. Elles désignent l'ensemble des informations sur le domaine d'application de l'agent ainsi que les informations sur les autres agents négociateur. Les compétences dépendent du rôle de l'agent et décrivent ce que l'agent sait faire. Elles sont les tâches de l'agent augmentées de conditions d'utilisation permettant de modifier ses propres croyances ou de conduire à des interactions avec les autres agents afin de modifier les leurs. Les buts représentent la liste des objectifs à satisfaire (adoptés) par l'agent en utilisant ces compétences. Cette liste peut contenir aussi des objectifs nécessaires aux actions que l'agent promet à un autre agent de réaliser (engagements sociaux). Enfin, les plans représentent la liste des plans choisies a fin d'atteindre les objectifs courants ou en cours de réalisation (i.e. les intentions de l'agent).

• **Expertise :** représente la connaissance de l'agent sur la résolution de problème. Alors que les compétences dans l'état intellectuel décrivent ce que sait faire l'agent, l'expertise détaille le comment de ces compétences. En effet, cette composante décrit les différentes tâches réalisables par l'agent sous forme d'une bibliothèque de plans d'actions. Un plan caractérise une tâche bien définie et spécifie une séquence d'actions tels que la mise à jour de l'état intellectuel, les calculs et l'envoi de message, etc. Notons que les protocoles d'interactions et de négociations sont représentés par des plans dans l'expertise et ne sont pas définis directement dans le module de communication.

• **Communication :** est le mécanisme permettant aux agents d'échanger des informations et des résultats nécessaires à la réalisation de leurs tâches. Ce mécanisme gère l'envoi et la réception des messages vers et en provenance des autres agents. La gestion concerne essentiellement l'expression et l'interprétation des messages. Le langage de communication *FIPA-ACL* est utilisé ici comme langage d'expression des messages. Le mécanisme de communication intègre deux interfaces de communication, chacune d'elles gère l'envoie et la réception d'un type de messages (messages *FIPA-ACL* pour la négociation et pour la paramétrisation). L'emploie d'un langage de communication conforme permet aux agents négociateurs d'échanger et de comprendre les actes communicatifs *FIPA-ACL (CallForPropose, propose, requestForPropose, Accept, reject, etc.)* d'une manière flexible et fiable.

• **Décision :** représente le noyau de l'agent. Ce module assure l'application des connaissances pour mettre à jours l'état intellectuel. Il consiste en un ensemble de fonctions permettant de contrôler l'exécution des actions au niveau de l'expertise. Suite à la détection d'un événement ou à la réception d'un message, une décision est prise par la fonction de décision. Cette décision porte sur le traitement des informations reçues et la mise à jour des différents éléments de l'état intellectuel (croyances, engagements, etc.). Dans certains cas, ce traitement nécessite l'adoption de nouveaux buts par l'agent. La fonction de décision gère ainsi la création et la suppression des buts dans l'état intellectuel (liste de buts). A la création d'un but, la fonction de planification élabore un plan d'exécution (plan d'actions) relatif à ce but et l'ajoute dans la liste des plans. Pour ce faire, elle choisit un plan parmi les plans applicables de la bibliothèque de plans. La fonction de planification se charge également de l'exécution des plans crées.

• **Apprentissage participatif :** Ce module assure l'application de l'approche de simulation participative pour mettre à jour l'état intellectuel, ainsi l'expertise de l'agent négociateur. Il consiste en un ensemble de fonctions permettant de contrôler l'exécution des actions d'un agent négociateur par un utilisateur pour améliorer et adapter le comportement de nos agents à des nouvelles configurations du système,

5.2. Caractéristiques générales de notre outil de simulation

Nous nous basons sur ce que [Drogoul et al 03] et [Nguyen-Duc 05] utilisent pour caractériser un outil de simulation typique qui permet l'utilisation des JdR afin d'affiner les comportements des agents:

Interface homme-machine avec assistant

Son but est de faciliter l'intégrité des experts et les apprentis dans le simulateur. Elle fournit ainsi une perspective sur la façon dont les agents prennent leurs décisions. Chaque interface dédiée à un expert/apprenti est attachée à un agent qui peut assister seul cet expert/apprenti. Les agents d'interface ou assistants sont dotés de la capacité d'expliquer les raisons de leurs décisions basée sur un mécanisme d'interaction humain-agent efficace, enrichiront énormément l'apprentissage de l'agent.

Interactions utilisateurs (experts/apprentis) et agents

Dans notre simulateur, un expert peut participer a une simulation premièrement pour comprendre les mécanismes de décision des agents, deuxièmement pour valider ce mécanisme et finalement pour l'améliorer. Il observe les actions exécutées par d'autres utilisateurs. Il peut demander à l'assistant la raison d'une action à effectuer ou valider une autre action proposée par un agent. Cette relation expert/agent mène à un genre d'apprentissage interactif: l'agent est capable d'apprendre (ou d'améliorer) ses comportements en interagissant avec l'expert (*la modélisation de l'expertise humaine*). Pour les apprentis, ils peuvent participer a un joue de simulation seulement pour comprendre les mécanismes de décision des agents, et le fonctionnement globale de système (*l'approche de simulation participative dans ce cas sert à l'enseignement et l'apprentissage, l'utilisateur qui n'est donc dans ce cas qu'un apprenant, sera bien convaincu de la façon dont le système réel se comporte*).

Amélioration des comportements des agents

La participation des êtres humains à des jeux de simulation à pour but d'enrichir les connaissances de nos agents et donc leurs champs d'actions dans leurs environnement. Il réagiront plus justement aux évènements qui surviendront et cela grâce à son apprentissage assisté pour les rendront plus autonomes et diversifiés. Dans des situations inconnues, si le cas se présente, ils questionneront le joueur humain, ou attendront son aide, afin de réagir le mieux possible. Il est donc logique d'observer une amélioration de la fiabilité et de la richesse de la simulation grâce à cet *apprentissage participatif.*

Un processus composé de deux étapes est envisagé pour y parvenir [Nguyen-Duc 05]:

• L'agent est d'abord conçu pour pouvoir construire un plan d'actions validées/modifiées pendant des sessions d'interaction avec l'expert. Le plan formé doit être structuré d'une telle manière que le tuteur peut l'employer pour corriger les comportements d'agent;
• Le plan d'actions est restructuré de sorte que tout agent puisse l'employer pour apprendre ses comportements (les agents apprennent hors-ligne leurs comportements à partir des actions validées/modifiées conservées dans le plan. Des techniques d'apprentissage appropriées doivent être encore étudiées avant une vraie implémentation; et les techniques d'apprentissage sont appliquées en mode en ligne, c.-à-d. pendant des interactions expert/agent).

5.3. Modèle d'un simulateur multi-agent participatif

Tout d'abord, notre système doit être un simulateur participatif, dans lequel on doit pouvoir le contrôler en modifiant certains paramètres. Nous avons proposé un schéma de simulation dans notre modèle sur lequel un logiciel distribué est lancé sur plusieurs machines dont chacune est utilisée par un utilisateur. Les propriétés de ce type de simulateur participatif vont être présentées dans le chapitre suivant.

5.3.1. L'interface utilisateur

Le but d'une interface est de fournir à un participant tous les moyens pour participer à la simulation. La figure 6.9 présente l'interface de notre simulateur, qui intègre des experts de domaine

ou des apprentis à la boucle d'exécution. Chaque utilisateur utilise cette interface pour participer à certaines procédures de collaboration.

5.3.2. Distribution de l'interface utilisateur

Une des parties essentielles d'un simulateur distribué se situe dans son architecture logicielle (architecture réseau de simulation), et plus précisément dans la façon dont les différents composants logiciels, distribués sur les différentes machines, communiquent et se transmettent les données. La figure5.5 montre le schéma architectural de notre système. On peut voir sur chaque machine dédiée à un utilisateur un composant d'interface utilisateur installée, qui caractérise ce type de simulateur distribué.

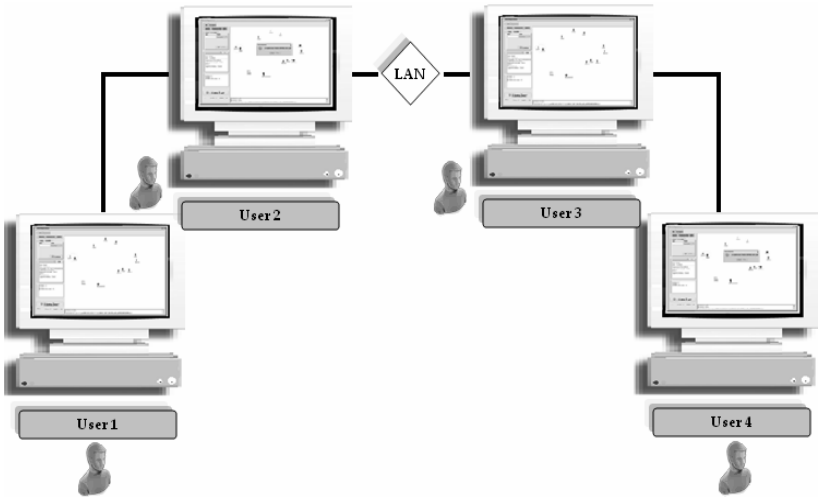


Figure 5.5. Architecture réseau de notre simulateur.

5.3.3. Sauvegarde des traces numériques

Il est nécessaire pour la conception participative de comportements d'agents dans une expérimentation informatique, de sauvegarder des traces numériques. pour permettre d'analyser la simulation, ou pour essayer de suivre les traces de la dernier simulation. Pour cela il faut mettre en œuvre des agents enregistreurs qui enregistrent l'historique des actions des utilisateurs, ainsi que leurs profils, et les différentes interactions entre les agents impliquer dans le processus de négociation, dans notre modèle c'est le rôle de l'agent d'interface, qui mémorise des négociations passées.

5.3.4. Utilisation des agents assistants (Ghost agents)

Les agents assistants jouent un rôle très important dans notre approche. Chaque assistant est attaché à une interface utilisateur dédiée à un expert, son rôle effectif est de montrer à l'utilisateur correspondant comment travailler avec le simulateur, d'observer les actions effectuées par l'utilisateur et de proposer des actions à effectuer. Ainsi, nous caractérisons l'utilisation d'un assistant en fonction du rôle que l'on souhaite lui affecter pour des experts de domaine ou des apprentis, la figure 6.9.7 représente une proposition d'action à un utilisateur.

5.3.5. Utilisation d'un module pour la paramétrisation de simulateur

Dans l'approche participative on doit pouvoir contrôler le simulateur en modifiant certains paramètres du modèle, pour cela il nous faut un module de paramétrisation, qui nous a permet un contrôle de la dynamique globale (ordonnancement des différents évènements d'un pas de temps du modèle).

6. Expérience de simulation

Dans toutes les expériences le modèle était complètement abstrait, sans employer n'importe quelles données réelles. Ainsi, aucune conclusion quantitative ne peut être prise par les résultats. Cependant, si on change les paramètres du modèle, on peut observer des propriétés de comportement du modèle intéressantes, et poser beaucoup question genre *"que ce qui ce passe si ?"* a partir de l'expériences.

Notre simulation commence par une initialisation nul, càd aucun groupe d'agents vendeurs à une relation avec aucun agents acheteur, donc on observe le comportement du modèle pour un plus grand temps de simulation. Le modèle est plutôt abstrait, ainsi l'unité de temps n'est pas définie. Dans notre modèle l'intervalle de temps pour les mouvements (le mouvement ici correspond au changement des propositions par les agents négociateurs) d'agent égal à 7 Unités de temps.

Les groupes d'agents (un agents acheteur et un ensemble d'agents vendeurs) sont créées dés la création de la structure des événements dans le processus d'agent. Chaque agent acheteur recherchera de manière permanente des autres agents vendeurs qui peuvent être impliquer dans le processus d'enchère. Dans notre modèle l'agent déjà impliqué dans un processus d'enchère ne peut pas être impliqué dans un autre, donc chaque groupe est représenté sous forme d'une arbre, où le nœud correspond à l'agent acheteur et les feuilles aux agents vendeurs, ainsi nous pouvant avoir dans notre simulateur plusieurs salles d'enchères (de cette façon que les structures se développent. Comme mentionné avant, nous pouvons avoir plusieurs salles d'enchères).

Le modèle entier est contrôlé par un certain nombre de paramètres. Une partie est fixée dans le code, et l'autre est introduit par les utilisateurs avant ou pendant le lancement de la simulation. Ces paramètres définissent le comportement du modèle. Finalement, une fois le modèle défini, on peut attribuer des valeurs aux paramètres et effectuer les simulations. Chaque simulation est caractérisée par un état initial et une succession de transformations et d'actions faites par les agents négociateurs. Au premier lieu on impliquant des experts de domaine pour améliorer et adapter le comportement de nos agents, En suite on peut tester et valider les performances de notre système à des nouvelles configurations on impliquant des utilisateurs pas forcement des experts.

