ANTS : an API for creating negotiation applications.

Philippe Mathieu & Marie-Hélène Verrons
Équipe SMAC, Laboratoire d'Informatique Fondamentale de Lille – CNRS UMR 8022
Université des Sciences et Technologies de Lille, Villeneuve d'Ascq, France
email : {*mathieu,verrons*}*@lifl.fr*

In this paper, we present a generic negotiation model for multi-agent systems, built on three levels : a communication level, a negotiation level and a strategic level, which is the only level specific to the application. XML files are used to configure the system, freeing the end-user with recompilations each time he wants to change a parameter. The aim of this paper is then to show that it is possible to describe precisely a generic model that we can use in several negotiation problems. This model has been implemented by a Java API called ANTS used to build our applications. ANTS is the only platform which enables the use of different communication systems and of negotiation strategies independent of any attribute like price. These researches on negotiation take place in software engineering works for artificial intelligence and multi-agent systems.
keywords : negotiation protocol, MAS, software engineering, XML

1 INTRODUCTION

With the progress of information technology, multi-agent systems and electronic market places, the need of automatic agents able to negotiate with the others on behalf of the user becomes stronger and stronger. Moreover, the utility of using an agent during negotiations is perfectly justified by the explosion of the number of messages exchanged between agents. In certain cases, specially with cascaded renegotiations, the number of messages can be in $O(m^n)$ if n is the depth of the cascaded process and m the number of agents involved in one negotiation.

Since several years, negotiation has been studied by many researchers ((Rosenschein and Zlotkin 1994; Sykara 1989; Kraus 2001)), and many negotiation systems have been achieved in specific domains like auctions or market places often in the aim of electronic commerce, let's cite Zeus (Nwana et al.) developed by British Telecommunications and works done at HP Laboratories (Bartolini et al. 2002a). It is not our aim, we want to offer a generic platform able to build any kind of negotiation applications like auctions, but also meeting scheduling or reservation systems. When studying such negotiation problems, we can see that many used notions are the same in many systems. For example, `contracts`, `resources`, `contractors`, `participants` have a semantic equivalent in all negotiation systems. Our aim in the software engineering field, is to show that these notions can be reified in a generic and open negotiation model and to build the corresponding API. This model should be wide enough to allow classical negotiation applications to be covered without an adaptation effort, and to possess enough parameters to adapt to different models, which is a difficult engineering problem.

Although it is difficult to define formally what is negotiation, we will base our arguments on the following consensual definition, which can be applied to many fields such as auctions, appointment taking systems, games or others.

definition : Negotiation is carried out on a *contract* to obtain common *resources* and on the request of an *initiator*. It brings together a set of *participants* and an *initiator* and runs until an agreement satisfying a percentage of participants is reached. Participants equally try to obtain the best possible solution for themselves while giving a minimum set of information to the others.

This definition is of course inspired of the Contract Net Protocol proposed by Smith (Smith 1980) in 1980, which is a fundamental of all negotiation works (Sandholm and Lesser 1995).

To conceive our model, three levels are necessary. The negotiation level which contains the negotiation protocol, the management of data structures and speech acts necessary for agents to evolve their knowledge; the communication level allowing agents to send messages in a centralised way if agents are on the same computer, or in a distributed way if they are on different computers; the strategic level allowing agents to reason on the problem and infer on the knowledge obtained from the others. In our work, each level can be changed independently of the others. We decided to separate these three levels in order to provide more facilities to adapt the negotiation system to applications as their common need is the negotiation level. As a matter of fact, each application has its own communication system and needs specific strategies of negotiation. It is for example possible to use ANTS in a round robin way with synchronous communication with all agents on the same computer to achieve a video game where virtual beings will negotiate turn to turn, and to use it in a distributed way with asynchronous communication for electronic marketplace. In our model, the negotiating agent is composed of reactive micro-agents, where each micro-agent manages a negotiation.

