Université de Montréal

Agents à base de règles pour les négociations électroniques: concepts et infrastructure

par

Hakim Alj

Département d'Informatique et de Recherche Opérationnelle

Faculté des Arts et des Sciences

Mémoire présenté à la Faculté des Études Supérieures
en vue de l'obtention du grade de
Maître ès Sciences (M.Sc.)
en informatique

Mars, 2002

Université de Montréal
Faculté des Études Supérieures

Ce mémoire intitulé:

Agents à base de règles pour les négociations électroniques:
concepts et infrastructure

présenté par:

Hakim Alj

a été évalué par un jury composé des personnes suivantes:

Président-rapporteur :	Peter Kropf
Directeur de recherche :	Rudolf K. Keller
Membre du jury :	Gilbert Babin

Mémoire accepté le: 15 mai 2002

Sommaire

Avec l'essor d'Internet, les négociations électroniques commencent à entrer dans les habitudes commerciales des entreprises et des particuliers. Aussi, beaucoup de travaux sont effectués pour développer des agents permettant d'automatiser le processus de négociation, notamment dans le contexte des enchères. Cependant ceux-ci mettent surtout l'emphase sur la mise au point d'algorithmes performants et efficaces qui sont souvent conçus pour fonctionner avec un ou deux types de négociation et qui sont généralement inadaptables à d'autres types. Peu se sont intéressés au développement d'une infrastructure agent capable de s'adapter à un vaste ensemble de négociations.

Nous avons choisi une approche à base de règles pour mettre au point une architecture pour une telle infrastructure. L'utilisation d'un système à base de règles permet de séparer les fonctionnalités de l'agent de son comportement et offre la possibilité de changer celui-ci de façon dynamique. Par ailleurs, le système intègre un blackboard permettant de coordonner les agents au besoin et une interface de surveillance des agents. Afin de valider cette architecture, une infrastructure nommée INSULA a été implémentée et des tests ont été menés avec pour contexte les enchères électroniques. Ceux-ci ont entres autres mis en valeur l'ambivalence des agents INSULA acquise grâce à notre approche.

Mots clés : commerce électronique, négociation, enchère, agent, moteur de règles, stratégies, coordination.

Abstract

With the growth of the Internet, electronic negotiations are becoming a part of commercial uses of enterprises and consumers. Therefore, many research works are being done in the way to develop agents that will permit to automate the negotiation process. Nevertheless, these works concern essentially the optimization of algorithms that work with only one or two types of negotiations and that generally do not adapt to other types. Few works are interested in the development of an agent infrastructure that is able to deal with a large set of negotiations.

We have chosen a rule based approach to design an architecture for such an infrastructure. The use of a rule based system allows us to separate the agent functionalities from its behavior and gives us the possibility to change this last one dynamically. Moreover, the system integrates a blackboard that is used when coordination is needed, and also an agent monitoring tool. In order to validate this architecture, an infrastructure named INSULA has been implemented and tests have been conducted in the context of electronic auctions. These tests have highlighted the versatility of the INSULA agents acquired thanks to our approach.

Keywords: electronic commerce, negotiation, auction, agent, rule engine, strategy, coordination.

Table des matières

Liste des tableaux

Liste des figures

Liste des sigles et abréviations

CIRANO	Centre Inter-universitaire de Recherche en ANalyse des Organisations
GNP	Generic Negotiation Platform
TEM	Towards Electronic Marketplaces

Je dédie ce mémoire à mes parents, qu'ils y voient la récompense de leurs sacrifices et de leur patience.

Remerciements

Je tiens à remercier Monsieur Rudolf K. Keller, professeur à l'Université de Montréal, pour avoir dirigé et supervisé la rédaction de ce mémoire, ainsi que pour m'avoir permis de faire partie de l'équipe du projet CONSENSUS. Sa patience, sa rigueur, ainsi que sa générosité m'ont été d'une aide inestimable tout au long de ce travail.

Je me dois aussi de remercier Monsieur Morad Benyoucef pour son soutien, ses encouragements et son amitié.

Cette recherche a pu voir le jour grâce à Bell[1] Canada, BCE[2] Emergis, CRSNG[3] (Conseil National de Recherches en Sciences Naturelles et en Génie) et CIRANO[4] (Centre Inter-universitaire de Recherche en ANalyse des organisations). Je les remercie.

Enfin, je voudrais remercier les chercheurs, les professionnels de recherche et le personnel de soutien du CIRANO pour leur support et leur disponibilité qui ont fait de mon passage parmi eux une expérience inoubliable.

[1] http://www.bell.ca
[2] http://www.emergis.com
[3] http://www.nserc.ca
[4] http://www.cirano.qc.ca

Chapitre 1: Introduction

1.1 Mise en contexte

E-commerce, e-negotiations, e-business, B2C[5], B2B[6], etc. Autant de nouveaux mots et d'acronymes apparus ces dernières années qui témoignent de l'émergence d'un nouveau paradigme en terme de commerce. Internet a déjà commencé à bouleverser le monde des affaires et du commerce et va sans doute le révolutionner. C'est dans cette perspective qu'un travail effervescent en recherche impliquant des domaines aussi divers que les sciences informatiques, les sciences économiques, la psychologie et le droit a lieu en ce début de 21ème siècle auquel on référera sans doute comme l'époque des premiers balbutiements du commerce électronique.

De nos jours, le commerce est encore principalement dirigé par des actions et décisions humaines : on décide du montant qu'on est prêt à dépenser pour acheter un produit, ainsi que où et quand on va effectuer l'achat. L'entrée d'Internet dans notre vie quotidienne a permis à des mécanismes économiques, tels que les enchères, trop chers ou pas assez efficaces de se développer. Ainsi, les enchères, autrefois réservées au marché de l'art ou encore de l'attribution des licences pour les plages de fréquences en téléphonie, connaissent un succès sans cesse grandissant auprès des consommateurs comme en témoigne le célèbre site d'enchères eBay. Les infrastructures de commerce électronique permettent l'essor de ces mécanismes en éliminant certaines de leurs contraintes. Ainsi, comme le mentionne Parkes [PUF99], dans le cas d'une enchère, l'Internet fait en sorte qu'il n'est pas besoin de

[5] Business-To-Customer
[6] Business-To-Business

rassembler les produits et les personnes en un même lieu, ce qui diminue considérablement les coûts de participation et permet de joindre un large auditoire.

Mais qu'est-ce que le commerce électronique ? De façon informelle et un peu simpliste, on peut dire que le commerce électronique se résume à la contribution des technologies de l'informatique et des télécommunications au commerce traditionnel. Plus formellement, il peut être défini par le modèle du laboratoire AMEC du Massachusetts Institute of Technology. Ce modèle, nommé *Consumer Buying Behavior* (CBB) qu'on peut traduire par *comportement d'achat du consommateur*, identifie six étapes dans une transaction de commerce électronique [MGM99] :

1. **Identification du besoin** : éventuellement stimulé par de l'information reçue sur des produits, l'acheteur prend conscience d'un besoin.
2. **Courtage du produit** : à ce stade, on recherche de l'information sur le produit et évalue les différentes alternatives afin de cibler le produit à acheter.
3. **Courtage du marchand** : à partir du résultat de l'étape précédente et en fonction des informations propres aux marchands, on identifie le vendeur chez qui on va acheter.
4. **Négociation** : cette étape consiste à déterminer les termes de la transaction. Selon les cas, le prix et d'autres aspects du produit peuvent être négociés.
5. **Achat et livraison** : l'achat et la livraison du produit peuvent terminer la négociation ou survenir plus tard.
6. **Service et Évaluation** : cette dernière étape implique le service à la clientèle et l'évaluation du produit par l'acheteur.

Dans le cadre de ce mémoire, on s'intéresse à la quatrième étape de ce modèle, la négociation. En effet, la négociation est une activité qui peut s'avérer complexe et coûteuse en temps pour les participants. C'est pourquoi, on désire automatiser cette étape pour l'acheteur, en respect avec ses besoins et ses contraintes.

Dans ce contexte, l'objectif de ce mémoire est de développer une infrastructure agent basée sur un système à base de règles.

1.2 Contributions principales

La finalité de cette recherche consiste en deux contributions majeures.

a. Explorer l'approche des agents à base de règles dans le contexte des négociations électroniques. Certes, l'utilisation des systèmes à base de règles dans des applications logicielles date des années 80, avec le boom alors de l'intelligence artificielle, dont le concept est issu d'ailleurs, mais l'idée consistant à combiner les concepts d'agent et de système à base de règles pour les appliquer au commerce électronique, qui, rappelons-le, est un domaine qui émerge à peine, est relativement originale. On pourra s'en apercevoir lorsqu'on rapportera les divers travaux connexes au chapitre 3.

b. Réaliser et tester une infrastructure agent, nommée INSULA, dans un cadre réel (le projet CONSENSUS) afin de valider nos idées. Plus précisément, on donne une description détaillée de l'infrastructure et on relate les tests effectués ainsi que les leçons apprises de cette expérience.

Comme retombées de cet effort, on remarquera également les contributions suivantes :

- introduction aux concepts des négociations électroniques et des négociations automatisées
- l'infrastructure réalisée est opérationnelle sur la plate-forme de négociation GNP (cf. chapitre 3) pour les négociations suivantes : enchère anglaise, enchère hollandaise, enchère à enveloppes scellées.

1.3 Publications

Les activités de l'auteur au sein du projet CONSENSUS ont contribué aux publications suivantes.

L'article [BAVK01] introduit de façon globale le projet CONSENSUS et les idées qui s'y rapportent, tandis que la publication [BAK01] couvre le développement de INSULA, l'infrastructure agents réalisée dans le cadre de cette recherche. Enfin, l'article [BALK02]

présente les expérimentations qui ont été menées pour tester l'utilisation de règles dans divers scénarios de négociation.

1.4 Organisation du mémoire

Ce mémoire est organisé en deux principaux axes.
Le premier axe est théorique et introduit notions et idées tandis que le deuxième axe couvre leur mise en œuvre.

En premier lieu, au chapitre 2, sont présentés les agents et les systèmes multi-agents, avant de définir le concept de négociation automatisée et de présenter l'éventail d'enchères qui représentent le type de négociation électronique que l'on rencontre le plus fréquemment sur les sites commerciaux ou dans les travaux de recherche. Ces deux survols permettront de saisir pleinement la notion d'agent de négociation.

Viennent ensuite, au chapitre 3, une description du contexte dans lequel ce travail a été mené et, en particulier, une description du projet CONSENSUS dans lequel il s'insère, sans oublier une revue des principaux travaux liés aux agents de négociation ou aux contexte de la négociation automatisée.

Pour terminer ce premier axe, on présente, au chapitre 4, notre approche ainsi que notre architecture pour des agents génériques de négociation. Ce chapitre clôt le premier axe.

Au chapitre 5, on propose une implantation de l'infrastructure définie dans le chapitre précédent. Entre autres, les choix effectués et une description détaillée sont donnés pour chacun des modules.

Par la suite, au chapitre 6, on décrit les tests qui ont été menés pour valider l'implantation et donc l'architecture. On y précise notamment quels sont les types de négociation qui ont été choisis et quelles sont les règles de stratégie qui ont servi pour les tests.

Enfin, le chapitre 7 clôt ce mémoire par une discussion des résultats obtenus suite aux tests et par une discussion des forces et faiblesses de notre infrastructure.

Chapitre 2: Agents de négociation

Ce chapitre a pour objectif de définir le concept d'agent de négociation. En effet, les termes *agent* et *négociation* sont des termes génériques qui prennent un sens différent selon les contextes et les gens qui les emploient, faute de consensus général sur leurs définitions. D'ailleurs, dans la littérature, les travaux ayant trait aux agents commencent souvent - et devraient si ce n'est pas le cas - par donner leur définition de ce concept afin d'éviter toute ambiguïté. Dans un premier temps donc, on commencera par donner un aperçu général de ce qu'est un agent avant de donner la définition retenue ici. Les principaux types d'architecture d'agents qu'on peut rencontrer seront également présentés, suivis d'une introduction aux systèmes multi-agents. Puis, dans un deuxième temps, nous discuterons des négociations et des types de négociation, ce qui nous amènera à définir le concept de négociation automatique. Enfin, le concept d'enchère sera développé pour terminer avec une description des enchères les plus connues. Ce sont ces enchères qui ont été choisies comme cadre d'expérimentation.

2.1 Agents logiciels

2.1.1 Définitions

Agent : *Celui qui agit* – Le Petit Robert

Qu'est-ce qu'un agent logiciel ? Le domaine des agents implique des disciplines aussi variées que l'intelligence artificielle, la théorie des jeux et les systèmes d'information. Il est par conséquent difficile de fournir une définition universelle de ce qu'est un agent. Il suffit de se reporter aux travaux de Franklin et Graesser pour se rendre compte qu'il y a autant de

définitions que de groupes de recherches spécialisés en la matière [FG96]. Toutefois, la majorité des chercheurs s'accordent pour définir un agent comme une entité combinant un certain nombre de propriétés. Selon le type d'applications dans lequel il est amené à évoluer, l'agent devra posséder des propriétés différentes. Une liste non exhaustive de ces propriétés est énumérée ci dessous [Gro00] :

- *Autonome* : capable d'agir sans l'intervention directe d'humains et a un certain contrôle sur ses actions et son état interne.
- *Interactif* : communique avec l'environnement et d'autres agents.
- *Adaptatif* : capable de modifier son comportement à partir de son expérience
- *Sociable* : l'agent est affable, agréable, aimable.
- *Mobile* : capable de se déplacer d'un environnement à un autre.
- *Mandataire* (proxy) : peut agir pour la gouverne d'une personne ou d'une entité. Autrement dit, agit comme un représentant, pour le bénéfice d'un autre.
- *Pro-actif* : n'agit pas simplement en réaction par rapport à son environnement mais est capable de prendre l'initiative.
- *Intelligent* : l'état de l'agent est formalisé par ses connaissances (croyances, buts, plans et suppositions) et il interagit avec d'autres agents via un langage symbolique.
- *Rationnel* : capable de choisir une action en se basant sur ses buts internes et les connaissances nouvelles que peuvent lui apporter ses actions.
- *Imprévisible* : capable d'agir de façon non prévisible et d'adopter un comportement non déterministe.
- *Temporairement continu* : c'est un processus s'exécutant de façon continue.
- *Caractère* : doté d'une personnalité et d'un état émotionnel.
- *Transparent* : doit être transparent lorsque requis et doit être capable de fournir une trace de ses activités sur demande.
- *Coordonnable* : capable d'effectuer une tâche dans un environnement partagé avec d'autres agents. Les tâches sont coordonnées via une planification, un workflow ou autres.
- *Coopératif* : capable d'être coordonné avec d'autres agents dans le but d'atteindre un but commun. N.B. : le terme *collaboration* est parfois utilisé comme synonyme de *coopération.*

- *Compétitif* : c'est le contraire de coopératif. Le succès d'un agent implique l'échec des autres.
- *Robuste* : capable de fonctionner avec des erreurs et des données incomplètes.
- *Fiable* : à qui on peut faire confiance.

La définition de Jennings, Sycara et Wooldridge [JSW98] est une des plus communément citées, aussi c'est celle que nous adopterons :
Un agent est un système informatique, situé dans un certain environnement, qui est capable d'agir de façon autonome et flexible afin d'atteindre les objectifs pour lesquels il a été conçu.
Les trois concepts clé de cette définition sont :

- *contextualisation* : l'agent peut recevoir des données sensorielles de la part de son environnement et il peut commettre des actes susceptibles d'avoir un impact sur ce dernier.
- *autonomie* (cf. ci-dessus).
- *flexibilité* : ce concept se définit par :
    - *Habilité sociale* : l'agent interagit avec d'autres agents et possiblement des humains via un langage quelconque de communication dans le but de résoudre son problème et d'aider les autres dans leurs activités.
    - *Réactivité* : l'agent perçoit son environnement et réagit aux changements qui y ont lieu dans un temps raisonnable.
    - *Pro-activité* (cf. ci-dessus).

Il est entendu que selon le contexte dans lequel l'agent évolue, ces caractéristiques ne s'exprimeront pas avec la même force.

Par ailleurs, dans le cadre du commerce électronique, on peut raffiner cette définition d'agent. Ainsi, Parkes et al. [PUF99] distinguent les agents *autonomes* des agents *semi-autonomes*. Un agent autonome doit avoir un ensemble complet d'informations et de préférences provenant de l'usager afin d'être en mesure de faire face (seul) aux différentes situations qui peuvent survenir, tandis que dans un cas de semi-autonomie, l'agent prend

des décisions lorsque l'usager lui a fourni l'information permettant de le faire et requiert l'intervention de celui-ci lorsque son action est mal définie par rapport à la situation courante. On parlera donc d'autonomie en termes de degrés ou de niveaux.

De même, dans le cas d'enchères électroniques, on fera une distinction supplémentaire entre un agent avec *prix de réserve* (« reserve-price agent ») et un agent avec *prix progressif* (« progressive-price agent »). Le premier est un agent autonome qui, lorsqu'il mise, ne peut dépasser le prix de réserve qui lui a été fixé. Ce type est approprié pour les usagers qui ont une évaluation précise de la valeur du bien qu'ils désirent. Le second est un agent semi-autonome qui est doté d'une borne inférieure et d'une borne supérieure sur le prix de réserve. Ainsi, l'agent se comporte de façon autonome tant que le prix demandé est en dessous de la borne inférieure et quitte la négociation lorsqu'il est au-dessus de la borne supérieure. Par contre, lorsque le prix est entre les deux bornes, l'agent sollicite l'intervention de l'usager en lui demandant soit de redéfinir les bornes, soit de quitter l'enchère. Ce deuxième type est adéquat lorsqu'il est difficile d'évaluer la valeur d'un bien qui est mis aux enchères [PUF99].

2.1.2 Familles d'agents

Comme on l'a vu, on peut différencier les agents par les caractéristiques et propriétés qui les composent. Mais on peut également classer les agents par leur mode de fonctionnement. On distingue en particulier trois grandes familles d'agents : les agents réactifs, les agents délibératifs et les agents hybrides [WJ95].

2.1.2.1 Agents réactifs

Un agent réactif est un agent dirigé par des événements extérieurs provenant de son environnement. Lorsqu'il perçoit un changement au sein de cet environnement, il réagit en conséquence, en respectant un schéma d'actions pré-établi, appelé comportement, sans effectuer de raisonnement sur l'environnement. C'est-à-dire qu'à chaque perception différente de l'extérieur est associé un comportement. Ces agents sont employés dans des

contextes où les contraintes de temps nécessitent une réaction rapide de type « action-réponse » et ont généralement une représentation interne du monde qui les entoure qui est très simplifiée, voire minimale. En fait, l'intelligence de ces agents est exprimée non pas en termes de capacité de raisonnement mais comme le produit de leur interaction avec leur environnement.

2.1.2.2 Agents délibératifs

Cette catégorie d'agent est un peu l'opposé de la catégorie précédente. En effet, dans cette approche, les agents possèdent, de façon explicite, une représentation symbolique de leur environnement. De plus, les décisions sont prises suite à un raisonnement (pseudo-) logique basé sur une manipulation symbolique et sur du filtrage (« pattern matching ») [WJ95]. Le raisonnement se fait en fonction du but de l'agent qui peut s'exprimer sous la forme d'une fonction d'utilité. Le raisonnement peut également prendre la forme d'un plan se présentant comme une suite d'actions qui, exécutées, permettront d'atteindre le but désiré.

Parmi les agents délibératifs, on retrouve un sous-ensemble particulier que constituent les agents BDI (acronyme anglais pour « Belief », « Desire », « Intentions »). Ceux-ci se définissent par un ensemble d'attitudes mentales : *croyances*, *désirs* et *intentions*. Par analogie, les croyances correspondent en fait aux informations que détient l'agent par rapport à son environnement. Les désirs représentent les buts de l'agent. Quant aux intentions, elles peuvent être associées aux actions que choisit l'agent.

2.1.2.3 Agents hybrides

Comme son nom l'indique, cette approche est une combinaison des deux précédentes. Cette architecture a pour but de répondre aux problèmes pour lesquels ni l'architecture réactive, ni l'architecture délibérative ne sont pleinement adéquates. Elle comporte donc aussi bien des aspects réactifs que des aspects délibératifs. Ceux-ci sont généralement séparés sous forme de couches structurées verticalement, auquel cas une seule couche est en contact avec l'environnement extérieur, ou horizontalement, auquel cas, toutes les couches sont en contact avec l'environnement extérieur.

2.1.3 Systèmes multi-agents

Un système multi-agents est un système mettant en œuvre plusieurs agents qui interagissent entre eux. Ce type d'infrastructure est généralement conçu pour la résolution de problèmes concurrents ou distribués. Ces problèmes sont généralement trop complexes pour les capacités et connaissances individuelles d'un seul agent, vu dans ce contexte comme un solutionneur de problème [DL89]. En fait, pour un système multi-agents non distribué, on pourrait très bien imaginer un agent capable de tout faire, mais un agent aussi gros en taille et en complexité serait assez déficient sur le plan de la performance, de la fiabilité, de la maintenance, etc., alors que diviser les fonctionnalités entre plusieurs agents présente les qualités suivantes : modularité, flexibilité, modifiabilité et extensibilité [Gro00].
Les systèmes multi-agents présentent les caractéristiques suivantes [JSW98] :

- chaque agent a des informations ou des capacités de résolution incomplètes. Ainsi, chaque agent a un point de vue limité;
- il n'y a pas de contrôle global du système;
- les données sont décentralisées;
- les calculs sont asynchrones.

Souvent, dans un système multi-agents, les agents sont amenés à interagir entre eux, c'est-à-dire à communiquer, à échanger de l'information ou encore, à surveiller les actions des autres agents. Ci-dessous, suivent les quatre principaux types d'interactions qu'on peut distinguer.

1) Coexistence

C'est, en quelques sorte, le cas où il y a absence d'interactions. Les agents évoluent dans le même environnement en ayant conscience ou non de l'existence d'autres agents. Toutefois, ils ne communiquent pas, ni épient les actions et décisions des autres. Pour résumer, on peut dire que les agents sont indifférents les uns aux autres.

2) Coordination

C'est un processus dans lequel les agents s'engagent dans le but d'assurer qu'une communauté d'agents individuels agissent de manière cohérente, c'est-à-dire, que leurs

actions ne soient pas conflictuelles les unes envers les autres [NLJ96]. Les différentes techniques de coordination seront passées en revue au chapitre 4.

3) Coopération

Des agents coopératifs sont des agents coordonnés qui, en plus d'avoir éventuellement un but individuel, partagent un but commun. Autrement dit, l'objectif principal d'un agent au sein d'un groupe d'agents coopérants est la réussite du groupe à atteindre le but collectif. La coopération des agents implique leur coordination, mais l'inverse n'est pas vrai. En effet, des agents peuvent avoir des buts individuels et être coordonnés, par exemple, pour l'utilisation d'une ressource commune.

4) Compétition

Dans un environnement compétitif, les agents agissent à l'opposé de la coopération et de la coordination. Ils ont pour objectif d'atteindre leur but individuel, même si cela se fait au détriment des autres. Cela peut avoir pour conséquence de générer des conflits entre les agents qu'ils devront résoudre.

2.1.4 Applications dans le commerce électronique

Les agents sont et peuvent être utilisés à plusieurs niveaux dans le commerce électronique. Pour faire le lien avec le modèle CBB[7], on peut dire que les étapes deux, trois et quatre du modèle, respectivement courtage du produit, courtage du marchand et négociation du produit, peuvent être assignés à des agents. Plus généralement, les activités de recherche et de collecte d'informations, de recherche de fournisseurs, de négociation, de surveillance de marché comme la bourse ou les enchères, mais aussi les activités inter- et intra-entreprises comme la gestion de processus d'affaires, sont toutes des activités propices à l'automatisation et donc sujettes à être confiées à des agents logiciels. En outre, ces tâches obéissent pour la plupart à un processus relativement bien défini, consomment beaucoup de temps, et accaparent des ressources qui pourraient être mieux utilisées. Des agents de type

[7] Consumer Buying Behavior, cf. section 1.1

délibératif ou hybride, coordonnés ou coopératifs lorsque le cas l'exige, seraient pertinents dans de tels cadres d'application.

2.2 Négociation automatique

Cette section se propose de définir le terme *négociation (électronique)*, explique le contexte dans lequel il est employé ici et définit en particulier ce qu'est une négociation automatique. On y décrit également les types de négociation auxquelles cet ouvrage fait référence. Enfin, on introduit le concept de négociation combinée. Notons qu'on désignera par les termes *item*, *produit*, *objet*, etc., aussi bien un service qu'un bien matériel, c'est-à-dire une instance qui possède une valeur et par conséquent à laquelle est associé un prix et qui peut donc faire l'objet d'une négociation.

2.2.1 Négociation

À l'instar des agents, il est difficile de trouver et donner une définition standard de ce qu'est une négociation. C'est un terme générique qui couvre plusieurs domaines et qui est employé dans divers contextes. Dans le cas présent, on se restreindra au cadre du commerce en général et du commerce électronique en particulier. D'ailleurs, dans ce dernier cas, on fait souvent usage des termes *négociations électroniques* ou *e-négociations* .

Leciwicki et al. [LSM99] décrivent les caractéristiques communes aux négociations comme suit :

- Il y a deux parties ou plus, c'est-à-dire plusieurs individus, groupes ou organisations.
- Il y a un conflit d'intérêt entre deux ou plus des parties, c'est-à-dire que ce que l'un veut n'est pas nécessairement ce que l'autre veut, et les parties doivent chercher une façon de résoudre ce conflit.
- Les parties négocient car elles pensent qu'elles peuvent exercer une certaine forme d'influence qui leur permettra de faire une meilleure affaire (« to get a best deal ») plutôt que de simplement prendre ce que l'autre offre volontairement.

- Les parties, au moins pour un moment, préfèrent essayer de trouver un accord plutôt que de s'affronter jusqu'à ce qu'une des parties finisse par capituler ou que les parties soient obligées de s'en remettre à une tierce partie pour trancher.
- Lorsqu'on négocie, on s'attend à des concessions.
- Une négociation réussie implique la prise en compte de critères implicites et la détermination de critères explicites (p. ex., le prix).

D'autre part, Guttman et al. [GMM98] présentent la négociation comme une forme de prise de décision où deux ou plusieurs parties cherchent conjointement dans l'espace possible des solutions avec pour but d'atteindre un consensus.

Enfin, Strobel affirme qu'une négociation a lieu lorsqu'un accord ne peut pas être conclu sur la base de l'offre initiale faite par une des parties ou parce que l'accord peut potentiellement être optimisé et les parties opérant la transaction sont prêts à discuter de leurs offres [Str99]. De plus les négociations peuvent se classer selon deux dimensions principales : elles peuvent être distributives versus intégratives et bilatérales versus multilatérales. Dans une négociation distributive, un seul critère est sujet à négociation et les parties impliquées ont des intérêts opposés, tandis que dans une négociation intégrative, plusieurs critères sont négociés et les parties impliquées n'ont pas les mêmes préférences quant à ces critères. Dans une négociation bilatérale, deux parties sont impliquées seulement versus une négociation multilatérale où plusieurs parties le sont. La négociation bilatérale peut être vue comme une négociation *un-à-un* tandis que la négociation multilatérale peut être vue comme une négociation *un-à-plusieurs* ou *plusieurs-à-plusieurs.* Cette classification n'est pas nécessairement fixe pour une négociation en cours dans le sens où pendant son déroulement elle peut évoluer, par exemple, de distributive à intégrative ou de multilatérale à bilatérale.

Strecker [Str01a], après recoupement de différentes définitions de négociation propose la définition suivante :

- c'est un processus de communication et de prise de décisions,
- dans lequel deux ou plusieurs parties,

- veulent résoudre un conflit d'intérêts,
- car chaque partie ne peut atteindre ses objectifs individuellement et
- chaque partie n'est pas prête à accepter ce que l'autre partie offre volontairement (de son plein gré)
- en cherchant et décidant conjointement d'une solution individuelle préférable.

Il est à noter que cette définition est suffisamment générale pour s'appliquer à d'autres domaines que le commerce tels que la politique, par exemple. Cette dernière définition ainsi que la classification de Strobel discutée ci-haut [Str99] sera conservée tout au long de ce mémoire.

Les deux principales formes de négociation sont le marchandage (« *bargaining* ») et les enchères (incluant les appels d'offre). Le marchandage a lieu entre deux parties seulement et par conséquent consiste en une suite d'offres et de contres-offres. Il peut concerner plusieurs aspects de l'item négocié et, à ce titre, peut être vu comme une négociation bilatérale intégrative tandis que les enchères qui impliquent plusieurs participants négociant principalement le prix de l'item peuvent être considérées comme des négociations multilatérales distributives [Str99]. Les enchères sont décrites en détail à la section 2.3.

2.2.2 Négociation électronique

Dans une négociation électronique [Str01b], les messages échangés par les participants se font par le biais d'un medium électronique tel que le courriel. Les participants peuvent aussi bien être des humains que des agents logiciels et l'échange des message peut être non structuré (par exemple : un clavardage) ou très structuré comme dans le cas d'une enchère électronique. Enfin, le processus peut être entièrement ou partiellement automatisé.

