Une forme de négociation entre entités virtuelles

Marie-Hélène VERRONS
LIFL (CNRS - UMR 8022), Université des Sciences et Technologies de Lille
Cité Scientifique, Bât M3, 59655 Villeneuve d'Ascq Cedex
verrons@lifl.fr

Résumé : La modélisation de phénomènes de groupe est au cœur de la problématique multi-agents. Parmi eux se trouvent les systèmes de négociation automatique qui ont été largement étudiés dans le domaine du commerce électronique pour modéliser notamment les enchères. Dans cet article, nous présentons un modèle général de négociation pour les systèmes multi-agents, appelé *GeNCA (Generic Negotiation of Contracts API)*, s'appuyant sur une décomposition à trois niveaux : un niveau de gestion de la communication, un niveau de négociation et un niveau stratégique, seul niveau spécifique à une application fixée. Le modèle de négociation présenté se veut général et paramétrable à différents types de négociations. Il a été implémenté sous la forme d'une API Java appelée aussi *GeNCA*, que nous avons utilisée pour construire nos applications. *GeNCA* est la seule plateforme qui permette d'utiliser différents systèmes de communication et des stratégies de négociation spécifiques aux applications réalisées. Ces recherches sur la négociation s'inscrivent dans le cadre de travaux en génie logiciel pour l'intelligence artificielle et les systèmes multi-agents.
Mots-clés : négociation, systèmes multi-agents, intelligence artificielle, génie logiciel.

1 INTRODUCTION

Avec les progrès des technologies de l'information, des systèmes multi-agents et avec le développement du commerce électronique et des places de marché sur Internet, le besoin d'agents automatiques capables de négocier avec les autres pour le compte d'un utilisateur devient de plus en plus fort. De plus, l'utilité d'utiliser un agent pendant les négociations est parfaitement justifié par l'explosion du nombre de messages échangés entre les agents. Dans certains cas, il peut être exponentiel.
Depuis plusieurs années, beaucoup de systèmes de négociations ont été réalisés dans des domaines spécifiques [Hafid, 1996] comme les enchères ou les places de marché, souvent dans un objectif de commerce électronique, citons notamment Magnet [Magnet] et le projet SilkRoad [SilkRoad]. Bien sûr, la négociation peut être utilisée dans d'autres domaines comme la prise de rendez-vous, les systèmes de réservation ou même les jeux vidéo, mais il semble que ces voies n'aient pas été réellement étudiées. Lors de la réalisation de telles applications, nous constatons que de nombreuses notions utilisées sont les mêmes dans la plupart des systèmes. Par exemple, les `contrats`, les `ressources`, les `managers` et les `participants` ont un équivalent sémantique dans tous les systèmes de négociation. Notre but dans le domaine du génie logiciel est de montrer que ces notions peuvent être réifiées dans un modèle de négociation général et ouvert, et de montrer qu'il est possible de construire l'API correspondante. Ce modèle, ainsi que l'API qui l'implémente, est appelé *GeNCA* [Mathieu, 2003b]. Il est suffisamment large pour permettre de traiter les applications classiques de négociation sans effort d'adaptation, et possède suffisamment de paramètres pour s'adapter aux différentes applications de négociation. C'est un modèle flexible qui permet à un nombre important d'agents de participer aux négociations.

type d'enchères	description
anglaises	enchères ascendantes, publiques
hollandaises	enchères descendantes, publiques
offres scellées au meilleur prix	enchères en un seul tour de parole privées, le vainqueur est celui qui propose le meilleur prix
offres scellées au second meilleur prix (Vickrey)	enchères en un seul tour de parole privées, le vainqueur est celui qui propose le meilleur prix mais ne paie que le second meilleur prix.

FIG. 1 – Descriptions des différentes enchères les plus couramment utilisées.

De nombreux types de négociation existent, les plus connues du grand public étant les *enchères* (Figure 1). D'autres types de négociation sont également répandus : on peut citer le *Contract-Net Protocol* (CNP) proposé par Smith en 1980 [Smith, 1980] (qui est un fondement des travaux sur la négociation en informatique), la négociation *à prendre ou à laisser*, les négociations *multi-attributs*, *combinées*, *à plusieurs étapes* et *par argumentation*.
Bien qu'il soit toujours difficile de définir formellement ce qu'est la négociation, nous allons baser nos arguments sur la définition consensuelle suivante [Smith, 1980, Faratin, 1999, Jennings, 2000], qui peut être appliquée à de nombreux domaines comme les enchères, les systèmes de prise de rendez-vous, les jeux ou autres.

Définition 1 *La négociation s'effectue sur un* contrat *pour obtenir des* ressources *à la demande d'un* initia-

teur. *Elle réunit un ensemble de* participants *et un* initiateur *et se déroule jusqu'à ce qu'un accord satisfaisant un pourcentage de participants soit trouvé. Les participants cherchent également à obtenir la meilleure solution possible pour eux tout en donnant le minimum d'informations aux autres.*

Définition 2 *Un contrat est l'entité qui sera négociée. Il comprend* l'initiateur *de la négociation, les* ressources *impliquées, le* délai d'attente *des réponses et une* réponse par défaut *pour le cas où un participant ne répondrait pas à temps.*

Cette définition de la négociation s'appuie sur le CNP. L'*initiateur* est l'agent qui décide d'entamer un processus de négociation avec d'autres agents appelés *participants*. Il se charge de la gestion de la négociation du contrat qu'il propose. Dans le cadre de notre étude, nous considérons qu'un minimum d'informations doit être divulgué aux autres agents, car lorsque toutes les informations sont connues, on se retrouve dans un cadre de résolution de problème, et non plus dans un cadre de négociation. D'autres algorithmes sont alors mieux adaptés, comme ceux utilisés pour les CSP (problèmes de satisfaction de contraintes).
Notre proposition s'appuie sur une architecture à trois niveaux pour permettre une réelle généricité.