7. Résultats et Conclusion

Dans ce chapitre, nous avons proposé une architecture multi-agents basée sur une simulation multi-agent participative qui est dédiée au développement d'un système multi-agent, d'une part pour automatiser le processus de négociation entre les acheteurs et les vendeurs au sein d'un environnement de commerce électronique, et d'autre part pour simuler les nouvelles procédures collaboratives des acteurs impliqués dans le processus de négociation. Cependant, elle possède deux

caractéristiques intéressantes, en ce qu'elle permet la participation des experts de domaine (professionnels du domaine) ou des apprentis à sa boucle d'exécution, et qu'elle utilise des agents assistants qui aident les participants à utiliser l'interface homme-machine. En profitant de ces caractéristiques, nous présentons, dans le chapitre suivant, comment nous avons développé notre outil de simulation qui est basé sur la norme *FIPA* pour le développement des *SMA* afin d'assurer l'interopérabilité avec les systèmes respectant cette norme. La réalisation effective de cette architecture constitue une description plus technique de son application. Pour cela, la plateforme *JADE*, qui est une plateforme multi-agents compatible *FIPA*, sera employé dans le prochain chapitre comme un environnement d'implémentation et d'exécution des agents.

Chapitre 6
Implémentation du Modèle

1. Introduction

Dans le précédent chapitre de ce mémoire, nous avons présenté notre modèle qui est basé sur une approche de simulation participative basée agent pour le protocole de négociation dans le commerce électronique. Dans le présent chapitre, nous présentons la mise en oeuvre du ce modèle proposé pour une application des simulations participatoires a base des systèmes multi-agents comme nouveau genre d'expériences participatoires combinant des agents et la participation d'un certain nombre d'utilisateurs, en utilisant comme exemple les protocoles d'enchères qui offrent les mécanismes les plus connus du grand public et les plus compétitifs pour la négociation dans le e-commerce. Parmi ces protocoles d'enchères on trouve également les enchères multicritères qui permettent de négocier sur d'autres critères tels que la qualité et le délai de livraison, …etc. Dans notre modèle nous avons proposer un protocole d'enchères anglaises inversées multi-attributs. Après la définition de l'application choisis, nous présentons les choix techniques retenus afin de l'implémenter ensuite. Nous justifions ces choix lorsque cela s'avère nécessaire et nous présentons également des alternatives de développement que nous avons envisagées, mais que nous n'avons pas pris en compte, afin de converger vers un environnement adapté à la complexité de notre modèle.

2. Choix techniques effectues

Tout au long des développements, nous avons veillé à utiliser le plus de librairies ou d'outils standard et librement distribuables afin de faciliter le développement, l'extension et la diffusion de notre application.

2.1. Langage d'implémentation

Notre but est de proposer un modèle pour un simulateur multi-agents participatif. Ce modèle à été implémenté en langage Java. Donc le premier choix que nous avons dû effectuer fut celui du langage de programmation, nous avons choisi le langage Java qui est à la fois un langage de programmation et une plateforme d'exécution. Le langage Java a la particularité principale d'être portable sur plusieurs systèmes d'exploitation tels que *Windows*, *MacOS* ou *Linux*. C'est la plateforme qui garantit la portabilité des applications développées en Java. Aussi Java permet de développer des applications autonomes mais aussi, et surtout, des applications client-serveur.

Donc les applications Java peuvent être exécutées sur tous les systèmes d'exploitation pour lesquels a été développée une plateforme Java, dont le nom technique est *JRE* (*Java Runtime Environment*). Cette dernière est constituée d'une *JVM* (*Java Virtual Machine*), le programme qui interprète le code Java et le convertit en code natif. Mais le *JRE* est surtout constitué d'une bibliothèque standard à partir de laquelle doivent être développés tous les programmes en Java. C'est la garantie de portabilité qui a fait la réussite de Java dans les architectures client-serveur en facilitant la migration entre serveurs, très difficile pour les gros systèmes. Le langage Java est très adapté pour la réalisation d'applications structurelles, nécessitant une approche objet comme les applications graphiques ou réseaux. Ce langage est aussi très prisé pour le développement d'applications du domaine de l'intelligence artificielle pour la représentation de neurones par exemple. Il existe d'ailleurs énormément de projets accessibles sur Internet traitant de ce domaine.

Les créateurs de Java ont écrit un livre blanc qui présent les caractéristiques fondamentales de Java. Ce livre est articulé autour des 10 termes suivants :

• **Distribué** Java possède une importante bibliothèque de routines permettant de gérer les protocoles *TCP/IP* tels que *HTTP* et *FTP*. Les applications Java peuvent charger et accéder sur Internet via des URL avec la même facilité qu'elles accèdent à un fichier local sur le système.

• **Fiabilité** Java a été conçu pour que les programmes qui l'utilisent soient fiables sous différents aspects. Sa conception encourage le programmeur à traquer préventivement les éventuels problèmes, à lancer des vérifications dynamiques en cours d'exécution et à éliminer les situations génératrices d'erreurs...etc.

• **Orienté objet** Pour rester simples, disons que la conception orientée objet est une technique de programmation qui se concentre sur les données (les objets) et sur les interfaces avec ces objets.

• **Simple** Nous avoirs voulu créer un système qui puisse être programmé simplement sans nécessiter un apprentissage ésotérique, et qui tire parti de l'expérience standard actuelle. En conséquence.

• **Sécurité** Java a été conçu pour être exploité dans des environnements serveur et distribués. Dans ce but, la sécurité n'a pas été négligée. Java permet la construction de systèmes inaltérables et sans virus.

• **Architecture neutre** Le compilateur génère un format de fichier objet dont l'architecture est neutre – le code compilé est exécutable sur de nombreux processeurs, à partir du moment où le système d'exécution de Java est présent. Pour ce faire, le compilateur Java génère des instructions en bytecode qui n'ont de lien avec aucune architecture particulière. Au contraire, ces instructions ont été conçues pour être à la fois faciles à interpréter et faciles à traduire en code natif.

• **Portable** A la différence du *C/C++*, on ne trouve pas les aspects de dépendance de la mise ne oeuvre dans la spécification. Les tailles des types de données primaires sont spécifiées, ainsi que le comportement arithmétique qui leur est applicable.

• **Interprété** L'interpréteur *Java* peut exécuter les bytecode directement sur n'importe quelle machine sur laquelle il a été porté. Dans la mesure où la liaison est un processus plus incrémentiel et léger, le processus de développement peut se révéler plus rapide et exploratoire.

• **Performances élevées** En général, les performances des bytecodes interprétés sont tout à fait suffisantes, il existe toutefois des situations dans lesquelles des performances plus élevées sont nécessaires. Les bytecodes peuvent être traduits à la volée en code machine pour l'unité centrale destinée à accueillir l'application.

• **Multithread** Les avantages du multithread sont une meilleure interréactivité et un meilleur comportement en temps réel.

Donc le principal avantage du Java comme nous l'avons dit précédemment est qu'il permet de créer des applications utilisables par tous les systèmes d'exploitations existants, et ce sans regénérer son code : Java fonctionne sur une machine virtuelle (programme qui réalise le traitement du code). Il est donc très facile de développer une application et de la distribuer. Nous avons choisi le langage Java pour l'ensemble des raisons suivantes [Clavel et al 00] :

- Langage orienté objets ;
- La richesse des bibliothèques de base (l'*API Java*) ;
- L'indépendance vis-à-vis des plateformes d'exécution (Portabilité excellente) ;

- La richesse et la disponibilité de projets tiers (en particulier, les plateformes agent, les API, etc.) ;
- La disponibilité de la plateforme du téléphone portable *(J2ME)* aux serveurs *(J2EE);*
- Langage puissant;
- *JDK* très riche;
- Langage de haut niveau;
- Enfin, notre application doit être au final intégré à *JADE* qui est programmé en *Java,* en plus on doit utiliser l'outil *JESS* qui nous permettant de géré la stratégie de nos agents, et qui est a leur tour de rôle intégré avec *JADE* à l'environnement de programmation.

2.2. Plateforme agent

Afin d'implémenter les agents de notre modèle, nous avons fait le choix de la plateforme *JADE* présenté dans le deuxième chapitre. les implémentations de protocole de négociation et de communication permettent d'utiliser cette plate forme multi-agents, et d'utiliser l'outil *Jess* pour l'implémentation de la stratégie des agents de notre système. *JADE* a pour but de simplifier le développement des systèmes multi-agents tout en fournissant un ensemble complet de services et d'agents conformes aux spécifications *FIPA*, il répond mieux à nos attentes, car il a des caractéristiques que nous jugeons importantes :

- Il prend en compte les spécifications de la *FIPA* pour l'interopérabilité des systèmes multi agents;
- Il est totalement neutre vis à vis la définition d'un agent et ne traite que des capacités bas niveau permettant la mise en oeuvre de communication et d'interaction entre les agents ;
- Il est entièrement écrit en *Java* et propose une large gamme de bibliothèques de classes implémentant les fonctionnalités de base des agents *(identification, comportements, communication, protocoles d'interaction, ontologies, mobilité, etc.)* ;
- Il offre une interface graphique permettant la gestion des agents et facilitant leur déboguage ;
- Il inclut un environnement d'exécution distribué dans lequel des agents peuvent s'exécuter;
- Il inclut une bibliothèque de classe permettant le développement d'agents;
- Il permet l'intégration efficace du moteur d'inférence *Jess;*
- Il a la bonne documentation avec une liste d'expédition (mailing list) très actif ;
- Il est libre, et même open source (i.e. le code source peut être facilement réutilisé pour construire de nouveaux systèmes) ;
- Enfin, il est largement répandu pour le développement orienté agent, et il a été employé avec succès dans les différents milieux.

2.3. L'outil *JESS*

JESS[1] *(Java Expert System Shell)* est un programme permettant la manipulation de systèmes experts. Il est écrit en java de Sun par Ernest Friedman-Hill à Sandia National Laboratories à Livermore, CA, il possède de nombreuses connexions avec le langage java (utilisation d'objets java dans les règles dans un sens, utilisation de systèmes experts dans des programmes java dans l'autre sens).

[1] Le programme, sous forme de fichier exécutable jar, peut se charger sur http://herzberg.ca.sandia.gov/jess/ La documentation en ligne se trouve à l'adresse http://herzberg.ca.sandia.gov/jess/docs/70/ ou encore sur http://herzberg.ca.sandia.gov/jess/manual.pdf Décompressez le fichier, puis lancez le programme avec : java -classpath jess.jar jess.Main ou java -classpath jess.jar jess.Main nom de fichier.clp.

Dans notre simulateur nous avons développé des class d'agents négociateurs avec la plate forme multi-agents *JADE* qui offre une classe *JessBehaviour* qui permet l'intégration avec le moteur d'inférence *JESS. JADE* fournit le noyau de l'agent et garantit la conformité avec les normes *FIPA*, alors que *JESS*[1] est le moteur d'inférence de l'agent qui exécute le raisonnement nécessaire pour la résolution du problème en utilisant une base de connaissance interne qui est déterminée dans l'architecture générale du système dans le chapitre précédant.

2.4. L'environnement de programmation

Pour exécuter notre application, nous avons besoin de l'environnement *JBuilder7* avec l'intégration de la plate-forme multi-agent *JADE 3.3* (il existe une nouvelle version de *JADE 3.4* Mai 2006), Ainsi nous avons utilisé le kit de développement Java *(JDK 1.4.1)* qui dispose de tous les composants fournis par ces environnements (cf. figure 6.1). *JBuilder7* propose des améliorations majeures pour la productivité du développeur, ainsi qu'une interface utilisateur plus claire, plus intuitive et des améliorations remarquables des performances. L'environnement *JBuilder7* gère entièrement le développement, le test et le débogage d'applications utilisant les caractéristiques du *JDK 1.4.1.*

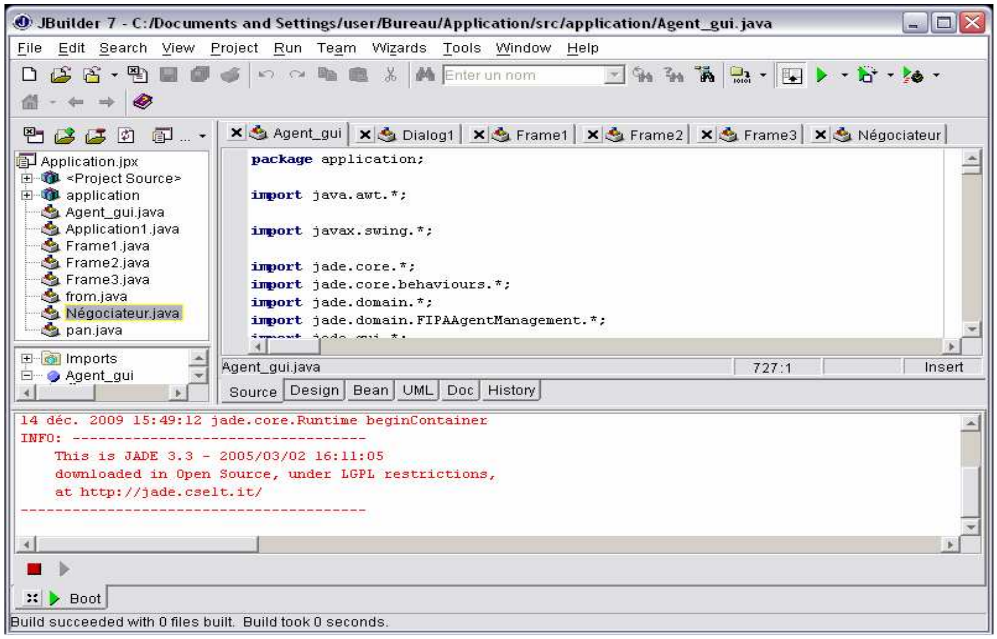


Figure 6.1. L'environnement JBuilder7 avec une intégration de la plate forme agents *JADE*.