2.2.3 Négociation automatisée

Parfois désignée sous les appellations de *négociation automatique* ou de *négociation assistée par agents* (« agent-mediated negotiation »), la négociation automatisée est un

sous-ensemble des négociations électroniques qui implique exclusivement des agents logiciels. Le degré d'autonomie que l'usager donne aux agents détermine si la négociation est complètement automatisée ou semi-automatisée. Ainsi, si, durant le processus de négociation, l'agent sollicite l'intervention de l'usager, on est alors dans une configuration semi-automatisée, tandis que si l'agent conduit la négociation jusqu'à son terme sans intervention aucune d'un humain, on est alors dans une configuration automatisée.

Jennings et al. clament qu'il y a trois aspects importants à considérer sur le plan des négociations automatiques [Jen01]:

a. les protocoles de négociation

C'est l'ensemble des règles qui gouvernent l'interaction. Cela couvre les types de participants permis (p. ex. : les vendeurs et les acheteurs), les états de la négociation (p. ex. : soumission des mises en cours, négociation fermée), les événements qui déclenchent la transition d'un état à un autre (exemple : plus de soumission de mises, offre acceptée) et les actions valides des participants étant donné un état de la négociation (p. ex. : quels messages peuvent être envoyés à qui, par qui et à quel étape).

b. les objets de la négociation

Ce sont les critères sur lesquels un accord doit être conclu. L'objet peut se limiter à un seul critère (comme le prix) ou peut couvrir un ensemble de critères (prix, qualité, quantité, termes et conditions, etc.). Les participants négocient un accord, représenté par une structure satisfaisante de l'objet, par le biais d'actions qui respectent le protocole établi à l'avance. Ces actions peuvent modifier les valeurs de la structure de l'objet, ou encore l'altérer, si le protocole le permet.

c. les modèles de prise de décision des agents

C'est l'appareil de prise de décision que les participants emploient pour atteindre leurs objectifs tout en respectant le protocole de négociation. La complexité du modèle dépend du protocole, de l'objet de la négociation et des opérations permises au cours du processus. Le modèle de prise de décision correspond à ce qu'on désigne plus

communément sous le nom de *stratégie*. Dans la suite de ce mémoire, on emploiera aussi le terme *comportement* pour désigner la stratégie d'un agent. Ce terme est cependant plus générique et désigne de façon générale comment un participant réagit en termes d'actions pendant la négociation.

L'importance relative de ces trois aspects varie selon le type de négociation et le contexte environnemental. Ainsi, dans le cas d'une enchère, l'emphase est mise sur le protocole, le nombre d'opérations possibles est relativement restreint et le modèle de prise de décision peut se résumer à soumettre son évaluation du prix dans le cas d'une enchère à enveloppes scellées (cf. section 2.3.3).

Par ailleurs, comme le soulignent Beam et Segev [BS97], il y a plusieurs difficultés à surmonter dans le cadre de la négociation automatisée, dont la définition d'une ontologie et la question des stratégies que les agents doivent adopter. L'ontologie permet de classer et catégoriser des objets de façon à ce qu'il soient sémantiquement significatifs pour un agent. Elle permet ainsi d'assurer que deux agents réfèrent exactement au même produit. Quant au problème de la stratégie de négociation de l'agent, c'est un problème majeur. En effet, c'est toute la qualité de la stratégie qui fera en sorte que l'agent soit capable d'acquérir le produit demandé au prix le plus avantageux pour l'usager qu'il représente. Une stratégie trop simpliste est une stratégie qui est facilement devinable par les adversaires (agents ou humains) de l'agent et qui peut donc l'amener (et donc le client) à payer un prix trop élevé ou encore, dans un environnement de compétition, l'amener à ne pas obtenir le produit désiré.

2.2.4 Système de support à la négociation

Maintenant que nous avons défini ce qu'est une négociation, on peut définir ce qu'est un système de support à la négociation (SSN). Comme son nom l'indique, un SSN est un outil de support électronique pour les négociations. Un SSN doit éliminer les différentes barrières de communication qui existent entre les parties et doit offrir un service d'analyse et d'aide à la prise de décision. [JF89]

2.2.5 Place de marché

Pour finir, nous définissons brièvement le concept de place de marché, ce terme étant quelquefois employé dans la suite de ce mémoire. Par définition, le marché est la résultante de la rencontre entre l'offre et la demande d'un bien. De là, par extension, la place de marché est le lieu où acheteurs et vendeurs échangent. La place de marché désigne par conséquent un lieu où se tiennent des négociations. La bourse est un parfait exemple de place de marché dans laquelle des acheteurs et des vendeurs s'échangent des actions. Dans le commerce électronique, la place de marché est bien entendu virtuelle et correspond généralement à un portail Internet.

2.3 Les enchères

Les enchères ou ventes aux enchères ou encore ventes à l'encan, sont un type particulier de négociation [KF98]. Ce sont des mécanismes simples et éprouvés qui permettent la vente de produits qu'on ne trouve pas couramment sur le marché, qui sont présents en nombre limité et dont la valeur est incertaine [PUF99]. Wolfsetter définit une enchère comme un mécanisme de mises décrit par un ensemble de règles qui spécifient comment un gagnant est déterminé et combien il doit payer [Wol94]. À ce titre, les appels d'offre peuvent être considérés comme des enchères, dans la mesure où le mécanisme est le même, à la différence qu'il s'agit d'une proposition d'achat pour laquelle des vendeurs font des offres de vente. Mais ce cas n'est pas traité dans ce mémoire.

De fait, les enchères étaient très prisées sous l'empire romain et sont encore très répandues de nos jours où elles sont utilisées pour la vente d'objets d'art par des maisons telles que Christie's ou Sotheby's, pour l'attribution de fréquences de radiocommunications aux entreprises de télécommunication notamment par le FCC américain (« Federal Communication Commission » [Fcc02]) ou encore pour la vente en gros du poisson dans les ports de pêche. Toutefois, avec l'avènement d'Internet, les enchères se sont ouvertes à un vaste public et touchent une gamme plus large de produits, comme la vente d'ordinateurs par exemple. Dans une enchère, l'encanteur, appelé également commissaire

priseur ou encore vendeur aux enchères, est la personne qui procède à l'estimation et à la vente des items.

Il existe plusieurs types d'enchères [Ago01]: elles peuvent être progressives (ascendantes ou descendantes), ouvertes (publiques) versus fermées (à enveloppes scellées) et peuvent être au premier prix versus au second prix. Par enchère ouverte, on entend que chaque offre est connue des autre participants à l'inverse de l'enchère fermée.

Enchère	Sens	Type	Prix payé	Exemple
Double continue	_	ouverte	_	Bourse
Hollandaise	descendante		premier prix	denrée périssable (poisson, fleurs)
Anglaise	ascendante			pièce de collection (eBay)
Enveloppes scellées		fermée		appel d'offre
Vickrey			second prix	FCC

Tableau 1 - Caractéristiques des principales enchères

Les sections qui suivent présentent les quatre enchères[8] de base, telles qu'établies par Vickrey [Vic61] ainsi que l'enchère double (cf. tableau 1) qui est très utilisée par les institutions financières américaines. D'autres types d'enchères existent également, mais ce sont souvent des variantes de celles-ci et sont moins répandues.

Mais avant d'aller plus loin, il faut définir la notion de *prix de réserve* afin d'éviter toute confusion par la suite. Si on est acheteur, le prix de réserve désigne le prix maximal que l'on est prêt à payer pour l'item que l'on désire acquérir. Il représente en quelque sorte notre évaluation de sa valeur. Similairement, si on est vendeur, le prix de réserve désigne le

[8] le monde de la finance a parfois des définitions différentes de ces enchères, ce qui peut amener une certaine confusion

prix minimum pour lequel on est disposé à céder l'item. Il représente donc la valeur seuil en dessous de laquelle on ne peut ou veut vendre.

2.3.1 L'enchère anglaise

C'est la plus connue et la plus répandue. Elle est ouverte, ascendante et au premier prix. Milgrom la définit de la façon suivante [Mil89]:

> L'encanteur commence par annoncer le prix acceptable le plus bas, nommé prix de réserve, puis procède en sollicitant des mises de plus en plus élevées des participants jusqu'à ce que plus personne ne soumette de mise. L'item est alors adjugé (vendu) au plus offrant.

Chaque nouvelle mise doit respecter un incrément minimum *i*, c'est-à-dire que la mise *n+1* doit être supérieure à la mise *n* d'au moins *i*. La somme de la mise *n* et de *i* est ce qu'on appelle le prix demandé. Parfois, l'enchère débute au montant zéro, le prix de réserve est alors conservé secret et l'item n'est adjugé que si le prix de réserve a été atteint ou dépassé.

L'enchère anglaise est caractérisée par une compétition très forte de la part des participants, ce qui a souvent pour conséquence qu'un item soit vendu à un prix bien supérieur à sa valeur réelle, particulièrement lorsque certains des participants sont inexpérimentés. Ce phénomène est appelé « la malédiction du gagnant ». C'est donc, on s'en doute, une enchère très appréciée par les vendeurs.

2.3.2 L'enchère hollandaise

Utilisée notamment pour la vente de fleurs et la vente de poissons, c'est une enchère ouverte, descendante et au premier prix. Ici, l'encanteur débute en annonçant un prix initial très supérieur à la valeur de l'item. Puis, à intervalles réguliers, il décrémente le prix d'un montant donné, jusqu'à ce que une offre d'achat soit soumise. Dans ce cas donc, l'adjudication est faite lorsque le prix correspond à l'estimation d'un des participants. Il peut arriver que plusieurs participants désirent l'item au prix annoncé, il y a alors deux

façons de résoudre le conflit. La première est la politique du premier arrivé, premier servi, où c'est le premier participant à avoir fait une offre qui l'emporte. La deuxième consiste à démarrer une enchère anglaise à partir du prix sur lequel l'enchère s'est arrêtée.

2.3.3 L'enchère à enveloppes scellées au premier prix

Comme l'enchère anglaise, elle est ascendante et un prix de réserve est annoncé. Toutefois, ici les participants ne soumettent qu'une seule mise qui n'est pas dévoilée aux autres participants, d'où le nom de l'enchère. L'adjudication est également faite au plus offrant. Étant donné que les participants n'ont droit qu'à une seule mise, on se doute qu'il est important de bien réfléchir à celle-ci.

2.3.4 L'enchère Vickrey, dite enchère à enveloppes scellées au second prix

Cette enchère fonctionne comme l'enchère présentée à la section précédente à la différence qu'au moment de l'adjudication l'item est toujours attribué au participant ayant soumis la plus haute mise, sauf que contrairement au premier prix, le gagnant va payer le montant soumis par le deuxième plus haut enchérisseur. Ce système permet d'encourager la compétition en encourageant les participants à soumettre des mises qui reflètent leur estimation réelle de la valeur de l'item. En effet, les compétiteurs n'ont plus à craindre la malédiction du gagnant puisqu'ils sont assurés de payer le second prix le plus haut. Autrement dit, le prix que va payer le gagnant ne dépend pas de son unique action, mais bien du comportement des autres participants.

2.3.5 L'enchère double continue

L'enchère double [Str01b] admet plusieurs acheteurs et vendeurs en même temps. Une correspondance est faite entre les mises (achat) et offres (vente) dans l'ordre dans lequel elles arrivent. Ainsi, lorsqu'une nouvelle mise est traitée, l'encanteur vérifie si le prix offert correspond avec la moins chère des offres de vente, et inversement. Lorsqu'une correspondance est établie, l'encanteur retire le prix courant et publie un nouveau prix. Une transaction est complétée lorsqu'une mise (ou offre) est acceptée par un des participants.

L'enchère double continue est particulièrement utilisé dans des marchés boursiers où il y a ventes et achats d'actions simultanément.

2.3.6 Enchères multi-items

Dans les enchères précédentes, les objets en vente sont généralement disponibles en quantité unique seulement ou sous la forme d'un ensemble indissociable. Cependant, il est possible de vendre plusieurs unités à la fois d'un même produit. Il est bien sûr possible de faire une enchère pour chacune de ces unités, mais le plus souvent elles sont offertes dans une même enchère et distribuées entre les participants selon leurs mises [Ben01].

Dans le cas d'une enchère anglaise, l'acheteur ayant soumis la plus haute mise obtient à l'adjudication la quantité qu'il a demandé, et les acheteurs ayant les plus hautes mises suivantes obtiennent le maximum d'unités possibles par rapport à leur demande et la quantité qui reste. Chacun paye le montant exact de son offre.

Pour une enchère hollandaise à plusieurs items, les items sont adjugés au fur et à mesure que les offres sont faites et l'enchère prend fin quand ils sont tous vendus. Il existe deux variantes : le cas discriminatoire dans lequel les gagnants payent le montant de leur offre et le cas uniforme où les gagnants payent le montant de l'offre du dernier gagnant.

Par ailleurs, similairement à l'enchère anglaise, dans l'enchère à enveloppes scellées au premier prix, les objets sont répartis entre les participants par ordre décroissant des plus hautes mises et ces derniers payent le montant de leur mise.

Pour finir, dans une enchère de Vickrey multi-items où l'on suppose qu'il y a k items à vendre et n participants désirant chacun un seul item, les k premiers participants obtiendront chacun un item par ordre décroissant du montant de leur mise et payeront tous le montant de la $(k+1)$ ième plus haute offre.

2.3.7 Enchères combinatoires

Dans une des enchères présentées ci-dessus, si une personne désire acheter un ensemble d'items séparés mais complémentaires, elle doit s'engager dans plusieurs enchères qui se déroulent de façon séquentielle ou parallèle. Cela devient très vite complexe à gérer. En effet, pour savoir combien miser dans une des enchères il faut pouvoir estimer combien d'argent on va dépenser dans les autres enchères. Dans le cas séquentiel, c'est très difficile car les autres enchères n'ont pas encore débuté et dans le cas parallèle le problème peut paraître moins complexe mais il l'est car même si on peut surveiller les autres enchères parallèles il faut prévoir le comportement des autres acheteurs. L'enchère combinatoire permet à un acheteur de soumettre une offre d'achat sur une combinaison de produits complémentaires. Ainsi, on n'est plus obligé de prévoir ce qui va se passer plus loin dans le temps au niveau des autres enchères. Par contre, la difficulté réside du côté de l'encanteur qui doit déterminer qui est le gagnant à partir des combinaisons de prix qu'il reçoit [San00]. Ce modèle a notamment été adopté pour les enchères du FCC (cf. section 2.3). Toutefois, ce modèle est rarement appliqué du fait de la complexité de l'allocation des biens.

2.3.8 Quelques sites d'enchères en ligne

Plusieurs centaines de sites de vente aux enchères existent maintenant sur Internet, dont plusieurs rencontrent un fort succès. Le format de l'enchère anglaise est le plus répandu, mais on trouve aussi des enchères hollandaises, des enchères à enveloppes fermées et des enchères doubles. Voici un aperçu des plus populaires.

eBay [eBay02] est certainement le site le plus connu et celui qui génère le plus de revenu ($70,000,000 en 1998). Sur eBay, des particuliers vendent leurs articles via une enchère anglaise à d'autres particuliers. On peut ainsi vendre toutes sortes d'articles, du sac de billes aux timbre de collection en passant par les motos. Par ailleurs, eBay offre, comme beaucoup, un service de *proxy bidding*, c'est-à-dire un agent à qui on spécifie notre prix de réserve, et qui sans aucune intelligence, va miser automatiquement en augmentant le prix

de l'incrément minimum jusqu'à ce que le prix de réserve soit atteint ou que le produit lui soit adjugé. Généralement, l'enchère a une durée de vie de quelques jours.

EggHead [Egg02], anciennement onSale, propose lui aussi des enchères anglaises. Toutefois, il s'agit d'un site marchand, c'est-à-dire qui vend ses propres produits. On peut ainsi acheter essentiellement des produits électroniques comme des ordinateurs, des appareils photos, etc., mais aussi des voyages. À la différence de eBay, eggHead simule le fameux *une fois, deux fois, trois fois adjugé!* (*going going... gone!*): lorsqu'il reste moins d'une minute avant la fin de l'enchère et que quelqu'un mise (humain ou agent logiciel), alors l'enchère est relancée pour une courte période, appelée période d'extension, afin de laisser l'opportunité à d'autres acheteurs de répliquer. Autrement, il est facile d'attendre les dernières secondes de l'enchère pour miser et être sûr d'emporter la mise. Il y a d'ailleurs un terme anglais pour désigner ce comportement : le « sniping ».

Enfin, Yahoo!Auctions [Yahoo02] propose lui aussi un large éventail d'enchères, toutes anglaises également. Cependant, pour ce qui est de la période d'extension d'une enchère, Yahoo!Auctions se distingue de ses deux concurrents en offrant le choix au vendeur. En effet, celui-ci décide si l'enchère est prolongée ou non en cas d'activités dans la période précédant la fermeture de l'enchère.

Pour plus de détail, il est conseillé de se reporter à une étude menée en août 1999 sur 142 sites d'enchères en ligne, couvrant les différents aspects de ce domaine [Luc99].

2.4 Négociation combinée – définition

Dans le cadre du projet CONSENSUS, nous avons introduit un nouveau type de négociation qui se définit comme suit :

> *L'usager s'engage dans plusieurs négociations pour obtenir les items d'un ensemble (package) où :*
> *Les items sont :*
> - *inter-reliés et*
> - *négociables.*
>
> *Les négociations sont :*
> - *indépendantes les unes des autres et*

- *possiblement de différents types.*

Il est à noter que dans le contexte électronique, les négociations peuvent se dérouler sur un même serveur ou sur des serveurs différents, en autant que le critère d'indépendance des négociations soit respecté.

2.5 Popularité des enchères

On pourrait se demander pourquoi les enchères sont populaires au sein du commerce électronique et sont appelées à devenir une norme pour les achats effectués via Internet. En fait, les enchères se réduisent souvent à une négociation sur le prix. Elles sont une interaction entre un vendeur et plusieurs acheteurs et les règles de négociation sont généralement simples, claires et éprouvées. Pour toutes ces raisons, il apparaît facile d'automatiser le fonctionnement d'une enchère et de créer des agents qui joueraient le rôle des participants. Les agents n'ont en effet pas de difficulté à obéir à des règles relativement stables et énoncées clairement. Toutefois, comme le soulignent Beam et Segev [BS97], il existe certaines difficultés à l'emploi d'agents dans les négociations dont le manque d'ontologie et la difficulté de doter les agents de bonnes stratégies.

Chapitre 3: Contexte et environnement de recherche

Ce chapitre présente le cadre de recherche dans lequel s'inscrit le travail présenté ici. En premier lieu, on introduit le cadre générique, puis le cadre spécifique et enfin les travaux connexes.

3.1 « Towards Electronic Marketplaces »

Le projet CONSENSUS décrit à la section 3.2 est une sous-partie du projet de recherche « Towards Electronic Marketplaces » (TEM). Ce dernier est le fruit d'un partenariat entre l'industrie et le milieu universitaire qui a vu le jour en 1999 et qui implique aussi bien des chercheurs en économie et en recherche opérationnelle qu'en génie logiciel. Un des axes majeurs de TEM est la conception de mécanismes génériques de négociation pour les marchés électroniques. Dans cette optique, un environnement de négociation générique du nom de GNP (« Generic Negotiation Platform ») [BKL+00] a été développé. Cette plate-forme est présentement fonctionnelle et supporte plusieurs types de négociation, comme l'enchère anglaise et l'enchère hollandaise pour ce qui est de celles qui nous intéressent ici. En outre, elle offre aux participants deux manières de négocier :via une interface web ou via des agents logiciels. C'est cette dernière option, exploitée dans CONSENSUS, qui a fait de GNP notre choix comme serveur de négociation. Une description détaillée de tous les axes de TEM se retrouve dans [BCG+01]; ci-dessous, on s'intéresse exclusivement à l'axe GNP de TEM.

3.1.1 Description de GNP

Dans la description qui suit, nous avons conservé les termes anglais car ce sont les termes originaux employés dans la conception et la documentation de la plate-forme.

Sur le plan technique, GNP repose sur les technologie Java [Java02], J2EE [J2EE02] et XML [XML02]. Les mécanismes permettant à des agents logiciels de négocier sur cette plate-forme seront présentés plus en détail au chapitre 5.

Sur le plan architectural, GNP se découpe en deux couches principales. La première, générique, gère les aspects de communication, les événements et la notion de temps (timing). Quant à la deuxième, elle est fonction du type de la négociation et gère les documents impliqués à partir de règles préalablement rédigées sous la forme d'un script. L'exécution de ce script joue le rôle d'encanteur. Les documents en question se répartissent en trois catégories : documents d'entrée, documents internes et documents de sortie.

- Documents d'entrée :
    - *Rules* : contient les informations statiques et régit la négociation.
    - *Product Reference* : contient les informations sur le produit à négocier.
    - *Order* : le participant soumet un ordre de vente ou d'achat pour un produit dans une négociation. C'est équivalent à une mise lorsque c'est un ordre d'achat.

  Le document nommé annonce (*announce*) est composé de ces trois documents et est utilisé par un participant pour initier une nouvelle négociation.

- Documents internes :
    - *Negotiation* : contient les informations dynamiques sur une négociation en cours. Ce document n'est pas accessible aux participants.

- Documents de sortie :
    - *Quote* : ce document est construit suite à chaque ronde et contient les informations accessibles et éventuellement affichées aux participants.
    - *Response* : ce document est construit à chaque fois qu'un ordre parvient à l'encanteur.
    - *Adjudication* : ce document est construit lorsque survient la fin de la négociation. C'est une association entre un ordre de vente et un ordre d'achat.

Une négociation dans GNP fonctionne comme une séquence de rondes. À chaque ronde, les participants (humains ou agents logiciels) reçoivent un ensemble d'informations publiques et privées et peuvent effectuer un ensemble d'actions prédéfinies. Dépendamment du type de négociation, une nouvelle ronde est initiée en fonction de facteurs tels que le temps écoulé ou le nombre de participants ayant interagi durant la ronde.

Plusieurs types de marché ont déjà été implantés dans GNP [GTLT00]. Il y a entre autres l'enchère anglaise ouverte simple multi-unitaire, l'enchère au premier prix à enveloppes scellées également multi-unitaire, l'appel d'offre mono-unitaire, l'enchère double continue mono-unitaire et l'enchère double périodique mono-unitaire.

3.1.2 Scénario d'utilisation de GNP

Ci-dessous, à la figure 1, est présenté le principal cas d'utilisation de GNP, où l'on peut voir tous les acteurs et leurs rôles au sein du processus de négociation [GTLT00].

Reader

Le *reader* (lecteur) est un participant générique. Les *sellers* (vendeurs), *buyers* (acheteurs), *announcers* (annonceurs) et les *submitters* (soumetteurs) sont des *readers*. Ils peuvent parcourir la liste des produits, demander de l'information sur les négociations en cours, regarder les cotes publiques et finalement sélectionner celles sur qui ils vont réagir et soumettre une offre.

Announcer

Un *announcer* initie la négociation. Il le fait en trois grandes étapes:

1. Il utilise le rôle de *Rules Editor* (éditeur de règles) pour choisir un patron d'enchère et définir les paramètres variables comme le temps de fermeture de l'enchère.
2. Il utilise le rôle de *Product Editor* (éditeur de produit) pour donner la référence à la description complète du produit.
3. Il utilise le rôle de *Negotiator* (négociateur) pour définir la quantité et le prix initial (ou minimum) et émettre l'ordre initial.

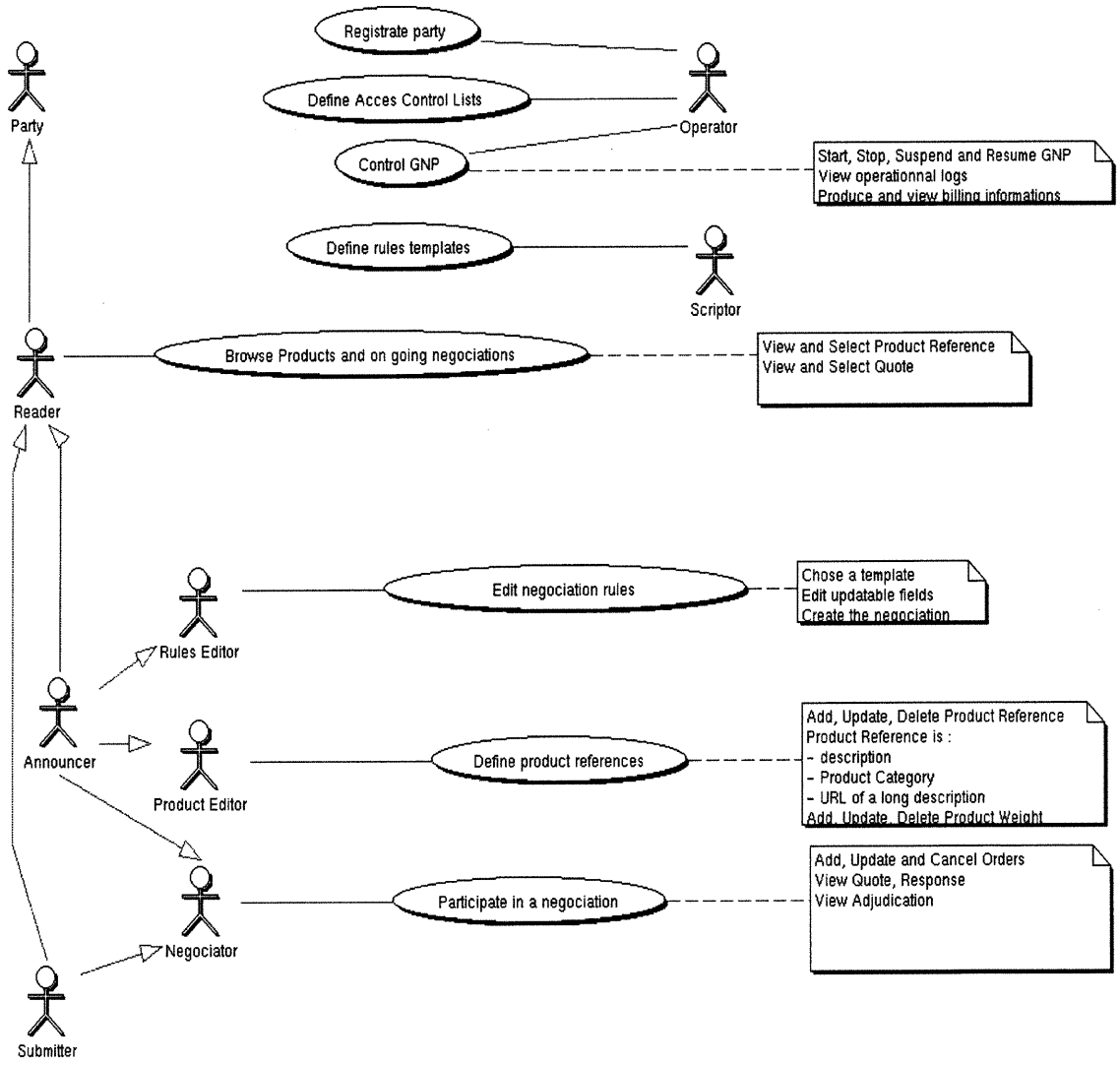

Figure 1 - Scénario d'utilisation de GNP [GTLT00]

Negotiator* et *Submitter

Le *negotiator* émet les ordres d'achat ou de vente. Il peut être soit un vendeur, soit un acheteur. Il peut également être l'*announcer* faisant sa mise initiale pour vendre ou acheter. Enfin, il peut être un *bidder* (miseur) qui soumet des ordres qui spécifient le prix et la quantité et la référence du produit.

Administrator* et *Operator

Les administrators et les operators peuvent gérer:

- les parties,
- les marchés et
- les négociations.

Ils peuvent ajouter, modifier et supprimer soit les propriétés des parties soit les marchés. En plus, ils peuvent suspendre et reprendre des marchés ou des négociations.

3.2 CONSENSUS

Dans cette section, nous allons présenter le projet CONSENSUS [BAVK01] dans lequel s'inscrit le travail rapporté dans ce mémoire. Plus précisément, nous allons introduire le concept de négociation combinée, exposer les besoins particuliers créés par ce type de négociation et décrire comment CONSENSUS va essayer de les combler. On précisera également la position de CONSENSUS par rapport à GNP (cf. la section 3.1).

3.2.1 Introduction

Comme il a été dit à la section précédente, CONSENSUS est un sous-projet de TEM.
Le projet s'intéresse aux négociations combinées. Dans ce type de négociation, une personne désire acquérir un ensemble de plusieurs items, produits ou services, et pour cela s'engage dans plusieurs négociations en même temps. Celles-ci peuvent être de types différents (enchère anglaise, enchère hollandaise, etc.) et sont généralement indépendantes les unes des autres. Au contraire, les produits impliqués dans la négociation combinée son

interdépendants. Pour illustration, on peut prendre l'exemple d'une personne - on la nommera Paul - désirant s'offrir des vacances de ski. Il y a trois items à se procurer : un billet de transport (avion, bateau, train, etc.), une chambre d'hôtel et enfin le forfait de ski. Il est clair que ces trois produits sont liés par un ensemble de contraintes et dépendances. Par exemple, notre vacancier doit se rendre à destination avant que ne commence son forfait de ski et ne repartir qu'après qu'il soit fini. Supposons maintenant, que les trois parties qui composent le voyage soient négociables sur des serveurs présents sur Internet. Rappelons que le type de négociation peut différer d'un serveur à l'autre et que les serveurs étant indépendants les uns des autres, les négociations le sont également. Il appartient alors à Paul de gérer et de coordonner les négociations. Ceci n'est pas forcément aisé et peut mener le vacancier à manquer de belles opportunités. On peut facilement imaginer le cas où pendant que Paul est en train de négocier sa chambre d'hôtel, il soit en train de manquer un tarif spécial sur un billet d'avion.