– **Le niveau de négociation** qui contient la gestion des structures de données et les actes de langage nécessaires aux agents pour faire évoluer leur connaissance ;
– **le niveau de gestion de la communication** qui permet aux agents d'envoyer des messages de façon centralisée s'ils sont sur le même ordinateur, ou de façon distribuée sinon ;
– **le niveau stratégique** qui permet aux agents de raisonner sur le problème et d'inférer leurs décisions en fonction de la connaissance obtenue des autres agents négociateurs.

L'intérêt d'une telle décomposition est que chaque niveau peut être changé indépendamment des autres à la manière de briques logicielles. Il est par exemple possible d'utiliser *GeNCA* en round-robin (les agents forment une ronde et chacun parle l'un après l'autre) avec des communications synchrones avec tous les agents sur le même ordinateur pour réaliser un jeu vidéo dans lequel les individus virtuels négocieront tour à tour, ou de l'utiliser de façon distribuée avec des communications asynchrones pour des places de marché électroniques avec les agents sur des machines différentes, ou encore au sein d'une application existante nécessitant l'ajout d'une fonctionalité de négociation au sein des agents. D'autres applications utilisant des communications asynchrones en local ou synchrones en distribué sont également envisageables.
Dans cet article, nous détaillons tout d'abord le protocole de négociation utilisé dans notre modèle. Puis nous présentons en détail les trois niveaux de notre modèle. Nous indiquons ensuite comment utiliser le paquetage pour réaliser une application avec *GeNCA* et nous terminons en comparant notre modèle à d'autres, applicatifs ou formels, afin d'en montrer les avantages et inconvénients.

2 LE PROTOCOLE DE NÉGOCIATION ET SES PROPRIÉTÉS

Nous présentons ici le protocole de négociation utilisé dans notre modèle et les paramètres permettant de le spécialiser. L'objectif du protocole est de définir les messages que les agents pourront s'envoyer avec la dynamique opérationnelle associée. Le protocole de négociation que nous proposons est caractérisé par une suite de messages échangés entre un initiateur et des participants comme dans le cadre du CNP.

2.1 Primitives de négociation

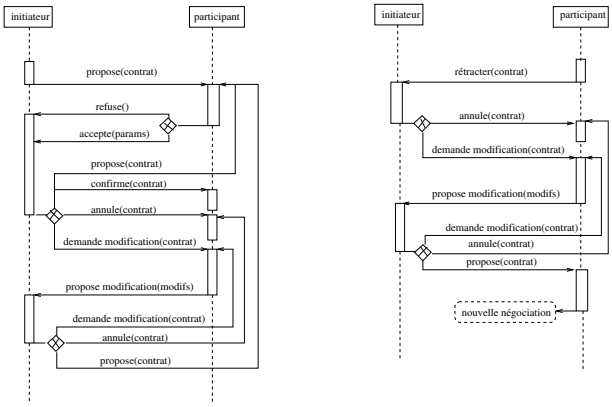

FIG. 2 – Graphes d'interaction entre un initiateur et un participant (formalisme AUML 1.0) : à gauche, le séquencement des messages pour la négociation d'un contrat entre un initiateur et des participants. Par souci de clarté, un seul participant est représenté ici. A droite, le séquencement des messages pour la renégociation d'un contrat pour lequel un participant s'est rétracté.

Pour mener à bien un processus de négociation entre agents, il est nécessaire de définir des primitives spécifiques à l'initiateur et des primitives spécifiques aux participants. Notre objectif ici n'est pas de faire communiquer un de nos agents avec n'importe quel autre agent issu d'une plateforme différente (ce qui nécessiterait une plateforme "FIPA compliant" ou plus simplement des agents communiquant via ACL), mais de faciliter la réalisation d'une application avec nos agents. Le séquencement de ces primitives est représenté à la Figure 2. Examinons plus en détail ces primitives.

2.1.1 Primitives de l'initiateur

L'initiateur possède quatre primitives de communication :

– *propose (contrat)* : c'est la première primitive que l'initiateur envoie aux participants pour leur proposer un contrat. Elle est également utilisée au cours de la négociation lorsqu'une nouvelle proposition est formulée.
– *demande modification (contrat)* : ce message indique aux participants que le contrat ne peut être conclu sous sa forme actuelle et qu'il faut le modifier. L'initiateur demande aux participants de lui indiquer une ou plu-

sieurs modifications possibles du contrat afin d'en proposer un nouveau convenant mieux à l'ensemble des participants. Une modification du contrat consiste en une nouvelle instanciation des paramètres du contrat, par exemple pour la prise de rendez-vous ce serait un autre horaire. La demande de modification peut également être un moyen de raffiner le contrat.
– *confirme (contrat)* : ce message indique aux participants que le contrat est confirmé. La négociation a été un succès.
– *annule (contrat)* : ce message indique aux participants que le contrat est annulé. La négociation a échoué.