We have identified many criteria to describe a negotiation, among which we can find the number of rounds in a negotiation process, the minimum number of agreements needed to confirm the contract, the retraction possibility and the answer delay. Many of them have been taken into account to build ANTS.

A human user has two ways to use his agent. Manually, it is then a help-decision tool which shows the state of all the concurrent negotiations. In such case, it is the human user who agrees a query. Automatically, this time the agent is hidden and proposes or answers queries by itself.

In ANTS, the general server has an XML configuration file which allows to define the general notions like retraction possibility, number of rounds in a negotiation process, for example. Each agent can also have its own XML file to define the parameters of its owner (minimum number of agreements needed to confirm the contract, answer delay . . .). Having XML files to configure the system makes it easier for the user to define a negotiation problem.

In this paper, we first detail each level of ANTS. Then, we give ANTS' features and the different ways to use it. Finally, we compare our works to others achieved on the same subject.

2 ANTS COMMUNICATION LEVEL

This level is responsible for communication between agents, and defines the primitives that all agents must be able to understand to negotiate with ANTS.

2.1 *Presentation*

ANTS uses a mechanism of subscription to the system for locating participants and collecting resources. A names' server has been defined to retain participants with their address and resources that will be negotiated. Each agent wishing to participate in the negotiation system has to register itself to the names' server, giving its name, address and its own resources that it will negotiate. Following its subscription, it receives the list of participants already present in the system and the resources that will be negotiated. Each participant already present is notified of the arrival of a new participant and his resources.

Each agent can connect/disconnect the system as it wants, then notifying it to the names' server.

Each agent wishing to send a message to a set of participants asks the names' server to send it. As a matter of fact, participants only know the name of the other participants, not their address. So, they have to ask the names' server to send the message because it is the only one who knows participants' addresses.

Each agent needs a communicator to interact with the names' server. This communicator is responsible for subscribing to the system, connecting/disconnecting it, and for asking the names' server to send a message to a set of participants.

The names' server also needs a communicator in order to send a message to an agent, given its address.

These communicators must implement the *Communicator* interface defined in ANTS. This allows the system to run on many platforms like Magique or Madkit, but also in a round robin way with threaded agents. In order to run an application, the developer has to define a "router" agent that plays the role of the names' server.

Let's now have a look to the different implementations of the communicator that we provide with ANTS.

2.2 *Implementations achieved*

Three implementations have been achieved : one with the Magique platform (mag), one with the Madkit platform (mad), and one free of any platform with threaded agents interacting in a round robin way.

For Magique and Madkit, that are two MAS platforms, the communication primitives of these platforms were used to communicate. For the simple agents version, communication is done through procedure calls on the agents that are objects.

3 ANTS NEGOTIATION LEVEL

This level contains all the objects needed for agents to negotiate, and of course contains the negotiation protocol used in ANTS. The protocol we propose here aims to define the messages that agents can send to each others with the operational dynamics associated. This negotiation protocol (Figure 1) is characterised by successive messages exchanged between an initiator (the agent who initiates the negotiation) and participants (the agents who participate to the negotiation) as in the Contract Net Protocol framework (Smith 1980). We first describe the phases that compose our negotiation protocol, and then we present the different kinds of applications that can be achieved with this protocol. The internal objects needed to the implementation of ANTS are described in (Mathieu and Verrons 2002).