3. Mise en Œuvre

Nous décrivons, ci-après, l'implémentation effective des différents éléments du modèle. Chacun des agents est un agent *JADE*, constitué d'un certain nombre de classes qui héritent généralement des classes du package *JADE*. La classe principale, permettant le lancement de l'agent, étend la classe prédéfinie *Agent* (située dans le package *jade.core*). Elle implémente en quelque sort le cycle (ou le corps) de l'agent et interagit directement ou indirectement avec les autres classes.

[1] www.disi.unige.it/person/MascardiV/Software/jessInJADE-Tutorial.pdf

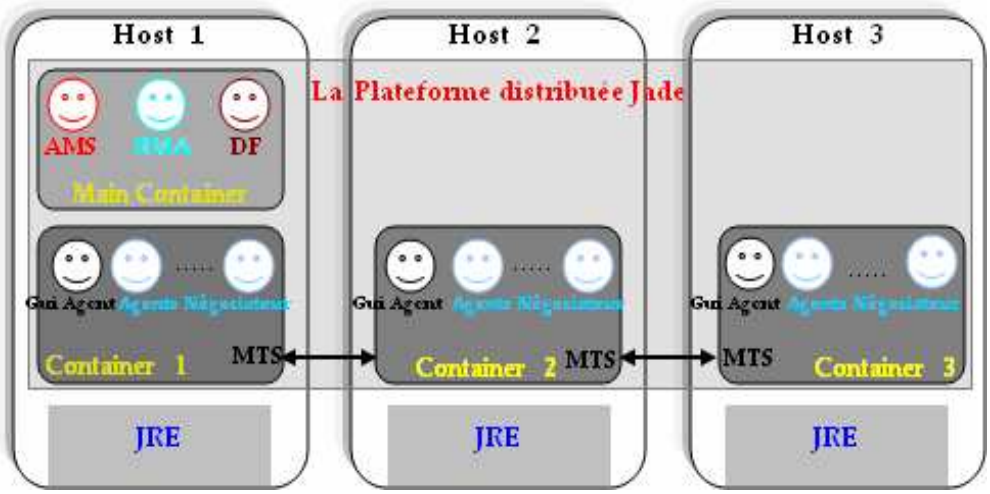


Figure 6.2. Architecture de notre simulateur sous la plate forme agents *JADE*.

Pour les deux agents d'annuaire *AMS (Agent Management System)* et *DF(Directory Facilitator)* leur rôle consiste à répondre à des recherches d'information afin de minimiser les flux de communication, en évitant au maximum l'envoie multiple de messages *(broadcasting)*. L'agent *AMS* offre un service de pages blanches en maintenant à jour les inscriptions des agents *(noms et adresses des agents).* Il permet la recherche des agents par nom pour être contactés pour la première fois. L'agent *DF* sert d'agent de pages jaunes qui gère une liste d'agents et de leurs compétences. Il assure ainsi la recherche des agents par compétences et l'envoie multiple de messages à ces agents. Les agents d'annuaire, issues de la norme *FIPA* pour la conception des *SMA* [Fipa 02a], sont implémentés dans toute plateforme agent compatible *FIPA.* Ainsi, l'utilisation de telle plateforme, ce qui est notre cas, permet d'éviter leur développement. L'agent *RMA* permet de contrôler le cycle de vie de la plate-forme et tous les agents la composant.

3.1. Description des classes utilisées

Pour notre application, on a utilisé 4 classes : Agent_Gui, négociateur, stratégies, Dessin:

3.1.1. Class négociateur

Le *négociateur* c'est la classe principale de notre modèle. Il est donc le seul à posséder tous les contrats proposés, toutes les propriétés des contrats qu'il a initié et les résultats des négociations. C'est donc naturellement lui qui fournit des indications sur les négociations passées, informations utiles pour la création de stratégies.

la classe principale de cet agent *négociateur* hérite de la classe *Agent* (située dans le package *jade.core*, cette classe représente une super classe commune pour tous les agents définis par l'utilisateur. Du point de vue programmeur, la conséquence, est qu'un agent *JADE* est simplement une classe *JAVA* qui étend la clase de base *Agent*). La classe négociateur (cf. figure 6.3) qui présente le code java, pour la définition de la class Négociateur, cela est fait par l'extension de la class de base *Agent* définie dans *JADE)* crée les différents types d'agents (acheteurs/vendeurs), elle contient les variables suivantes :

Type : le type d'agent (acheteurs/vendeurs);
Prob : la probabilité, puissance de capture de chaque agent;
Prix : position x;

Qualité : position y;
nbreTours: le nombre de tours pour l'enchère;
maxTours: le nombre maximum de tours pour l'enchère;
Timer: boolean initialisé à false;
Préférences: tableau contient tous les préférences d'un tel agents négociateur;
Lien : une matrice de 3*10 qui contient les subalternes de cet agent, Ex: les subalternes de l'agent acheteur sont des agents vendeurs (groupe G de n vendeurs); …..etc.

Les méthodes de la classe *Agent* que nous avons utilisé sont :

- *Setup ()* : sert à l'initialisation de l'agent.
- *tackDown ()* : sert à la dés-initiatisation de l'agent. Elle est appelée juste avant la suppression de l'agent de l'*AMS*.
- *doDelete ()* : fait passer l'agent à l'état *"Deleted"*, ce qui aura pour effet de supprimer l'agent de l'*AMS* (et appeler la méthode *tackDown ()* juste avant).
- *Send (ACLMessage)* : lorsque un agent souhaite envoyer un message, il doit créer un nouvel objet *ACLMessage*, compléter ses champs avec des valeurs appropriées et en fin appeler la *méthode Send ().*
- *Receive (MessageTemplate)* : lorsque un agent souhaite recevoir un message, il doit employer la méthode *Receive ().*
- *addBehaviour(Behaviour)* : permet d'ajouter un comportement (Behaviour) à la file de comportement d'un agent.

La class agent inclut les capacités d'un agent, le type de message et des événement aux quels il répond, et les plans[1] qu'il utilisera pour réaliser ses buts.

[1] Les plans d'un agent sont les fonctions de cet agent ce sont les instructions que l'agent suit pour essayer d'accomplir ses buts et de manipuler ses événements indiqués.

```
package application;
import jade.core.Agent;
…….;
public class Négociateur extends Agent{
// déclaration de méthodes pour la négociation et des variables;
private long tMaxNego, wakeupTime;
private boolean fini; //Ce booléen sera utilisé pour déterminer si la négociation est terminée ou non
private ACLMessage msgOffre;
private ACLMessage msgContreOffre;
Plan checkPlan ;//plan que l'agent va utiliser ;
…….. ;
event    cfpEvent     cfpEvt ; // événement reçus par l'agent ;
event    registerEvent     regEvt ; // événement envoyés  par l'agent ;
event    cancelEvent     cancelEvt ; // événement envoyés  par l'agent ;
…….. ;
public void onStart() {///Cette méthode s'exécute automatiquement au début:elle nous permet l'initialisation
tMaxNego = Strategie.initierNegociation(msgOffre.getSender());
wakeupTime = (tMaxNego < 0 ? Long.MAX_VALUE : System.currentTimeMillis() + tMaxNego);
}// La méthode done est appelée automatiquement après chaque exécution de la méthode action : tant qu'elle
retourne la valeur false on va exécuter la méthode action une autre fois.
public boolean done() {
if (fini) ModuleDeDonnees.modifierProfil(Strategie.getNouveauProfil(msgOffre.getSender()));
return fini;
}
protected void setup(){
    addBehaviour(new CyclicBehaviour(this){
            public void action()
                {
                    msgOffre = myAgent.receive(templateNegociation); // attente des messages d'offre
                    Strategie.calculProfilNegociateur (msgOffre) ;
                    ……..…….. ;
                    // Les actions de l'agent;
                    ……..…….. ;
                }
    });
 }
}
```

Figure 6.3. Extrait de code en Java, provenant de la class négociateur.java.

Donc la class *Négociateur* nous permette de créer deux types d'agents (acheteur/vendeur) qui son exécutés dans seule hôte inclut un ou plusieurs conteneurs. On suppose que les agents sont exécutés dans un même hôte, les hôtes sont interconnectés utilisent une infrastructure Internet/Intranet. Si un agent reçoit un message de la part de l'agent d'interface, selon le début de ce message qu'il reconnaît son sens. Ex : Si l'agent reçoit « 1 », ça veut dire qu'on demande ces informations. Il répond par : 1#name#type#prob#prix#qualité#lien#ddv#, Ce dernier message est codé, il sera décodé par la suite par l'agent d'interface. Si l'agent Gui détecte une interaction entre deux agents, pour que puisse entrer dans l'enchère, ces deux agents recevant des messages, et selon le contenue de ce message, ils réagiront comme il le faut.

Les agents négociateurs qui sont crées à partir de la class *négociateur* s'échangent des messages de type *NegotiationMessage*. Ce message comprend en plus de l'émetteur du message, de la primitive de négociation employée et des paramètres nécessaires pour le traitement de cette primitive, l'identifiant de la négociation ainsi que l'état de la négociation. Les différentes primitives utilisées sont du type *NegotiationPrimitive*, ce sont celles décrites dans le protocole de négociation, à savoir :*PROPOSE*, *ACCEPT*, *REFUSE*, *MODIFICATION_REQUEST*, *PROPOSE_MODIFICATION*, *CONFIRM*, *ANNUL*, et *CANCEL*.

sendMessage(to,msg) : cette méthode est invoquée par un agent négociateur pour envoyer un message de type *NegotiationMessage*:

```
public void sendMessage(String msg,String to){
    ACLMessage mssg = new ACLMessage(ACLMessage.INFORM);
    mssg.setContent(msg);
    mssg.addReceiver( new AID( to,AID.ISLOCALNAME  ));
    send(mssg);
 }
```

Figure 6.4. Extrait de code en Java, pour la méthode sendMessage.

Nous détaillons ici les différents composants d'un agent négociateur à savoir :

• **L'agent acheteur:** Celui-ci enverra un appel à propositions aux agents vendeur. Il se compose des événements *cfpEvent*, *reject_proposalEvent*, *accept_proposalEvent*, *des plans registerPlan*, *cfpPlan*, *respondPlan*, *inform_donePlan*, *inform_donePlan*, *failurePlan* définis ci-dessous.
Exemple : *ACLMessage cfpEvent AppelOffres = interfaceAcheteur.getConfigurationAppelOffres (ACLMessage. QUERY_REF);* qui nous permet la création d'une requête dont la performative QUERY_REF signifie l'attente d'une réponse *inform.*

• **L'agent vendeur:** reçoit les appels à propositions de l'agent acheteur. Il vérifie dans leurs bases de croyances s'ils peu répondre à ces demandes. Si c'est le cas, il envoie leur proposition à l'agent acheteur. Il se compose des événements *registerEvent*, *checkEvent*, *respondEvent*, *inform_doneEvent*, *failureEvent* et des plans *checkPlan*, *acceptPlan*, *rejectPlan* définis ci-dessous.