2.1.2 Primitives du participant

Les messages envoyés par les participants sont uniquement destinés à l'initiateur. C'est un choix que nous avons fait afin que les autres participants n'aient pas connaissance de ces messages. De plus, les participants ne connaissent pas la liste des participants conviés à la négociation, ils ne peuvent donc pas former de coalition au cours de la négociation.
Le participant possède trois primitives de communication :
– *accepte (paramètres)* : ce message répond à la proposition de contrat faite par l'initiateur. Le participant indique par ce message à l'initiateur qu'il accepte le contrat tel qu'il est. Il peut y avoir des paramètres dans le cas où le contrat est partiellement instancié. C'est le cas dans les enchères Vickrey où les participants doivent proposer un prix pour l'article mis en vente.
– *refuse* : ce message répond à la proposition de contrat faite par l'initiateur. Le participant indique à l'initiateur qu'il refuse le contrat.
– *propose modification (liste modifications)* : ce message répond à une demande de modification de la part de l'initiateur. Le participant envoie à l'initiateur une liste de modifications possibles du contrat. Le nombre de modifications contenues dans la liste est un paramètre de la négociation. Cette liste peut être vide s'il n'existe pas de modifications possibles.

Une primitive de communication est commune aux initiateurs et aux participants :
– *rétractation (contrat)* : le contrat avait été confirmé mais le participant ou l'initiateur ne peut (ou ne veut) plus l'honorer. Il décide donc de se rétracter.

2.2 Paramètrage du protocole

Nous avons mentionné différents types de négociation et proposé un protocole de négociation général au sens où il est facilement adaptable. Nous décrivons ici différents paramètres permettant, à partir du noyau de négociation, de formaliser différents types de négociation. Nous avons choisi de configurer ces paramètres dans un fichier au format XML pour l'ouverture du système à l'extérieur. Nous avons donc conçu un fichier DTD (Figure 3) afin de pouvoir valider les fichiers de configuration. Il y a un fichier pour configurer l'application et chaque agent peut redéfinir les valeurs des paramètres dans un fichier qui lui est propre.

```
<!ELEMENT protocol (answer-delay,default-answer,
minAgreements,nbRounds,nb-modifications-by-round,
retraction-allowed,nbRenegotiations)>
<!ELEMENT answer-delay (#PCDATA)>
<!ELEMENT default-answer EMPTY>
<!ATTLIST default-answer value (accept | refuse)
 "refuse">
<!ELEMENT minAgreements (#PCDATA)>
<!ELEMENT nbRounds (#PCDATA)>
<!ELEMENT nb-modifications-by-round (#PCDATA)>
<!ELEMENT retraction-allowed EMPTY>
<!ATTLIST retraction-allowed value (true | false)
 "true">
<!ELEMENT nbRenegotiations (#PCDATA)>
```

FIG. 3 – Fichier DTD pour la configuration du protocole de négociation.

Lors de négociations distribuées comme c'est le cas lorsqu'il s'agit d'agents qui travaillent pour le compte d'un utilisateur, il se peut qu'un participant ne réponde pas à la proposition de l'initiateur, soit parce qu'il est absent, soit parce qu'une panne est survenue. Il faut alors que la négociation ne soit pas bloquée. Afin de permettre à la négociation de continuer, un mécanisme de temps d'attente des réponses est mis en place, et lorsque ce temps est écoulé, l'initiateur considère une réponse par défaut pour le participant. Si un participant envoie une réponse avant la fin du délai mais que cette réponse n'arrive pas à l'initiateur avant le délai, l'initiateur considère la réponse par défaut car le délai en question est un délai de réception des réponses et non un délai d'émission des réponses. Si c'était un délai d'émission, le problème de l'attente infinie des réponses des participants ne serait pas réglé alors que ce problème est la raison d'être du délai que nous avons mis en place. Si l'initiateur désire recevoir les réponses à ses propositions sous 10 minutes et considère un refus par défaut, les paramètres seront les suivants : `<answer-delay>10</answer-delay>` et `<default-answer value=''refuse''/>`.
Pour que l'initiateur puisse décider de confirmer ou d'annuler le contrat suite aux réponses fournies par les participants, un paramètre fixant le nombre minimal d'accords nécessaires pour confirmer le contrat est mis en place. Ce nombre est soit un pourcentage, soit un nombre fixe. Par exemple, pour une enchère, il suffit qu'un seul participant accepte le contrat, tandis que pour d'autres applications, tout le monde doit accepter la proposition : `<minAgreements>100%</minAgreements>`.
Afin de ne pas avoir une phase de conversation infinie, nous avons défini le nombre de tours de négociation, c'est-à-dire le nombre de fois où les participants pourront proposer une modification du contrat, faire une contre-proposition. Ce nombre de tours sera fixé à zéro si la négociation est du type "à prendre ou à laisser", ou si ce sont des enchères à offres scellées au meilleur ou au second meilleur prix : `<nbRounds>0</nbRounds>`. A chaque tour de parole, les participants peuvent proposer un certain nombre de modifications à apporter au contrat. Ce nombre est fixé

par le paramètre `<nb-modifications-by-round>1 </nb-modifications-by-round>`.

Nous avons évoqué la nécessité de pouvoir se rétracter d'un contrat pris dans certains types de négociation. Pour cela, un paramètre booléen indique si la rétractation est possible ou non, et si elle l'est, un autre paramètre fixe le nombre maximal de fois où il est possible de renégocier le contrat : `<retraction-possible value=''true''/>` et `<nbRenegotiations>5</nbRenegotiations>`.

Tous ces paramètres nous permettent donc de décrire une négociation spécifique à partir du noyau commun. Enlever des paramètres rendrait le système moins puissant, moins général. Prenons l'exemple du nombre de tours dans le processus de négociation, si nous l'enlevions et ne faisions donc plus que de la négociation en une seule proposition, nous ne pourrions plus implémenter les enchères anglaises ou hollandaises. Notre proposition consiste donc à offrir un système fournissant ce noyau et prenant en compte ces paramètres afin d'instancier différentes négociations.