De là, apparaît le besoin d'avoir une infrastructure de support pour les négociations combinées qui aide l'usager à gérer l'ensemble des négociations qui se déroulent en parallèle. Ce système est désigné sous l'appellation CNSS (« Combined Negotiation Support System »).

3.2.2 Négociations-OU et Négociations-ET

Dans une négociation combinée, on peut s'engager dans plusieurs négociations pour l'obtention d'un item. Par exemple, dans le cas du voyage de ski, plusieurs négociations peuvent être engagées pour le billet d'avion. Ces négociations sont appelées *négociations-OU* (« OR-negotiations »). De même, les négociations pour chacun des différents items qui composent le forfait de vacances sont appelées *négociations-ET* (« AND-negotiations »).

Négociations-OU

Les négociations-OU sont des négociations parallèles effectuées pour un seul item. Elles sont démarrées en même temps afin de pouvoir gagner du temps (versus des négociations séquentielles) et de maximiser les chances d'obtenir le meilleur prix possible. De plus,

l'usager peut suivre l'évolution de chacune de ces négociations et décider de participer (en misant, faisant une offre, etc.) dans la plus prometteuse d'entre elles. Il faut toutefois s'assurer que les engagements pris sont mutuellement exclusifs, c'est-à-dire qu'un seul engagement est pris à la fois. Sinon, on peut se retrouver à conclure plusieurs transactions pour le même item. Bien sûr, rien n'empêche l'usager de démarrer les négociations-OU de façon séquentielle, en démarrant une nouvelle négociation à chaque fois que la négociation en cours se termine par un échec.

Négociations-ET

Là aussi, l'usager a le choix d'exécuter les négociations de façon séquentielle ou parallèle. Cependant, l'exécution parallèle permet à l'usager d'exploiter les événements qui surviennent dans une négociation pour prendre des décisions optimales dans une autre.

3.2.3 Scénario d'utilisation de CONSENSUS

Ce scénario est basé sur l'exemple précédemment exposé. L'usager, par l'intermédiaire de CONSENSUS s'engage, mettons, dans une négociation pour l'hôtel, deux négociations différentes pour le transport et enfin dans une dernière pour le forfait de ski. Par la suite, toujours à l'aide de CONSENSUS, il construit un workflow modélisant la négociation combinée. Ce dernier permet de représenter le séquencement ou le parallélisme entre les négociations ainsi que leurs dépendances. Un exemple de dépendances modélisables de cette façon est la somme d'argent déjà dépensée et la somme d'argent encore disponible. Une fois le workflow achevé, quatre agents logiciels sont instanciés et assignés aux quatre négociations impliquées. Bien entendu, l'instanciation de chaque agent respecte le type de négociation et le type de serveur sur lequel celle-ci se déroule. L'usager peut également doter les agents de règles de stratégies telles que : « si tes opposants ont un comportement agressif alors soit moins agressif ». Il peut ensuite démarrer la négociation combinée et en suivre la progression avec possibilité d'intervention via un outil de visualisation adéquat.

3.2.4 Architecture de CONSENSUS

Au sein du projet, nous avons développé et implanté une architecture (cf. figure 2) pour un CNSS, appelée CONSENSUS, qui est basée sur les technologies[9] de systèmes de gestion de workflow, la technologie agent et sur les moteurs d'inférence de règles. Pour rappel, un workflow est l'automatisation d'un processus d'affaire, en partie ou dans sa totalité, durant laquelle des documents, des informations ou des tâches sont transmises d'un participant à un autre en respect avec un ensemble de règles procédurales [WMC99]. Un processus d'affaire peut se définir comme un ensemble d'une ou plusieurs activités, qui, collectivement concrétisent un objectif. Un système de gestion de workflow est donc un système qui permet de définir, exécuter et gérer des workflows.

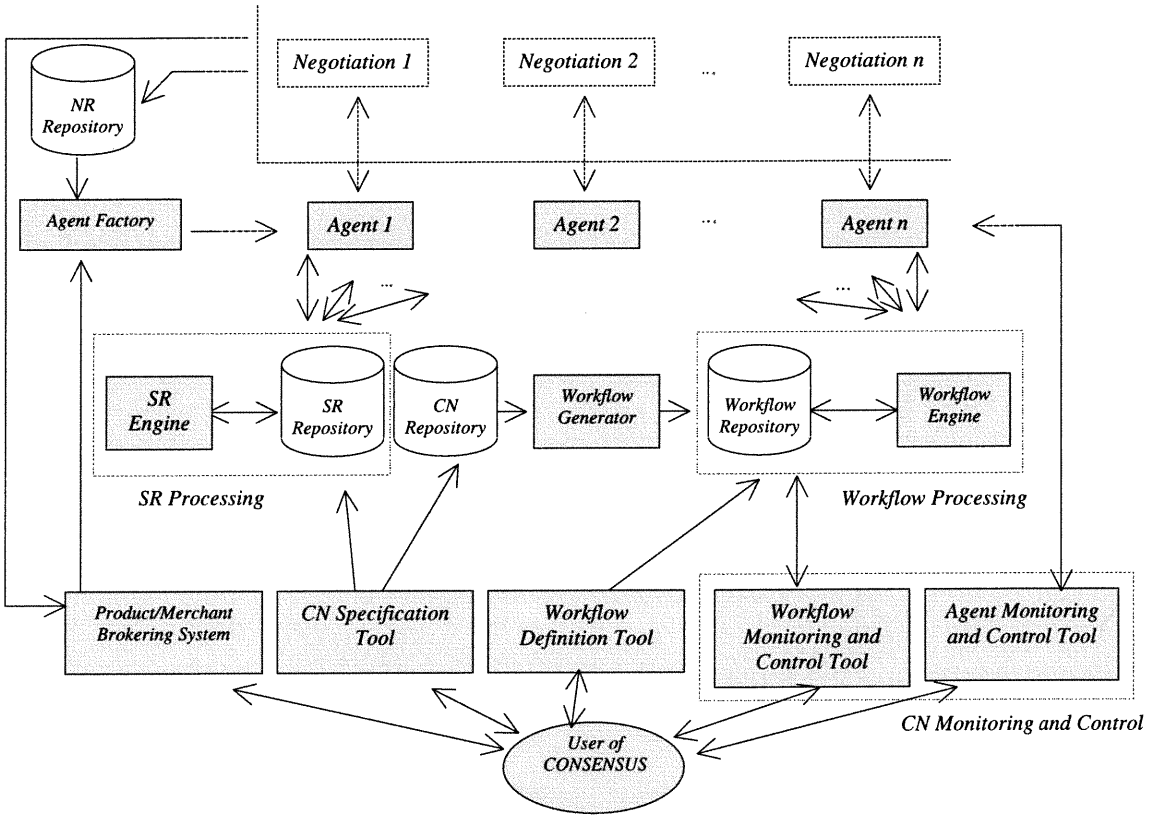

Figure 2 - Architecture de CONSENSUS [BAVK01]

[9] Les principaux choix technologiques et architecturaux de CONSENSUS [BAVK01] ont été effectués dans le cadre d'un projet de doctorat encore en cours de réalisation au moment de la rédaction de ces lignes.

Commençons par décrire le module de courtage, nommé *Product/Merchant Brokering System*. Il est utilisé par l'usager pour sélectionner les produits et les fournisseurs avec qui il désire négocier. Le système retourne alors la liste possible des négociations s'accordant avec sa sélection et l'usager choisit celles qui l'intéressent. Par la suite, cette liste est passée au module de fabrique des agents, appelé *Agent Factory* qui va télécharger les règles de négociation (NR) des serveurs de négociation correspondants. Les règles sont stockées dans un entrepôt, le *NR Repository*, et seront utilisées par la fabrique d'agents pour instancier les agents responsables des négociations individuelles telles que choisies par l'usager. Chaque agent sera instancié en fonction du type de négociation pratiqué par le serveur de négociation auquel il sera assigné. Après avoir été instanciés, les agents se connectent et s'enregistrent sur leurs serveurs associés. L'usager emploie alors le module de spécification des négociations combinées, le *CN Specification Tool*, qui se présente sous la forme d'une interface graphique. Un langage de contraintes permettra de représenter, manipuler et enregistrer les contraintes de la négociation combinée (CN). Ces dernières sont enregistrées dans un dépôt nommé *CN Repository*. Par ailleurs, une autre activité de la spécification concerne les règles de stratégies (SR). Celles-ci sont enregistrées dans le dépôt associé, le *SR Repository*. Une fois la spécification de la négociation combinée achevée, le workflow la modélisant est généré et est stocké dans le *Workflow Repository*. L'usager peut alors visualiser le workflow via le *Workflow Definition Tool* et y apporter des corrections s'il y a lieu. Ceci met fin à la phase de modélisation et le workflow généré peut être exécuté. Son exécution peut être suivie à l'aide du *Workflow Monitoring and Control Tool*. L'exécution se déroule comme suit : Le moteur de workflow (*Workflow Engine*) distribue les tâches aux agents (considérés comme acteurs dans le workflow) et s'assure du séquencement des activités au sein du workflow. L'usager peut également suivre les négociations individuelles de chacun des agents via le *Agent Monitoring and Control Tool*. Enfin, chaque agent instancie un moteur d'inférence, le *SR Engine* qui infère à partir des règles se trouvant dans le SR repository.

3.2.5 Position de CONSENSUS par rapport à GNP

GNP se veut une plate-forme générique de négociation et en ce sens a pour objectif de permettre à ses usagers de conduire tous types de négociation. Cependant, les négociations combinées ont la particularité d'impliquer des serveurs de négociation indépendants les uns des autres et répondants à des protocoles de communication et de négociation différents. Comme on l'a vu, des outils et des techniques spécifiques sont nécessaires pour conduire ce type de négociation, et, on le rappelle, c'est là le but de CONSENSUS : fournir une infrastructure adéquate. CONSENSUS se présente donc comme un complément des services offerts par GNP, avec le même critère de généricité. Réciproquement, GNP fournit l'environnement nécessaire en termes de serveurs de négociation au développement initial de CONSENSUS.

3.3 Travaux connexes

Cette section présente les principaux travaux de recherche réalisés sur les infrastructures agents dédiées au commerce électronique et plus précisément aux enchères en ligne ainsi que les techniques employées pour modéliser les stratégies de négociation lorsque celles-ci sont proposées et implantées dans les systèmes.

3.3.1 KASBAH

Le laboratoire AMEC (« Agent-Mediated Electronic Commerce ») du MIT (Massachusetts Institute of Technology) a été un des précurseurs de l'utilisation des agents logiciels dans le cadre du commerce électronique. Les chercheurs ont développé un système multi-agents, KASBAH [CM96], pour une place de marché destinée aux annonces classées. Le type de négociation en jeu est donc le marchandage. Via une interface web, les usagers peuvent créer des agents vendeurs ou des agents acheteurs, selon qu'ils veulent vendre ou acheter quelque chose. Lorsqu'un usager crée un nouvel agent, il spécifie un certain nombre de paramètres. Par exemple, pour un agent acheteur, on spécifiera quand on veut acheter le produit, le prix que l'on désire payer et le prix maximum que l'on est prêt à payer. Ces

paramètres sont modifiables par intervention de l'usager en tout temps après la création de l'agent. De plus, certaines options de contrôle peuvent être ajoutées. Ainsi, on peut requérir que l'agent demande l'approbation de l'usager avant de conclure une entente ou encore qu'un courriel soit envoyé pour notifier qu'une entente a été faite. En dehors de ça, les agents sont autonomes et sont libres, en quelque sorte, de choisir la façon dont ils vont atteindre leur objectif. En fait, les agents ne sont pas des agents intelligents. Ils n'emploient pas de techniques d'intelligence artificielle ou autre mais agissent plutôt en fonction d'heuristiques relativement simples. Par exemple pour un agent vendeur, le comportement serait le suivant : *Offre le produit au prix spécifié par une fonction. Si il n'y a pas d'acheteur, baisse le prix pour susciter plus d'intérêt. À la date désirée de vente, le prix demandé doit être égal au plus bas prix acceptable.* Il est toutefois possible de spécifier une stratégie choisie parmi trois possibles : prudent (anxious), équilibré (cool-headed) et très intéressé (frugal) qui correspondent respectivement à des fonctions linéaires, quadratiques et exponentielles d'incrémentation des mises pour un agent acheteur ou de décrémentation du prix demandé pour un agent vendeur. Sur le plan conceptuel, les agents sont faits de trois composantes : paramètres de contrôle (présentés ci-dessus), état interne ainsi qu'historique de la négociation. L'état interne détient les informations nécessaires à la prise de décision (« contacts potentiels » et leurs dernières offres, prix demandé (prix de réserve)). L'historique enregistre chaque conversation effectuée avec d'autres agents. La figure 3 donne une représentation textuelle d'un agent KASBAH. En outre, la seule contrainte que KASBAH impose aux agents est d'utiliser le langage commun (les protocoles) à la place de marché, ce qui permet à des agents autres que des agents KASBAH de participer dans la place de marché.

Agent ID : 1 – type vendeur
Agent créé le : Mer 24 Jan 21:20:14 1996
Paramètres de contrôle :
Vendre le : Mer 24 Jan 21 :35 :00 1996
Prix désiré : 100
Prix le plus bas acceptable : 50
Fonction de décrémentation : linéaire
Description du produit : As de Pique

Figure 3 - Représentation textuelle d'un agent KASBAH (traduit de [CM96])

3.3.2 Fishmarket

Dans le projet Fishmarket [GGRG98], une plate-forme multi-agents d'expérimentations a été développée dans le cadre des enchères électroniques. La plate-forme, nommée FM97.5, anciennement FM96.5, est un environnement basé sur la technologie Java qui simule une maison d'enchère traditionnelle [RMN+98]. Comme son nom l'indique, Fishmarket s'intéresse particulièrement à la vente de caisses de poissons qui se fait traditionnellement sous la forme d'enchères hollandaises. Aussi, c'est ce dernier mécanisme qui a été implanté dans FM96.5 et FM97.5. À l'instar de son pendant traditionnel, dans la version électronique de la maison d'enchère, il y a plusieurs *scènes* qui se déroulent de façon concurrente. Par exemple, la *scène* d'inscription des acheteurs, la *scène* d'inscription des vendeurs, etc. Ainsi les agents vendeurs enregistrent leurs produits avec un agent chargé de l'admission des vendeurs, alors qu'un agent encanteur s'occupe de la vente et que les agents acheteurs sont enregistrés par un agent chargé des acheteurs et misent sur les biens en vente. Dans le système donc, les agents communiquent via des inter-agents dont la raison d'être est la gestion des protocoles de conversation qui garantissent que les agents suivent les règles du marché. Les inter-agents agissent en quelque sorte comme intermédiaires et comme garants du respect des règles. Ces expérimentations ont pour but de définir et d'évaluer différents scénarii appelés tournois. Ces derniers définissent en fait un ensemble standard de conditions sous lesquelles les agents compétitionnent avec pour objectif de maximiser leurs profits.

Pour cela, les agents sont dotés de stratégies évoluées. L'approche repose sur le raisonnement par cas, la logique floue ainsi que la théorie des possibilités et les modèles de prises de décision. Le modèle est un peu complexe mais peut se résumer ainsi : dans un premier temps, pour chaque mise d (vue comme une décision) de l'ensemble possible des mises et en se basant sur l'historique des enchères précédentes, la probanilité de gagner pour chacun des acheteurs (incluant l'agent) est calculée. Et, pour chacune de ces probabilités, son utilité est calculée. Dans un deuxième temps, à partir du modèle de décision, l'utilité globale de d est calculée en combinant les probabilités et les utilités. L'agent choisira alors la décision ayant la plus grande valeur en termes d'utilité globale.

Il est à noter qu'à l'instar de KASBAH, FM97.6 n'impose pas une architecture ou une implantation particulière des agents et qu'un outil de traçage permet de conserver l'ensemble des communications et transactions effectuées par les agents durant la simulation d'une enchère, ce qui permet d'évaluer et d'analyser l'évolution de leurs performances.

3.3.3 Michigan AuctionBot

AuctionBot [WWW98] a été un des premiers serveurs d'enchères ouverts aux agents à être disponible sur Internet. Via une interface web, il permet à des humains de créer des enchères et de soumettre des mises, de même qu'à des agents logiciels de le faire via TCP/IP à l'aide d'une API (Application Program Interface). De plus, il supporte plusieurs types d'enchères dont l'enchère anglaise, l'enchère hollandaise et l'enchère de Vickrey. AuctionBot a également été la première plate-forme à offrir une API permettant à des agents de communiquer de façon directe avec le serveur et c'est d'ailleurs ce qui a fait sa popularité. Grâce à l'interface TCP/IP, les agents peuvent exécuter les opérations suivantes : créer une enchère, soumettre une mise, demander et obtenir de l'information sur une enchère et vérifier l'état de leur compte. De plus, l'API est suffisamment complète pour permettre à des développeurs externes de construire leurs propres agents.

Un travail parallèle [HRW00] a été fait pour développer des agents compatibles avec AuctionBot pour un type particulier d'enchère double dénommée enchère Mième-prix (« Mth-price auction »). Trois types d'agents sont offerts, tous utilisant une fonction d'utilité. Le premier, appelé agent *compétitif*, a pour stratégie de toujours miser son vrai prix de réserve. Les deux autres types d'agents ont en commun qu'ils choisissent leurs mises en fonction de leurs possibles influences sur le marché. Les deux utilisent l'historique des enchères. Toutefois, l'un étudie les prix auxquels les transactions se sont faites dans les enchères précédentes pour essayer de déterminer à quel prix la prochaine transaction se fera, tandis que l'autre examine les mises effectuées par les concurrents afin de prévoir leurs prochaines mises. Pour cela, il conserve une trace de leurs mises précédentes. Dans les deux cas, les agents soumettent la mise la plus appropriée par rapport à leur prédiction.

3.3.4 ATTac

Depuis l'année 2000, une compétition, la *Trading Agent Competition* (TAC) [TAC02], a lieu une fois par an sur Internet et repose sur la plate-forme AuctionBot. Cette compétition met en jeu des agents de négociation développés par différentes équipes de développeurs [SG00]. Dans les éditions 2000 et 2001, les agents sont des agents de voyage qui ont pour but d'assembler des forfaits de voyage comprenant un billet de transport, une réservation d'hôtel et des places pour un événement de divertissement (par exemple : match de baseball). Chacun des éléments du forfait est vendu sur des marchés différents répondant à des règles de négociation différentes et se déroulant simultanément. Voici les détails sur le déroulement des enchères de l'édition 2000 : il y a une enchère pour chaque combinaison produit/jour et étant donné que la durée maximale du séjour est fixée à une période de cinq jours, il y a au total vingt huit enchères. Pour les billets d'avion, la vente se déroule ainsi : le prix varie aléatoirement toutes les 30-40 secondes d'une valeur choisie de façon uniforme dans l'intervalle [-10$, +10$]. Ce n'est pas une enchère mais plutôt un marché de type à prendre ou à laisser (*continuous clearing auctions*). En ce qui concerne l'hébergement, il y a seize chambres en vente par jour par catégorie d'hôtel (il y en a deux) et le marché est une enchère anglaise multi-items qui est caractérisée par une date de fin mais aussi par une période d'inactivité au bout de laquelle l'enchère est fermée et les adjudications sont faites. Cette dernière fonctionnalité est là pour éviter que toutes les mises soient faites à la fin. Enfin les billets pour les spectacles sont vendus dans une enchère double continue qui permet de vendre et acheter simultanément des billets, et, au lieu de provenir d'un (ou plusieurs) vendeur(s), ceux-ci sont aléatoirement répartis entre les participants, avant que les négociations ne débutent. Pour terminer, chaque agent représente huit clients ayant des préférences différentes et les scores de chaque agent sont évalués en fonction du respect de ces préférences et du montant net des dépenses.

Une douzaine d'équipes ont participé à cette simulation et la compétition a été remportée par ATTac, issue des laboratoires ATT. Du point de vue conceptuel, leur stratégie d'allocation des produits obtenus aux clients est globale. Autrement dit, l'agent essaye de satisfaire tous ses clients en même temps, contrairement à la majorité de ses concurrents qui essayent de satisfaire un client à la fois (algorithme vorace). Quant à la stratégie globale (cf.

figure 4) de soumission des mises, sa force réside dans ses différents éléments d'adaptabilité. Ainsi, ATTac est doté de deux modes de soumission des mises : le mode passif et le mode actif. En mode passif, l'agent essaye de conserver le plus d'options ouvertes possible et par conséquent n'achète pas encore de billet d'avion afin d'avoir le plus de liberté au niveau des contraintes. Soit T_m le temps préalablement calculé nécessaire à l'agent pour soumettre une mise et T_l le temps qui reste dans l'enchère. Lorsque $T_l < 2 \times T_m$, alors l'agent bascule en mode actif et procède à l'achat des billets d'avion et soumet des mises élevées pour les chambres d'hôtel afin d'avoir le plus de chances de les obtenir, sachant que l'enchère est sur le point de fermer. Donc, comme on peut le voir, l'objectif de la stratégie est de retarder les achats le plus possible.

Il est à noter enfin, que pour chacun des éléments du forfait, étant donné qu'ils sont vendus selon des règles de marché différentes, une stratégie propre est adoptée. Ainsi, pour l'hôtel, puisqu'il s'agit d'une enchère anglaise et que le prix tend à monter jusqu'au prix de fermeture (prix d'adjudication), la stratégie, en mode passif, est de miser faiblement, afin que si l'enchère ferme suite à une période d'inactivité, les chambres soient obtenues à un bas prix ou encore afin de maintenir une activité si aucune chambre n'est désirée pour le moment pour que l'enchère ne ferme pas prématurément. En ce qui concerne les billets de spectacle, la stratégie se décompose comme suit (cf. figure 4). Tous les billets détenus sont mis en vente. Soit G^* l'allocation optimale des produits en fonction de ceux détenus par l'agent et les prix en vigueur. Si le billet est alloué dans G^*, alors il est offert à la vente à un prix décroissant dans le temps mais supérieur au prix évalué par le client à qui il est attribué dans G^*, l'objectif étant d'essayer de gagner de l'argent (c'est un genre de spéculation). Par contre, si le billet ne figure pas dans G*, il est offert à la vente à un prix également décroissant dans le temps mais dont le prix au départ correspond à l'évaluation maximale faite par l'ensemble des clients. En mode actif, les billets non désirés sont vendus pour un bas prix fixe, et s'il n'y a toujours pas preneur alors ces billets inutiles sont conservés pour éviter que les concurrents ne fassent un trop gros profit.

1. **Tant que les enchères sont ouvertes :**
   - obtenir la dernière mise à jour des prix du marché
   - calculer G^*: l'allocation optimale des produits étant donné ce qui est détenu par l'agent et les prix en vigueur
   - Miser selon un des deux modes suivants
     - **Passif:** miser de façon à conserver le maximum d'options ouvertes
     - **Actif:** à la fin, miser agressivement

2. **Allocation :**
   - calculer G^* une fois les enchère fermées et alloués les produits achetés aux clients.

Figure 4 - Stratégie globale employée par ATTac (traduit de [SLSK00])

3.3.5 Anthony & al.

Généralement, de même qu'on a tendance à visiter plusieurs magasins avant d'acheter un produit, dans l'espoir d'obtenir le meilleur prix possible, une personne qui désire faire un achat via un site web de vente aux enchères voudrait surveiller ce qui se passe dans différents sites vendant un produit équivalent à celui qu'il désire et miser seulement dans les enchères les plus intéressantes. Toutefois, cela pose certaines difficultés, à cause notamment du fait que les enchères ont un début et une fin, et que ces derniers diffèrent d'une enchère à l'autre. De plus, c'est un processus très exigeant sur le plan du temps. Pour toutes ces raisons, le consommateur choisit généralement de participer dans l'enchère qui lui semble la plus intéressante, mais qui ne garantit aucunement qu'il obtiendra le meilleur prix (dans l'hypothèse où il gagne). Dans ce contexte, l'équipe de recherche de Nicolas Jennings a mis au point un agent capable de participer dans plusieurs enchères à la fois [AHDJ01]. Cet agent agit donc pour le compte d'un client qui lui spécifie entre autres un prix de réserve pour l'item à acheter et un temps limite pour acquérir ce dernier. Il utilise, en outre, des tactiques et des stratégies pour savoir quand, combien et dans quelles enchères il va miser. Il peut participer à des enchères anglaises, hollandaises et de Vickrey et son

comportement est dicté par l'algorithme suivant (cf. figure 5). L'agent identifie toutes les enchères actives, c'est-à-dire qui ont débuté et qui ne sont pas encore terminées, et récolte les informations pertinentes. Puis, il calcule le prix maximum que l'agent peut soumettre à l'instant donné. C'est en quelque sorte un prix de réserve temporaire qui est bien entendu inférieur au prix de réserve du client. En fonction de ce montant, l'agent sélectionne les enchères dans lesquelles il peut potentiellement miser et pour chacune d'entre elles, il calcule le montant à miser. L'enchère à qui est associée la mise maximisant l'utilité attendue est identifiée comme étant l'enchère cible, dans laquelle l'agent va miser.

Plusieurs contraintes sont prises en compte dans le calcul de la mise maximale pour un instant donné, dont le temps qui reste pour acquérir l'item, le nombre d'enchères encore actives et le degré d'inquiétude de ne pas obtenir l'item. Formellement, la mise maximale est donnée par

$$M(t) = \sum w_j(t) f_j(t)$$

où $w_j(t)$ est le poids associé à la contrainte j au temps t et $f_j(t)$ est la fonction polynomiale associée à la contrainte j qui suggère le montant à miser en fonction de cette contrainte. Il y a quatre types de fonctions qui sont identifiées comme étant des tactiques. Avec la tactique du temps restant (« Remaining Time Tactic »), plus le temps donné t s'approche du temps limite t_{max}, plus l'agent doit miser proche du prix de réserve, et il mise éventuellement celui-ci lorsque $t = t_{max}$. Similairement, avec la tactique des enchères restantes (« Remaining Auctions Tactic »), moins il reste d'enchères et plus la mise est proche du prix de réserve. Lorsque la tactique de la bonne affaire (« Desire For Bargain Tactic ») est choisie, l'agent maintient son incrément de mise à un minimum de t à t_{max} et mise éventuellement son prix de réserve lorsque $t = t_{max}$. Enfin, la tactique du désespoir (« Desperateness Tactic »), comme son nom l'indique, s'applique lorsque l'agent (son client) désespère d'obtenir le produit. Dans ce cas, il mise proche de son prix de réserve à t = 0 et éventuellement mise ce dernier lorsque $t = t_{max}$. Plus de détails sur les fonctions sont disponibles dans [AHDJ01] et dans [FSJ98] dont elles sont inspirées.

La valeur de la mise maximale calculée ainsi est utilisée pour sélectionner les enchères dites potentielles et par la suite, l'enchère cible. Cette sélection ne se fait que si l'agent n'est pas meneur (leader) dans une enchère anglaise ou s'il n'a pas déjà soumis de mise dans une enchère hollandaise ou de Vickrey. Ceci a pour but de ne pas acquérir plus d'un item. Donc, pour les enchères anglaises actives, seules celles dont le temps de fermeture est proche et dont le prix demandé additionné de l'incrément minimum est inférieur à la valeur de la mise maximale sont sélectionnées. Pour les enchères hollandaises, les élues sont celles dont le prix demandé est inférieur à la valeur de la mise maximale. Enfin, les enchères de Vickrey sélectionnées sont celles qui se terminent dans l'instant donné. Quant au montant que l'agent doit miser, dans l'enchère anglaise, il s'agit du prix demandé additionné de l'incrément minimum, c'est-à-dire la mise minimale demandée et pour l'enchère de Vickrey, il s'agit de la mise maximale calculée. L'enchère cible est déterminée en calculant, pour chaque enchère potentielle, l'utilité qui est fonction de la mise qui lui est associée : c'est celle dont la fonction donne la valeur maximale.

Tant que ($t \leq t_{max}$) et (Item_NA = vrai)
    Construire la liste des enchères actives.
    Calculer la mise maximale pour l'instant donné à partir de la stratégie de l'agent.
    Choisir les enchères dans lesquelles l'agent peut potentiellement miser.
    Choisir une enchère cible comme étant celle qui maximise l'utilité attendue de l'agent.
    Miser dans l'enchère cible en prenant la mise maximale comme prix de réserve pour l'instant donné.