• **Événements:** les différents événement sont les suivants :

Register Event: lorsqu'un agent vendeur inscrit auprès d'un agent acheteur, il envoie un tel acte d'inscription à ce dernier.
Cfp Event: lorsque l'acheteur veut envoyer une demande à des vendeurs enregistrés auprès de lui, il envoie un tel acte *Cfp* à ses vendeurs.
Respond Event: après avoir reçu un appel à propositions de type *Cfp*, les vendeurs vérifient s'ils peuvent y répondre. Dans le cas positif, ils font une proposition au moyen de l'acte *Propose Event*, sinon, ils refusent le message *Cfp.*
Propose Event : il s'agit de la proposition du vendeur en réponse à l'appel à *proposition*.
Accept_proposal Event: l'agent acheteur choisit la meilleure proposition, et contacte l'agent vendeur choisi comme performeur pour la suite par un tel acte d'acceptation de proposition.
Reject_proposal Event: l'agent acheteur envoie cet acte de rejet aux vendeurs non choisis.
Failure Event: si la proposition d'un vendeur est acceptée, celui ci re-vérifie s'il est toujours en mesure de l'assumer. S'il n'est plus capable de le faire ou s'il change de décision de par les conditions changeantes de l'environnement, il le signifie à l'acheteur par cet acte d'échec.

Inform_done Event: si le vendeur est toujours capable d'assumer sa proposition il signifie son accomplissement à l'acheteur lorsqu'elle est réalisée.

• **Plans :** les différents plans sont les suivants :

Register Plan: lorsque l'acheteur reçoit l'inscription d'un vendeur, il l'enregistre dans une base de données.
Check Plan: après avoir reçu un *CfpEvent,* un vendeur vérifie, par ce plan, ses bases de données de manière à déterminer s'il est capable de répondre à l'acheteur. Cet événement pourrait être réalisé par d'autres plans ajoutés ultérieurement.
Respond Plan: après avoir reçu les réponses des vendeurs, l'acheteur, s'il y a plusieurs propositions choisit, par ce plan, celles qui satisfont les conditions requises et les empilent dans une base de données.
Accept Plan: ce plan implémente comme intention le désir exprimé en termes *BDI* par l'événement ***Accept_proposal:*** le vendeur reçoit cet événement d'acceptation de sa proposition, il vérifie alors s'il peut ou il veut toujours effectuer ce service. Cet événement pourrait être réalisé par d'autres plans ajoutés ultérieurement.
Reject Plan: ce plan implémente comme intention le désir exprimé en termes *BDI* par l'événement ***Reject_proposal:*** le vendeur l'utilise pour recevoir le rejet de l'acheteur. Cet événement pourrait être réalisé par d'autres plans ajoutés ultérieurement.
Failure Plan: ce plan implémente comme intention le désir exprimé en termes *BDI* par l'événement *Failure.* Il est utilisé par l'acheteur pour signifier, en réponse au participant, que la proposition a bien été déclarée comme échouée. Cet événement pourrait être réalisé par d'autres plans ajoutés ultérieurement.
Inform_done Plan: ce plan implémente comme intention le désir exprimé en termes *BDI* par l'événement *Inform_done.* Il est utilisé par l'acheteur pour signifier, en réponse au vendeur, que la proposition a bien été déclarée comme réalisée. Cet événement pourrait être réalisé par d'autres plans ajoutés ultérieurement.

• **Croyances:**

BDD Items_acheteur: base de données utilisée pour stocker les données de l'acheteur.
BDD Items_vendeur: base de données utilisée pour stocker les données des vendeurs.
BDD vendeurs: base de données utilisée pour stocker les vendeurs qui se sont inscrits auprès de l'acheteur.
BDD Responses: base de données utilisée pour stocker les propositions valides.

Nous avons simuler maintenant l'interaction entre l'agent acheteur et les agents vendeurs, l'interaction est basées sur l'envoi de message *ACL* entres les différents agents de système. La Figure 6.3. présente cette interaction, l'agent acheteur *Ach* envoi un appel d'offres (*CFP*) aux agents vendeurs *Vend1, Vend2* , et *Vend3,* les agents *Vend1, Vend2* refusent de participer dans l'offre tandis que l'agent *Vend3* propose une offre, l'agent acheteur l'accepte.

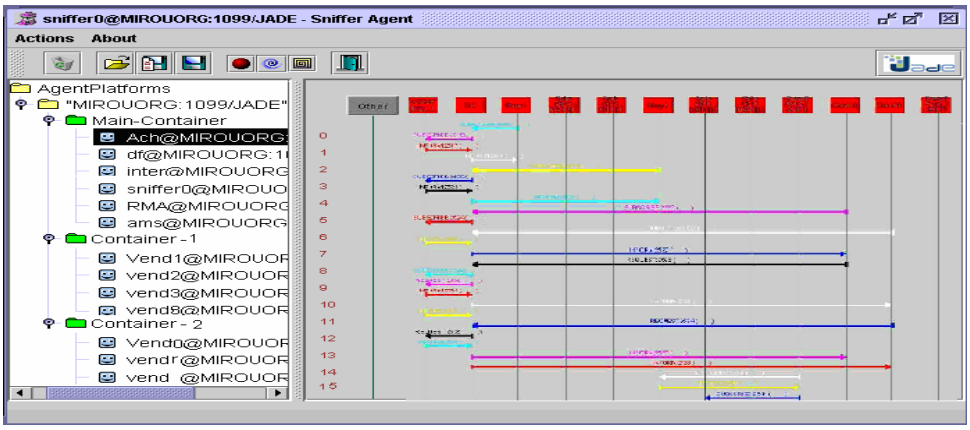


Figure 6.5. Protocole d'interaction entre un agent acheteur et plusieurs agents vendeur utilisant la plate forme agents *JADE*.

3.1.2. Class stratégies

Dans cette section on va donner le corps général de la classe Strategie et ses méthodes (cf. figure 6.6), La stratégie d'un tel agent lui permettre de raisonner sur le problème et d'inférer leurs décisions en fonction de la connaissance obtenue des autres agents négociateurs, elle lui permet de décider de confirmer ou d'annuler le contrat et de synthétiser les différentes propositions de modifications des vendeurs afin de proposer un nouveau contrat. Élaborer une bonne stratégie nécessite en effet une expertise du domaine que nous pouvons fournir par notre approche de simulation participative.

```
package application;
import jade.core.Agent;
…….;
public class Strategie {
private static AID[] courantsNegociateursAID;
private static Profil[] profilAgents;
private static Tactique tactique = new Tactique(); //elle sera modifiée à chaque itération de la négociation.
private static String avisAcheteurDeConcession;// initierNegociation et retourner Tmax pour la négociation
public static long initierNegociation(AID agentID) { // initialisation et préparation initial de la tactique
if (!appartientAgentsNegociateurs(agentID)) {
courantsNegociateursAID[courantsNegociateursAID.length] = agentID; // Charger l'ancien profil croyance de cet agent vis à vis agentID
profilAgents[profilAgents.length] = ModuleDeDonnees.getProfil(agentID);}
Profil profil = getProfilAgents(agentID);
ConfigNegociation configNegociation = InterfaceAch.getConfigNegociation(agentID);…….. ;}
public static Tactique calculeStrategique(AID agentID) {
Profil profil = getProfilAgents(agentID); //détermination du type de tactique à générer en plus de l'initialisation du début.
char degreConcession = profil.getDegreConcession(); // utilisation de la croyance de l'agent
…….. ;}
public static void calculProfilNegociateur(ACLMessage msgOffre) {
Profil profil = getProfilAgents(msgOffre.getSender()); // degre de concession avec laquelle l'antagoniste procède
profil.modifierConcession(examinerDegeConcession(msgOffre), msgOffre.getSender());}
public static Profil getNouveauProfil(AID agentID) {//sera exécuté à la fin du processus de la négociation
Profil profil = getProfilAgents(agentID);// Modifier les croyances vis à vis l'antagoniste en calculant l'estimation de EC à l'aide du processus passé
profil.setNouveauEspaceConcession( (float) profil.calculeEspaceConcession());
supprimerProfilAgents(agentID);
supprimerCourantNegociateur(agentID);
return profil;}} // Fin de la classe
```

Figure 6.6. Extrait de code en Java, provenant de la class strategie.java.

3.1.3. Class Agent_gui

Notre but est de fournir un système qui pourra être utilisé de façon automatique ou guidée par un ensemble d'utilisateur humain. Nous voulons en effet qu'un utilisateur puisse gérer ses négociations via notre modèle. Afin de pouvoir interagir avec son agent, une interface graphique est proposée à l'utilisateur. Nous avons conçu une interface utilisateur. A travers cette interface, l'utilisateur peut créer des agents négociateurs, des contrats, suivre l'avancement de ses négociations et répondre aux propositions de contrat. Nous décrivons par la suite chaque panneau de l'interface graphique.

Cette class permet de crée un agents de type *Gui agent*, ce qui veux dire un agent d'interface, cette agent est la colonne vertébral de l'application. Car il contrôle tous les agents. A l'exception des agents précédents, la classe principale de cet agent *Agent_gui* hérite de la classe *GuiAgent* (située dans le package *jade.gui*), qui est une simple extension de la classe *Agent.* La classe *Agent_gui* gère l'ensemble des comportements de l'agent *(ReceiveMessages)* ainsi qu'un certain nombre d'interfaces graphiques *GUI.* L'héritage de la classe *GuiAgent* offre deux méthodes

spécifiques : *postGuiEvent()* et *onGuiEvent()*. Ce sont les deux méthodes permettant l'interaction entre l'agent et son interface graphique. Les deux fragments de code, ci-après, illustrent la réalisation de ce mécanisme, successivement, au niveau de l'interface *Frame1* et l'agent *Agent_gui*. Tout d'abord, l'interface *Frame1* traite les actions de l'utilisateur par l'intermédiaire de sa méthode abstraite *actionPerformed()*. Elle signale ainsi des évènements graphiques de type *GuiEvent*, avec éventuellement leurs paramètres nécessaires, via la méthode *postGuiEvent()* (cf. figure 6.7*)*.

```
package application;
import java.awt.*;
import java.awt.event.*;
import jade.gui.*;
......
public class Frame1 extends JFrame {
……
protected Frame1 f;
public Agent_gui agent; // Pointeur sur l'agent d'interface
…….
public Frame1(Agent_gui a) {
agent = a;
…….
enableEvents(AWTEvent.WINDOW_EVENT_MASK);
……..
try {  jbInit();}
catch(Exception e) { e.printStackTrace();} }
.......
jButton1.addActionListener(new java.awt.event.ActionListener() {
public void actionPerformed(ActionEvent e) {
jButton1_actionPerformed(e);
}
});
…….
void jButton1_actionPerformed(ActionEvent e) {
GuiEvent ge = new GuiEvent(agent, 0);
agent.postGuiEvent(ge);
…….
  }
……..
} //class Frame1
```

Figure 6.7. Extrait de code en Java qui représente l'interface utilisateur, provenant de la class Frame1.java.

Ensuite, l'agent *Agent_gui* emploie sa méthode abstraite *onGuiEvent()* pour traiter les événements *GuiEvent* en provenance de *Frame1*. Le traitement d'un événement graphique nécessite souvent l'activation d'un comportement et/ou la manipulation de l'interface *Frame1* elle-même *(*cf. figure 6.8*)*. Et enfin voici les fonctionnalités de l'agent Gui:

```
package application;
import jade.gui.*;
.....
public class Agent_gui extends GuiAgent {
.....
protected void setup () {
..... // Creation de l'interface utilisateur
int command;
protected Frame1 f; // Instance de la GUI
…….
public Agent_gui() {
f=new Frame1(this);
}
protected void setup(){
 f.setVisible(true); }
protected void onGuiEvent(GuiEvent ev) { // interaction avec l'utilisateur à travers le GUI
command = ev.getType ();
if (command = 0) {
..... // un ensemble d'actions
}
……
if (command = n) {
.....// un ensemble d'actions
}
} } //class Agent_gui.
```

Figure 6.8. Extrait de code en Java qui représente l'implémentation de l'interface utilisateur utilisant la plate forme *JADE,* provenant de la class Agent_gui.java.

L'intermédiaire : L'agent gui est le seul agent intermédiaire entre tous les agent et la plateforme, celui qui ce charge d'échanger les informations entre les autres agents et la plateforme.

Demande d'informations des autres agents : A chaque fois que l'utilisateur clique sur le Botton *simulation*, l'agent Gui diffuse un message de demande de propositions(selon les critères les plus importants dans les enchères qui sont le prix et la qualité) aux agents de la simulation. Chaque agent reçu ce message répond par un autre message qui contiens tous les information nécessaire comme la proposition, le type, …. Etc. le message sera décomposer puis se mette dans une liste.

Vérification des interactions : L'agents gui, parcoure la liste des agents, et cherche les agents peuvent crées des interaction entre eux.

L'affichage : A chaque fois, l'agent gui parcoure la liste des agents, et affiche une image symbolique de chaque agent, cette image est affichée sur la position de l'agent, suivant leurs proposition (prix, qualité). Aussi il indique aux utilisateurs l'emplacement des interactions entre agents.