2.3 Propriétés de notre protocole

2.3.1 Renégociation automatique

Souvent, lors de négociations, certains contrats ne peuvent plus être honorés et doivent être renégociés. C'est par exemple le cas lors de la prise de rendez-vous. Un participant peut être amené à annuler un rendez-vous à cause d'un imprévu mais voudrait en obtenir un autre. Pour cette raison, nous proposons de renégocier automatiquement les contrats qui doivent l'être. Lorsque la renégociation est autorisée, l'initiateur d'un contrat peut automatiquement le renégocier pourvu que le nombre maximum de renégociations ne soit pas atteint pour ce contrat. Dès qu'un initiateur reçoit une rétractation pour un contrat, il peut choisir de l'annuler pour tous les participants et éventuellement d'entrer dans une nouvelle phase conversationnelle afin de trouver un nouvel accord. L'initiateur peut également ne rien faire si le nombre d'accords reste supérieur au nombre minimal d'accords nécessaires pour le succès de la négociation.

2.3.2 Cardinalité de la négociation

La cardinalité de la négociation est une notion importante pour les SMA. Il s'agit de savoir combien d'agents négocient entre eux. Différents types de cardinalité de négociation existent [Guttman, 1998], depuis la négociation de 1 vers 1 jusqu'à n vers m. Kasbah [Chavez, 1996] est un exemple de négociation de 1 vers 1 : un acheteur négocie un article avec un vendeur à la fois. Cette forme de négociation n'est utile que lorsque deux personnes seulement sont impliquées dans la négociation. Mais lorsque la négociation implique plusieurs participants avec un initiateur, il s'agit de négociation de 1 vers n. Notre protocole permet à plusieurs initiateurs de proposer un contrat à un ensemble de participants simultanément, il s'agit donc de négociation de n vers m agents, ou plus exactement n négociations de 1 vers m simultanées, contrairement à Kasbah ou à Zeus.

2.3.3 Deadlocks

Un participant à une négociation peut être l'initiateur d'une autre négociation simultanée alors que les ressources sont partagées sans qu'il ne se produise un deadlock. En effet, les paramètres `<answer-delay>` et `<default-answer value="refuse"/>` du fichier de configuration XML permettent de limiter l'attente des réponses des participants et donc de débloquer une situation où un agent ne répondrait pas par exemple pour cause de crash. Il est donc possible de paramétrer *GeNCA* pour qu'aucun blocage ne se produise.

2.4 Évaluation de notre protocole

Notre protocole se voulant général, nous décrivons ici quelques types d'applications réalisables avec ce protocole. Cette liste n'est bien sûr pas exhaustive mais reprend les types de négociation évoqués dans l'introduction.

Comme nous l'avons mentionné auparavant, notre protocole s'inspire du Contract Net tout comme le FIPA-Contract-Net proposé par la FIPA [FIPA]. La différence principale entre notre protocole et celui de la FIPA est que nous ajoutons une phase optionnelle de conversation. Ainsi, il est possible pour les contractants de formuler des contre-propositions et ainsi converger vers une solution plus rapidement qu'avec le FIPA-Contract-Net où seul le manager formule des propositions et ne peut donc pas s'appuyer sur les préférences des contractants pour aboutir à une solution.

Notre protocole décrit les messages pouvant être échangés entre les agents et plus spécialement l'ordre de ces messages et les tours de parole des agents, mais n'impose pas le contenu des messages (il n'impose pas de prix, par exemple), il peut par conséquent être utilisé dans des domaines divers, ce qui n'est pas le cas de plusieurs autres protocoles dont celui utilisé dans Zeus [Nwana] qui est dédié aux places de marché.

Ce protocole peut, par exemple, convenir pour une négociation du type *à prendre ou à laisser* si la phase de conversation n'est pas utilisée, c'est-à-dire si le paramètre `nbRounds` est fixé à 0. Ce protocole permet également d'implémenter des enchères (Figure 1), que ce soient des enchères à offres scellées ou des enchères hollandaises. Dans le cas des enchères, les ressources qui seront négociées sont les articles mis en vente. L'argent que possèdent les agents ne constitue pas, pour nous, une ressource à négocier (il ne fait pas partie intégrante du contrat proposé). Le paiement de l'enchère ne fait pas partie de notre système de négociation, il doit être effectué par un service de paiement externe à notre modèle. Pour les enchères hollandaises, l'initiateur propose un article à un prix très élevé, et si personne n'accepte la proposition, l'initiateur propose à nouveau l'article en baissant le prix sans demander de modification aux participants. Le processus se termine lorsqu'un participant accepte la proposition, ou lorsque le prix atteint le prix minimum voulu par l'agent, ou encore lorsque le nombre de tours défini par l'initiateur est atteint.

Il est également possible d'utiliser notre protocole pour la négociation *multi-attributs*. Ce type de négociation