Figure 5 – Algorithme de Anthony et al. (traduit de [AHDJ01])

3.3.6 Hewlett Packard Trusted E-Services Laboratory

Chris Preist des laboratoires HP s'est également intéressé aux agents participants dans plusieurs enchères simultanées [Pre00]. Dans le contexte d'expérimentations, les enchères sont des enchères anglaises multi-items. Dans le cas où les enchères se terminent toutes en

même temps, l'algorithme ne concerne que l'aspect de coordination des mises. Plus précisément, il détermine quelles sont les enchères où miser ainsi que les mises à soumettre dans ces enchères. Dans ce contexte, une mise est dite active si, à supposer que l'enchère prenne fin immédiatement, elle se traduit par une transaction, c'est-à-dire par un achat. Soit L le nombre de mises actives détenues par l'agent et soit m le nombre d'items qu'il doit acquérir. À tout moment le nombre de mises actives qui manquent à l'agent est m-L. D'autre part, si l'agent doit détenir j mises actives dans l'enchère a_i, il doit battre les j plus faibles mises actives de a_i. Ce sous-ensemble de mises est appelé la *j-battable* liste pour l'enchère a_i. Désignons par b_{ij} (notation différente de [Pre00]) la plus haute mise de ce sous-ensemble. Alors, pour battre ces j mises, l'agent doit soumettre j mises de valeur $b_{ij}+\delta$ où δ est l'incrément minimum. À la soumission de ces mises est associé un *coût incrémental*. Ce coût est égal à $[j * (b_{ij}+\delta)]$ – {somme des mises détenues dans *j-battable*}. À partir de là, l'agent construit des ensemble de mises. Ces derniers sont des ensembles de listes *j-battable* tels qu'ils contiennent exactement une liste *j-battable* de chaque enchère et qu'il y a au total exactement m-L mises soumises par des participants autre que l'agent. C'est-à-dire que chaque ensemble de mises représente une façon de placer des mises qui assure que l'agent va obtenir les m-L mises actives qui lui manquent. L'ensemble de mises qui sera généré et soumis est l'ensemble qui possède le coût incrémental total le plus faible. Lorsque les autres participants soumettent des mises, le processus est recommencé afin de maintenir m mises actives au coût le plus bas.

Cet algorithme n'est optimal que si les enchères se terminent toutes en même temps. En effet, si elles ne se terminent pas en même temps, les mises seront toujours plus élevées dans les enchères sur le point de finir que dans celles qui débutent et si l'on suppose par exemple qu'une enchère débute toutes les demi-heures, il est fort possible que l'agent ne fasse jamais de transaction car il changera toujours pour les nouvelles enchères qui sont moins chères. Il doit donc déterminer s'il doit rester dans une enchère qui est sur le point de s'achever ou s'il doit se lancer dans de nouvelles enchères qui débutent avec des mises beaucoup plus basses. Pour réaliser cela, il doit être capable de faire un compromis sur le plan de l'espérance de gain entre la certitude relative de rester dans une enchère sur le point de fermer et le risque de participer dans une nouvelle enchère, celle-ci pouvant résulter sur

un achat à meilleur prix ou un achat à prix plus élevé. Le mécanisme employé ici repose sur l'apprentissage à base de connaissances et la théorie de l'utilité. L'apprentissage à base de connaissances permet de construire un modèle de l'étendue des évaluations détenues par les participants dans différentes maisons d'enchères [Pre00]. À partir de ces évaluations, l'agent calcule l'utilité d'une participation prometteuse dans les enchères persistantes et la compare avec le revenu attendu de l'enchère finissante. Si cette dernière possède une utilité plus élevée, alors l'agent demeure dans l'enchère et effectue l'achat, sinon il se retire et s'engage dans d'autres enchères.

3.3.7 Experienced Based Negotiator

Des chercheurs du CSIRO (« Commonwealth Scientific & Industrial Research Organisation ») proposent une approche différente pour les agents de négociation [WZKA00]. En se basant sur le fait que les meilleurs négociateurs étaient généralement des gens d'expérience en la matière, ils ont choisi l'approche du raisonnement par cas, désignée par l'acronyme anglais CBR (« Case Based Reasoning »). L'idée consiste donc à utiliser les expériences et stratégies de négociation passées. Le cadre de cette recherche s'est limité au marché des voitures. La négociation est vue comme une suite d'épisodes de prises de décision consistant en l'évaluation d'une offre, le choix d'une stratégie et la génération d'une contre-offre. Comme un choix de stratégie est effectué à chaque épisode, l'agent peut changer de stratégie d'un épisode à l'autre. Ces stratégies sont reflétées par des concessions. Étant donné une série d'offre (*O(1)*, *O(2)*, ..., *O(n)*), la concession *C(i+1)* appliquée dans la stratégie de l'épisode *S(i+1)* est un pourcentage calculé comme suit :

$$C(i+1) = [O(i+1)-O(i)] / O(i) * 100$$

Ainsi, les stratégies employées dans les négociations précédentes sont décrites sur la base des concessions et contre-concessions faites aux offres et contre-offres.

Le système se décompose en un négociateur à base de cas (« Case-Based Negotiator »), un navigateur de cas (« Case Browser »), un module statistique (« Statistics component ») et

un module de maintenance des cas (« Case maintenance component »). Étant donné une offre, le moteur de négociation évalue s'il accepte ou non l'offre. Si c'est non, le négociateur fait alors l'association du scénario courant de négociation avec des cas précédents réussis et sélectionne le cas le plus similaire. Une fois cette sélection faite, la prochaine concession qui fût employée dans le cas le plus similaire est suggérée à l'agent pour générer une contre-offre. Quelques adaptations peuvent être faites si besoin est. Ainsi, si la contre-offre est supérieure au prix de réserve du client, la concession est alors ajustée afin d'arriver à une contre-offre inférieure ou égale au prix de réserve. Par ailleurs, les négociations qui se traduisent par un succès sont automatiquement ajoutés à la base de cas. Enfin, il est à noter que cette approche suppose que les agents de négociation (humains ou logiciels) ont un comportement rationnel et que la base de cas ne contient que des cas de négociation valides et représentatifs.

3.3.8 Su et al.

Su et al. ont développé un serveur web de négociation permettant de conduire des négociations de type marchandage [SHH00]. À la différence de beaucoup de travaux, ils n'ont pas choisi une approche orientée agent mais plutôt une approche de base de données orientées objet pour le développement de leur serveur. Les acheteurs et les vendeurs choisissent un serveur de négociation parmi plusieurs et c'est celui-ci qui va conduire la négociation. L'usager donne une description du produit qu'il veut acquérir (ou vendre si c'est un vendeur) ainsi que ses préférences et spécifie la stratégie de négociation à suivre. Le serveur prend alors le relais et recherche un fournisseur qui correspond à sa demande. Lorsqu'il en trouve un, il commence un marchandage jusqu'à ce qu'une entente soit conclue ou qu'un des participants mette fin unilatéralement à la négociation. Un processus de satisfaction des contraintes est utilisé pour évaluer les offres et contre-offres par rapport aux préférences et contraintes de l'usager. En outre, une approche déclarative est employée pour représenter les stratégies de négociation. Le fonctionnement des règles et leur représentation au sein du système seront revus plus en détail au chapitre 4.

3.3.9 Autres travaux

D'autres travaux ont été effectués sur les négociations électroniques. En voici un bref aperçu.

Zeng et Sycara ont proposé un système expérimental nommé BAZAAR [ZS98] modélisant les négociations comme une tâche séquentielle de prise de décisions. Ce système utilise les probabilités bayesiennes comme mécanisme sous-jacent d'apprentissage. C'est également un framework permettant de structurer et coordonner des agents.

MAGMA [TGMW97] est une architecture basée sur des agents permettant la simulation de places de marché. Cette approche prend en considération la publicité pour les produits ainsi que la gestion financière des transactions. Les agents sont composés de quatre modules. Le *portefeuille* conserve une trace des comptes de banques et des transactions effectuées. L'*inventaire* conserve une trace des produits détenus par l'agent. L'*annonceur* place des publicités sur le serveur pour les produits de l'inventaire à vendre. Enfin, le *négociateur* est responsable de la recherche de produits et d'effectuer la négociation si le mode automatique est sélectionné, ou de faire le médiateur entre les agents si le mode manuel est choisi.

Le *shopbot* (contraction des termes anglais « shop » et « robot ») [GK99] est un agent qui recherche sur Internet de l'information sur les prix et caractéristiques d'un produit à la demande d'un usager. Il permet donc à un acheteur de comparer les offres des fournisseurs et de faire la meilleure affaire possible. Son pendant respectif, le *pricebot* (contraction des termes anglais « price » et « robot ») [GK99], permet quant à lui aux vendeurs d'ajuster leurs prix dynamiquement sur Internet selon un algorithme pré-déterminé qui permet donc de maximiser les profits. Ces deux types d'agents ont été développés par IBM et sont opérationnels sur Internet.

3.3.10 Résumé

Les principales caractéristiques qui ressortent de ces travaux sont présentés dans le tableau 2. La colonne *plate-forme agent* indique si une infrastructure agent a été développée dans le projet ou non. Si ce n'est pas le cas, cela signifie que l'emphase du travail de recherche a surtout été mise sur le développement d'une stratégie de négociation et sur la validation de celle-ci.

Projets	Plate-forme agent	Type de négociation	Complexité des stratégies	Représentation des stratégies	Remarques
KASBAH [CM96]	oui	Marchandage	Simple	Fonction mathématique simple	On a le choix entre 3 stratégies
FishMarket [GGRG98]	oui	Enchère hollandaise	Évoluée	Fonction d'utilité, logique floue, CBR	-
AuctionBot [WWW98]	oui	Enchère anglaise, enchère hollandaise, enchère Vickrey	-	-	API (en C++) permettant de connecter des agents
ATTac [SLSK00]	non	Enchère anglaise multi-items, enchère double continue, marché à prendre ou à laisser	Évoluée	Algorithme, fonction d'utilité	Agent et stratégies conçus spécifiquement pour un cas très particulier . Fonctionne sur AuctionBot.
Anthony & al. [AHDJ01]	non	Enchères anglaise, hollandaise, vickrey	Évoluée	Algorithme, fonction d'utilité	1 seul agent participant simultanément dans plusieurs enchères de types différents
HP Trusted E-Services [Pre00]	non	Enchère anglaise multi-items	Évoluée	Algorithmes	1 seul agent participant simultanément dans plusieurs enchères
Experienced Based Negotiator [WZKA00]	non	Marchandage	Variable	CBR	-
Su et al. [SHH00]	non	Marchandage	Variable	Règles	Les règles ne sont pas modifiables à l'exécution

Tableau 2 - Résumé des travaux connexes

Comme on peut le voir et comme l'a constaté Pattie Maes [MGM99], la majorité des systèmes font appel à des stratégies de négociation prédéfinies et non adaptables. De plus, beaucoup de travaux se concentrent sur un seul type de négociation, à l'exception des travaux de Anthony et al. qui considèrent trois types d'enchères. ATTac considère également plusieurs types de négociation, mais pas d'un point de vue générique. En effet, ATTac a été conçu spécifiquement pour le scénario du TAC (cf. section 3.3.4). En outre, on peut remarquer qu'à part le projet « Experienced Based Negotiator » et le prototype de Su et al., les stratégies consistent en algorithmes couplés avec des fonctions d'utilités et figés dans le code, ce qui explique pourquoi les stratégies sont statiques, donc non modifiables. Pour pallier à cette faiblesse, les systèmes comme KASBAH (cf. section 3.3.1) et les agents de Anthony et al. (cf. section 3.3.5), proposent différentes stratégies pour leurs agents. Il est à noter également que dans l'approche de raisonnement par cas employée dans le « Experienced Based Negotiator », les stratégies et leur complexité dépend grandement de la richesse de la base de cas. Par contre, l'approche déclarative de Su et al. permet à l'usager de spécifier la stratégie de négociation, donc de l'adapter au besoin en fonction des paramètres de la négociation. Cependant, il n'est pas possible de changer de stratégie au cours de la négociation et on remarquera l'absence d'agents, contrairement à la majorité des autres projets.

Notre approche qui a, entre autres (cf. section 4.2), été inspirée par les travaux de Su et al. se veut déclarative, adaptable et générique. Par là, on entend une approche qui permette à un agent de participer à plusieurs types de négociation et qui soit très flexible sur le plan de stratégies de négociation. En effet, on veut aussi bien faire participer notre agent dans une enchère anglaise que dans une négociation de type marchandage et on veut pouvoir adapter sa stratégie, non seulement en fonction du type de la négociation mais aussi en fonction des paramètres de la négociation. On veut également permettre d'encoder les stratégies sous forme de règles instinctives afin qu'il soit aisé pour l'usager de les formuler. Mais on veut aussi laisser à des économistes ou des professionnels de recherche opérationnelle la possibilité de coder des stratégies de façon formelleet, pourquoi pas, de permettre de combiner l'approche algorithmique et l'approche par règles. Enfin, au lieu d'avoir un agent

participant dans plusieurs négociations, on se propose d'avoir un agent par négociation et de les coordonner, lorsque requis, via des règles de coordination.

Les avantages et inconvénients de notre approche seront discutés tout au long de la suite de ce mémoire et plus particulièrement au chapitre 7.

Chapitre 4: Concepts et architecture

Ce chapitre constitue la clé de voûte de ce mémoire. On y décrit les idées et les concepts qui constituent l'originalité de ce travail. On présente notamment les différents aspects qui caractérisent notre approche ainsi que les choix qui ont été effectués. Pour finir, nous introduisons l'architecture du système qui résulte de cette vision.

4.1 Exigences

Le système qu'on désire concevoir se présente comme suit : une interface graphique permet à l'usager de créer et contrôler ses agents de négociations ainsi que de suivre le déroulement de leurs exécutions. De plus, un module permet à l'usager de spécifier sa stratégie de négociation sous forme de règles. Enfin, l'usager a l'option de coordonner les agents, à l'aide de règles également.

Avant tout, il est important de donner les exigences par rapport à un tel système. Voici la liste des critères les plus importants :

- Le système doit pouvoir fonctionner pour plusieurs types de négociation.
- Le système doit pouvoir interagir avec différents serveurs de négociation, régis par des protocoles distincts.
- Le système doit être intégrable dans l'architecture de CONSENSUS, ce qui implique que :
    - Les agents doivent pouvoir être coordonnés.
    - Il doit y avoir un outil de surveillance permettant de suivre les actions des agents.

- Les règles déterminant le comportement et les stratégies des agents doivent être réutilisables.
- Le comportement ou la stratégie d'un agent doit pouvoir être modifié en cours d'exécution de l'agent.

Chacun de ces points sera détaillé dans les sections qui suivent.

4.2 Approche choisie

Ci-dessous est présentée l'approche qui a été choisie pour développer notre architecture. On s'apercevra que les exigences énoncées à la section précédente ont, logiquement, une large part d'influence et on évoquera certains travaux ou publications qui ont influencé nos choix. Notre approche se veut déclarative, orientée objet, générique et délibérative.

4.2.1 Une approche déclarative

L'utilisation d'une approche déclarative pour encapsuler le comportement de nos agents de négociation est un des points forts du travail rapporté ici. L'idée nous a été principalement inspirée de deux travaux connexes effectués dans le domaine du commerce électronique.

4.2.1.1 Travaux d'IBM

Grosof et al. du laboratoire IBM T.J. Watson [GLC99] ont travaillé sur la représentation des règles d'affaires inhérentes aux contrats de transaction. Ils ont en effet constaté que les termes des contrats s'expriment souvent sous la forme de relations conditionnelles facilement exprimables sous la forme de règles d'affaires, appelées également politiques d'affaires. Mais avant d'aller plus loin, il est peut-être important de définir ce qu'on entend par le terme *règle*. Grosof et al. décrivent une règle comme étant une implication de type SI-ALORS (IF-THEN) dans laquelle l'antécédent (c'est-à-dire la partie SI) peut contenir plusieurs conditions nécessaires (jointes par une conjonction ET). On peut étendre cette définition en ajoutant qu'une règle est une instruction déclarative qui dirige l'activité dans

une application logicielle en décrivant la ou les actions à exécuter lorsqu'un ensemble spécifié de conditions est satisfait [MB00]. Un exemple d'une telle règle est :

« Si un acheteur retourne un produit, dans l'année qui suit son achat, parce qu'il est défectueux, alors le prix total de l'achat sera remboursé. » [GLC99].
Plusieurs façons sont rapportées pour représenter les règles :

- programmation impérative (ex.: C ou Java), sous la forme d'instructions if-then.
- programmation logique (ex.: Prolog),
- vues SQL (proche de Prolog),
- règles événement-condition-action / règles actives / déclencheur qu'on trouve dans beaucoup de systèmes de bases de données,
- règles de production,
- systèmes experts, systèmes à base de connaissances, etc.

Comme aucune de ces approches ne satisfaisait pleinement leurs besoins, Grosof et al. ont développé une nouvelle approche basée sur la logique ordinaire et l'ont appelée logique courtoise (courteous logic). Celle-ci permet de traiter les conflits de priorités des règles par des règles spéciales, nommées règles de priorité.

4.2.1.2 Travaux de Su et al.

Le travaux de Su et al., présentés à la section 3.3.8, sont la deuxième source d'inspiration de notre approche [SHH00]. Les stratégies de négociation sont exprimées sous forme de règles Événement-Déclencheur (« Event-Trigger-Rules ») désignées par l'acronyme anglais ETR. Une ETR est composée de trois parties : (1) la condition qui doit être vérifiée pour que la règle s'applique, (2) les actions que le serveur de négociation doit exécuter lorsque la condition est satisfaite et (3) des actions optionnelles, alternatives, pour le cas où la condition n'est pas rencontrée. Les règles sont activées lorsque surviennent des événements spécifiques lors du processus de négociation. Le lien entre les événements et les règles est fait par le biais de déclencheurs. Au lieu d'utiliser un moteur d'inférence de règles, la gestion des événements et des règles est faite par un serveur ETR. L'exemple qui suit est

une ETR qui dit que si la livraison ne peut se faire avant au moins dix jours, alors il faut diviser le temps de livraison par deux. Autrement, une intervention humaine est requise.

TriggerEvent

SupplierComputer_Systemdelivery_day

SR1 :

***Condition**: getLowBound("delivery_day")>10*

***Action**: downLowBound("delivery_day", 2);*

***Alternative**: unResolvable("delivery_day can not be satisfied");*

4.2.1.3 Notre vision

On désire reprendre l'idée d'une approche déclarative en intégrant des moteurs de règles aux agents de négociation afin de capturer leurs comportements et, plus particulièrement, leurs stratégies dans le processus de négociation. La technologie des moteurs d'inférence n'est pas nouvelle et a été très employée il y a une vingtaine d'années dans les systèmes experts, les systèmes à base de connaissance et autres systèmes issus de l'intelligence artificielle. Brièvement, un moteur d'inférence ou moteur de règles est une application qui essaie de faire correspondre des faits à des patrons qui sont les conditions de la partie IF d'une règle IF-THEN. Ceci détermine quelles règles de la base de règles (lieu virtuel où sont stockées les règles) sont applicables. Cette sélection faite, les règles élues sont exécutées engendrant de nouveaux faits, et le processus se répète ainsi de suite. Peu employée autrement, on assiste depuis peu, avec l'émergence d'un nouveau paradigme pour le commerce, à un regain d'intérêt pour cette technologie. Sur le plan commercial, les moteurs de règles sont déjà utilisés pour faire ce qu'on appelle du profilage de consommateurs. Ceci consiste à déterminer, via une collecte d'informations, les préférences et les goûts des usagers d'un site web afin d'en adapter le contenu à leurs profils.

Parmi les différents avantages que peuvent apporter les règles, on retiendra qu'elles permettent d'avoir un niveau élevé d'abstraction. On entend par là que la connaissance humaine reproduite sous forme de règles est reproduite sous une forme explicite, facile à comprendre. De là, il est aisé de détecter des erreurs et d'apporter les corrections ou les

améliorations qui s'imposent. Ceci reste vrai même si la connaissance ainsi reproduite est subjective car elle demeure exprimée de façon explicite. On retiendra également que les règles peuvent être continuellement mises à jour au fur et à mesure que la connaissance qu'elles expriment évolue. Ceci peut se faire en modifiant les règes actuelles ou en ajoutant ou supprimant des règles. Enfin, soulignons qu'il est assez aisé de modifier les règles dynamiquement [GLC99].

À l'image du travail de Su et al., on veut donc offrir la possibilité aux usagers de spécifier eux-mêmes les stratégies de négociation qu'ils souhaitent utiliser. Cependant, on remarquera certaines différences notables dans notre approche. Premièrement, comme ils le soulignent eux-mêmes, Su et al. ont travaillé à développer un serveur web de négociation et non à développer un système multi-agents. Par conséquent, l'utilisation de leur approche est limitée à leur système, tandis que nous prétendons que nos agents seront capables d'interagir avec différents serveurs de négociation. Par ailleurs, il apparaît que les ETR ne peuvent être composées que d'une seule condition, ce qui est quelque peu restrictif. Nous n'envisageons pas de limites au nombre de conditions et d'actions que peut contenir une règle. Voici à quoi ressemblerait une telle règle pour un agent participant dans une enchère anglaise :

Si

Je ne suis pas leader, ***et***

Je n'ai pas encore atteint mon prix de réserve

Alors

Je peux miser

Enfin, le serveur de Su et al. se limite à un seul type de négociation qui est du type offres et contre-offres, alors qu'on se propose de supporter plusieurs types de négociation et en particulier plusieurs types d'enchère. Pour finir, nous pensons utiliser les règles pour fournir un minimum de coordination entre les agents de négociation (cf. section 4.3.3).

4.2.2 Une approche orientée objet

Dans la définition et la description d'un agent qui a été donnée au chapitre 2, rien n'indique de quelle manière un agent devrait être implanté. Souvent, cette implantation va dépendre de l'environnement dans lequel l'agent va être amené à évoluer. Cependant, l'approche orientée objet est une approche fréquemment choisie pour le développement d'agents. En effet, conceptuellement, comme le stipule l'OMG [Gro00], les agents sont des objets dans le sens où les agents ont une *identité,* ont un état interne et un comportement qui leur est propre et ont des interfaces qui leur permettent de communiquer entre eux ou avec d'autres entités. En outre, à l'instar des objets, même si les agents peuvent être de n'importe quelle taille, il est généralement conseillé de les développer comme des applications légères, c'est à dire qui s'exécutent rapidement et qui ne sont pas gourmands en mémoire. De plus, l'OMG suggère que si les objets et les agents sont quand même deux entités distinctes, cela n'empêche pas les agents d'être composés d'objets, d'autant plus que ceux-ci ont plusieurs points en commun comme le fait d'avoir une identité, un état, un comportement et des interfaces qui leurs permettent de communiquer.

Wooldridge et Ciancarini [WC00] font également une analogie entre objets et agents tout en mettant en évidence les caractéristiques qui les distinguent, à savoir :

- Les agents possèdent une notion plus forte d'autonomie.
- Les agents peuvent avoir un comportement flexible allant de réactif à proactif en passant par social.
- Dans un système multi-agents, chaque agent est supposé avoir au moins un « thread » de contrôle.

Enfin, Shoham [Sho93] abonde lui aussi dans le sens d'une similarité entre agents et objets. En effet, il affirme qu'un agent est essentiellement *un objet avec une attitude.* Autrement dit, il considère les agents comme des objets actifs, c'est-à-dire comme des objets capables de modifier leurs comportements par eux-mêmes.

Comme on peut le constater, conceptuellement, objets et agents offrent beaucoup de similitudes, et des organismes comme l'OMG [OMG02] travaillent à exploiter ces

similitudes pour développer des paradigmes communs. En fait, on peut voir un agent comme un objet avec des capacités et des comportements étendus. De plus, plusieurs infrastructures agents ont été développées avec succès dans des langages de programmation orientés objet comme Java. Pour toutes ces raisons, le choix d'une approche orientée objet pour le développement de notre infrastructure paraît évident.

4.2.3 Une approche générique

Comme il a été spécifié précédemment, on veut idéalement que l'infrastructure proposée puisse couvrir un large éventail de types de négociation et puisse s'adapter à divers contextes, c'est-à-dire à divers produits et services. Ceci implique que nos agents devraient pouvoir interagir avec différents serveurs de négociation qui imposent des protocoles différents de communication, mais cela implique également une certaine modularité de l'architecture afin qu'il soit suffisamment aisé de prendre en compte un nouveau type de négociation. On ne désire pas que les agents soient spécifiques au contexte particulier d'une négociation, comme c'est le cas des agents Fishmarket ou ATTac par exemple (cf. sections 3.3.2 et 3.3.4 respectivement).

4.2.4 Une approche délibérative

On considère que nos agents sont essentiellement des agents délibératifs (cf. section 2.1.2) car son activité consiste en partie à construire une représentation du monde extérieur que constituent les négociations et leurs participants ainsi qu'une représentation interne de ses états et des préférences de l'usager. À partir de ces représentations symboliques, il est à même d'effectuer un raisonnement grâce à la composante système à base de règles et de procéder aux actions adéquates qui affecteront son environnement.

4.3 Coordination

4.3.1 Nécessité

Tel qu'évoqué au chapitre 3, il nous apparaît important qu'il y ait un mécanisme de coordination entre les agents en raison du caractère combiné des négociations auxquelles on s'intéresse dans le cadre de CONSENSUS. L'exemple qui suit permet de mettre en évidence cette nécessité. Supposons que l'usager dispose de la somme de 500 $ pour négocier la réservation de la chambre d'hôtel du forfait décrit à la section 3.2.1 et que trois agents soient instanciés et lancés sur des négociations indépendantes pour obtenir le meilleur prix possible. Il faut qu'il y ait un mécanisme permettant de s'assurer qu'un seul agent sur les trois va, au bout du compte, effectuer une réservation, et que celle-ci soit la moins chère des trois possibles. Ce mécanisme est clairement un mécanisme de coordination et peut être mis en place à deux niveaux : au niveau du workflow gérant la négociation combinée (cf. section 3.2.2) ou bien au niveau des agents eux-mêmes. Il y a également une troisième possibilité qui consiste à attribuer une partie du contrôle de la coordination au workflow et l'autre partie aux agents. La meilleure solution n'a pas encore été établie dans le cadre de CONSENSUS et c'est dans cette optique qu'un mécanisme de coordination est incorporé dans l'architecture décrite ici. Toutefois, cela ne constitue pas une composante majeure du travail présenté ici. La section qui suit donne un aperçu des principales techniques de coordination. Ensuite, notre choix sera présenté.

4.3.2 Survol des techniques

Il y a plusieurs façons de classer les techniques de coordination entre agents. Nwana en rapporte trois catégories [NLJ96]:

- structure organisationnelle,
- passage de contrat,
- planification multi-agents.

4.3.2.1 Structure organisationnelle

C'est un scénario de coordination simple qui repose sur la structure organisationnelle du système. Cette structure définit les responsabilités et les capacités des agents, et, de plus, fournit le cadre d'application pour les activités et les interactions des agents via la définition de rôles ou de relations d'autorité. Il y a deux grandes façons d'implémenter cette technique. Premièrement, si la structure est plutôt hiérarchique, alors le modèle maître-esclave s'applique bien. On peut ainsi avoir un agent maître qui planifie et partage les tâches entre les agents esclaves. Ceux-ci peuvent éventuellement communiquer entre eux mais doivent rapporter leurs résultats à l'agent maître. La seconde méthode repose sur une architecture « blackboard ». Les agents lisent et écrivent dans le « blackboard » et il peut y avoir un agent céduleur (ou agent maître) qui gère les lectures et écritures des autres agents. Cette approche est adéquate lorsque le problème à résoudre est distribué, lorsqu'un agent céduleur est déjà présent ou lorsque les tâches des agents sont prédéfinies. L'architecture « blackboard » est décrite en détail à la section 4.3.3.

4.3.2.2 Passage de contrat

Cette approche s'adresse plus particulièrement aux structures de marché décentralisées et concerne l'allocation des tâches et des ressources entre les agents. Deux rôles sont possibles pour un agent. Il peut être un gestionnaire qui va diviser le problème en plusieurs sous-tâches et chercher des contractants pour les exécuter. La supervision de la résolution globale fait également partie de ce rôle. De l'autre côté, il peut jouer le rôle du contractant qui réalise une sous-tâche. Il peut alors devenir gestionnaire en subdivisant sa tâche et ainsi de suite. Donc les deux rôles peuvent être cumulés par un agent, hormis celui qui est au début de la chaîne de décomposition du problème et ceux qui sont à la fin de cette chaîne. La recherche de sous-contractants se fait par le biais d'un mécanisme d'appel d'offre.

4.3.2.3 Planification multi-agents

Avec cette technique, les agents établissent un plan détaillé des actions et interactions requises pour parvenir à leurs buts. Cette approche permet d'éviter les inconsistances et les conflits qui peuvent survenir avec la technique de passage de contrats. Dans le cas d'une

planification multi-agents centralisée, un agent coordonnateur analyse les plans que lui soumettent les autres agents dans le but d'identifier les incohérences et les conflits potentiels. Si il en détecte, il tente alors de modifier ces plans, puis combine ces derniers en un plan global sans conflits. Par contre, dans la planification multi-agents distribuée, chaque agent possède un modèle des plans des autres agents. Les agents communiquent alors pour construire et mettre à jour leurs plans ainsi que leurs modèles, et ce jusqu'à ce que tous les conflits soient éliminés.

4.3.3 Notre choix : architecture « blackboard »

L'architecture « blackboard » est issue du domaine de l'intelligence artificielle. Son nom (tableau noir) évoque le tableau autour duquel un groupe d'experts se réunit pour résoudre un problème complexe. Comme il a été mentionné à la section 4.3.2.1, cette architecture s'inscrit dans la catégorie des techniques de coordination dite de structure organisationnelle. La structure imposée par le modèle du « blackboard » est expliquée dans la description qui suit.