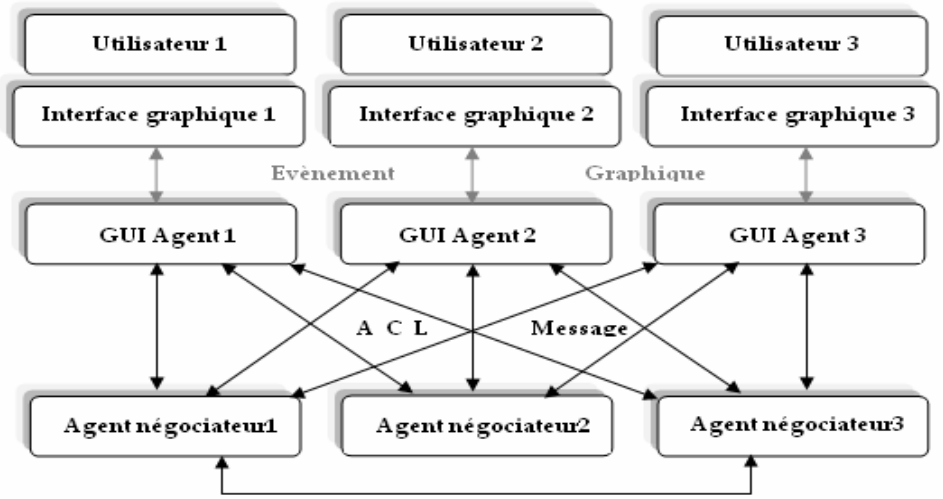


Figure 6.9. Interaction entre l'interface graphique et les agents utilisant des *GUI agents* dans un réseau LAN.

3.1.4. Class Dessin

C'est la class qui gère les grapheurs de courbes au sein de notre plate forme de simulation, elle représente l'ensemble des agents en interaction, et leurs différentes situations.

3.2. Présentation de l'application

Notre application fournit également une interface graphique pour la négociation, qui permet de créer des agents négociateur, de visualiser les différents messages envoyés et reçus par ces agents, de visualiser les différents agents vendeurs pris par l'agent acheteur, d'avoir une vue sur les négociations en cours sur les ressources, …etc. Dans notre application, l'utilisateur humain a la possibilité d'utiliser son agent, et de le contrôler par la modification de certains paramètres au moment d'exécution de modèle, c'est alors un outil d'aide à la décision qui montre l'état de toutes les négociations en cours et, dans ce cas, c'est l'utilisateur humain qui répond à une proposition de contrat. Par la suite les agents de modèle peuvent répondre automatiquement aux propositions sans intervention de l'utilisateur.

Afin de mettre notre application facile a comprendre, et a manipuler on a proposer ce désigne, qui est compréhensible dés la première utilisation:

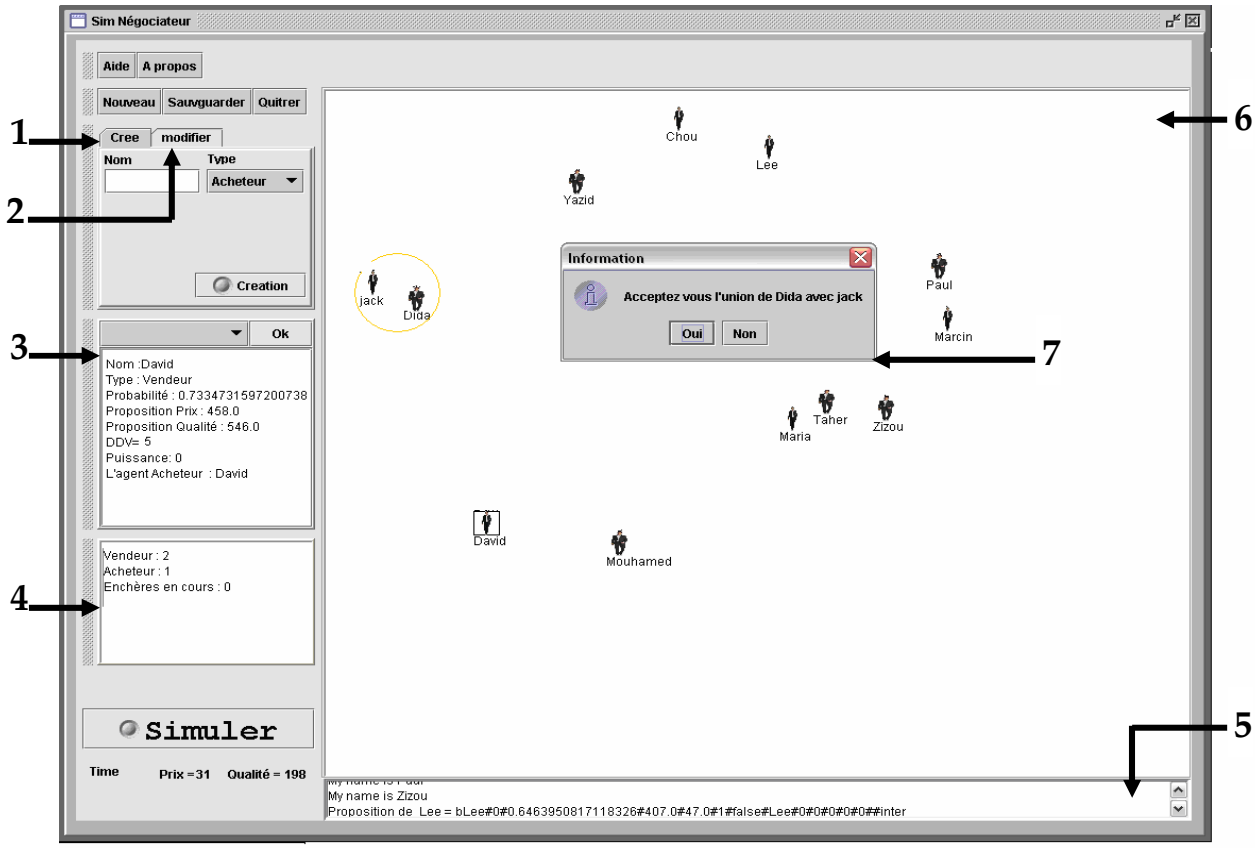


Figure 6.10 : L'interface de notre plate forme de simulation.

3.2.1. Le panneau crée

Pour crée un agents l'utilisateur doit mettre le nom d'agent dans la zone du nom d'agents, puis choisir son type, en fin cliquer sur le buttons création. Figure 6.10.1.

3.2.2. Le panneau modifier

L'utilisateur peut changer quelque paramètre des agents qu'il contrôle, pour cela il doit accéder au panneau modifier. Figure 6.10.2. et Figure 6.11. Dans ce panneau l'utilisateur peut changer la proposition de l'agent sélectionner par introduire par exemple les valeurs prix, qualité correspond a la nouvelle proposition désirer, puis cliquer sur le buttons *Modifier* ce dernier envoi un message a l'agents sélectionner et l'ordonner de changer ça proposition au prix, qualité reçu dans le message.

Figure 6.11 : le panneau modifier.

3.2.3. Le panneau information

Dans ce panneau l'utilisateur peut visualiser les agents qu'ils contrôle, pour cela, il doit choisir l'un de ces agents puis cliquer sur ok, ce panneau fournis quelque information concernant l'agents sélectionner, comme son nom, et leur type … etc. comme le montre la figure 6.10.3.

3.2.4. Le panneau statistique

Dans ce panneau l'utilisateur, peut observer les statistiques consternant la simulation. Les statistique contenant le nombre d'agents vendeurs, acheteurs, et le nombre d'enchère en cours….etc. figure 6.10.4.

3.2.5. Le panneau historique

Dans ce panneau l'utilisateur, accéder a l'historique de la simulation courante, il peut aussi voir les informations sur le déroulement de la simulation et la liste des messages échangés(cf. Figure 6.10.5. qui représente un extrait de la liste des messages émis et reçus par un agent), l'historique sera sauvegarder par la suite dans un fichier, pour permettre d'analyser la simulation, ou pour essayer de suivre les trace de la dernier simulation. Figure 6.10.5.

3.2.6. La plateforme de la simulation

Les utilisateurs peuvent crée, modifier et contrôler les agents qu'ils sont crée, et pour être dans la simulation on proposer une plateforme graphique qui montre la position de chaque agent, pour voir changement de la simulation l'utilisateur doit cliquer sur le buttons *simulation*, ce dernier cherche touts les agents disponible, et affiche une image qui représente le type d'agents dans ça position x, y (selon les critères de proposition les plus importants *Prix, Qualité*). Figure 6.10.6.

3.3. Les points forts de notre simulateur

Plusieurs points forts peuvent être dégagés pour notre simulateur:

- La participation des utilisateurs à des jeux de simulation ce qui permet l'amélioration et l'adaptation de nos agents à des nouvelles configurations de système,
- Interface utilisateurs dédiée pour permettre à ces derniers de contrôler un composant du modèle simulé, les manipuler à leurs grés durant l'exécution, en modifiant certains paramètres du composant,
- Possibilité de guider les utilisateurs dans le processus de simulation on utilisant des *GhostAgents*,
- On peut voir plusieurs négociations simultanément,
- Possibilité de fonctionnement en réseaux hétérogènes,
- Communication asynchrone,
- Négociation automatique,

3.4. Matérielles utiliser pour le développement

Pour la phase de développement de l'application :

- Un Pc de type Pentium 4. 3Ghz, Ram 1 Go, Windows XP SPII
- Pour la phase de test on a utilisé :
  Un réseau local (LAN) de 3 PC de type
  - 3 PC type Pentium 4. 3Ghz, Ram 1GO.
  - Switch de 100 Mbits/s.
  - Windows XP SPII.

3.5. Problèmes rencontrés

Au cours de développement de cette application, on a trouvé pas mal de problème :

- Le manque de la documentation ou des tutorial pour l'outil de développement jade.
- Beaucoup de temps perdu au moment de test, car les pc ne sont pas très puissants, et le *jbuilder* est très gourment aux ressources mémoire surtout au lancement des simulations.
- l'indisponibilité de version académique de l'outil de l'intelligence artificielle *JESS*, version limiter à 10 jours.

4. Simulations et exploration

Afin de valider notre proposition, et une fois que le modèle informatique implémenté sous forme de code, éventuellement à l'aide d'une plateforme dédiée (dans notre cas la plate forme multi-agents *JADE*), le modélisateur peut passer à la phase de simulation et exploration de son modèle. Dans notre exemple, nous simulons un certain nombres d'enchères.

Pour la première expérience, nous avons utilisé trois ordinateurs. Sur le premier ordinateur, le conteneur principal *Main-Container* est initialisé, et qu'il contient les trois agents *AM, RMA,* et *DF*, une autre conteneur doit être aussi lancer sur cet ordinateur qui est *Container-1,* elle contient l'agent d'interface *GuiAgent,* et les agents *négociateurs* qui sont crées par l'utilisateur de cet premier ordinateur. Sur le deuxième et le troisième ordinateur, les deux conteneurs *Container-2* et *Container-3* qui sont liées à la *Main-Container* sur le premier ordinateur ont été démarré. Sur les trois ordinateurs(Au lancement du système seul l'agent d'interface est actif pour chaque hôte dans notre réseau, puis chaque utilisateur qui participe à la simulation doit créer des agents négociateurs sur leur propre ordinateur dans une *container-i* qui contient aussi l'agent d'interface *GuiAgent* qui gère les interactions entre les agents de nos système, et qui permet à l'utilisateur un contrôle directe de ces agents, voir La Figure 6.2).

Dans cette expérience nous avons choisi un scénario simple de négociation, avec un agent acheteur *Ach*, et trois agents vendeurs *Vend1, Vend2, et Vend3,* afin de permettre la concurrence des prix, et d'autre attributs (dans notre cas c'est seulement la qualité). La figure 6.3 illustre ce scénario. Les agents Vendeurs souscrivant au niveau de l'agents Acheteur *Ach* pour commencer la négociation par l'envoi de *CallForPropose* aux *Vend1, Vend2, et Vend3* (dans notre simulateur des nuages de points sont créer et qui représentent un ensemble d'enchères entre un agent acheteur et plusieurs vendeurs).

Les vendeurs intéressés vont répondre par des propositions après une analyse du contenu de cette commande diffusée. Ces propositions peuvent être compatible avec la commande diffusée comme elles peuvent être différentes en quelques ou tous les attributs. Ils accompagnent leurs propositions d'une liste des attributs négociables qui peut être différente d'un vendeur à un autre. Puis l'agent Acheteur doit négocier chaque proposition indépendamment de l'autre par l'échange alternatif de proposition et contre proposition en faisant des concessions dans chaque alternative. Parmi les résultats de ces négociations, il y en a des accords non définitifs. L'agent fait des comparaisons et il choit la proposition la plus bénéfique pour lui. Ensuite, il envoie une confirmation à un ou plusieurs vendeurs et il demande l'établissement des contrats. Selon les règles et conditions, les membres sont obligés de respecter leurs contrats avec leurs contraintes et leurs échéances.