consiste à négocier un contrat contenant plusieurs attributs à négocier. Par exemple, pour l'achat d'un voiture, on peut négocier le type, la motorisation, la couleur, la présence d'airbags, etc.
Notre protocole n'est néanmoins pour l'instant pas adapté aux négociations qui doivent être traitées en plusieurs étapes : par exemple pour négocier l'achat d'une voiture, on peut négocier le modèle puis la couleur puis le prix, etc. Notre protocole ne permet pas de séquencer un contrat afin de le négocier point par point. Il permet cependant de traiter chaque étape du contrat. Nous tentons donc d'ajouter un mécanisme permettant à l'initiateur de négocier chaque étape du contrat l'une après l'autre, c'est-à-dire lui permettre de créer un contrat par étape, et le succès de la négociation d'une étape entraînerait le début de la négociation de l'étape suivante.
Les négociations combinées ne peuvent pas non plus être implémentées avec ce protocole car elles nécessitent un lien entre plusieurs contrats. On ne peut pas créer deux contrats et dire que les deux doivent être pris ou aucun. Si l'on veut obtenir plusieurs ressources d'une même personne, il faut les mettre dans un même contrat, mais si l'on veut plusieurs ressources de personnes différentes, il faut créer un contrat par personne/ressource mais on ne peut pas spécifier que tous les contrats doivent être pris ou aucun. Pour pouvoir utiliser ce type de négociation, il faudrait ajouter un mécanisme de liaison entre les contrats, qui permettrait de savoir si les différentes négociations ont échoué ou sont en attente du succès des autres et ainsi pouvoir confirmer tous les contrats en même temps, ou terminer la négociation des contrats dès que la négociation de l'un d'eux est en échec. Cette modification fait partie de nos perspectives proches.
La négociation par argumentation n'est pas incluse dans *GeNCA*, bien que le protocole pourrait convenir. En effet, les paramètres des différentes méthodes peuvent être des arguments. Nous pensons donc qu'il suffirait d'utiliser *GeNCA* avec des agents sachant argumenter et de faire interagir le négociateur avec l'agent pour pouvoir appliquer la négociation par argumentation. Cela fait aussi partie de nos perspectives proches.

3 LES 3 NIVEAUX DE NOTRE MODÈLE

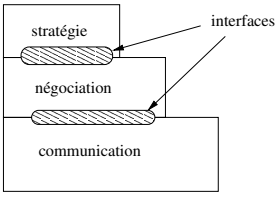

FIG. 4 – Les trois niveaux de notre modèle général.

Comme nous l'avons indiqué précédemment, notre modèle général de négociation s'appuie sur trois niveaux (gestion de la communication, négociation, stratégique), indépendants les uns des autres (Figure 4) afin de faciliter l'adaptation de *GeNCA* à une application donnée. Par construction, il est possible de modifier un niveau sans que cela ait de conséquence sur les autres. Dans cette section, nous présentons ces trois niveaux.

3.1 Le niveau de gestion de la communication

Ce niveau est responsable de la communication entre les agents. Il définit les primitives de base que tous les agents doivent implémenter pour négocier avec *GeNCA*.
Notre modèle utilise un service d'abonnement à un serveur qui gère les participants et les ressources qui seront négociées. Ce serveur permet de conserver, selon le type de ressources négociées (articles pour une vente aux enchères, tranches horaires pour une prise de rendez-vous), les différents participants et les ressources en jeu. Lorsqu'un agent s'inscrit, le serveur lui communique la liste des participants inscrits pour la même négociation que lui, ainsi que la liste des ressources qui seront négociées. Son arrivée est elle aussi communiquée aux participants déjà inscrits à la même négociation, ainsi que les ressources apportées. Les participants connaissent uniquement les noms (pseudos) des autres participants, mais pas leur identifiant physique. Ils ne peuvent donc communiquer entre eux que par l'intermédiaire du serveur qui est le seul à connaître les identifiants physiques. Les participants peuvent à tout moment se connecter ou se déconnecter du serveur. Lorsqu'ils sont déconnectés, les messages qui leur sont destinés sont stockés dans une boîte aux lettres et leur sont transmis lors de leur prochaine connexion.
Ce niveau est décrit dans une interface qu'il faut implémenter pour une application donnée. Actuellement, nous en fournissons juste quatre (cf. Section 4).

3.2 Le niveau stratégique

Les stratégies utilisées pendant la négociation sont spécifiques à l'application réalisée. En effet, négocier l'obtention de ressources partagées et négocier les prix dans une place de marché ne se fait pas de la même façon. C'est pourquoi nous avons choisi de séparer ce niveau de conception et de laisser aux utilisateurs la responsabilité de la création des stratégies adaptées à leur domaine d'application. Élaborer une bonne stratégie nécessite en effet une expertise du domaine que nous ne pouvons fournir a priori. Nous proposons néanmoins deux stratégies par défaut (l'une pour l'initiateur et l'autre pour le participant) qui sont générales et adaptables à de nombreuses applications.
Nous avons vu que notre protocole distingue deux rôles : initiateur et participant. La stratégie de négociation n'est pas la même selon le rôle de l'agent, il y a donc deux sortes de stratégies. La stratégie de l'initiateur lui permet de décider de confirmer ou d'annuler le contrat, et de synthétiser les différentes propositions de modifications des participants afin de proposer un nouveau contrat. L'autre stratégie est celle du participant qui l'utilise pour décider d'accepter ou de refuser la proposition de contrat et de trouver une modification pour le contrat lorsque l'initiateur en demande une. Chaque agent possède donc deux stratégies : l'une utilisée lorsqu'il tient le rôle d'ini-

tiateur, et l'autre utilisée lorsqu'il tient le rôle de participant.

C'est dans ce niveau que s'effectuent les choix entre plusieurs contrats. Ainsi, la spécification de contraintes est placée dans ce niveau, ce qui permet de choisir, si l'on prend l'exemple de la prise de rendez-vous, entre un rendez-vous le 10 à 10 heures à Paris et un rendez-vous le 10 à 11 heures à Lille. C'est au niveau de la stratégie qu'un agent vérifie la compatibilité des contrats, soit sur l'exemple s'il est possible d'assister à un rendez-vous à Paris à 10 heures et d'être de retour à Lille pour un rendez-vous à 11 heures.