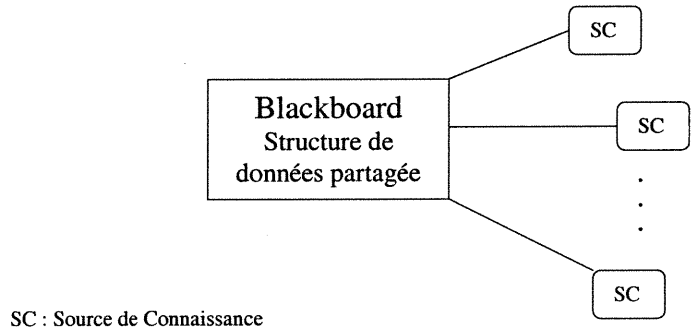

Figure 6 - Architecture blackboard de base

L'architecture de base distingue deux types de composantes, comme on peut le voir sur la figure 6. D'une part, il y a le « blackboard » lui-même qui est une structure de données servant de dépôt central pour des informations et qui est accessible de façon globale. Et,

d'autre part, il y a ce qu'on appelle les sources de connaissances (« knowledge sources ») [PH98]. La source de connaissance exécute un processus (calcul, traitement) à partir du contenu du « blackboard » et le résultat de ce processus peut lui-même altérer le « blackboard ». Conceptuellement, les sources de connaissance ne peuvent communiquer entre elles directement : toute communication s'effectue par le biais du « blackboard ». Les informations contenues dans le « blackboard » peuvent aussi bien représenter des faits que des suppositions ou encore des déductions. En outre, certaines architectures ajoutent une troisième composante, appelé céduleur (« scheduler »). Ce module contrôle l'accès des sources de connaissances à la structure partagée.

Initialement, l'architecture « blackboard » a été conçue pour les systèmes de résolution de problème, mais le concept est applicable et est appliqué d'ailleurs dans d'autre contextes également tel que les systèmes en temps réel multi-tâches distribués. Notons que dans notre cas, les sources de connaissance sont les agents négociateurs. Par ailleurs, comme le souligne Nwana [NLJ96], les architectures « blackboard » sont adéquates dans une architecture multi-agents lorsque le nombre d'agents est relativement restreint. En effet, en raison de l'approche centralisée, si le nombre d'agents est trop élevé, il y a risque d'apparition d'un goulot d'étranglement au niveau des interactions entre les agents et le « blackboard ».

En résumé, nous avons donc opté pour la solution « blackboard » pour coordonner les agents de négociation car il s'agit d'une approche déjà éprouvée et reconnue pour sa simplicité, car les tâches de nos agents sont prédéfinies (cf. section 4.3.2.1) et, enfin, car le nombre d'agents à coordonner est assez limité. De plus, étant donné que chaque agent est couplé avec un système à base de règles et que les règles offrent beaucoup d'avantages qui ont été énoncées précédemment, la coordination se fera également par le biais de règles. Celles-ci s'appliqueront en fonction des informations que les agents rendront disponibles dans le « blackboard ». Un exemple de règle de coordination pour une négociation-OU (cf. section 3.2.3) du type enchère anglaise (cf. section 2.3.1) est donné ci-dessous.

Si

les autres agents ne sont pas leaders et
ne sont pas en train de miser ***et***
je ne suis pas leader ***et***
je n'ai pas atteint mon prix de réserve

Alors

je mise

Cette règle permet de coordonner les agents qui l'exécutent de la façon suivante : chaque agent, avant de miser, va s'assurer qu'aucun autre agent n'est leader dans son enchère, c'est-à-dire, qui ne soit proche d'une transaction. On est ainsi sûr qu' au plus un et un seul item sera acheté.

4.4 Présentation de l'architecture

4.4.1 Vue générale

Le schéma présenté à la figure 7 illustre l'architecture globale de l'infrastructure que nous proposons pour les agents de négociation. La figure 7 décrit l'architecture générale tandis que la figure 8 présente les modules composant un agent. Le schéma sera explicité dans la section suivante par le biais d'un scénario d'utilisation. Les détails du schéma ainsi que sa réalisation feront l'objet du chapitre 5.

4.4.2 Scénario d'utilisation

L'usager a à sa disposition deux modules principaux pour gérer ses agents (cf. figure 7). Le premier, le module de gestion des agents, lui permet de créer et contrôler les agents avec les caractéristiques qu'il désire, tandis que le module de gestion des règles lui sert à déterminer le comportement et les stratégies que les agents vont adopter pour les négociations. Imaginons un scénario où une personne, qu'on nommera Lydia, désire acheter une édition limitée d'un roman par le bais de librairies spécialisées en ligne. Pour cela, elle est prête à

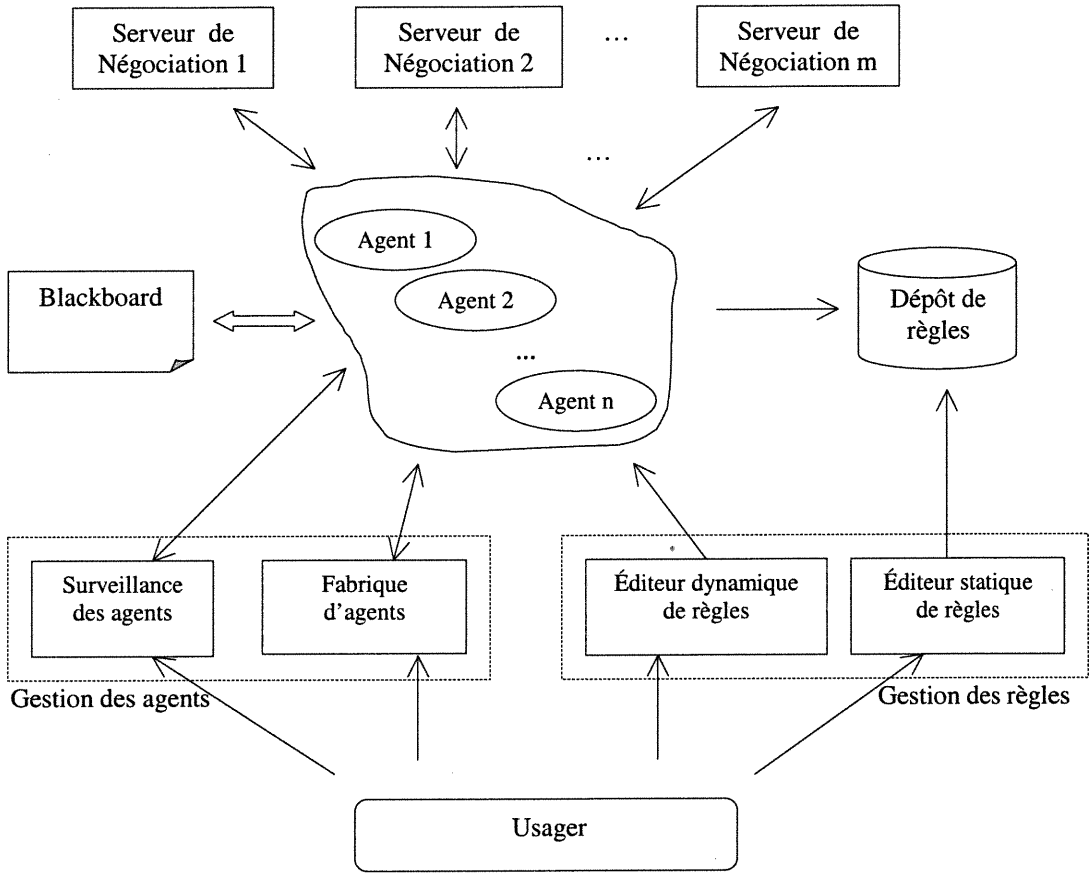

Figure 7 - Architecture globale de l'infrastructure

payer au maximum 100$.

Dans un premier temps, supposons que Lydia ait trouvé un site web qui vend le roman aux enchères et que l'enchère soit une enchère anglaise. Lydia commence par saisir les règles qui vont représenter le comportement et la stratégie de l'agent. Pour cela, elle utilise l'éditeur statique de règles. Voici un exemple de règle que Lydia écrit :

Si

le prix demandé du livre atteint 80$ ***et***

il reste moins de 5min avant la fin,

alors

je mise le prix de réserve.

Les règles ainsi entrées sont mémorisées de façon permanente dans le *dépôt de règles*; par ailleurs, Lydia n'est pas obligée de réécrire des règles à chaque fois et peut utiliser des règles déjà existantes dans le dépôt. Ceci fait, Lydia peut utiliser la fabrique d'agents pour créer son agent négociateur. Elle spécifie notamment l'adresse du serveur de négociation et le code identifiant la négociation, ainsi que les informations de base comme le prix de réserve, le nombre d'exemplaires qu'elle désire acquérir, etc. Enfin, dernière étape de la création, Lydia choisit les règles de stratégies parmi celles qui sont disponibles dans le dépôt. Idéalement, celles-ci seront groupées par paquets de manière à représenter un certain type de comportement. Lydia peut maintenant lancer son agent et, grâce au module de surveillance, suivre ses actions et observer le déroulement de la négociation. En fonction de ce qu'elle voit, Lydia peut décider d'intervenir sur l'exécution de l'agent en le suspendant ou l'arrêtant complètement, par exemple. Mais elle peut également intervenir directement sur sa stratégie. En effet, l'éditeur dynamique de règles lui permet de modifier, d'ajouter ou de retirer des règles pendant que l'agent procède à la négociation. Et c'est là un des gros avantages de cette infrastructure.

Supposons maintenant que Lydia ait trouvé trois librairies en ligne qui vendent son livre aux enchères. Deux des enchères sont anglaises et l'autre est une enchère hollandaise (cf. section 2.3.2). Les trois agents n'auront de ce fait pas le même comportement et la même stratégie que l'on qualifiera d'individuelles. De plus, le besoin se fait sentir de coordonner ces trois agents. En effet, il faut s'assurer qu'on n'obtienne qu'un seul exemplaire du livre. Dans le cas où les règles de l'enchère lui permettent de retirer de son engagement, c'est assez simple, il suffit d'essayer de gagner dans les trois enchères et de retirer ses offres pour garder la moins chère. Cependant, le plus souvent, il n'est pas possible d'annuler son engagement ou alors cela se fait en retour d'une compensation financière. Il faut alors

coordonner les agents de façon intelligente, ce que Lydia peut faire dans une certaine mesure pour ses trois agents. Elles peut en effet les doter de règles de coordination qui sont semblables, sur le principe, aux règles individuelles. Un exemple de ce type de règle qui pourrait être donné aux trois agents, pour le cas qui nous occupe, serait :

Si

un des autres agents est leader dans son enchère ***et***
je ne suis pas leader ***et***
ma mise serait supérieure à celle de l'autre agent

alors

je ne mise pas.

Cette règle devrait normalement assurer qu'il y ait toujours un seul agent leader dans son enchère et que ce soit avec la mise la plus basse. Pour le reste, le processus de création et d'exécution des agents reste le même, si ce n'est que dans les paramètres que Lydia donne dans la *fabrique d'agents,* elle spécifie, pour chacun des agents, qu'il doit être lié au « blackboard » qui contiendra les informations communes aux trois agents et sur lesquelles reposent les règles de coordination.

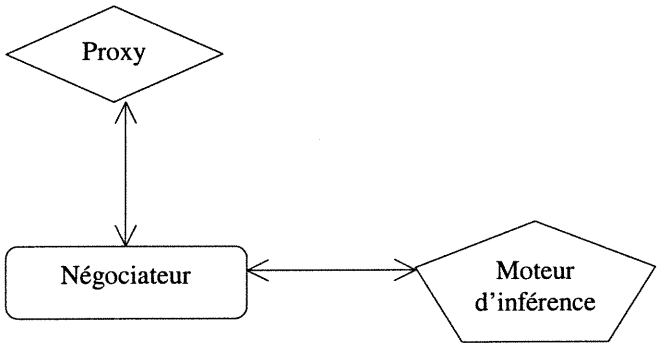

Figure 8 - Architecture de l'agent de négociation

Pour terminer, en ce qui concerne l'architecture même de l'agent, illustrée dans la figure 8, elle est assez simple comme on peut le constater. Le proxy est le module qui gère la communication et l'échange d'informations entre l'agent et le serveur de négociation. Cette

séparation rejoint l'objectif de généricité car elle fait en sorte que les agents peuvent se connecter à des serveurs de négociation ayant un protocole de communication différent sans avoir à altérer le comportement de l'agent. Il suffit pour cela de changer de proxy lorsqu'on change de type de serveur. Le moteur d'inférence représente, avec les règles, l'intelligence de l'agent puisque c'est lui qui procède et veille à leur bonne exécution. Enfin, le module appelé négociateur représente le noyau de l'agent et fait le lien avec les deux modules précédents.

Chapitre 5: Implantation de l'infrastructure INSULA

Nous décrivons dans ce chapitre une implantation de l'infrastructure proposée au chapitre précédent. Ce système a été nommé INSULA (*INfraStructure for rULe-driven negotiating software Agents*) et comporte l'ensemble des modules de l'architecture de la figure 7. Java [Java02] a naturellement été choisi comme langage de développement étant donné que c'est le langage par excellence pour faire de la programmation orientée objet. De plus, le projet TEM, auquel, rappelons-le, le projet CONSENSUS est rattaché, utilise également les technologies Java pour le développement des applications, ce qui facilite la compatibilité et l'intégration d'INSULA au sein de CONSENSUS et de TEM.

Pour commencer, nous donnerons un aperçu général de la conception d'INSULA. Vient ensuite la description des modules principaux tels qu'illustrés dans la figure 7 . Enfin, quelques données d'implantation closent le chapitre.

5.1 Vue globale

Cette section présente la façon générale dont le code de l'implémentation a été conçu et décrit en quoi consiste l'exécution d'un agent INSULA.

5.1.1 Conception du code

Afin de réaliser le caractère générique de l'infrastructure, c'est-à-dire la facilité de pouvoir écrire des « proxy » pour des serveurs de négociation différents et de traiter différents types de négociation, on a opté pour une conception reposant sur le système des interfaces de Java. Rappelons qu'une interface est une collection de constantes et de méthodes abstraites

(c'est-à-dire de méthodes sans corps) et ne peut être instanciée. Une interface est utilisée pour définir formellement un ensemble de méthodes ou fonctions qu'une (ou plusieurs) classe(s) peut(vent) implémenter. En outre, les interfaces permettent de faire du polymorphisme. C'est de cette caractéristique qu'on tire avantage ici. En effet, le nom d'une interface peut être utilisé pour désigner le type d'une variable et de là, on peut faire référencer cette variable sur n'importe quel objet dont la classe implémente l'interface en question. Dans notre cas, par exemple, nous avons une interface nommée NegotiationProxy qui décrit l'ensemble des fonctions que doit permettre le proxy utilisé par l'agent pour communiquer avec le serveur de négociation. Pour chaque nouveau serveur de négociation, il suffit alors de créer une nouvelle classe implémentant NegotiationProxy, sans qu'il soit nécessaire de modifier le reste du code de l'agent, celui-ci faisant référence au proxy via le nom de l'interface. De même, une interface NegotiationAgent a été définie pour décrire les fonctionnalités communes aux négociations. Pour chaque type de négociation, une classe implémentant NegotiationAgent doit être développée et être enrichie des caractéristiques propres au genre de la négociation.

5.1.2 Principes de fonctionnement

Cette partie correspond à la description du fonctionnement du module Négociateur de l'architecture conceptuelle de l'agent présentée à la figure 8. L'agent fonctionne en exécutant une série d'actions de façon cyclique. Chaque cycle, que nous nommons pulsation (« pulse »), peut être décrit par le pseudo-code suivant :

```
Répéter
    SuspendreExécution(p);
    ObtenirInformations();
    Penser();
    Agir();
Jusqu'à ce que (état = gagnant ou état = perdant ou état = quittant)
```

La première étape, SuspendreExécution(p), fait rentrer l'agent dans une période *p* d'inactivité. Cette étape est nécessaire pour éviter d'effectuer des pulsations inutiles car trop rapprochées dans le temps et donc d'accaparer le processeur inutilement. Par la suite, lorsque l'agent redevient actif, à l'étape ObtenirInformations(), il communique avec le serveur et met à jour ses informations. À partir de ces nouvelles informations, dans l'étape Penser(), des règles sont exécutées tout en déterminant quelle action doit être effectuée par l'agent. Enfin, l'étape Agir() consiste en l'exécution de cette action.

Ce mode de fonctionnement est imposé par GNP, qui, comme la majorité des serveurs de négociation, supporte le mode « pull » (modèle du tirer). Le mode « pull » signifie que c'est à l'utilisateur (ici, l'agent logiciel) de chercher ou de demander l'information dont il a besoin, à l'image d'une personne qui, à partir de son navigateur, rafraîchit une page web pour voir si son contenu a été modifié. L'alternative à ce type de fonctionnement est le mode « push » (modèle du pousser). Dans ce modèle, toute nouvelle information est transmise directement et automatiquement par le serveur aux utilisateurs.

Plusieurs états ont été définis pour l'agent (les termes originaux anglais sont conservés) :

- Trailing : l'agent n'est pas leader dans sa négociation.
- Leading : l'agent est leader dans sa négociation.
- Bidding : l'agent est en train de soumettre un ordre.
- Losing : l'enchère a pris fin alors que l'agent était dans l'état trailing.
- Winning : l'enchère a pris fin alors que l'agent était dans l'état leading.
- Out : l'agent a quitté la négociation.

Notons que ces états ont été définis pour le cas des enchères (anglaises). Ils doivent éventuellement être généralisés pour supporter des enchères plus complexes et d'autres types de négociation comme le « bargaining » par exemple.

Quant aux actions, il y a cinq possibilités :

- Undefined : c'est l'action par défaut. L'agent ne fait rien.
- Wait : cette action va plonger l'agent dans une période d'inactivité préalablement déterminée.

- Bid : l'agent (acheteur) soumet une offre d'achat à partir de paramètres préalablement spécifiés, comme le prix et la quantité.
- Sell : l'agent (vendeur) soumet une offre de vente à partir de paramètres préalablement spécifiés, comme le prix et la quantité.
- Drop : l'agent quitte la négociation.

5.2 Module de règles

Ce module est le plus important car c'est lui qui permet de doter les agents d'une certaine intelligence et c'est également sur lui, à terme, que l'usager va intervenir pour créer de nouvelles stratégies de négociation ou modifier le comportement d'un agent. Par ailleurs, les moteurs d'inférences sont des applications complexes qui requièrent une grande expertise pour les développer. Aussi, la possibilité de développer notre propre moteur d'inférence a été rapidement éliminée et l'horizon s'est plutôt porté vers des applications commerciales.

5.2.1 Produits disponibles

Le langage de développement étant Java, et INSULA étant avant tout un prototype donc devant être réalisé dans un laps de temps assez raisonnable, les choix ont été restreints aux moteurs facilement intégrables à une application Java. Deux produits récents et relativement répandus répondent à ce critère : JESS et ILOG JRules.

5.2.1.1 JESS

JESS (Java Expert System Shell) [JESS02] est, comme son nom l'indique, un système expert écrit en Java qui, en principe, s'ajoute aisément à un programme Java. Il utilise un langage dont la syntaxe est proche de celle du langage Lisp [Lisp02], c'est donc une syntaxe quelque peu complexe pour les programmeurs non avertis. Un exemple de règle JESS extrait de [LMM+01] est:

```
Jess> (defrule example-7
      (person ?x)
      (not (married ?x))
      =>
      (printout t ?x " is not married!" crlf))
```

À la base, JESS n'offre qu'une simple console en guise d'interface de programmation. Il n`y a pas d'outil permettant la gestion des objets ou la gestion des règles. Cependant, JESS est un outil gratuit pour un usage académique ou de recherche et dont les usagers peuvent contribuer à son amélioration par le développement de divers modules, comme par exemple un mode JESS pour l'éditeur de texte Emacs ou une interface graphique pour le développement. Enfin, JESS possédant son propre langage, le moteur d'inférence ne peut interagir directement avec des objets Java. Pour ce faire, il faut écrire une interface entre les objets JESS et les objets Java.

5.2.1.2 ILOG JRules

JRules [ILOG02] est un logiciel commercial clés-en-main de la compagnie ILOG qui se présente sous la forme de librairies de classes Java. Toutefois, en dehors de la syntaxe des règles, JRules n'a pas de langage de développement qui lui est propre, le moteur d'inférence manipule directement des objets Java et exécute du code Java. En plus d'avoir une syntaxe pour les règles lisible et relativement simple, JRules vient avec un ensemble d'outils graphiques favorisant entre autres l'édition des règles, le développement d'une application et le débogage du code. Il est même possible de développer et d'associer aux règles un modèle de syntaxe basé sur le contexte d'affaires qui permet d'écrire et d'afficher les règles d'une façon plus naturelle et plus lisible. Voici un exemple d'une règle JRules présentée sous sa forme source d'abord, puis sous une forme plus naturelle :

- Forme source:

```
when {
     // Classes used in the rules
   ?shoppingCart:ShoppingCart();
   ?customer:Customer();
   ?session:Session();
```

```
    // Conditions
  evaluate(?shoppingCart.containsItemsInRange(2, 4));
  evaluate(?customer.value > 100);
  evaluate(?customer.getCategory() equals "Gold");
}
then {
    // Actions
  ?shoppingCart.applyDiscount(15);
  ?session.displayMessage("We're giving you a nice discount!");
}
```

- Forme plus lisible:

```
If
   the shopping cart contains between 2 and 4 items
   and the purchase value is greater than 100
   and the customer category is Gold
Then
   apply a 15% discount
   and display the message 'We're giving you a nice discount!'
```

Plus de détails sur le fonctionnement de JRules sont disponibles à la section suivante.

Les avantages de JRules par rapport à JESS ressortent clairement de la comparaison entre les deux. Ceci est d'ailleurs confirmé par une étude comparative récente menée par Lamma et al. [LMM+01]. Dans cette étude, trois outils de programmation à base de règles ont été comparés dans le contexte du développement d'un système expert pour la validation de données d'un laboratoire micro-biologique. Les trois outils comparés sont KAPPA-PC [KAP01], JESS et JRules, et le tableau récapitulatif issu de l'étude est reproduit ci dessous (tableau 3). On peut constater que sur tous les critères, à l'exception du dernier sur lequel les deux sont égaux, JRules est meilleur que JESS (et KAPPA-PC). Notre choix s'est donc porté sans hésitation sur le produit de la compagnie ILOG.

Outil	Complexité de la syntaxe des règles	Outils de dével.	Complexité de programmation	Inférence	Performance du moteur
K-PC	faible	très bon	faible	acceptable	suffisante
JESS	élevée	pauvre	élevée	acceptable	bonne
JRules	médium	bon	médium	très bon	bonne

Tableau 3 - Tableau comparatif entre trois systèmes à base de règles.
(traduit de [LMM+01])

5.2.2 Fonctionnement de JRules

Une application fonctionnant avec JRules est typiquement composée des éléments suivants :

- une base de règles (*rule base*),
- un moteur d'inférence (*inference engine*),
- une collection d'objets Java contenue dans une mémoire de travail (*working memory*),
- un agenda qui contient les règles éligibles classées par ordre de priorité (il y a quatre niveaux de priorité).

La base de règles est un ensemble de règles de la forme *SI conditions ALORS actions* où les conditions réfèrent à des objets Java et où les actions consistent en la création, le retrait ou la modification des objets dans la mémoire de travail. Par ailleurs, les règles peuvent être regroupées par paquet qu'on peut activer ou désactiver à loisir, ce qui peut être très pratique comme on le verra au chapitre 6.

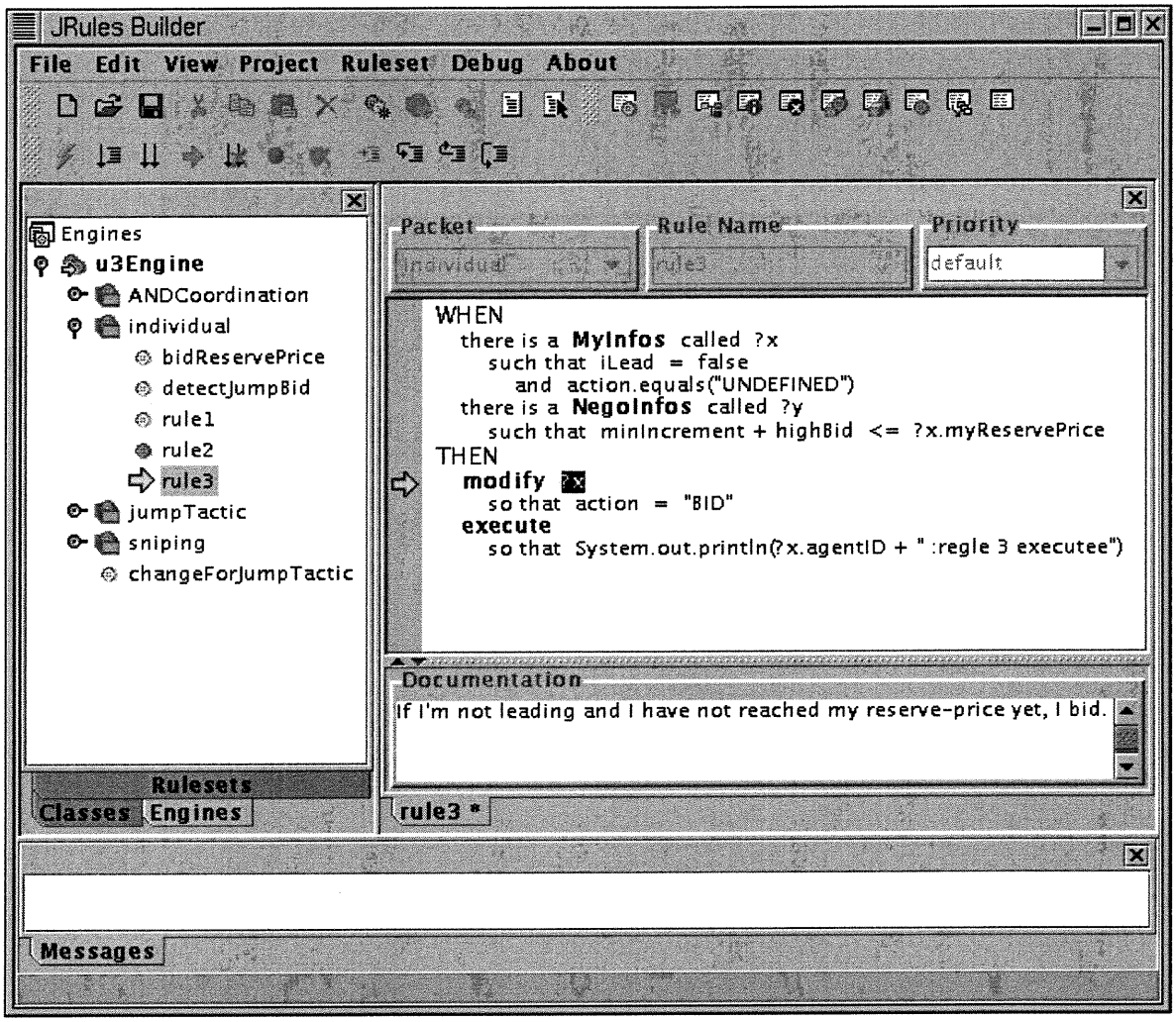

Figure 9 - L'outil JRulesBuilder

Une base de règles, une mémoire de travail et un agenda constituent ce qu'on appelle un contexte. Ce contexte sert d'interface entre le moteur et l'application Java. Il associe une base de règles à une mémoire de travail et implémente le moteur d'inférence qui gère les interactions entre l'application Java et la base de règles. Plus précisément, le moteur analyse le contenu de la mémoire de travail et les règles de production pour déterminer quelles règles sont éligibles, décide laquelle de ces règles va être exécutée et, enfin, réorganise la mémoire de travail pour refléter l'effet de chaque règle exécutée. Le contexte est implémenté comme une instance de la classe Java IlrContext fournie par JRules.

Enfin, comme on l'a mentionné un peu plus haut, un environnement de développement est fourni, le JRulesBuilder (cf. figure 9), qui permet d'éditer et d'exécuter des règles, mais aussi de visualiser entre autres le contenu de l'agenda et de la mémoire de travail (cf. figure 10).

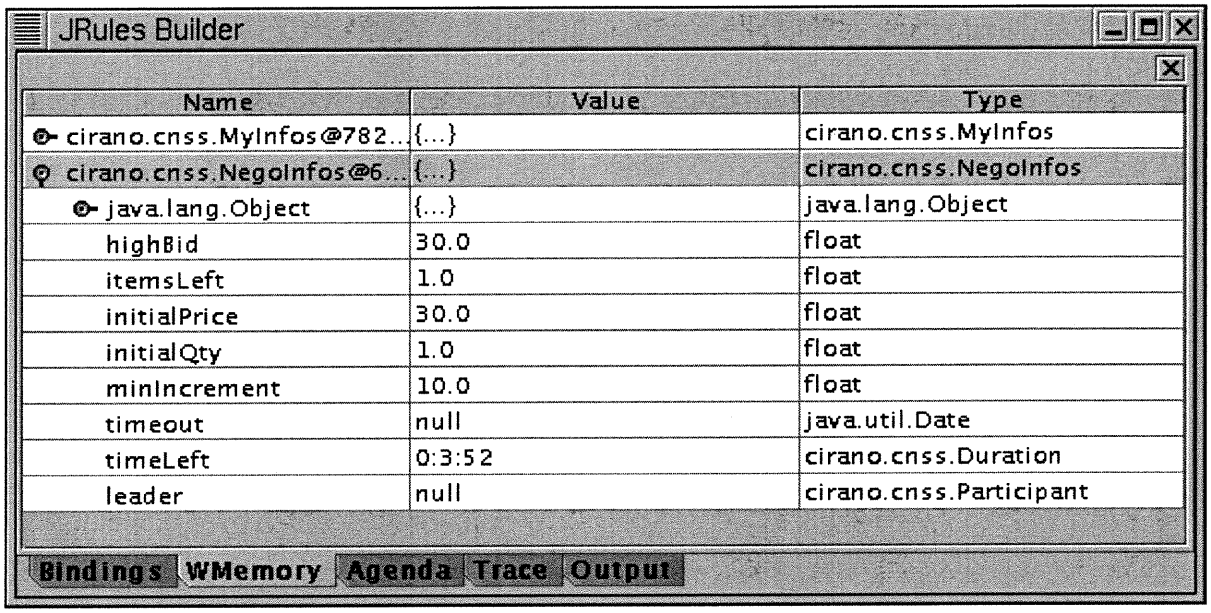

Figure 10 - Visualisation de la mémoire de travail avec JRulesBuilder

5.3 Le « blackboard »

Cette section commence par une discussion des implémentations possibles du « blackboard », suivie d'une description de l'implémentation retenue.