5. Conclusion

Nous avons présenté dans ce chapitre l'implémentation et la validation de notre modèle de négociation. Nous avons opté pour *JADE* pour le faire, vu ses avantages pour la programmation agents. nous avons réalisé une plate forme de simulation appelée *SimNégociate* qui comporte une interface graphique qui permet à l'utilisateur de gérer des négociations dans les enchères multi-attributs inversés. Les différents choix techniques nécessaires ont été présentés avant de décrire l'implémentation des différentes composantes de l'application. Ainsi, nous avons préféré d'utiliser les outils distribuables (*JAVA, JADE, JESS*) pour mettre en oeuvre le paradigme multi-agents en utilisant un environnement de développement orienté agent compatible *FIPA*.

Les avantages de notre application sont nombreuses, nous allons donc citer ici les plus importantes. Premièrement, cette application aide l'utilisateur à enchérir et enchérit à sa place quand il n'est pas là. Deuxièmement, cette application peut facilement être étendue à un autre type d'enchères, comme les enchères anglaises, hollandaises, Vickrey, etc. Et troisièmement, cette application est portable, en effet, les agents peuvent être placés sur un réseau hétérogène, sur des PDAs, etc. Bien que notre prototype ne soit pas encore (au moment de la rédaction de ce document) complètement mis en oeuvre, il assure ce minimum de tâches. Ce prototype apparaît ainsi comme un outil permettant de faire la synthèse de notre travail sur ***les simulations participatives à bases des systèmes multi-agents pour le protocole de négociation dans le e-commerce.***

Conclusion générale et Perspectives

Conclusion et Perspectives

Le travail de recherche que nous avons effectué implique deux domaines, d'une part la nouvelle approche pour la simulation qui est la simulation participative, et d'autre part l'intelligence artificielle distribuée et les systèmes multi-agents, et plus particulièrement, la négociation dans le domaine de commerce électronique. Notre objectif de travail était de proposer un protocole de négociation dans le e-commerce utilisant l'approche de simulation participative à base d'agents. Pour ce faire, nous avons adopté une approche multi-agents qui s'adapte bien à ce type de problèmes, en adaptant les techniques d'intelligence artificielle nécessaires pour la tâche de simulation.

Chronologiquement, nous avons fait un état de l'art sur les approches d'intelligence artificielle distribué et notamment l'approche multi-agents. Nous avons bien souligner le champ de cette étude détaillé, à savoir le domaine du commerce électronique, et plus particulièrement la négociation dans ce domaine. Cette dernière constitue aujourd'hui un domaine de recherche à part entière, et une étude qui avait pour objectifs de dégager et d'identifier les différents éléments nécessaires à l'utilisation de cette approche pour répondre à notre problématique.

En suite en a utiliser un model ouvert pour mieux comprendre l'approche participative, en combinant la simulation multi-agents, et la simulation participative, ce qui donne une simulation multi-agents participative, cette approche consiste a contrôler certain nombre d'agents, par des utilisateurs, ceci nous donne de nouvelle vue sur le model, et de nouvelles connaissances apparus ce qui s'appelle l'émergence. Nous avons adopter cette nouvelle approche au domaine du e-commerce, et en particulier à la négociation. En se basant sur l'étude des différentes techniques et approches utilisées dans la négociation, ainsi les travaux de la simulation participative, nous avons proposé un protocole de négociation basé sur cette nouvelle approche qui est la simulation participative. Nous avons utilisé dans notre modèle un mécanisme de négociation qui est basé sur un protocole d'enchères anglaises inversées, où on a proposé un nouveau modèle d'agent négociateur basé sur la participation des utilisateurs durant l'exécution de modèle, ces derniers sont au premier lieu des experts de domaine pour améliorer et adapter le comportement de nos agents en utilisant un apprentissage participatif, en suite on peut tester et valider les performances de notre système à des nouvelles configurations on impliquant des utilisateurs pas forcement des experts. Donc dans notre modèle, des agents dotés de capacités d'apprentissage et d'une certaine forme d'autonomie décisionnelle seront utiles pour automatiser ce processus d'extraction de connaissances d'une part, et pour automatiser le processus de négociation d'autre part. L'expert joue, par le biais d'une interface dédiée, le rôle qui lui a été confié au sein de la simulation pour contrôler nos agents et développer leur base comportementale, l'agent observe, puis propose des comportements qui peuvent être améliorés par l'expert. Cela revient à mettre en oeuvre un système d'*apprentissage participatif*.

Le succès d'utilisation des environnements de développement orientée agents sont nécessaires pour mieux adopter l'approche multi-agents. Par ailleurs, les efforts de standardisation de ces environnements représentent un grand pas dans le domaine et permet de renforcer le succès de la technologie multi-agents. Ainsi, nous avons présenté les spécifications *FIPA* pour les *SMA,* quelques plateformes agents et plus particulièrement l'environnement distribué *JADE*. Ce dernier est une plateforme agents respectant les standards *FIPA* pour le développement de systèmes multi-agents interopérables. Il a été largement utilisé avec succès dans plusieurs domaines. Pour cela, nous sommes intéressé à cette plateforme pour implémenter notre modèle. Nous avons également proposé une architecture développée dans un environnement distribué *(JADE/JAVA)*

afin de tester les comportements des agents en fonctions des différents paramètres et stratégies à simuler et de leur capacité de négociation.

Finalement, la mise en oeuvre efficace des différentes entités de cette simulation a été faite en analysant les différents aspects techniques nécessaires. Ainsi, nous avons exploité l'environnement *JADE* pour réaliser un prototype totalement en langage *Java.* L'implémentation avec *JADE* est toujours en développement, mais elle est prometteuse pour fournir quelques résultats intéressants.

De là on peut dire que l'approche de simulation participative peut être appliquée facilement à d'autres phénomènes sociaux, biologiques, …etc. Un tel travail trouve son utilité dans beaucoup de domaines comme l'enseignement /l'apprentissage, la prise de décision en prévoyant les mesures nécessaires face à certains comportements prévisibles ainsi que dans la recherche pour mieux comprendre certains phénomènes, valider certaines théories ou en conclure de nouvelles.

Ce mémoire constitue une base de travail à partir de laquelle de nouvelles activités de recherche peuvent être lancées afin d'améliorer le travail présenté. Nous envisageons de nombreuses perspectives, ainsi les efforts d'approfondissement qui restent encore à faire, et qui peuvent donc s'orienter vers les directions suivantes:

- ✓ La première perspective à moyen terme consiste à enrichir notre modèle afin de pouvoir représenter de nouvelles formes de négociation. Le protocole que nous avons défini reste assez simpliste, et il faudrait le doter de mécanismes plus précieux afin de prendre en compte les différents critères de la négociation;

- ✓ Dans notre approche nous avons proposé un modèle d'agent basé sur un apprentissage participatif, et nous avons montré son importance, ainsi leur efficacité, mais nous avons pas détaillé ce mécanisme. Donc une deuxième perspective à long terme consiste à enrichir ce mécanisme et de l'adapter a d'autre domaine de recherche comme le *e-learning*;

- ✓ Comme notre système est ouvert, les recherches dans ce domaine peuvent être continué, soit par nous même, ou par nos successeur. A ce stade la bonne utilisation de l'outil *JESS* donnera de bonne amélioration de ce système au niveau de l'intelligence des agents. Ainsi l'utilisation de la plate forme *Zeus* présentée dans le chapitre 2 section2.2.2 qui nous permet de développer des agents à une certaine capacités de raisonnement et des algorithmes d'apprentissage, et qui décrit les relations et les interactions entre les agents. L'utilisation de *Zeus* pour le développement de *SMA* est cependant conditionnelle à l'utilisation de la méthodologie *role modeling*. De plus, L'outil est assez complexe et sa maîtrise nécessite beaucoup de temps.

- ✓ Pour valider ce type de travail, il est nécessaire de s'appuyer sur un cas concret permettant la mise en œuvre de notre modèle. C'est pour quoi nous aimerions réaliser une application de négociation de grande importance, telle que des salles d'enchères sur Internet, une telle application nous permettrait une bonne validation de notre modèle et de montrer son utilité et sa facilité d'adaptation.

Ceci ne constitue qu'un aperçu de toutes les perspectives que l'on peut envisager pour notre travail, mais résume bien nos ambitions et tout le travail qu'il reste à faire.

BIBLIOGRAPHIE

Bibliographie

[Agha 86] Agha, Actor: "*A Model of Concurrent Computation in Distributed Systems*", MIT Press, 1986.

[Alessio et al 01] Alessio R. Lomuscio, Michael Wooldridge, Nicholas R. Jennings, "*A classification scheme for negociation in electronique commerce*" Department of computing, Imperial College of Science, Technology and Medecine, 2001

[Anderson 89] Anderson J.R."*Simulation : méthodology and application*" Agricultural economics, pp.3-54, 1989.

[Axelrod 97] Axelrod R."*Advancing the Art of Simulation in the Social Sciences*" Simulating Social Phenomena, Hegselmann R. and Terna P. (Eds.) Berlin: Springler, pp. 21-40, 1997.

[Bakos 97] Bakos, Y., "*Reducing Buyer Search Costs: Implication for Electronic Marketplaces*" Management Science, Volume 43, Number 12,pp.1676-1692, 1997.

[Barreteau 98] Barreteau O."*Un Système Multi-Agent pour explorer la viabilité des systèmes irrigués: dynamiques des interactions et modes d'organisation*". Thèse de doctorat, Ecole nationale du Génie Rural, des Eaux et des Forêts (ENGREF), 1998.

[Barreteau et Bousquet 01] Barreteau O. et Bousquet F., "*Role-playing games for opening the black box of multi-agent systems: method and lessons of its application to Senegal River Valley irrigated systems* ", dans Journal of Artificial Societies and Social Simulation, 2001.

[Beam et Segev 97] Beam, C. & Segev, A. "*Automated Negotiations: A Survey of the State of the Art*", [online] available at: http://www.haas.berkeley.edu/citm/publications/papers/, 1997.

[Becu et al 08] Becu N., Neef A., Schreinemachers P. and Sangkapitux C. *" Participatory computer simulation to support collective decision-making: potential and limits of stakeholder involvement"*. Land Use Policy, 25(4):498-509. (ScienceDirect), 2008.

[Bedou 97] I. Bedou, " *Modélisation de la coopération entre systèmes d'information: une approche cognitive basée sur la négociation* ". congrés INFOSID'97, pp. 427-442, 10-13 Juin, 1997.

[Beer et al 98] Beer M., d'Inverno M., Luck M., Jennings N., Preist C., Schroeder M. "*Negotiation in Multi-Agent Systems*" Workshop of the UK Special Interest Group on Multi-Agent Systems (UKMAS'98),1998.

[Bellifemine et al 99] Bellifemine F., Poggi A., Rimassa G., "*JADE -- A FIPA-compliant agent framework*", CSELT internal technical report. Part of this report has been also published in Proceedings of PAAM'99, London, pp.97-108, April 1999.

[Bellifemine et al 00] Bellifemine F., Giovani G., Tiziana T. Rimassa G, "*Jade Programmer's Guide*" Jade version 2.6 (http://www.fipa.org/specs/fipa00001/), 2000.

[Bellifemine et al 04] Bellifemine F., Caire G., Trucco T. and Rimassa G., "*JADE administrator's Guide* ", http://sharon.cselt.it/projects/jade/doc/, TILab, July 2004.

[Bellosta et al 04] M. Bellosta, I. Brigui, S. Kornman, and D. Vanderpooten. *"A multicriteria model for electronic auctions"*. In ACM Symposium on Applied Computing , SACŠ04, pages 759–765, 2004.

[Ben Said 03] Ben Said L., " *Simulation multi-agent des comportements des consommateurs dans un contexte concurrentiel* ", Thèse de doctorat, Université de Paris 6, 2003.

[Benyoucef et al 00] Benyoucef, M., Keller, R. K., Lamouroux, S., Robert, J., and Trussart, V. "*Towards a Generic E-Negotiation Platform*". In Proceedings of the Sixth International Conference on Re-Technologies for Information Systems, pages 95–109, Zurich, Switzerland, 2000.

[Bichler et al 02] M. Bichler, M. Kaukal, and A. Segev. "*Multiattribute auctions for electronic procurement*". In First IBM IAC Workshop on Internet Based Negotiation Technologies, volume 44, pages 291–301, 2002.