Afin de pouvoir élaborer des stratégies, nous avons mis en place dans le modèle deux listes de priorités, une pour les ressources et une pour les personnes. Ainsi chaque agent peut définir un ordre sur les ressources et les personnes grâce auquel il pourra choisir le contrat qu'il souhaite prendre en cas de conflits. Ces listes de priorités permettent également aux initiateurs de pondérer les modifications proposées par les participants selon leur priorité et les priorités des ressources qu'ils proposent.

Nous fournissons également des méthodes permettant de consulter les négociations précédantes en fonction de la personne et/ou des ressources impliquées afin de prendre en compte les résultats des négociations passées dans la prise de décision pour une nouvelle négociation.

Ce niveau est décrit dans deux interfaces (l'une pour l'initiateur, l'autre pour le participant), qu'il faut implémenter pour une application donnée. Nous fournissons une implémentation de chaque interface.

3.3 Le niveau de négociation

Ce niveau est au coeur de notre modèle. Il contient le protocole de négociation (cf. Section 2) et les structures de données nécessaires à la gestion des négociations. Une description de ces structures de données est présentée dans [Mathieu, 2002]. Nous présentons ici les modes de gestion des négociations entrant en conflit.

Lorsqu'une personne doit négocier de nombreux contrats, il peut être préférable d'en négocier plusieurs simultanément afin de gagner du temps. Cela ne pose pas de problèmes lorsque les ressources mises en jeu dans les contrats sont différentes, c'est pourquoi nous effectuons toujours les négociations de ces contrats simultanément. En revanche, il peut s'avérer souhaitable de négocier séquentiellement les contrats portant sur des ressources communes. Notre modèle propose deux façons de gérer les négociations de contrats portant sur des ressources communes : la gestion séquentielle et la gestion en parallèle. Le mode de gestion est un paramètre de l'application de négociation. Lorsque l'utilisateur choisit la gestion en parallèle, aucune restriction n'est faite sur la disponibilité des ressources, elles peuvent déjà être en cours de négociation pour un autre contrat. En revanche, lorsque l'utilisateur choisit la gestion séquentielle, le système utilise la matrice présentée dans [Mathieu, 2002] afin de gérer les négociations. Le principe de cette matrice est le suivant : les colonnes représentent les ressources et les contrats à négocier s'empilent dans les colonnes correspondant aux ressources impliquées dans le contrat. Seuls les contrats à la base de la matrice (c'est-à-dire proposés en premier) sont en cours de négociation, les autres attendent leur tour. Lorsque la négociation d'un contrat est terminée, ce dernier est retiré de la matrice ce qui perment à un autre contrat d'entamer son processus de négociation.

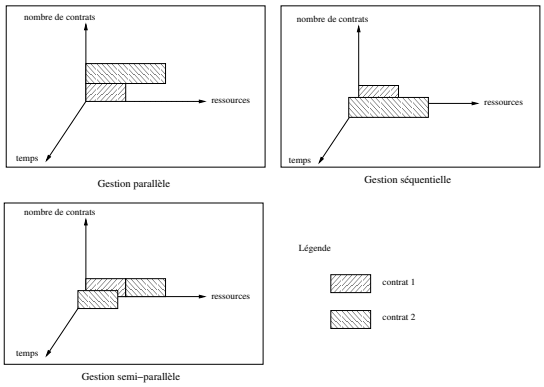

FIG. 5 – Les différents cas de gestion des négociations portant sur des ressources communes.

La Figure 5 représente les différents cas qui peuvent se présenter pour la négociation de deux contrats ayant une ressource commune $r1$. Dans le cas de la gestion parallèle, on négocie tous les contrats simultanément, sans se préoccuper du conflit. En revanche dans le cas de la gestion séquentielle, on négocie tous les contrats les uns à la suite des autres. Le cas de la gestion semi-parallèle consiste à décomposer une négociation portant sur n ressources en n négociations portant sur une seule ressource. Chacune de ces négociations sera réalisée dès que la ressource sur laquelle elle porte n'est plus en cours de négociation. Si l'on reprend l'exemple de la Figure 5, cela consiste à négocier le premier contrat qui porte sur la ressource $r1$ et à négocier en même temps la ressource $r2$ du deuxième contrat. La négociation de la ressource $r1$ du deuxième contrat s'effectue lorsque la négociation du premier contrat est terminée.

Notre modèle ne propose pas de gestion semi-parallèle des négociations portant sur des ressources communes car ce type de gestion peut aboutir à des négociations inutiles sur plusieurs ressources d'un contrat alors que ce dernier n'a aucune chance d'être accepté.

4 UTILISATION DU PAQUETAGE POUR UNE APPLICATION

Notre but est de proposer un modèle général pour négocier des contrats quels qu'ils soient. Ce modèle, que nous avons appelé *GeNCA*, a été implémenté en langage Java afin de fournir une API pour la création d'applications de négociation de contrats. Le paquetage que nous fournissons implémente intégralement la couche de négociation et le serveur de noms de la couche de gestion de la communication et donne des implémentations par défaut des interfaces des couches de communication

et de stratégie.

Les implémentations de la couche de gestion de la communication que nous fournissons permettent d'utiliser les plates-formes Magique et Madkit, ainsi que des agents centralisés parlant à tour de rôle et des agents communiquant par e-mail. Nous fournissons également l'agent système auprès duquel les agents des utilisateurs s'enregistrent et qui est responsable de l'envoi des messages. Pour ces quatre modes d'utilisation, une classe permettant de lancer l'application est fournie. Pour toute autre mode de communication, il est nécessaire d'implémenter l'interface *Communicator* et d'avoir un agent intégrant le serveur de noms implémenté dans le paquetage. Chacune des deux interfaces de la couche stratégique possède une implémentation qui sont certes basiques mais néanmoins génériques.