5.3.1 Implantations possibles

Pour les mêmes raisons qui ont mené le choix du moteur d'inférence, on se restreint aux solutions « blackboard » écrites en Java. De plus, étant donné que, dans le cas présent, le nombre d'agents à coordonner est relativement restreint (on estime qu'un usager pourrait s'engager dans au maximum une dizaine de négociations parallèles, donc qu'une dizaine d'agents au maximum devraient être coordonnés) et que le « blackboard » à implémenter

est assez simple dans sa conception (il n'y a pas de céduleur et seules des informations y seront consignées), on désire une application facilement intégrable et facilement utilisable.

À partir de là, deux options sont intéressantes. La première est le service Java Spaces de la technologie Jini [Jini02] (technologie réseau permettant de créer des communautés d'applications Java). Ce service fournit un des espaces de stockage pour des objets Java et, à ce titre, peut être employé comme « blackboard » partagé où l'on dépose et retrouve des objets. Ces objets peuvent simplement représenter des données ou des tâches à exécuter. Java Spaces est particulièrement adapté aux application distribuées et semble être aisé à intégrer ainsi qu'à utiliser. En effet, il offre un service de recherche (« lookup ») et les opérations de base sont simples et similaires à celles attendues pour un « blackboard ». De plus, c'est un service qui semble robuste et les problèmes de conflits d'accès et de cohérence sont normalement traités, ce qui devrait soulager quelque peu l'application l'utilisant. De fait, Java Spaces semble adapté pour des applications distribuées et costaudes.

L'autre option qui se présente est d'implanter notre propre solution « blackboard ». En effet, en raison de l'utilisation limitée supposée du « blackboard », du moins dans un premier temps et de la simplicité de son architecture, une implantation ad hoc serait simple à réaliser. De plus, elle offrirait l'avantage de tenir compte du fait que le « blackboard » est essentiellement manipulé au sein des règles JRules. En effet, il faut s'assurer que chaque mise à jour du « blackboard » se traduise par une mise à jour des contextes JRules, afin que les règles exécutées soient cohérentes avec le « blackboard ».

Par conséquent, dans un premier temps, l'implémentation ad hoc a été préférée mais la solution Java Spaces est envisageable si, dans le futur, il était décidé de basculer sur un système distribué.

5.3.2 Implantation ad hoc

Ci-dessous on explique de façon détaillée comment le « blackboard » a été implanté au sein d'INSULA. On y explique également son fonctionnement et de quelle façon il est manipulé par les règles.

5.3.2.1 La classe blackboard

La méthode ad hoc la plus simple consiste à implanter le « blackboard » comme un objet représentant une structure de données sur laquelle certaines opérations peuvent être effectuées. Par définition, il y a deux catégories d'opérations pouvant être exécutées sur un

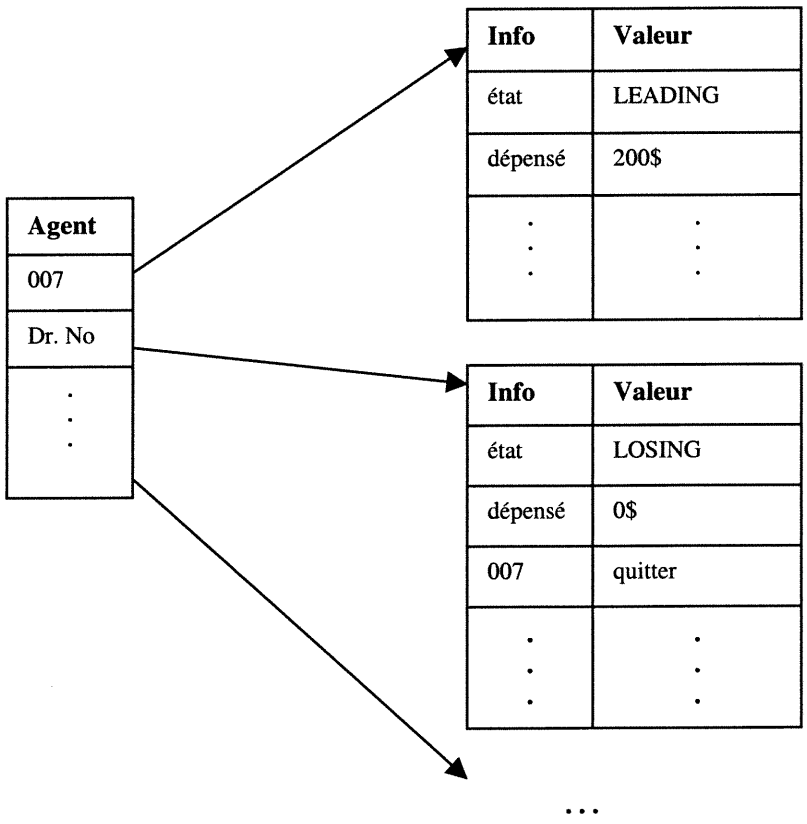

Figure 11 - Représentation schématique du blackboard

« blackboard »: les opérations d'écriture et les opérations de lecture. En outre, il est difficile de prévoir quels types d'information peuvent être écrits dans le « blackboard ». En effet, il

peut très bien s'agir de valeurs numériques, de valeurs booléennes ou de valeurs textuelles. Il peut même s'agir de données plus complexes. Il faut donc s'assurer d'être en mesure d'écrire et de lire n'importe quel type de données au besoin. Pour réaliser cela, il a été décidé de représenter le « blackboard » comme une table de hachage dont les clés correspondent aux identités des agents et les valeurs associées sont également des tables de hachage (cf. figure 11). Pour ces dernières, les clés désignent le genre d'information sous une forme textuelle (un objet de type String en Java) et les valeurs associées constituent la valeur de l'information elle même. Ainsi, un agent, pour écrire dans le « blackboard », doit spécifier son identité, le genre d'information qu'il écrit et l'information elle-même. Par exemple, un agent identifié par le code 007, qui voudrait écrire dans le « blackboard » qu'il est leader dans l'enchère anglaise à laquelle il participe, procéderait ainsi : *écrire (« 007 », « statut », « leader »)*. Réciproquement, pour lire, il faut spécifier l'identité de l'agent dont on veut lire une information et le genre de l'information : *lire (« 007 », « statut »)*.

Cette solution offre l'avantage de permettre l'écriture et la lecture de n'importe quel type de données comme désiré, et ce de façon illimitée. Cependant, c'est à l'usager de s'assurer qu'une information lue par un agent est bien rédigée par un autre, notamment au niveau de la rédaction des règles. Au total, une opération d'écriture (« write ») a été implantée ainsi que deux opérations de lecture : une opération qui permet simplement de lire (« read ») et une autre qui lit et retire (« take ») l'information du « blackboard ».

5.3.2.2 Le « blackboard » et JRules

Étant donné qu'on a opté pour une approche déclarative en ce qui concerne la coordination des agents et qu'on dispose déjà d'un moteur de règles pour l'aspect comportemental, la solution la plus simple consiste à utiliser le même moteur d'inférence pour les règles de coordination, c'est-à-dire le moteur JRules. Il faut que l'objet « blackboard » soit accessible à tous les agents devant être coordonnés et comme les manipulations du « blackboard » vont se faire via des règles JRules, il faut que l'objet « blackboard » soit accessible à tous les moteurs d'inférence de ces mêmes agents. Ceci est réalisé en l'intégrant au contexte JRules via la classe IlrContext (cf. section 5.2.2). Il faut toutefois ajouter une opération au « blackboard ». En effet, à chaque fois qu'une modification est faite, il faut avertir les

autres moteurs de règles que l'objet « blackboard » de leur contexte a été modifié et qu'ils doivent donc se mettre à jour. La fonction de mise à jour d'un contexte est fournie dans l'API de JRules. De là, la méthode de notification a facilement été ajoutée à la classe « blackboard ». Elle doit être appelée après chaque appel de méthode qui modifie l'objet « blackboard ». Pour finir, il est à signaler que l'ensemble des informations qui sont mises à jour régulièrement dans le « blackboard », sont écrites pour des raisons pratiques évidentes au niveau du code de l'agent, à l'étape *ObtenirInformations* du cycle d'exécution décrit à la section 5.1.2. Ainsi, les informations mises à jour à cette étape sont également recopiées dans le « blackboard ». En effet, on ne peut pas prévoir à l'avance quelles informations partagées seront utilisées par les règles, celles-ci n'étant pas fixes. Cependant, ceci n'empêche pas que des règles écrivent de nouvelles informations dans le « blackboard ».

5.4 Le proxy pour GNP

N'ayant à notre disposition qu'une seule plate-forme de négociation offrant une API Java permettant à des agents de se connecter, nous n'avons développé qu'un seul type de proxy. Pour comprendre comment implémenter le proxy, il faut en même temps comprendre les mécanismes régissant la plate-forme GNP. On commencera donc par expliquer l'architecture de la partie serveur de GNP, puis on expliquera comment les informations échangées entre le proxy et le serveur sont représentées.

5.4.1 EJB et GNP

La partie serveur de la plate-forme GNP repose sur une architecture EJB [EJB02]. La spécification Enterprise Java Beans (EJB) est une composante de Java 2 Enterprise Edition (J2EE) [J2EE02] qui définit une architecture pour le développement et le déploiement de composantes logicielles distribuées et transactionnelles. Avant de décrire les Beans, il faut définir ce qu'est un conteneur (« container »). Le conteneur est le lieu qui héberge une Bean et qui gère les interactions d'une application cliente avec elle. Le conteneur gère également tous les problèmes de sécurité, de persistance, de concurrence et de transaction. En fait, il n'y a aucun accès direct par une application à une Bean, tous les accès se font via

le conteneur par l'entremise de deux interfaces : Remote (distance) et Home (chez-soi). L'interface Remote représente les services de la Bean auprès des applications distantes tandis que l'interface Home concerne son cycle de vie (création, destruction, etc.). Home est utilisée pour obtenir une référence vers l'interface Remote de la Bean. Par ailleurs, on distingue deux catégories de Bean. La Bean entité est utilisée pour représenter les données d'une base de données. C'est une sorte d'interface orientée-objet pour les données. Quant à la Bean session, elle gère les interactions entre les Beans entités et les autres Beans session et exécute des tâches à la demande du client.

Dans GNP, à chacun des documents énumérés dans la brève description de GNP (cf. section 3.1) correspond une Bean entité, à savoir : Adjudication, Annonce, Negotiation, Order, ProductReference, Quote, Response, Rules. Et pour chaque entité XXX, il y a, bien entendu, une interface XXXHome correspondante. De même, aux différents rôles sont associées les Beans entités suivantes : AnnouncerSession, ReaderSession, NegotiatorSession et AdministratorSession. GNP fournit une API (« Application Programming Interface ») qui permet entre autre de manipuler ces Beans. Et c'est ainsi que fonctionne le proxy de l'agent implanté pour GNP : en utilisant les méthodes de l'API. La première opération devant être appelée sur le proxy est l'opération de connexion au serveur. Celle-ci nécessite un nom d'usager et un mot de passe valide, et a pour effet de lui donner une référence vers le conteneur. Par la suite, les fonctions directement liées au processus de négociation font essentiellement appel aux Beans ReaderSession et NegotiatorSession.

5.4.2 Échange de documents entre GNP et le proxy

Par ailleurs, les documents Adjudication, Annonce, etc., à l'interne de GNP, sont représentés sous forme de documents XML. Aussi, afin qu'il soit plus facile de manipuler l'information contenue dans ces documents, une classe Java a été implémentée pour chacun d'entre eux. Ces classes jouent le rôle de petites structures de données contenant les informations pertinentes. Ainsi, lorsque nécessaire, le proxy ne requiert qu'une seule fois les documents auprès de GNP, et, à partir de leur contenu, instancie les objets correspondants. Ceux-ci peuvent alors être accédés plusieurs fois au besoin, sans qu'il soit

nécessaire de demander à nouveau les documents au serveur et de les traiter une nouvelle fois. Il est à noter que la majorité des documents sont transférés du serveur vers le proxy. Par exemple, pour soumettre une mise, c'est-à-dire un ordre, le proxy construit une chaîne de caractères (un String en Java) dont la syntaxe est la syntaxe XML imposé par GNP pour les documents de type Order. Cette chaîne de caractères est ensuite soumise au serveur, qui, après avoir vérifié la syntaxe, l'utilisera pour créer un nouveau document Order.

5.5 Interface de surveillance des agents

L'interface de surveillance des agents permet de visualiser l'évolution des négociations dans lesquelles les agents sont engagés et notamment les actions de ces derniers. Il s'agit donc essentiellement d'une interface graphique qui se compose d'une table résumant les diverses informations et d'un espace affichant une trace d'exécution des agents (cf. figure 12).

5.5.1 Modèle de l'interface

Le modèle régissant l'interface fonctionne par un système d'événements et d'écouteurs (« listeners »). En effet, l'interface s'enregistre comme écouteur auprès des proxies de chacun des agents et auprès de la fabrique d'agents. Ceux-ci, en retour, dès qu'un événement survient, notifient les écouteurs (ici l'interface de surveillance). Dans le cas de la fabrique d'agents, les événements sont liés à la création et la destruction des agents, tandis que dans le cas du proxy, ils sont liés à leur connexion-déconnexion au serveur, à leur engagement-désengagement dans une négociation, au changement des informations courantes (prix, quantité disponible, etc.) d'une négociation et, enfin, à la soumission d'un ordre par un agent. Suite à chaque notification, les informations de la table sont mises à jour de même que la trace d'exécution.

5.5.2 L'interface GUI

Commençons par décrire le tableau. Dans la figure 12, chaque ligne correspond à un agent. En colonnes, dans l'ordre, on retrouve l'identité de l'agent, l'identificateur de la négociation dénommé « Global Primary Key » (GPK) dans GNP, l'état de la négociation (ouverte ou fermée), le prix de réserve de l'agent, le prix actuellement demandé, le prix soumis, la quantité d'items temporairement allouée et à quel prix, le prix auquel les items ont été adjugés à l'agent (ce champ est rempli seulement lorsque la négociation est terminée et qu'il y a eu une adjudication faite pour l'agent concerné) et, pour finir, l'état de l'agent (cf. section 5.1 .2).

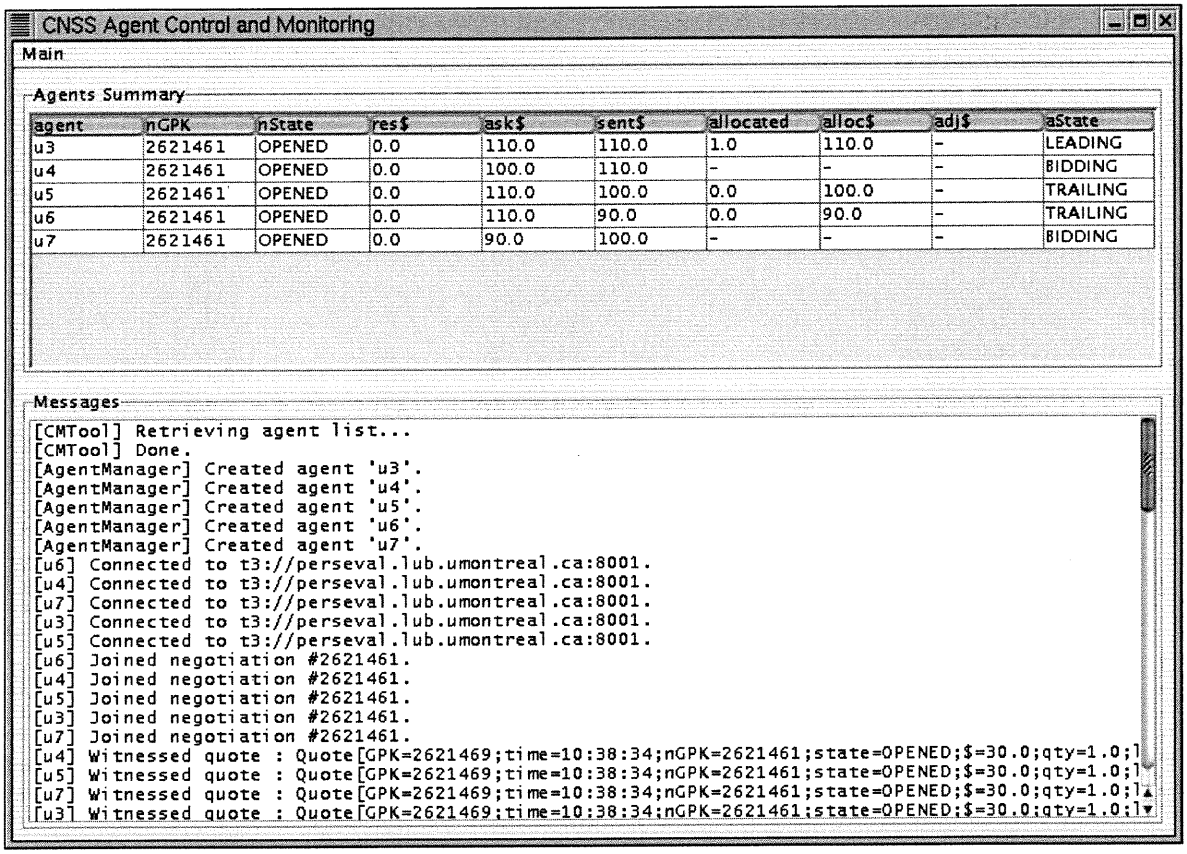

Figure 12 - Interface de surveillance des agents

Dans la partie textuelle, les messages rapportant les événement des agents suivis sont affichés. Il y a ainsi des messages indiquant les connexions, les engagements dans une négociation, les soumissions d'ordres, les mises à jour des informations des négociation,

etc. À ce sujet, chaque agent construit une historique de ces actions qui est enregistrée, à la fin de son exécution, dans un fichier identifié par son nom. De plus, JRules permet, au besoin, d'obtenir une trace de l'exécution du moteur d'inférence.

Par ailleurs, comme on peut le constater sur la figure 12, cette interface est surtout adaptée au contexte des enchères. Pour la rendre plus générique, quelques modifications au niveau des informations qui y sont affichées devraient y être apportées. En fait, l'idéal serait de la paramétrer en fonction du type de négociation. Mais, l'interface, dans son état, est pour le moment suffisante et remplit son rôle. Pour finir, on remarquera l'absence de boutons d'interventions qu'on a préféré situer (pour des raisons pratiques) au niveau du module de fabrique d'agents.

5.6 La fabrique d'agents

La fabrique est le lien direct entre l'usager et l'infrastructure. C'est par son intermédiaire que l'usager crée de nouveaux agents, les engage dans une négociation, les relie à un « blackboard », etc.

5.6.1 Le module de fabrique d'agents et CONSENSUS

La fabrique a été implantée de façon minimale, pour permettre le fonctionnement simultané de plusieurs agents et d'effectuer certains tests (cf. chapitre 6). Aussi, plusieurs fonctionnalités et options n'ont pas été prises en compte. En effet, le module de fabrique des agents fait partie intégrante de l'architecture de CONSENSUS (cf. section 3.2). Par conséquent, un travail de réflexion et de spécification est prévu prochainement pour le raffinement de ce module. Sur le schéma d'architecture de CONSENSUS (cf. figure 2), on peut notamment voir que la fabrique doit être reliée à un dépôt contenant les règles constituantes des protocoles de négociation et au système de courtage. Elle joue en quelque sorte un rôle de charnière entre ces composantes. Pour toutes ces raisons, dans l'optique d'une intégration complète de l'infrastructure INSULA à CONSENSUS, le module de fabrique d'agents proposé ici devra être modifié pour prendre en compte d'autres critères.

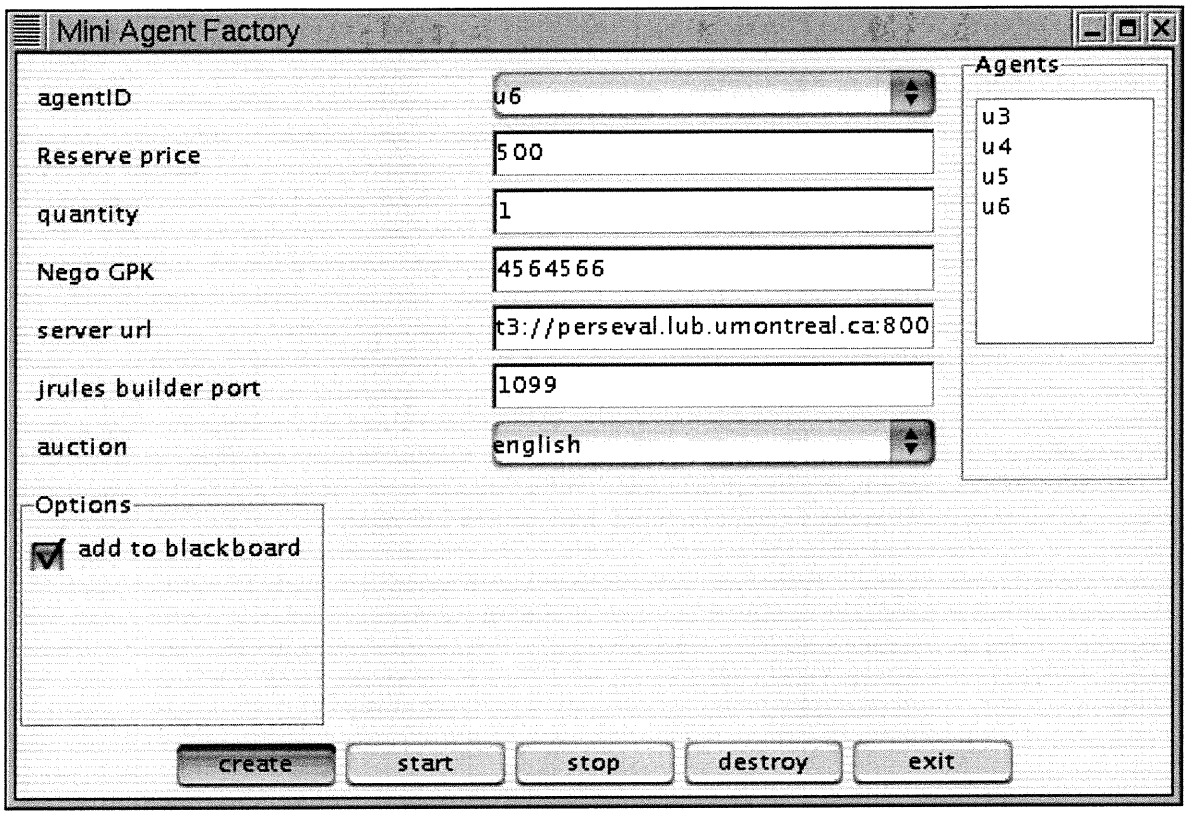

Figure 13 - Interface graphique de la fabrique d'agents

5.6.2 Scénario d'utilisation

Pour permettre la création d'agents, la fabrique doit permettre de spécifier les paramètres qui varieront d'une négociation à l'autre. Ainsi, pour chaque nouvel agent, l'usager doit spécifier son identificateur, l'adresse du serveur de négociation sur lequel il est appelé à négocier, le type de la négociation, l'identificateur de la négociation sur le serveur, son prix de réserve et le nombre d'items qu'il désire acquérir (cf. figure 13). Dans les cas où on désire coordonner des agents, la fabrique doit également s'occuper de la création du « blackboard ». Une option pour lier l'agent à ce dernier doit de plus être disponible.

Une fois toutes ces spécifications faites, l'usager n'a qu'à appuyer sur le bouton de création. Une fenêtre de dialogue est alors lancée (cf. figure 14) et permet à l'usager de sélectionner le fichier de règles JRules qui correspond à la stratégie désirée. Ceci fait, l'instanciation de l'agent est achevée et son identité apparaîtra dans la liste des agents

affichée dans la partie droite de la console (cf. figure 13). Pour le moment, un seul « blackboard » peut être créé et on ne peut visualiser quels agents y sont reliés. À partir de la liste des agents, on peut sélectionner un groupe d'agents et, au choix, les activer en appuyant sur le bouton *start* ou les arrêter ou encore les éliminer avec respectivement les boutons *stop* et *destroy*.

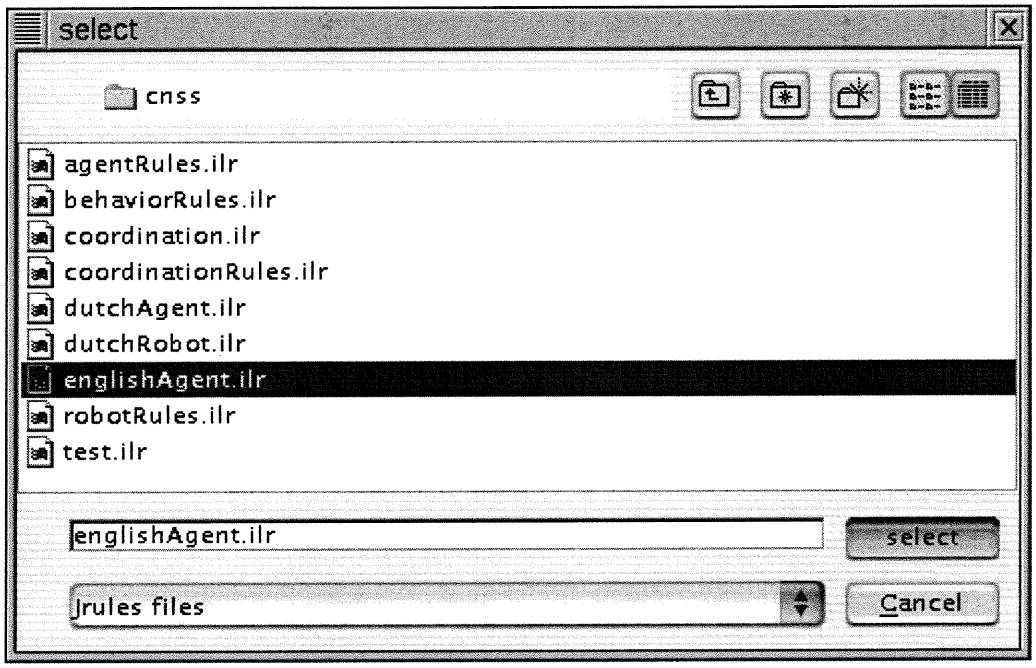

Figure 14 - Sélectionneur des fichiers de règles

5.7 Exploitation et traitement des informations

L'information est en quelque sorte le nerf de la négociation, surtout de la négociation automatisée. En effet tout comportement, toute stratégie et donc toute décision est prise en fonction de l'information que l'on détient. Généralement, dans le cadre d'une négociation automatisée, les participants sont égaux en ce qui a trait aux informations découlant du processus de négociation. Chacun peut cependant détenir des informations particulières sur le produit à acquérir. Pour illustrer cela, prenons le cas de la vente dans une enchère anglaise d'une voiture de marque Béta. En fonction de leurs connaissances de la marque en question et de leur expérience en matière automobile, les acquéreurs vont avoir une évaluation différente de la valeur de la voiture et donc vont avoir des prix de réserve

différents. Par contre, au cours de la négociation, ils ont accès aux mêmes informations, à savoir la mise la plus haute, la date de fin de l'enchère, etc.

Dans une négociation, on peut distinguer deux catégories d'informations, les informations publiques, partagées par l'ensemble des participants, et les informations personnelles, privées.

5.7.1 Informations disponibles

Les informations publiques disponibles vont varier d'un type de négociation à un autre. En effet, les protocoles de négociation régissent également l'information accessible aux participants. Mais d'une façon générale, les participants ont accès au prix courant, à la quantité disponible et aux conditions d'arrêt de la négociation (le plus souvent une date de fin). Quant aux informations privées, on retrouve normalement le prix de réserve et la quantité d'items à acheter et le prix à soumettre.

5.7.2 Représentation

Les informations sont utilisées et exploitées au niveau des règles. Comme les règles JRules manipulent directement des objets Java, la solution la plus simple a été d'encapsuler les informations dans des objets. Il y a notamment un objet pour les données personnelles et un autre pour les données publiques. Ces objets ne servent que de structure de données pour le moment. En outre, une classe nommée Flag permet de représenter n'importe quel type d'information. Similairement à la classe blackboard, cette structure de données est composée d'une chaîne de caractères servant à nommer l'information et d'une table de hachage permettant de placer un nombre indéterminé de valeurs de types différents. Les objets de cette classe doivent normalement être créés par des règles, suite à un raisonnement quelconque et être utilisés comme condition par d'autres règles. C'est donc en quelque sorte un cadre générique pour représenter des données.