[Bond et Gasser 88] Bond A. H. and Gasser L." *Readings in distributed artificial intelligence*". Kaufmann M., (Eds.) San Mateo, California, 1988.

[Bousquet 94] Bousquet F."*Des milieux, des poissons, des hommes : étude par simulations multiagents*". Le cas de la pêche dans le Delta Central du Niger. Thèse de doctorat, Université Claude Bernard- Lyon, 1994.

[Bousquet et al 01] F. Bousquet, O. Barreteau, C. Mullon, J. Weber, "*Modélisation d'Accompagnement: Systèmes Multi-Agents et Gestion des Ressources Renouvelables*". Mémoire pour l'obtention de l'Habilitation à Diriger les Recherches de l'Université de Lyon 1 Soutenue le 22 mars 2001**.**

[Bousquet et Le Page 01] Bousquet F., et Le Page C."*Systèmes multi-agents et écosystèmes*" Principes et architectures des systèmes multi-agents, J-P Briot et Y.Demazeau, (Eds.) Paris : Hermes, Vol. 1, pp. 235-266, 2001.

[Bratley et al 87] Bratley P., Fox B. et Schrage L., " *A Guide to Simulation*", 2e édition, NY: Springer-Verlag, 1987.

[Brousseau 00] Brousseau E., "*Commerce Electronique : ce que disent les chiffres et ce qu'il faudrait savoir*", Séminaire INSEE - Économie et Statistique, n°339-340, pp. 147-170, 2000.

[Cassandras et Lafortune 99] Cassandras C. and Lafortune S. "*Introduction to discrete event systems* ". Kluwer Academic Publishers, Boston, MA, 1999.

[Castelfranchi et Conte 92] Castelfranchi C. et Conte R. "*Mind is not Enough: Precognitive Bases of Social Interaction*". Pages 93-110 of:Proceedings of1992 Symposium on Simulating Societies , April 1992.

[Chadès 03] I. Chadès, "*Planification distribuée dans les systèmes multi-agents à l'aide de processus décisionnels de Markov* ", Thèse de doctorat, Université Henri Poincaré-Nancy 1, 2003.

[Chaib-Draa et Jarras 01] Chaib-Draa B., Jarras I., Moulin B. " *Principes et architectures des systèmes multi-agents* ". In: Y.Demazeau, J.-P.Briot, (Eds.), "Systèmes multi-agents : principes généraux et applications". Hermès, 2001.

[Chaib-Draa et Jarras 02] B. Chaib-draa et I. Jarras, " *Aperçu sur les systèmes multi-agents* ", Série scientifique de CIRANO, ISSN 1198-8177, Montréal (Canada), Juillet 2002.

[Cheikhrouhou 02] Dr-Ing. Naoufel Cheikhrouhou Laboratoire de Gestion et Procédés de Production, " *La simulation à événements discrets* ", 2002.

[Clavel et al 00] G. Clavel, N. Mirouze, S. Munerot, E. Pichon, M. Soukal et S. Tiffanneau," *JAVA, la synthèse : des concepts objet aux architectures Web* ", 3ème édition, Dunod, ISBN 2-10-005379-5, 2000.

[CNIS 01] CNIS (Conseil National de l'Information Statistique) (2001), "*Rapport du groupe de travail du CNIS sur l'observation statistique du développement des Technologies de l'Information et de la Communication et de leur impact sur l'économie*", CNIS, Paris, n° 63, février 2001.

[Cohen 01] Cohen J."*Les places de marché cherchent encore leur modèle économique*", Magazine 01-Informatique, n°1633, 11 mai, pp. 20-21, 2001.

[Collins et al 98] Collins J., Tsvetovat M., Mobasher B., and Gini M. "*MAGNET: A Multi-Agent Contracting System for Plan Execution*". In Workshop on Artificial Intelligence and Manufacturing: State of the Art and State of Practice, pp 63-68, AAAI Press, Albuquerque, NM, August 1998.

[Conte 97] Conte R." *The necessity of intelligent agents in social simulation*" Applications of Simulation to Social Sciences. Ballot G. and Weisbuch G., (Eds.) France: Hermes, pp.19-38, 1997.

[David 00] David Y., "*L'impact des places de marché sur les relations B-TO-B*", Séminaire INSEE Méthodes -Net-enterprises, pp. 27-31, 2000.

[David et al 02] E. David, R. Azoulay-Schwartz, and S. Kraus. "*An english auction protocol for multi-attribute items*". In AMEC 2002, pages 52–68, 2002.

[Davis et Smith 80] Davis R., Smith R. G."*Negotiation as a Metaphor for Distributed Problem Solving*" Readings in Distributed Artificial Intelligence. Bond A. H. and Gasser L., (Eds.), pp. 333-356, 1980.

[Demazeau 96] Demazeau Y, Costa A-R., "*Populations and organisations in open multi agent systems*". In 1st Symposium on Parallel and Distributed AI, Hyderabad, India, 1996.

[Doran et al 90] Doran J., Carvajal H., Choo Y.J. et Li Y., "*The MCS Multiagent testbed, Devlopments and experiments* ", in Cooperating knowledge based systems, Springer Verlag, 1990.

[Doran et al 92] Doran J., Palmer M., et Gilbert N., "*The EOS Project : Modelling Upper Palaeolithic Social Change, Actes de Simulating Societies Symposium*", University of Surrey, Guilford (Angleterre), pp31-47, 1992.

[Drogoul 92] Drogoul A., Ferber J. "*Multi-Agent Simulation as a Tool for Modelling Societies:Application to Social Differentiation in Ant Colonies*" Decentralized A.I. 4, Elsevier North-Holland, 1992.

[Drogoul 93] Drogoul A. "*De la simulation multi-agent à la résolution collective de problèmes*". Une étude de l'émergence de structures d'organisation dans les systèmes multi-agents. Thèse de doctorat, Université P et M Curie Paris, le 23 Novembre 1993.

[Drogoul 00] Drogoul A."*Systèmes Multi-Agents situés*" Mémoire d'habilitation à diriger des recherches (habilitation thesis), Pierre et Marie Curie (Paris 6), 2000.

[Drogoul et al 02] Drogoul A., Meurisse T. et Vanbergue D., *"Multi-Agent-Based Simulations : Where are the Agents ?"*, dans Multi-Agent-Based Simulation, Bologna, 2002.

[Drogoul et al 03] Drogoul A., Vanbergue D. et Meurisse T., " *Simulation Orientée Agent: où sont les agents?* ", dans Actes des Journées de Rochebrune: Le status épistémologique de la simulation, Rencontres interdisciplinaires sur les systèmes complexes naturels et artificiels, Megève, 2003.

[Durfee et Lesser 89] Durfee E.H, and Lesser V.R. "*Negotiating Task Decomposition and Allocation Using Partial Global Planning*", In Gasser, L. and Huhns, M. N., editors, Distributed Artificial Intelligence, volume II, pages 229–243. Morgan Kaufmann, San Mateo, California, 1989.

[Duvallet 01] Duvallet C. " *Des systèmes d'aide à la décision temps réel et distribué : modélisation par agents*". PhD thesis, Université du Havre, 2001.

[El Fallah-Seghrouni 01] El Fallah-Seghrouni A., *"Modèles de coordination d'agents cognitifs"*, In Principes et architecture des systèmes multi-agents ,JP. Briot et Y Demazeau, Ed. Hermes, Lavoisier, ISBN 2-7462-0336-7, pp 139-172, 2001.

[Ferber 95] J.Ferber. " *Les systèmes multi-agents, vers une intelligence collective* ". Inter Editions, Paris, 1995.

[Feuillette 01] Feuillette S. "*Vers une gestion de la demande sur une nappe en accès libre : exploration des interactions ressource usages par les systèmes multi-agents*" Sciences et techniques du Languedoc. Montpellier, France, 343 Pp, 2001.

[Fipa 02a] Foundation for Intelligent Physical Agents, "*FIPA Agent Management Specification* ", http://www.fipa.org/specs/fipa00023/, December 2002.

[Fipa 02b] Foundation for Intelligent Physical Agents, "*FIPA ACL Message Structure Specification* ", http://www.fipa.org/specs/fipa00061/, December 2002.

[Fipa 02c] Foundation for Intelligent Physical Agents, "*FIPA Communicative Act Library Specification*", http://www.fipa.org/specs/fipa00037/, December 2002.

[Fipa 02d] Foundation for Intelligent Physical Agents, "*FIPA Ontology Service Specification*", http://www.fipa.org/specs/fipa00086/, December 2002.

[Fipa 02e] Foundation for Intelligent Physical Agents, "*FIPA SL Content Language Specification*", http://www.fipa.org/specs/fipa00008/, December 2002.

[Fipa 03] Foundation for Intelligent Physical Agents, "*FIPA Interaction Protocol Library Specification*", http://www.fipa.org/specs/fipa00025/, February 2003.

[Gamble et Sen 94] Gamble, R. et Sen, S. "*Using Formal Specification to Resolve Conflicts between Contracting Agents*". In Proc. AAAI-94 Wokshop on Conflict Management in Cooperative Problem Solving, pages 33–38, Seattle,Washington, 1994.

[Garneau et Delisle 02] T. Garneau et S. Delisle, " *Programmation orientée-agent : évaluation comparative d'outils et environnements* ", JFIADSMA 2002, Lille (France), 28-30 Octobre 2002.

[Gerin-Lajoie 00] Gerin-Lajoie, R. *"Architecture informatique de gnp version 1.0"*. Technical report, CIRANO, 2000.

[Guessoum 96] Z. Guessoum, " *Environnement de développement et de conception de systèmes multi-agents*", Thèse de doctorat, Université Paris 6, Mai 1996.

[Guessoum 03] Z. Guessoum, " *Modèles et architecture d'agent et de systèmes multi-agents adaptatifs*", Thèse d'habilitation de l'Université Paris 6, Laboratoire d'informatique, 2003.

[Guyot et Drogoul 04] Guyot P. et Drogoul A., *"Designing multi-agent based participatory simulations"*, dans Proceedings of the 5th Workshop on Agent Based Simulations, Lisbon, 2004.

[Guyot 06] Paul Guyot " *Simulations multi-agents participatives* ", Faire interagir agents et humains pour modéliser, explorer et reproduire les comportements collectifs. Thèse de doctorat, Université PARIS 6, Soutenue le 27 juin 2006.

[Guyot et Honiden 06] Guyot, P. et Honiden S. *"Agent-based participatory simulations: Merging multi-agent systems and role-playing games"*. Journal of Artificial Societies and Social Simulation, 9(4). Retrieved January 11, 2007, from http://jasss.soc.surrey.ac. uk/9/4/8. html, 2006.

[Hammond et Sun 03] Hammond P. et Sun Y., " *Monte Carlo simulation of macroeconomic risk with a continuum of agents: the symmetric case* ", dans Economic Theory, 21, pages 743-766, 2003.

[Jade 00] Java Agent DEvelopment framework. (http://www.fipa.org/specs/fipa00001/), 2000.

[Jade 02] Jade P.G., Bellifemine F., Caire G., Trucco T., Rimassa G., "*JADE Programmer's Guide*", Février 2002.

[Jennings 93] Jennings N. R. "*Commitments and Conventions: The Foundation of Coordination in Multi-Agent Systems* ", The Knowledge Engineering Review, 8 (3), 223-250, 1993.

[Jennings 96] N. R. Jennings, "*Coordination techniques for distributed artificial intelligence*", Foundations of distributed artificial intelligence, Wiley and sons, 1996.

[Jennings 00] Jennings, N. R. "*on agent-based software engineering* ". Artificial Intelligence Journal, 2000.

[Jennings et al 01] Jennings, N.R., Faratin, P., Lomuscio, A.R., Parsons, S., Sierra, C. and Wooldridge, M. "*Automated Negotiation: Prospects, Methods and Challenges*", International Journal of Group Décision and Negotiation, vol. 10, n° 2, pp. 199-215, 2001.

[Jonker et Treur 01] Jonker, C. M. and Treur, J. "*An Agent Architecture for Multi-Attribute Negotiation*". In Nebel, B., editor, Proceedings of the 17th International Joint Conference on AI, IJCAI'01, pages 1195 – 1201, 2001.

[Kalakota et Whinston 97] Kalakota, R., A. B. Whinston, " *Frontiers of Electronic Commerce*", MA: Addison Wesley, Reading, 1997.

[Kersten et al 01] Kersten, G. E., Noronha, S. J., and Teich, J. "*Are All Ecommerce Negotations Auctions ? "* In Proceedings of the Fourth International Conference on the Design of Cooperative Systems (COOP'2000), Sophia-Antipolis, France, 2001.

[Kraus et Schwartz 97] Kraus S., and Schwartz R., " *Negociation on Data Allocation in Multi-agents Environments*". Proc. Of the AAAI-97, pp.29-35, 1997.