Le paquetage fournit également une interface graphique pour la négociation, qui permet de créer un contrat, de visualiser les différents messages envoyés et reçus par l'agent, de répondre à une proposition de contrat si l'utilisateur a choisi le mode manuel, de visualiser les contrats pris par l'agent, d'avoir une vue sur les négociations en cours sur les ressources et de se rétracter d'un contrat pris précédemment.

Dans notre paquetage, l'utilisateur humain a deux façons d'utiliser son agent. Manuellement, c'est alors un outil d'aide à la décision qui montre l'état de toutes les négociations en cours et, dans ce cas, c'est l'utilisateur humain qui répond à une proposition de contrat. Automatiquement, cette fois l'agent est caché et répond lui-même aux propositions sans intervention de l'utilisateur.

Afin de configurer l'application de négociation, et plus particulièrement les agents, deux types de fichiers XML sont utilisés : l'un est commun à tous les agents, il fournit la liste des ressources communes pour l'application, les paramètres du protocole, la gestion par défaut des négociations et les stratégies utilisées par défaut. Ce fichier est défini par le concepteur de l'application. L'autre est propre à chaque agent et optionnel. Il permet à un utilisateur de redéfinir les valeurs par défaut des paramètres du fichier commun aux agents.

Pour écrire une application avec ce paquetage, il suffit donc d'implémenter les interfaces de communication et de stratégies qui sont spécifiques à l'application (si les implémentations par défaut ne conviennent pas), et de définir le fichier commun de configuration en XML des agents et bien sûr d'écrire ses agents en leur incorporant le *négociateur* du paquetage. Lorsque l'application concerne la négociation de contrats portant uniquement sur les ressources, il n'y a rien d'autre à faire. Toute la gestion des négociations se fait automatiquement. En revanche, si le contrat nécessite d'autres paramètres (quantité, prix, etc), il faut alors étendre la classe *Contrat* et modifier l'interface graphique de création de contrat afin d'y faire apparaître les nouveaux paramètres.

Différentes applications ont été réalisées grâce à cette API, notamment un système de prise de rendez-vous et un système d'enchères. Ces applications sont décrites et peuvent être téléchargées à l'adresse suivante : *http ://www.lifl.fr/SMAC/projects/genca*.

5 COMPARAISON AVEC D'AUTRES MODÈLES

Nous ne sommes évidemment pas les seuls ayant pour objectif la négociation entre agents et la proposition d'une architecture générale pour l'accomplir.

Claudio Bartolini et ses collègues([Bartolini, 2002]) des laboratoires HP veulent créer un framework général pour la négociation automatique dédié aux mécanismes de marché. Ce framework comporte un *protocole général* de négociation qui se paramètre par des *règles de négociation*. Mais ce protocole ne permet pas de se rétracter et de renégocier automatiquement un contrat qui devrait l'être, ce que nous proposons dans *GeNCA*.

La Generic Negotiation Platform (GNP) est réalisée par Morad Benyoucef, Rudolf Keller et leurs collègues ([Benyoucef, 2000]) au département IRO de l'université de Montréal. C'est une plate-forme générique de négociation d'entreprises à entreprises orientée enchères pour les places de marché. Nous sommes donc très proches de ces travaux sur le GNP, mais notre plate-forme se veut plus générique encore, car nous ne nous sommes pas préoccupés uniquement des marchés et des ventes aux enchères, mais aussi aux négociations non commerciales telles que la prise de rendez-vous, par exemple.

Le projet SilkRoad [Ströbel, 2001] réalisé par IBM a pour but de faciliter la conception et l'implémentation de systèmes de support de négociation pour des domaines d'applications spécifiques. Ce projet se concentre sur les négociations commerciales où sont échangées des offres de vente et des offres d'achat. Dans SilkRoad, il n'y a pas, a priori, de communication entre les agents : un serveur regroupe les offres et s'occupe de les apparier. Notre approche est de fournir aux agents un cadre de négociation au sein duquel ils peuvent s'échanger des propositions de contrat. Chaque agent gère donc ses négociations, et il est possible de définir de nouvelles stratégies de négociation.

La plateforme de négociation Magnet [Collins, 1998] est un banc de tests pour la négociation multi-agents, implémenté sous la forme d'une architecture généralisée de marché et développé à l'université du Minnesotta. Le protocole de négociation pour la planification par la prise de contrats consiste en trois phases : un appel à propositions, une récolte des propositions et un choix de propositions. En revanche, notre protocole permet à l'initiateur de l'appel à propositions de faire des contre-propositions jusqu'à ce qu'un accord soit atteint. Ceci permet d'aboutir plus rapidement à un accord et de prendre en compte les souhaits des participants. Dans Magnet, la rétractation est possible moyennant une pénalité à payer, mais la renégociation ne se fait automatiquement, alors que c'est le cas dans notre modèle.

Il en existe encore bien d'autres comme Kasbah, AuctionBot, Fishmarket ou Jasa, qui permettent de créer des applications de ventes aux enchères uniquement. Ces précédents travaux, comme les nôtres, se basent sur le modèle général du CNP qui fonctionne sur le modèle d'un appel à propositions d'un agent manager vers des agents contractants. De tous ces travaux, Magnet (pour

la plateforme) et les travaux de M. Benyoucef (pour l'aspect formel) sont probablement les plus proches de ce que nous présentons. Néanmoins, aucun d'eux ne prend en compte à la fois les aspects génériques, la renégociation automatique et un mécanisme de gestion des conflits entre des négociations simultanées, que nous proposons dans *GeNCA*. De plus, *GeNCA* est la seule plateforme qui sépare les niveaux de communication, de négociation et de stratégie.