5.8 Données d'implantation

Ci-dessous sont énumérés quelques détails de l'implantation d'INSULA :

- Langage : Java
- Système à base de règle : ILOG JRules
- Nombre de classes et d'interfaces : 40
- Estimation du nombre de lignes de code : 4900
- Modules implantés par rapport à l'architecture conceptuelle: tous

En résumé, on peut dire qu'INSULA implémente l'ensemble des modules de l'architecture conceptuelle illustrée à la figure 7. INSULA fonctionne exclusivement en Java ou avec des produits Java. De plus, INSULA respecte dans sa conception le caractère générique de l'infrastructure. Cependant, certaines améliorations sont à prévoir pour la rendre pleinement générique. Une réflexion un peu plus poussée serait notamment nécessaire pour le module de fabrique. En effet, le développement d'INSULA s'est essentiellement fait dans un contexte d'utilisation limité aux enchères classiques et les améliorations en question permettraient d'étendre ce contexte aux autres négociations.

Chapitre 6: Validation

Ce chapitre couvre les tests qui ont été menés pour évaluer l'infrastructure INSULA. On introduira dans un premier temps l'objectif de ces tests, puis on décrira le contexte dans lequel ils se sont déroulés. Enfin, on terminera par la description elle-même des tests ainsi que de leurs résultats.

6.1 Objectif

Il est important de souligner ici qu'il ne s'agit pas d'expérimentations ayant pour but d'évaluer des stratégies de négociation. En effet, ce type de tests relève plus des domaines de l'économie et de la recherche opérationnelle et constituerait un travail de recherche en soi. L'objectif consiste plutôt à démontrer que l'infrastructure INSULA est opérationnelle et donc à démontrer les possibilités qu'offre la combinaison agent-système à base de règles dans le cadre des négociations automatisées. C'est, rappelons-le, un des objectifs de ce mémoire.

Les tests ont été élaborés pour permettre, dans la mesure du possible, l'évaluation des dimensions suivantes :

- différents types de négociation,
- changement dynamique de stratégie,
- coordination d'agents,
- achat d'items versus vente d'items

Certains des scénarios de négociation décrits dans les sections 6.3 à 6.9 ont été inspirés des travaux d'Anthony et al. [AHDJ01] et de Stone et al. [SLSK00] présentés au chapitre 3. Pour chacun des cas où cela s'applique, une référence plus précise est donnée.

6.2 Contexte

Le contexte général des tests est principalement caractérisé par le serveur de négociation, la négociation elle-même et les participants.

6.2.1 Serveur de négociation

Les tests se sont déroulés avec la plate-forme GNP (version 1.1). Dans le cas des négociations parallèles, n'ayant à notre disposition qu'un seul type de serveur (GNP), elles ont été lancées sur la même instance de GNP afin de réduire l'exploitation des ressources, puisque l'utilisation de plusieurs serveurs GNP n'aurait rien apporté de plus aux expériences.

6.2.2 Négociations

Deux types de négociation ont été utilisés, l'enchère anglaise (cf. section 2.3.1) et l'enchère hollandaise (cf. section 2.3.2). Les deux enchères sont des enchères multi-items. Cependant, l'enchère anglaise a principalement été utilisée pour la vente d'un seul produit. De plus, dans chacun des tests, les enchères ont une durée limitée. Pour la hollandaise, la négociation peut prendre fin avant son échéance s'il n'y a plus d'items disponibles. Le type des produits en vente est quelconque, aucune stratégie testée ne jouant sur leurs caractéristiques. Par contre, dans le cas d'une coordination OU, les produits en vente dans les enchères parallèles sont considérés comme identiques ou équivalents.

6.2.3 Participants

Pour chaque négociation, nous avons à notre disposition trois types de participant. Le premier, le *joueur*, désigne tout simplement un usager (humain) qui interagit avec le serveur par le bais d'une interface. Il est seulement utilisé lorsqu'on désire provoquer une situation particulière dans la négociation pour fins d'illustration. L'*agent* désigne tout simplement un agent INSULA, doté d'une stratégie particulière. Quant au *robot*, il s'agit en fait d'un agent dont le comportement est simple, prédéterminé et statique, il est en quelque sorte le pendant de ce qu'on appelle couramment le « proxy-bidding ».

Pour l'enchère anglaise à un item, le robot est doté des deux règles suivantes :

1. **Si** le prix demandé est supérieur à mon prix de réserve **alors** je quitte.
2. **Si** le prix demandé est inférieur à mon prix de réserve **alors** je mise.

Pour l'enchère hollandaise, le comportement du robot est représenté par les deux règles suivantes :

1. **Si** je n'ai pas acheté le nombre d'items que je dois acheter et que le prix courant a atteint mon évaluation **alors** je mise.
2. **Si** j'ai acheté le nombre d'items que je dois acheter **alors** je quitte.

En ce qui concerne le nombre de participants, il a varié de 5 à 20 d'un test à un autre. Mais le nombre de participants n'apporte rien par rapport aux objectifs des tests. Par contre, le nombre de participants serait déterminant si on devait tester la plate-forme GNP, ce qui n'est pas le cas ici. Par ailleurs, notons que le proxy-bidding est l'équivalent de la tactique de la bonne affaire d'Anthony et al. (cf. section 3.3.5).

6.3 Sniping dans une enchère anglaise

Dans ce test, on a doté l'agent du comportement couramment nommé « sniping » et déjà évoqué au chapitre 2. Le « sniping » a généralement lieu dans une enchère anglaise et consiste à demeurer dans le rôle d'observateur tout au long de l'enchère et d'attendre les

derniers instants avant la fin de celle-ci pour miser. Plus la mise est soumise proche de la fin de l'enchère, moins il y a de chances pour que les autres participants aient le temps de réagir et de contre-miser. On remarquera, en outre, que le « sniping » s'apparente à la tactique du temps restant employée par Anthony et al. (cf. section 3.3.5) ainsi qu'à la stratégie utilisée par les agents ATTac pour acquérir les chambres d'hôtel (cf. section 3.3.4). Ci-dessous on peut voir la règle JRules qui permet de réaliser le « sniping ».

```
/** I'm waiting until the last moment before bidding
*/
rule snipe
{
  packet = sniping;
  property activation = true;

  when
  {
        ?x: NegoInfos(?t:timeLeft);
        evaluate(?t.compareTo(new Duration(0,0,50)) < 1);
        ?y: MyInfos(iLead == false; action.equals("UNDEFINED");
?x.minIncrement + ?x.highBid <= myReservePrice);
  }
  then
  {
        modify ?y
        {
                priceToBid = ?x.highBid + myIncrement;
                action = "BID";
                refreshTime = 1000;
        }
  }
};
```

Cette règle peut se traduire ainsi :

> Lorsqu'il reste moins de 50 secondes avant la fin de l'enchère et que je suis en mesure de miser (le prix demandé est inférieur à mon prix de réserve) alors je mise et je raccourcis mon temps de rafraîchissement de l'information à une seconde.

Il est important d'avoir un temps de rafraîchissement assez court (rappel : l'agent fonctionne selon le modèle du tirer) dans les derniers instants d'une enchère. En effet, d'autres participants (incluant d'autres agents) peuvent avoir la même stratégie. Dans ce contexte, le « sniping » ne garantit pas de gagner. Il faut alors faire en sorte de maximiser le temps qu'on peut avoir pour réagir.

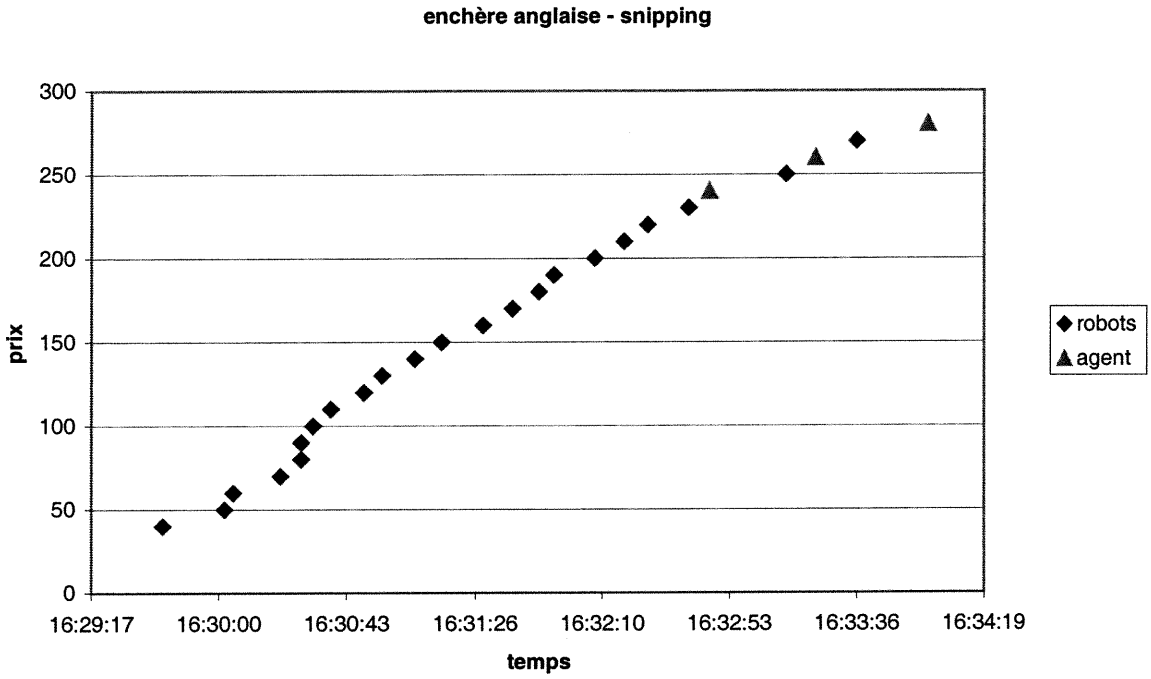

Figure 15 - Sniping dans une enchère anglaise

Le graphique de la figure 15 permet d'illustrer deux comportements, le « proxy bidding » des robots et le « sniping » par un agent. En effet, on remarque que tout au long de l'enchère seuls les robots participent et on peut alors constater que le prix augmente de façon régulière dans le temps, c'est-à-dire de façon linéaire et on peut aussi observer que l'agent ne participe que dans la dernière minute de l'enchère. Cette dernière partie du graphe illustre bien d'ailleurs pourquoi il est judicieux pour l'agent d'abaisser son temps de rafraîchissement de l'information. En effet, on voit bien que des robots ont quand même eu le temps de contre-miser une ou deux fois. On pourrait objecter qu'il suffit de raccourcir le moment du « sniping », mais c'est délicat car si le laps de temps est trop court et s'il y a beaucoup d'activités sur le serveur ou sur le réseau, ce qui ralentit un peu le traitement des mises par le serveur, il peut advenir que la mise de l'agent ne sera pas traitée à temps.

6.4 Enchère hollandaise

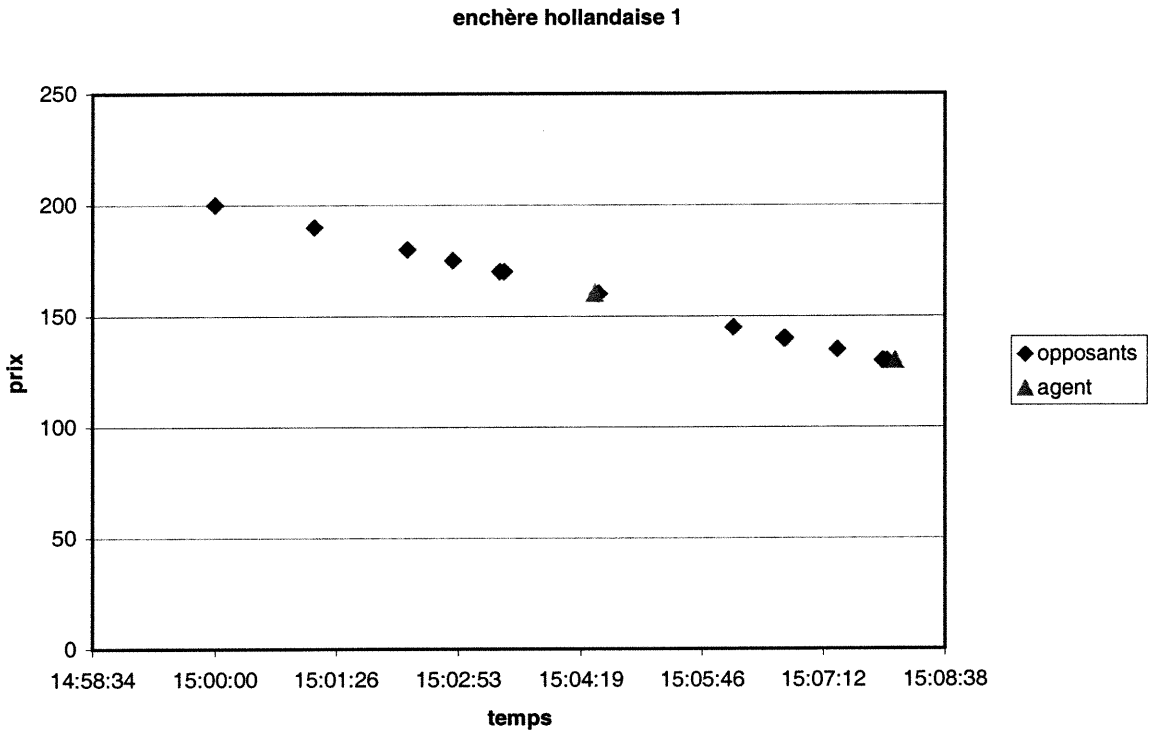

Figure 16 - Enchère hollandaise multi-items

Ce scénario a pour but de montrer qu'INSULA est effectivement polyvalent et s'adapte bien à un autre type de négociation, en l'occurrence à l'enchère hollandaise multi-items. On suppose que l'usager mandataire de l'agent a un nombre minimum x d'items à acquérir (x = 3 dans cet exemple), mais que si les prix sont intéressants, il est prêt à en acquérir jusqu'à $x + y$ (ici $y = 4$). La stratégie employée consiste alors à miser pour x items lorsque le prix atteint l'évaluation de l'usager, puis de laisser les prix continuer à descendre jusqu'à ce qu'il n'y ait plus qu'y items encore disponibles. Cette stratégie permet de garantir à l'usager d'acquérir son quota nécessaire et d'obtenir le reste à un prix minimum. Elle est illustrée par le graphique de la figure 16 où on voit l'agent miser une première fois, puis une seconde fois avant la fin de l'enchère. On peut noter d'ailleurs que l'enchère prend fin avec cette dernière mise car, conséquemment, il ne reste plus d'items disponibles.

La règle employée pour acheter le nombre maximum d'items est présentée ci-dessous :

```
/*
 * FEW ITEMS LEFT
 * if the number of items left is minimum,
 * I buy the missing items to reach my maximum number of items.
 */
rule rule3
{
  packet = individual;
  priority = high;
  property activation = true;

  when
  {
        ?x: MyInfos( );
        ?y: NegoInfos( ?x.myReservePrice >= highBid; itemsLeft <=
(?x.maxItems - ?x.itemsBought) );
  }
  then
  {
        modify ?x
        {
                itemsToBuy = ?y.itemsLeft;
                action = "BID";
        }
  }
};
```

6.5 « Jump bidding » dans une enchère anglaise

La stratégie décrite dans cette section est inspirée de la tactique du désespoir d'Anthony et al. (cf. section 3.3.5). On s'intéresse au cas où, dans une enchère anglaise, on voudrait que face à une forte activité de la part des participants (par exemple), l'agent s'affirme en misant bien plus haut que ne l'exige l'incrément minimum, ce qui créerait un écart avec les autres participants, d'où le nom de « jump bid » (littéralement : saut de mise) donné à cette stratégie. Ce comportement fait prendre le risque à l'usager de payer plus cher le produit, mais, en même temps, si celui-ci a une bonne évaluation de la valeur du produit, le risque est atténué. De plus, cela a l'avantage d'éliminer les participants plus frileux et de montrer aux autres sa détermination à obtenir le produit. Toutefois, les concurrents peuvent aussi interpréter ce geste comme un coup de bluff et surenchérir. Il faut alors que l'agent soit encore en mesure de répondre. Autrement dit, pour son saut de mise, il ne doit pas miser

son prix de réserve pour conserver une marge de manœuvre. Un exemple de saut de mise est visible sur le graphe de la figure 17.

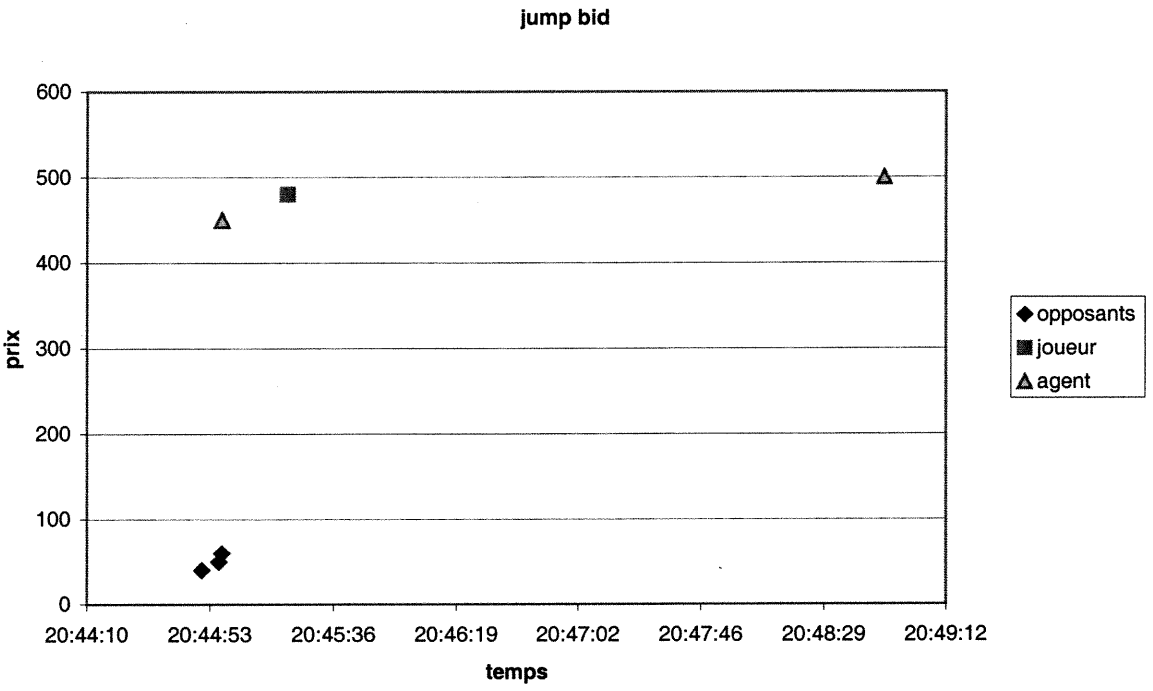

Figure 17 - Saut de mise dans une enchère anglaise

D'autre part, il peut aussi se produire la situation inverse, c'est-à-dire que l'agent détecte un comportement de saut de mise chez un participant. La stratégie consiste à ce moment-là à attendre pour faire un « sniping ».

6.6 Changement de stratégie dans une coordination ET

Ici, nous allons illustrer l'utilisation de règles pour la coordination d'agents. Plus précisément, dans ce cas, deux agents sont engagés dans des enchères anglaises pour obtenir deux items complémentaires, il s'agit donc d'une coordination ET (cf. section 3.2.3). Comme il est important d'obtenir les deux items (posséder un seul des deux a peu d'intérêt), il faut s'assurer que si on en obtient un, on obtienne le deuxième. La stratégie est donc la suivante pour les deux agents : miser prudemment, c'est-à-dire en utilisant les

mêmes règles que les robots. Si un l'emporte dans son enchère, il le communique alors grâce au « blackboard » et l'autre agent va alors changer sa stratégie pour une stratégie de saut de mise. Réciproquement, si l'un des deux perd, il écrit cette information dans le « blackboard » et l'autre agent prendra alors la décision de se désengager de son enchère, s'il le peut (soit l'enchère le permet avec une pénalité financière, soit il n'est pas en position de leader). Le graphique de la figure 18 met bien en valeur le changement de stratégie (saut de mise) qu'effectue l'agent A. On remarquera toutefois qu'il y a un certain décalage entre la dernière mise de l'agent B et le saut de mise de A. Ceci s'explique par le fait que l'enchère de B se termine à 15:41:07 seulement et qu'avant ce temps-là B ne peut pas encore affirmer qu'il a gagné et le communiquer à A par le « blackboard ».

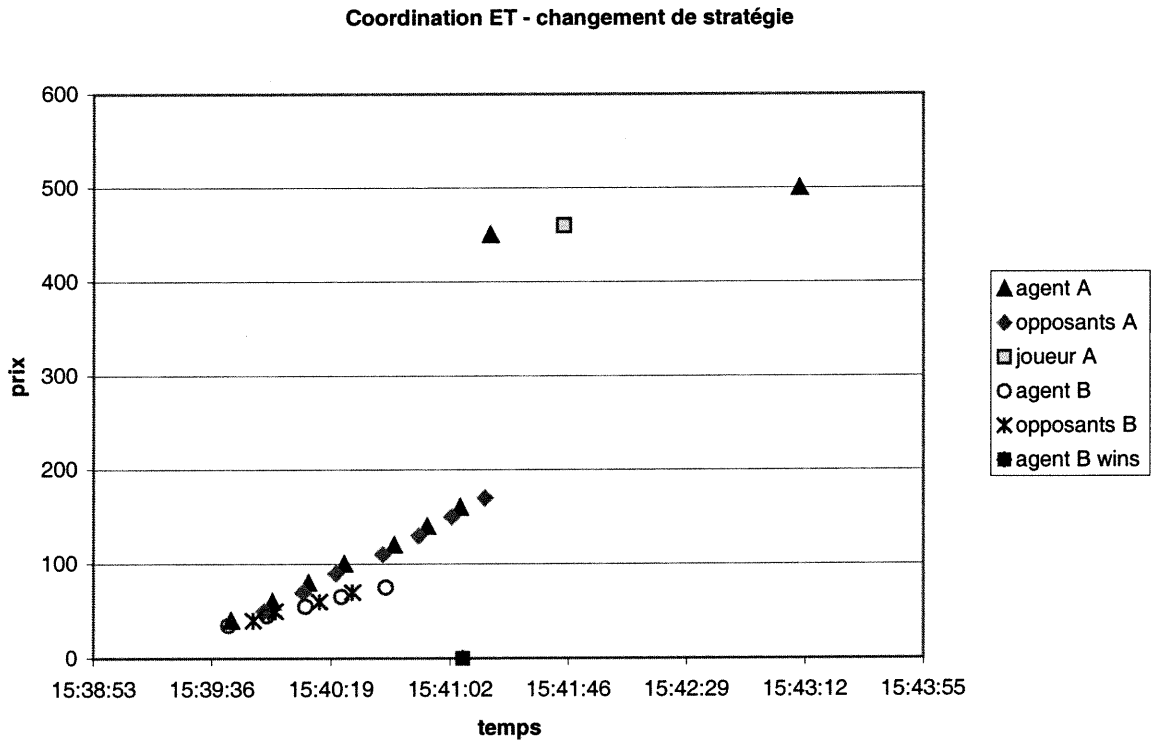

Figure 18 - Changement de stratégie dans une coordination ET

Lors de la description du fonctionnement de JRules (cf. section 5.2.2), on a mentionné que les règles peuvent être regroupées par paquets et c'est cette possibilité qui est exploitée ici; les règles de la première stratégie sont regroupées dans un paquet nommé « individual »

tandis que les règles de la seconde le sont dans un paquet nommé « jumpTactic ». Le changement de stratégie est réalisé grâce à la règle ci-dessous. Elle désactive le paquet « individual » et active le paquet « jumpTactic ».

```
/*
 *  "AND" negotiation
 *  If "u5"'s has won, I change my strategy for jump bidding.
 */
rule coordinate1
{
  packet = ANDCoordination;
  priority = high;
  property activation = true;

  when
  {
        ?x: MyInfos(iLead == false);
        not Flag(type.equals("u5 HasWon")) ;
        BlackBoard(?u3State:read("u5", "state")) from getBB();
        evaluate(?u3State.equals (new Integer(NegotiationAgent.WINNING)));
        IlrRuleset(?rulesToDeactivate:getPacketRules("individual")) from
?context.getRuleset();
        IlrRuleset(?rulesToActivate:getPacketRules("jumpTactic")) from
?context.getRuleset();
  }
  then
  {
        ?context.deactivateRules(?rulesToDeactivate);
        ?context.activateRules(?rulesToActivate);
        assert Flag("u5 HasWon")  ;
  }
};
```

6.7 Coordination (A et B) ou C

Un autre test de coordination a été effectué. Le scénario consiste à acquérir un billet d'avion pour une destination quelconque. On imagine qu'il y a deux possibilités : soit le voyage est direct et il n'y a qu'un seul billet à acheter, soit il y a une escale et, à ce moment-là, il y a deux billets achetés. Bien entendu, on cherche la solution la moins chère des deux. On est donc en présence d'une coordination OU pour exprimer l'exclusion des deux solutions et d'une coordination ET pour la solution avec escale où l'achat de deux billets est nécessaire. Symboliquement, on peut exprimer le scénario par l'expression (A et B) ou C. Pour réaliser le test, on a choisi trois enchères anglaises et les règles ont été écrites pour donner le comportement suivant aux agents :

- L'agent C ne mise que si ni A et ni B ne sont en position de leader dans leurs enchères respectives.
- De même, les agents A et B ne misent que si C n'est pas leader dans son enchère.
- C ne mise que lorsque le prix demandé dans son enchère est inférieur ou égal à la somme des prix demandés dans les enchères de A et B, et inversement pour A et B.
- Si A ou B gagne, alors C s'arrête et si C gagne alors A et B s'arrêtent.
- Si A perd alors B s'arrête et inversement.

Le graphique de la figure 19 illustre le cas ou l'agent C gagne. On peut constater, que l'exclusion des deux solutions est bien respectée et que les mises de C sont toujours effectuées quand le prix demandé est inférieur à la somme des deux autres, et inversement pour A et B.

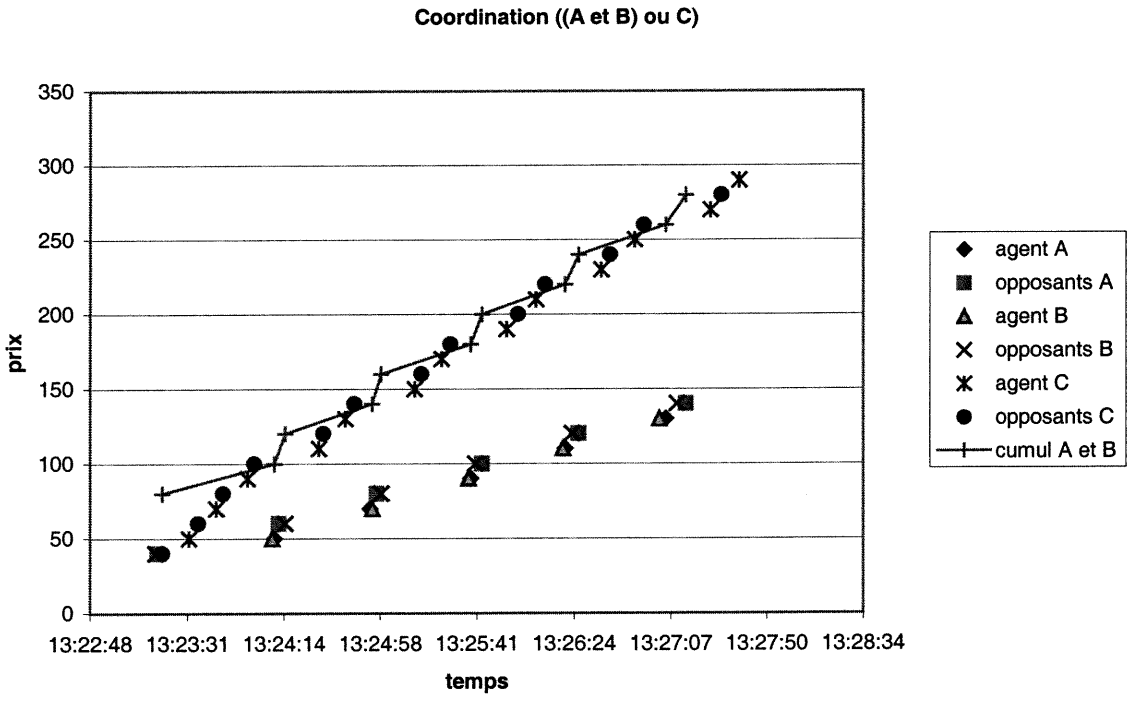

Figure 19 - Coordination (A et B) ou C

6.8 Agent vendeur

Pour finir, nous avons effectué un test où, cette fois-ci, au lieu d'avoir un agent qui achète des items, nous avons un agent qui vend des items. Pour cela, nous avons choisi une enchère hollandaise multi-items. Le but est de montrer que la conception d'INSULA ainsi que l'utilisation des règles est suffisamment flexible pour permettre de passer d'un agent vendeur à un agent acheteur en changeant seulement les règles qui régissent son comportement. Dans ce scénario, l'agent a notamment été doté de la règle suivante (syntaxe JRules) :

```
/*  if the price seems to be too expensive then I reduce the price
 */
rule decreasePrice
{
  packet = individual;
  priority = high;
  property activation = true;

  when
  {
        not Flag();
        ?x: MyInfos( );
        ?y: NegoInfos( itemsLeft >= initialQty * 0.75; highBid <= 200 );
  }
  then
  {
        modify ?x
        {
                priceToBid = 150;
                itemsToBuy = ?y.itemsLeft;
                action = "SELL";
        }
        assert Flag("Price decreased");
  }
};
```

En d'autres termes, cette règle dit :

> Si le prix atteint 200 et moins d'un quart des items ont été vendus, alors diminuer le prix à 150.