[Kreifelts et Von Martial 91] Kreifelts T., Von Martial F. "*A Negotiation Framework for Autonomous Agents*", Decentralized A.I. 2, Demazeau Y. and Müller J-P., (Eds.), pp. 71-87, 1991.

[Kuflik et Shoval 00] Kuflik T., Shoval P. " *User Profile Generation for Intelligent Information Agents – Research in Progress*". Proceedings of the Conference on Advanced Information Systems Engineering, CaiSE'00, Stockolm, Suède, 2000.

[Kuhn et Muller 93] Kuhn N. et Muller J., "*Simulating Cooperative Transportation Companies* ", in Modelling and Simulation : Proceedings of ESM'93, Simulation Councils, pp.333-337, 1993.

[Kuokka et Harada 98] Kuokka D. and Harada L. "*Matchmaking for Information Agents*", Readings in Agents, M. N. Huhns & M. P. Singh editors, 1998.

[Labidi 94] Labidi S., Lejouad W. " *De l'intelligence artificielle distribuée aux systèmes multi-agents*". Rapport de recherche, N°2004.INRIA, 1994.

[Lander et Lesser 93] Lander S. E. and R. Lesser. V. " *Understanding the Role of Negotiation in Distributed Search Among heterogeneous Agents*" International Joint Conference on Artificial Intelligence (IJCAI-93), 1993.

[Lant 94] Lant T. K. " *Computer Simulations of Organizations as Experiental Learning Systems: Implications for Organization Theory*". In Computational Organization Theory, Carley K. M. & Prietula M.J. (Eds.), London : LEA publishers, p. 195-215, 1994.

[Lashkari et Metral 94] Lashkari Y., Metral M. et Maes P. "*Collaborative Interface Agents*". Proceedings of the Twelfth National Conference on Artificial Intelligence, AAAI '94 (1994), Seattle, USA, AAAI Press, 1994.

[Le Bars 03] Marjorie LE BARS " *Un Simulateur Multi-Agent pour l'Aide à la Décision d'un Collectif*", Application à la Gestion d'une Ressource Limitée Agro-environnementale. Thèse de doctorat, Université PARIS IX-DAUPHINE, Soutenue le 27 mai 2003.

[Lee et al 03] K.Y. Lee, J.S. Yun, and G.S. Jo. Mocaas "*auction agent system using a collaborative mobile agent in electronic commerce*". Expert System with Applications, 24 :183–187, 2003.

[Lonborg 92] Lomborg B., "*An evolution of Cooperation* ", Rapport Technique de l'Université de Aarhus-Institute of Political Science, 1992.

[Lorentz 98] Lorentz F. "*Commerce Electronique : une nouvelle donne pour les consommateurs, les entreprises, les citoyens et les pouvoirs publics*", Rapport de Travail sur le Commerce électronique pour le Ministère de l'Economie, des Finances et de l'Industrie, Juillet, 18 p., <URL : http://www.finances.gouv.fr/mission_commerce_electronique/rapports/>, 1998.

[Mathieu et Verrons 05] Mathieu, P. and Verrons, M.-H. "*GeNCA : Un modèle général de négociation de contrats*". Revue d'Intelligence Artificielle. à paraître, 2005.

[Macarez 01] Macarez N. & Lesle F. "*Le Commerce Electronique*", Editions PUF - Collection Que sais-je ?, 127 p, 2001.

[Marsella et Johnson 98] Marsella S. et Johnson W. L., *"An Intelligent Assistant for Team Training in Dynamic Multi- Agent Virtual Worlds"*, dans Proceedings of Intelligent Tutoring Systems: Fourth International Conference (ITS'98), 1998.

[Mazouzi 01] Mazouzi. H. " *Ingénierie des protocoles d'interaction : des Systèmes Distribués aux Systèmes Multi-Agents*". PhD thesis, Université de Paris IX – Dauphine, 2001.

[Misse 00] Misse B. "*Vente aux enchères et Internet*", Décisions Marketing, n°21, Septembre-Décembre, pp. 99-100, 2000.

[Müller 96] Müller J. H. (1996) "*Negotiation Principles*" Foundations of Distributed Artificial Intelligence, O'Hare G. M. P. and Jennings N. R., (Eds.), John Wiley & Sons, pp. 211-229, 1996.

[Muller 01] Muller A. "*La Net Economie*", Editions PUF -Collection Que sais-je ?, 127 p, 2001.

[Nguyen-Duc et al 04] Nguyen-Duc M., Duong V. et Drogoul A., "*Agent-based modeling and experimentation for Real-time Collaborative Decision-Making in Air Traffic Management*", dans Proceedings of the 24th Congress of the Int. Council of the Aeronautical Sciences (ICAS'04), Yokohama, 2004.

[Nguyen-Duc 05] Nguyen-Duc Minh " *Vers la conception participative de simulations sociales*", Application à la gestion du trafic aérien. Thèse de doctorat, Université PARIS 6, Soutenue le 1 février 2005.

[Noriega 98] Noriega, P. "*Agent mediated auctions : The Fishmarket Metaphor*". PhD thesis, University of Barcelona, 1998.

[Nowak et Latane 92] Nowak A. et Latane B., "*Simulating the emergence of social order from individual behaviour*", Actes de Simulating societies Symposium, Université de Surrey, Guidford (Angleterre), pp. 139-141, 1992.

[Oliveira et al 99] Oliveira E., Fonseca J.M., Steiger-Garçao A., " *Multicriteria negotiation in Multi-Agent Systems*", The First International Workshop of Central and Eastern Europe on Multi-agent Systems (CEEMAS'99), St. Petersbourg, juin 1999.

[Parunak 96] Parunak, H.V.D. "*Applications of distributed artificial intelligence in industry*", in O'HARE, G. and JENNINGS, N. (Eds), Foundations of Distributed Artificial Intelligence, John Wiley & Sons, 1996.

[Pascalau 06] Laura Daniela PASCALAU *"Apprentissage automatique de comportements dans le cadre de simulations d'usages"* étude bibliographique sous la direction de Nicolas SABOURRET (OASIS, LIP6) 6 février 2006.

[Pierre 06] Pierre De Loor. "*Autonomisation de modèles pour les Simulations participatives*". Habilitation à diriger des recherches Université de Bretagne occidentale, Soutenue le 5 Décembre 2006.

[Quéau 86] Quéau P. " *Eloge de la simulation : de la vie des langages à la synthèse des images*". Edition Champ Vallon. Col. Milieux, 1986.

[Rejeb 05] L. Rejeb, "*Simulation multi-agents de modèles économiques: Vers des systèmes multi-agents adaptatifs*", Thèse de doctorat, Université de Reims Champagne-Ardennes, France, 2005.

[Ricordel et Demazeau 00] Ricordel P.-M., Demazeau Y. "*From Analysis to deployment: A Multi-Agent Platform Survey*" Proceedings of 1st International Workshop on Enginnering Societies in the Agents World (ESAW), ECAI'2000, A. Oinici R. Tolksdorf, and F. Zambonelli, (Eds.), Berlin, Germany, Springler Verlag pp 93-105, 2000.

[Romaric 03] Romaric C., " *Des agents intelligents dans un environnement de communication multimédia : vers la conception de services adaptatifs* ", doctorat de l'université Henri Poincaré- Nancy1. Décembre 2003.

[Rus et Gray 97] Rus D., Gray R. et Kotz D. " *Transportable Information Agent* ", Journal of Intelligent Information systems, vol. 9, n° 3, p. 215-238, 1997.

[Sandholm 93] Sandholm, T.W. "*An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations*". In Eleventh National Conference on Artificial Intelligence, AAAI-93, pages 256–262, Washington DC. (Acceptance Rate 24%), 1993.

[Sandholm 99] Sandholm, T.W. "*Distributed Rational Décision Making*", in WEISS, G. (Ed), Multi-Agent Systems: A Modem Approach to Distributed Artificial Intelligence,MIT Press, 1999.

[Sandholm et Lesser 95] Sandholm, T. and Lesser, V. "*Issues in Automated Negotiation and Electronic Commerce : Extending the Contract Net Framework*". In First International Conference on Multiagent Systems (ICMAS-95), pages 328–335, San Francisco, 1995.

[Sen et Durfee 94] Sen S. Durfee E.H. " *The role of commitment in cooperative negotiation* ". International Journal on Intelligent an Cooperative Information Systems, 3(1):67-81, 1994.

[Shoham 93] Y. Shoham, "*Agent-Oriented Programming*", Artificial Intelligence, 60(1):51-92, Mars 1993.

[Sierra 99] Sierra C. "*Agent-mediated electronic commerce*" a European viewpoint, In 7 th Journées Francophones pour l'Intelligence Artificielle Distribuée et les Systèmes Multi-agents, La Réunion, 9 November 1999.

[Smith 80] Smith. R.G, *The Contract Net Protocol : High Level Communication and Control in a Distributed Problem Solver.* IEEE trans. On Computers, Vol. 29 (12), Pages 1104-1113, 1980.

[Smith 88] Smith R. G. " *The Contract net Protocol: High-level Communication and Control in a Distributed Problem Solver* " Readings in Distributed Artificial Intelligence, Bond A. H. and Gasser L., (Eds.), pp. 357-366, 1988.

[Sycara 88] Sycara K. "*Resolving goal conflicts via negotiation*" 7th National Conference on Artificial Intelligence. Paul, Minnesota, 1988.

[Sycara 92] Sycara K. "*The PERSUADER*" The Encyclopedia of Artificial Intelligence, Shapiro J., (Eds.) New York, 1992.

[Tavoillot 00] Tavoillot J-A. "*Net-économie : l'âge de raison*", Magazine Enjeux Les Echos, Novembre, pp. 97-130, 2000.

[Terna 98] Terna P. "*Simulation Tools for Social Scientists: Building Agent Based Models with SWARM*" Journal of Artificial Societies and Social Simulation, Vol. 1, 9 Pp, 1998.

[Tsvetovatyy et al 97] Tsvetovatyy, M., Gini, M., Mobasher, B., and Wieckowski, Z. "*MAGMA : An Agent-Based Virtual Market for Electronic Commerce*". Journal of Applied Artificial Intelligence, 6, 1997.

[Vallin et Vanderpooten 02] Vallin P., Vanderpooten D., "*Aide à la décision : une approche par les cas*". Ellipses, Paris, 2000. Note: 2e édition, 2002.

[Vanbergue 00] Vanbergue, D., " *Modélisation de phénomènes urbains: Simulation des migrations intraurbaines* ", dans Pesty S. et Sayettat-Fau C. (éditeurs), Systèmes multi-agents, Hermes, 2000.

[Verrons 04] Verrons, M. "*GeNCA : un modèle général de négociation de contrats entre agents*". Thèse de doctorat, Université des Sciences et Technologies de Lille, Soutenue le 2 Novembre 2004.

[Vulkan et Jennings 00] Vulkan N., Jennings N.R., *"Efficient mechanisms for the supply of services in multi-agent environments"*, Decision Support Systems, vol 28, Issues 1-2, p. 5-19, mars 2000.

[Wooldridge et Jennings 95] Wooldridge, M.J., and Jennings, N.R. " *Agent Theories, Architectures and Languages: A Survey* ", in M. Wooldridge and N. R. Jennings, editors: Intelligent Agents - Theories,Architectures, and Languages I. Springer-Verlag Lecture Notes in AI Volume 890, February 1995.

[Wooldridge 97] Michael Wooldridge. "*Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence* ". The MIT Press, 1997.

[Wooldridge et al 98] M. Wooldridge, N. Jennings, K. Sycara. " *A roadmap of agent research and development* ". Autonomous Agents and Multi-agents systems**,** Vol.1, Num.1, pp.7-38, July 1998.

[Wooldridge 02] Michael Wooldridge, "*An Introduction to Multiagent Systems*", Wiley ed, 2002.

[Wurman et al 98] Wurman, P. R.,Wellman, M. P., and Walsh,W. E. "*The Michigan Internet AuctionBot : A Configurable Auction Server for Human and Software Agents*". In Proceedings of the Second International Conference on Autonomous Agents (Agents-98), Minneapolis, MN, USA, 1998.

[Wurman et al 02] P.R. Wurman, M.P Wellman, and W.E Walsh. *"Specifying rules for electronic auctions"*. AI Magazine, 23(3) :15–23, 2002.

[Zeng et Sycara 97] Zeng D. D., Sycara K. "*Benefits of Learning in Negotiation*" AAAI-97, 1997.

[Zeng et Sycara 99] Zeng D. D., Sycara K. " *Dynamic Supply Chain Structuring for Electronic Commerce Among Agents*" Intelligent Information Agents Mathias Klusch, (Eds.) Springer, 1999.