6 CONCLUSION ET PERSPECTIVES

Dans cet article, nous avons défini un modèle général pour la négociation de contrats (*GeNCA*), et son implémentation par une API Java. Notre modèle s'appuie sur trois niveaux indépendants les uns des autres : négociation, gestion de la communication et stratégie. Chaque niveau peut facilement être étendu par le développeur de façon à s'adapter à son application. Le niveau de négociation est le coeur de notre modèle, il inclut le protocole de négociation et les structures de données nécessaires à la réalisation d'une négociation. Le protocole se paramètre via un fichier XML afin de s'adapter à de nombreuses applications de négociation. Les deux autres niveaux qui sont spécifiques à une application fixée possèdent des implémentations par défaut afin de pouvoir utiliser le système immédiatement. Cette API permet la négociation de n vers m agents, la négociation simultanée de plusieurs contrats, et la gestion des deadlocks dans la conversation. Ces travaux font partie du génie logiciel et des travaux sur l'intelligence artificielle distribuée, avec les mêmes objectifs que la plateforme GNP de Montréal. La première perspective que nous avons est d'extraire le protocole du modèle afin de pouvoir en changer facilement et de proposer une bibliothèque de protocoles utilisables avec *GeNCA*. Nous allons pour cela étudier les travaux de Ishida [Ishida, 2002] sur le langage Q. De nombreuses perspectives d'implémentation de ces travaux sur différents supports software sont possibles (distribué, centralisé, WEB) et l'amélioration du niveau stratégique pour différents types de problèmes spécifiques est considéré. Nous tentons maintenant d'appliquer cette API à différents problèmes comme l'enseignement à distance, les jeux en réseau, ou les systèmes de workflow.

BIBLIOGRAPHIE

[Bartolini, 2002] Bartolini C., Preist C., Jennings N.R. : A Generic software framework for automated negotiation. Technical report HPL-2002-2, HP Laboratories Bristol (2002).

[Benyoucef, 2000] Benyoucef M., Keller R.K., Lamouroux S., Robert J., Trussart V. : Towards a Generic E-Negotiation Platform. Proc. of the Sixth Int. Conf. on Re-Technologies for Information Systems, p. 95-109, Zurich, Switzerland (2000).

[Chavez, 1996] Chavez A., Maes P. : Kasbah : An Agent Marketplace for Buying and Selling Goods. Proc. of the First Int. Conf. on the Practical Application of Intelligent Agents and Multi-Agent Technology, London, UK (1996).

[Collins, 1998] Collins J., Youngdahl B., Jamison S., Mobasher B., Gini M. : A Market Architecture for Multi-Agent Contracting. 2nd Int'l Conf. on Autonomous Agents, p. 285-292, Minneapolis (1998).

[Faratin, 1999] Faratin P., Sierra C., Jennings N.R., Buckle P. : Designing Responsive and Deliberative Automated Negotiators. Proc. AAAI workshop on negotiation settling conflicts and identifying opportunities, p. 12-18, Orlando, FL (1999).

[FIPA] http ://www.fipa.org

[Guttman, 1998] Guttman R.H., Maes P. : Cooperative vs. Competitive Multi-Agent Negotiations in Retail Electronic Commerce. Proc. of the 2nd Int. Workshop on Cooperative Information Agents, Paris, France (1998).

[Hafid, 1996] Hafid A., Bochmann G.V., Kerhervé B. : A QoS Negotiation Procedure for Distributed Multimedia Presentational Applications. IEEE International Symposium on High Performance Computing HPDC-5, Syracuse, USA (1996).

[Ishida, 2002] Ishida T. : Q : A Scenario Descripion Language for Interactive Agents. IEEE Computer, 35(11), p. 42-47 (2002).

[Jennings, 2000] Jennings N.R., Parsons S., Sierra C., Faratin P. : Automated Negotiation. Proc 5th Int. Conf. on the Practical Application of Intelligent Agents and M.A.S., PAAM-2000, p. 23-30, Manchester, UK (2000).

[Madkit] http ://www.madkit.org

[Magique] http ://www.lifl.fr/SMAC/projects/magique

[Magnet] http ://www.cs.umn.edu/Research/airvl/magnet/

[Mathieu, 2002] Mathieu P., Verrons M.H. : A generic model for contract negotiation. Proc. of the AISB'02 Convention, p. 1-8, London, UK (2002).

[Mathieu, 2003a] Mathieu P., Verrons M.H. : ANTS : an API for creating negotiation applications. Proc. of the 10th ISPE Int. Conf. on Concurrent Engineering : Research and Applications (CE2003), track on Agents and Multi-agent systems, p. 169-176, Madeira Island, Portugal (2003).

[Mathieu, 2003b] Mathieu P., Verrons M.H. : A Generic Negotiation Model for MAS using XML. Proc. of the ABA workshop Agent-based Systems for Autonomous Processing, held by the IEEE SMC Int. Conf., IEEE Press, p. 4262-4267, Washington, USA (2003).

[Nwana] Nwana H.S., Ndumu D.T., Lee L.C., Collis J.C. : ZEUS : A Toolkit for Building Distributed Multi-Agent Systems.

[Smith, 1980] Smith R.G. : The Contract Net Protocol : high-level communication and control in a distributed problem solver. IEEE Transactions on computers, C-29 (12), p. 1104-1113 (1980).

[Ströbel, 2001] Ströbel M. : Design of Roles and Protocols for Electronic Negotiations. Electronic Commerce Research, Special Issue on Market Design, 1(3), p. 335-353 (2001).

[SilkRoad] http ://www.zurich.ibm.com/csc/ebizz/oldprojects/silkroad.html