Elle exprime le fait que le vendeur soit inquiet d'avoir fixé un prix trop haut, ce qui fait en sorte qu'il a du mal à écouler sa marchandise ou que l'enchère va durer trop longtemps.

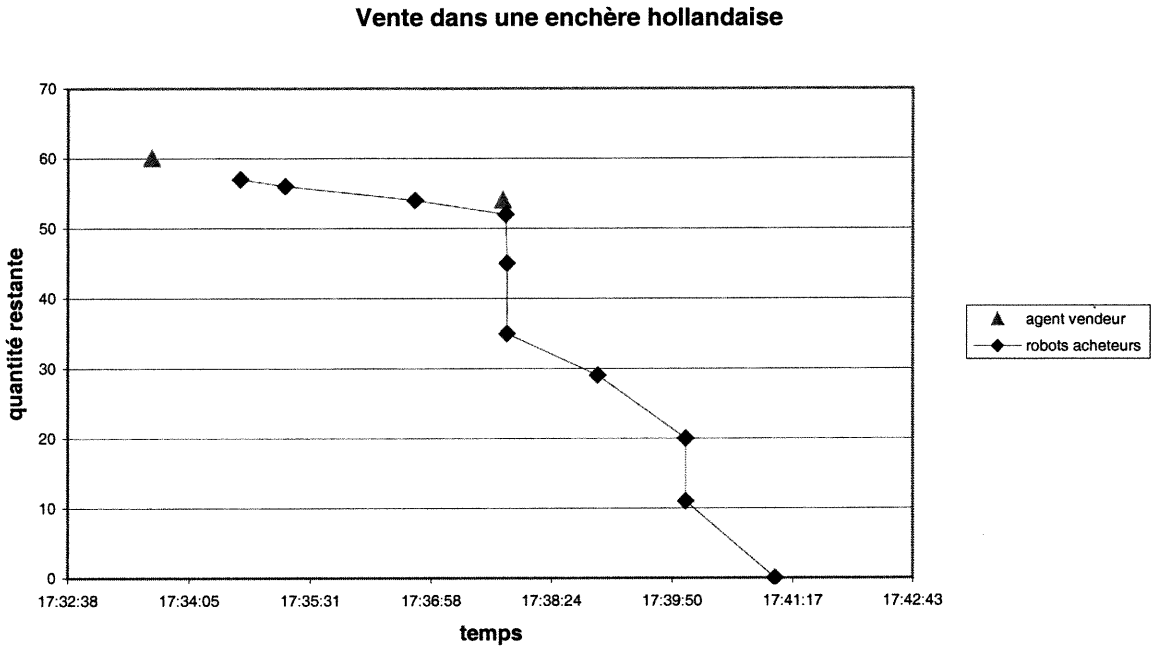

Figure 20 - Agent vendeur dans une enchère hollandaise (1)

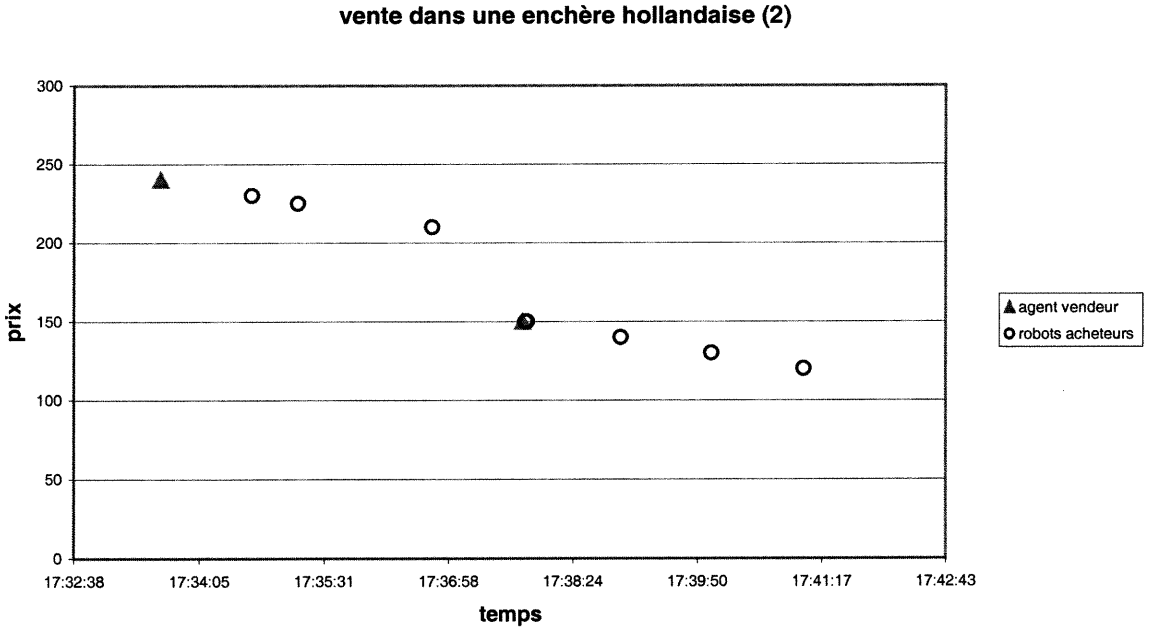

Figure 21 - Agent vendeur dans une enchère hollandaise (2)

Sur le graphique de la figure 20, on peut effectivement voir que lorsque l'agent diminue le prix à 150 (deuxième triangle sur le graphe) la quantité d'items chute brusquement. Avant ça, on peut remarquer que peu d'achats étaient effectués. La chute des prix, conséquemment à l'exécution de la règle est parfaitement visible sur le graphe de la figure 21, sur lequel on observe une fracture lorsque le prix passe brusquement de 210 à 150.

6.9 Autres tests

En marge des tests précédents, d'autres essais et réalisations ont été effectués. Parmi eux, on notera la réalisation d'un agent à prix progressif, c'est-à-dire semi-autonome (cf. section 2.1.1). Pour cela, il a suffi d'écrire une règle qui fait en sorte que, lorsque le prix atteint dans l'enchère (anglaise) est compris dans l'intervalle spécifié par l'usager, une fenêtre (cf. figure 22) apparaisse offrant l'option d'attendre, de modifier l'intervalle (si on a une idée plus précise de la valeur du produit) ou encore de quitter la négociation.

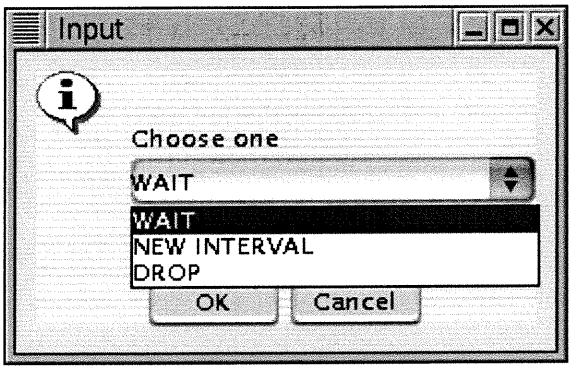

Figure 22 - fenêtre de sollicitation lancée par un agent à prix progressif

D'autre part, une version préliminaire et partielle d'INSULA a été intégrée avec succès au sein de CONSENSUS. La version en question ne comportait pas de « blackboard »; mais le scénario du voyageur décrit à la section 3.2 a pu être simulé, toute la coordination nécessaire étant assumée par le « workflow ».

6.10 Résumé

De tous ces tests, on retiendra qu'on a pu, en changeant uniquement les règles de comportement, passer d'un type de négociation à un autre, changer de stratégie en cours d'exécution, agir aussi bien comme acheteur que comme vendeur et coordonner plusieurs agents. On notera également que le nombre de règles nécessaire à chaque fois est assez restreint. Au chapitre suivant, à partir de l'expérience acquise dans ce travail, nous discuterons des avantages et des inconvénients de l'utilisation des règles par des agents de négociation.

Chapitre 7: Discussion et conclusion

Ce chapitre clôt ce mémoire en discutant des résultats obtenus et des contributions apportées. Il présente également les principaux axes qui vont orienter les travaux futurs avant de proposer une synthèse et de conclure.

7.1 Discussion

Cette section s'articule selon le modèle simple « avantages versus inconvénients ». On présentera donc en premier lieu les forces de notre approche, puis les faiblesses.

7.1.1 Forces

Parmi les nombreux avantages qu'on peut citer, on retiendra d'abord le haut niveau d'abstraction qu'offrent les règles. En effet, celles-ci permettent de ne prendre en considération que les éléments essentiels au processus de négociation, sans se préoccuper de détails de bas niveau. De même, la proximité du raisonnement par règles avec la compréhension humaine est un atout important.

D'autre part, l'approche à base de règles est d'une grande souplesse dans la mesure où elle permet à un agent de s'adapter à diverses situations de négociations sans apporter de modifications à ses fonctionnalités. On a ainsi vu que l'agent pouvait être acheteur ou vendeur, être acteur dans une enchère ascendante ou descendante et coopérer avec d'autres agents lorsque nécessaire. Pour que ceci soit réalisable, il faut s'être préalablement assuré de la modularité des composantes et des fonctionnalités de l'agent et donc avoir identifié ce qui variait d'une négociation électronique à une autre.

On retiendra également, sur le plan de la souplesse et du dynamisme, la possibilité de modifier le comportement d'un agent, c'est-à-dire de changer sa stratégie, durant le processus de négociation et ce, en utilisant seulement une règle .

Par ailleurs, les tests effectués amènent l'auteur à croire que pour donner un comportement intelligent à un agent de négociation, pour peu que les règles soient suffisamment expressives comme c'est le cas avec JRules, il suffit d'un nombre de règles suffisamment restreint (dans notre cas, pour chaque test, le nombre de règles est inférieur à dix) pour permettre une vision globale et cohérente du comportement de l'agent de la part de l'usager.

Pour finir, nous avons également mis en évidence qu'avec la même approche, combinée à une structure de type « blackboard », il est possible de coordonner un nombre restreint d'agents de négociations de façon simple et élégante, avec les mêmes avantages que ceux énumérés ci-dessus.

7.1.2 Faiblesses

Sur le plan des faiblesses, on notera en premier lieu la dépendance potentielle de la solution proposée vis-à-vis de l'expressivité des règles et donc vis-à-vis du moteur de règles choisi. En effet, moins les règles sont expressives, plus il faut de règles pour attribuer un comportement à l'agent. Et plus le nombre de règles est élevé, plus il est difficile de s'assurer de leur cohérence. De plus, il n'est pas garanti que les avantages susmentionnés demeurent valides avec un système à base de règles moins expressif et moins performant.

Le manque de performance est d'ailleurs un reproche qu'on pourrait également faire au moteur de règles. Là aussi, cela dépend du produit employé. Cependant, de manière générale et sans avoir mené de tests de performances, l'expérience de l'utilisation de JRules nous a montré que les règles peuvent être compilées et que des algorithmes d'optimisation éprouvés peuvent être employés pour obtenir un système pour améliorer grandement la performance.

Une autre difficulté qui se présente de manière générale pour les systèmes à base de règles est la résolution du problème de la cohérence des règles, lorsque celles-ci sont nombreuses. Dans notre cas le problème ne s'est pas vraiment posé, le nombre de règles étant relativement restreint. Toutefois, on pense qu'avec des outils graphiques permettant de faire du déboguage, une réflexion adéquate sur la conception des règles et le regroupement de celles-ci en paquets, cela ne devrait pas constituer un problème majeur, surtout si, comme on le prétendait précédemment, le nombre de règles à employer pour doter un agent de négociation d'un comportement intelligent devrait être raisonnable (dans nos tests, le nombre de règle était inférieur à 10).

Enfin, on pourrait se demander comment combiner la solution proposée avec les divers algorithmes d'optimisation semblables à ceux évoqués au chapitre 3. Une fois de plus, la solution dépend du moteur de règles, mais avec JRules, ce serait faisable en organisant ces algorithmes en procédures qui seraient appelés dans la partie action des règles. Ce dernier point nous amène aux travaux futurs envisageables à court et moyen terme.

7.2 Travaux à venir

On rappellera que sur le plan de l'implémentation nous nous sommes limités au développement d'un prototype et que parmi les prochaines étapes à venir, il faudrait évaluer les limites de notre approche comparativement à une approche algorithmique telle qu'employée dans les différents travaux rapportés au chapitre 3.

7.2.1 Évolution d'INSULA

Plusieurs axes d'amélioration et d'évolution sont possibles pour INSULA. En ce qui concerne le « blackboard », pour éviter d'y afficher des informations inutiles sous prétexte qu'on ne sait pas quelles informations vont être utilisées par les règles, on pense à une paramétrisation sous forme XML. Plus précisément, la structure du « blackboard » et donc les informations qu'il contiendrait seraient définis par un document XML, en fonction du

scénario de négociation. Le « blackboard » serait alors construit et mis à jour en fonction de ce document.

Par ailleurs, une intégration du prototype dINSULA, décrit dans ce mémoire, avec CONSENSUS est prévue prochainement. Cette intégration nécessitera entre autres une modification du module de fabrique d'agents, comme on l'a évoqué au chapitre 5.

Enfin, pour augmenter l'habileté des agents à s'adapter à un contexte particulier, on envisage éventuellement de chercher comment il serait possible avec notre approche de doter l'agent de capacités d'apprentissage. Autrement dit, le travail consisterait à évaluer comment on pourrait combiner la logique des systèmes à base de règles avec les techniques d'apprentissage.

7.2.2 Futurs tests

Il serait très intéressant d'effectuer des tests sur une autre plate-forme que GNP. Pour cela, il faudrait trouver un serveur de négociation offrant un API conçu pour permettre la connexion d'agents. La seule plate-forme publique connue ayant cette caractéristique est AuctionBot (cf. section 3.3.3). Malheureusement, AuctionBot n'est plus en service. Par ailleurs, les tests effectués avec INSULA se sont limités à un seul type de négociation : les enchères. Là aussi, il serait très pertinent d'étendre les tests à d'autres types de négociation, comme le « bargaining » (cf. chapitre 2) par exemple. Dans la même perspective, nous planifions d'effectuer des tests avec des scénarios de négociation issus d'un contexte B2B (« Business to Business »), c'est-à-dire dans un contexte de négociation inter-entreprises.

En outre, nous avons volontairement limité le champ des données sur lequel pouvaient s'appliquer les règles, en écartant notamment les descriptions des produits négociés. Ajouter cette dimension permettrait donc d'élargir le champ des règles et de raffiner les stratégies des agents, notamment sur le plan de la coordination.

Pour terminer, nous souhaitons soumettre une version stable d'INSULA à des économistes afin que ceux-ci poursuivent des expérimentations économiques et nous donnent leur appréciation de l'approche proposée.

7.3 Synthèse et Conclusion

Dans le cadre du projet CONSENSUS, nous avons voulu tester l'applicabilité d'une approche déclarative pour encoder le comportement d'agents oeuvrant dans le contexte de négociations électroniques. En effet, les systèmes à base de règles sont de nouveau en vogue dans les applications logicielles d'aujourd'hui, dans le commerce électronique entre autres, mais n'ont pas vraiment été employés dans le domaine auquel on s'est intéressé ici. De plus, on a constaté que dans la majorité des travaux et applications en rapport avec les agents de négociations, l'emphase a été mise sur le développement d'algorithmes optimaux, fixes et peu flexibles, conçus en réponse à des problèmes particuliers. Or les négociations offrent une multitude de scénarios et de situations et sont sujettes à des changement en cours de réalisation. Aussi, on a pensé qu'en isolant le comportement de l'agent de ses fonctionnalités, en définissant celui-ci par un ensemble de règles modifiables à souhait, on serait à même d'avoir un agent adaptable à un ensemble de situations de négociation très diversifié. Ceci requiert cependant que les fonctionnalités de l'agent soient définies de façon générique.

Pour réaliser cette approche, nous avons conçu une architecture conceptuelle d'agent respectant ce principe de généricité, du moins permettant d'adapter l'agent à une multitude de négociations. Par ailleurs, toujours dans cette optique de flexibilité et pour répondre aux besoins de CONSENSUS en matière de négociation combinée, nous avons intégré un schéma de coordination des agents par « blackboard » reposant également sur une approche déclarative par règles. Un prototype d'infrastructure respectant cette architecture a été développé sous le nom d'INSULA. Celui-ci a permis d'effectuer des tests avec plusieurs scénarios de négociation. Ces tests ont notamment mis en valeur l'ambivalence acquise par les agents, de même que le dynamisme de leur comportement .

Bibliographie

[Ago01] Agorics Inc. *Going going gone! a survey of auctions*, 2001. Available at http://www.agorics.com/.

[AHDJ01] Patricia Anthony, Wendy Hall, Viet Dung Dang, and Nicholas R. Jennings. *Autonomous agents for participating in multiple on-line auctions.* In IJCAI Workshop on E-Business and the Intelligent Web, August 2001.

[Ben01] Hussein Ben Ameur. *Enchères multi-objets pour la négociation automatique et le commerce électronique.* Master's thesis, Université Laval, Quebec, April 2001.

[BAK01] Morad Benyoucef, Hakim Alj, and Rudolf K. Keller. *An infrastructure for rule-driven negotiating software agents.* In Proceedings of the Twelfth International Workshop on Database and Expert Systems Applications (DEXA 2001), Munich - Germany, pages 737-741,September 2001. IEEE.

[BALK02] Morad Benyoucef, Hakim Alj, Kim Levy, and Rudolf K. Keller. A Rule-driven Approach for Defining the Behavior of Negotiating Software Agents. In Proceedings of the Fourth International Conference on Distributed Communities on the Web, Sydney, Australia, April 2002. Springer. LNCS. To appear.

[BAVK01] Morad Benyoucef, Hakim Alj, Mathieu Vézeau, and Rudolf K. Keller. *Combined negotiations in e-commerce: Concepts and architecture.* Electronic Commerce Research Journal, 1(3):277-299, July 2001. Special issue on Theory and Application of Electronic Market Design. Baltzer Science Publishers.

[BCG+01] Gilbert Babin, Teodor G. Crainic, Michel Gendreau, Rudolf K. Keller, Peter Kropf, and Jacques Robert. *Towards electronic marketplaces: A progress report.* In Fourth International Conference on Electronic Commerce Research (ICECR-4), pages 637-646, Dallas, TX, USA, November 2001.

[BKL+00] Morad Benyoucef, Rudolf K. Keller, Sophie Lamouroux, Jacques Robert, and Vincent Trussart. *Towards a generic e-negotiation platform.* In Sixth

International Conference on Re-Technologies for Information Systems, pages 95-109, Zurich, Switzerland, February 2000. Austrian Computer Society.

[BS97] Carrie Beam and Arie Segev. *Automated negotiations: a survey of the state of the art*. Wirtschaftsinformatik, 39(3):263-268, 1997.

[CIR02] Centre Inter-universitaire de recherche en Analyse des Organisations (CIRANO) website. http://www.cirano.qc.ca, 2002.

[CM96] Anthony Chavez and Pattie Maes. *Kasbah: An agent marketplace for buying and selling goods*. In Proceedings of the First International Conference on the Practical Appication of Intelligent Agents and Multi-Agent Technology, pages 75-90, London, UK, April 1996.

[DL89] E. H. Durfee and V. R. Lesser, *Negotiating task decomposition and allocation using partial global planning*, in: L. Gasser and M.N. Huhns, eds., Distributed Artificial Intelligence Vol. II, pages 229-243, Morgan Kaufmann, San Mateo, CA, 1989.

[eBay02] eBay Auctions. http://www.ebay.com, 2002.

[Egg02] EggHead website. http://www.eggHead.com, 2002.

[EJB02] Enterprise Java Beans. http://java.sun.com/products/ejb/docs10.html, 2002.

[Fcc02] Federal Communication Commission. http://www.fcc.goc, 2002.

[FG96] Stan Franklin and Art Graesser. *Is it an agent, or just a program?: A taxonomy for autonomous agents*. In Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, published as Intelligent Agents III, Springer-Verlag, pages 21-35, 1997.

[FSJ98] P. Faratin, C. Sierra, and N. R. Jennings. *Negotiation decision functions for autonomous agents*. Int. Journal of Robotics and Autonomous Systems, 24(3-4):159-182, 1998.

[GGRG98] Eduard Gimenez, Lluis Godo, Juan A. Rodrguez-Aguilar, and Pere Garcia. *Designing bidding strategies for trading agents in electronic auctions*. In Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS-98), pages 136-143, 1998.

[GK99] Amy R. Greenwald and Jeffrey O. Kephart. *Shopbots and pricebots*. In Agent Mediated Electronic Commerce (IJCAI Workshop), pages 1-23, 1999.

[GLC99] Benjamin N. Grosof, Yannis Labrou, and Hoi Y. Chan. *A declarative approach to business rules in contracts: Courteous logic programs in XML*. In

1st Conference on Electronic Commerce, Denver, Colorado, pages 68-77, November 1999.

[GMM98] R. Guttman, A. Moukas, and P. Maes. *Agent-mediated electronic commerce: A survey*. Knowledge Engineering Review, 13(2):147-159, June 1998.

[Gro00] Agent Platform Special Interest Group. *Agent technology, green paper.* Technical Report version 1.0, Object Management Group (OMG), 250 First Ave, suite 201, Needham, MA 02494, September 2000.

[GTLT00] Robert Gérin-Lajoie, Vincent Trussart, Sophie Lamouroux, and Isabelle Therrien. *GNP V1.0 Architecture Document.* Technical report v0.2 D, CIRANO, July 2002.

[HRW00] Junling Hu, Daniel Reeves, and Hock-Shan Wong. *Personalized bidding agents for online auctions.* In Proceedings of The Fifth International Conference on The Practical Application of Intelligent Agents and Multi-Agents, Manchester, UK, May 2000.

[ILOG02] ILOG JRules. http://www.ilog.com/products/jrules, 2002.

[J2EE02] Java 2 Enterprise Edition (J2EE). http://java.sun.com/j2ee, 2002.

[Java02] Java. http://java.sun.com/, 2002.

[Jen01] Nicholas R. Jennings. *Automated haggling: Building artificial negotiators.* In Pacific Rim International Conference on Artificial Intelligence, page 1, 2001.

[JESS02] Java Expert System Shell (JESS). http://herzberg.ca.sandia.gov/jess, 2002.

[JF89] M. T. Jelassi and A. Foroughi. *Negotiation support systems: An overview of design issues and existing software.* Decision Support Systems, (5):167-181, 1989.

[Jini02] Jini technology. http://www.sun.som/jini, 2002.

[JSW98] N. R. Jennings, K. Sycara, and M. Wooldridge. *A roadmap of agent research and development.* Journal of Autonomous Agents and Multi-Agent Systems, 1(1):7-38, 1998.

[KAP01] KAPPA-PC. http://www.intellicorp.com/, 2001.

[KF98] Manoj Kumar and Stuart I. Feldman. *Business negotiations on the internet.* Technical report, IBM Institute for Advanced Commerce, Yorktown Heights, NY, March 1998. Available at http://www.ibm.com/iac/.

[Lisp02] Lisp. http://www.lisp.org, 2002.

[LMM+01] E. Lamma, L. Maestrami, P. Mello, F. Riguzzi, and S. Storari. *Rule-based programming for building expert systems: a comparison in the microbiological data validation and surveillance domain*. Electronic Notes in Theoretical Computer Science, 59(4), 2001.

[LSM99] R. J. Leciwicki, D. M. Sanders, and J.W. Minton. *Negotiation : Readings, Exercises, and Cases*. Third edition. Columbus, OH: Richard D. Irwin and McGraw Hill, 1999.

[Luc99] David Lucking-Reiley. *Auctions on the internet: What's being auctioned, and how*. Technical report, Vanderbilt University, Nashville, August 1999.

[MB00] Colleen McClintock and Carole Ann Berlioz. *Implementing business rules in java*. Java Developers Journal, page 8, May 2000.

[MGM99] Pattie Maes, Robert H. Guttman, and Alexandros G. Moukas. *Agents that buy and sell*. Communications of the ACM, 42(3):81-91, 1999.

[Mil89] Paul Milgrom. *Auctions and bidding: A primer*. Journal of Economic Perspectives, 3(3):3-22, 1989.

[NLJ96] H. S. Nwana, L. C. Lee, and N. R. Jennings. *Coordination in software agent systems*. The British Telecom Technical Journal, 14(4):79-88, 1996.

[OMG02] The Object Management Group. http://www.omg.org/, 2002.

[PH98] K. Pfleger & B. Hayes-Roth. *An Introduction to Blackboard-Style Systems Organization*. Technical report, Knowledge Systems Laboratory, Stanford University, January 1998.

[Pre00] Chris Preist. *Algorithm design for agents which participate in multiple simultaneous auctions*. Technical report, HP Laboratories, jul 2000.

[PUF99] David C. Parkes, Lyle H. Ungar, and Dean P. Forster. *Agent Mediated Electronic Commerce, chapter Accounting for Cognitive Costs in On-line Auction Design*, pages 25-40. Springer-Verlag, 1999.

[RMN+98] Juan A. Rodriguez-Aguilar, Francisco J. Martin, Pablo Noriega, Pere Garcia, and Carles Sierra. *Towards a test-bed for trading agents in electronic auction markets*. AI Communications, 11(1):5-19, 1998.

[San00] Tuomas Sandholm. *eMediator : a next generation electronic commerce server*. In International Conference on Autonomous Agents, pages 341-348, 2000.

[SG00] Peter Stone and Amy Greenwald. *The first international trading agent competition: Autonomous bidding agents*. Electronic Commerce Research journal, 2000.

[SHH00] Stanley Y Su, Chumbo Huang, and Joachim Hammer. *A replicable web-based negotiation server for e-commerce*. In *Proceedings of the Thirty-Third Hawaii International Conference on System Sciences (HICSS-33),* IEEE Publishers. Hawaii, January 2000.

[Sho93] Y. Shoham. *Agent Oriented Programming*. Journal of Artificial Intelligence, 60(1), pages 51-92, 1993.

[SLSK00] P. Stone, M. Littman, S. Singh, and M. Kearns. *Attac-2000: An adaptive autonomous bidding agent*. Journal of Artificial Intelligence Research 15, AI Access Foundation and Morgan Kaufmann Publishers, pages 189-206, 2001.

[Str99] M. Strobel. *Effects of electronic markets on negotiation processes evaluating protocol suitability*. Technical Report 93237, IBM, Zurich Research Laboratory, Switzerland, 1999.

[Str01a] Stefan Strecker. *Electronic negotiations - a memo on taxonomy,terminology, and an introduction to literature*. Memo, Universitaet Karlsruhe, June 2001.

[Str01b] Stefan Strecker. *Engineering of negotiations - towards the structured design of electronic negotiations*. Proposal, Universitaet Karlsruhe, 2001.

[TAC02] Trading Agent Competition website. http://auction2.eecs.umich.edu/, 2002.

[TGMW97] M. Tsvetovatyy, M. Gini, B. Mobasher, and Z. Wieckowski. *Magma: An agent-based virtual market for electronic commerce*. Journal of Applied Artificial Intelligence, 11(6):501-523, September 1997.

[Vic61] William Vickrey. *Counterspeculation, auctions, and competitive sealed tenders*. Journal of Finance, 16:8-37, March 1961.

[WC00] Michael Wooldridge and Paolo Ciancarini. *Agent-oriented software engineering: The state of the art*. In AOSE, pages 1-28, 2000.

[WJ95] M. Wooldridge and N. R. Jennings. *Intelligent agents: Theory and practice*. The Knowledge Engineering Review, 10(2):115-152, 1995.

[Wol94] Elmar Wolfstetter. *Auctions: an introduction*. Journal of Economic Surveys, 10:367-420, 1994.

[WMC99] Workflow Management Coalition, Terminology and Glossary. WFMC-TC-1011, February 1999, 3.0. Available at http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf

[WWW98] Peter R. Wurman, Michael P. Wellman, and William E. Walsh. *The Michigan Internet AuctionBot: A configurable auction server for human and software agents*. In Katia P. Sycara and Michael Wooldridge, editors, Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98), pages 301-308, New York, 9-13, 1998. ACM Press.

[WZKA00] Wai Yat Wong, Dong Mei Zhang, and Mustapha Kara-Ali. *Negotiating with experience*. In KBEM-2001, Austin, TX, July 2000.

[XML02] Extensible Markup Language (XML). http://www.w3.org/TR/1998/REC-xml-19980210, 2002.

[Yahoo02] Yahoo! Auctions website. http://www.auctions.yahoo.com, 2002.

[ZS98] Dajun Zeng and Katia Sycara. *Bayesian learning in negotiation*. International Journal of Human Computer Systems, 48:125-141, 1998.