3.1 *Negotiation protocol phases*

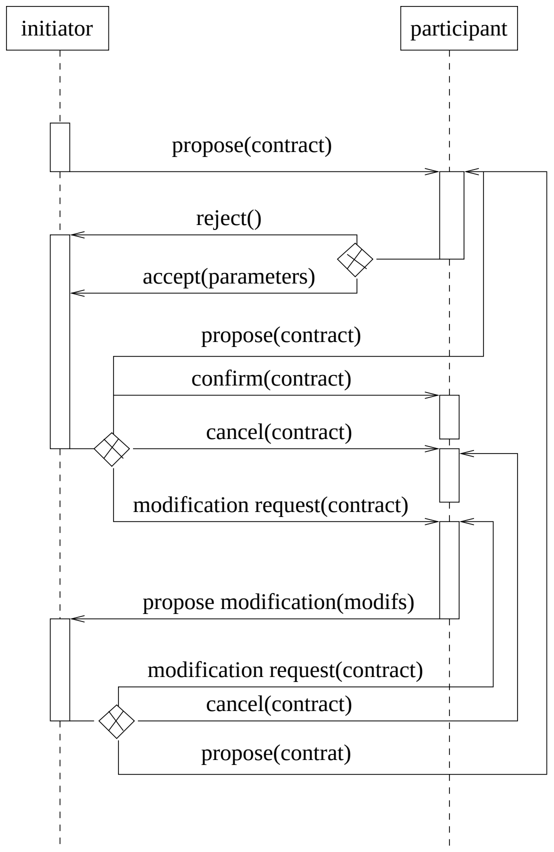


Figure 1: Negotiation protocol of ANTS

We distinguish three phases for our negotiation process : the first one is the proposition phase which initiates the negotiation process. Then, there is an optional phase named conversation phase. This phase consists of rounds of propositions and counter-propositions in order to converge to an acceptable contract for everyone. Finally, there is the final decision phase where the contract is either confirmed, either cancelled.

Proposition phase In this phase, the initiator proposes a contract to a set of participants and waits for their answer. In response to the proposition, each participant answers if he agrees or rejects it.

Conversation phase This phase is necessary if there was not enough participants who agreed the contract proposition. A conversation is then started between the initiator and participants during which modification propositions are exchanged. Following these propositions, the initiator proposes a new contract to participants, and a new proposition phase is entered.

Final decision phase This final decision phase comes to either a confirmation or a cancellation of the contract. This decision is taken by the initiator in response to participants' answers.

3.2 *Applications realisable with this protocol*

In this subsection, we present the type of applications realisable with this protocol, as it is aimed to be general.

As we mentioned before, this protocol is inspired of the Contract-Net, and it adds an optional phase of conversation. As the protocol describes messages exchanged between agents but especially the order of messages and agents' turn to talk, and not what is the content of the message (for example, always a price), it allows many different applications to use it, which is not the case of many protocols such as the one used in ZEUS which is dedicated to marketplaces.

For example, you can use it in a "take it or leave it offer" form if you don't use the conversation phase. If you want to make auctions applications, you can implement English auctions as well as Dutch auctions. For English auctions, the initiator proposes his articles and participants answer giving a price as argument of the accept message if they are interested in the article, or rejecting the proposition otherwise. If no participant has proposed a satisfying price for the initiator, a conversation phase is entered where each modification consists of a new bid. The process finishes when a satisfying price has been proposed or when nobody rebids or the maximum number of turns predefined by the initiator has been reached.

For Dutch auctions, the initiator proposes an article with a high price, and if no participant accepts the proposition, the initiator proposes again the article with a lower price without asking for a modification from participants. The process finishes when a participant accepts the contract, or when the price reaches the minimum price wished by the initiator, or when the maximum number of rounds defined by the initiator is reached.

This protocol is not adapted to negotiations that have to be processed on several levels, for example, for negotiating to buy a car, you can first negotiate the colour, and then the price . . . This protocol is not adapted to combined negotiations (Aknine 2002), where contracts need to be linked. For example, you can't create two contracts and say both of them must be taken or none. If you want several resources from the same person, you put them in a single contract, but if you want several resources from several persons, you'll need one contract per person/resource but you can't specify that all contracts must be taken or none. Despite the protocol could fit it, negotiation with argumentation (Parsons et al. 1998) is not included in ANTS. The protocol could be adapted since the parameters of acceptation or modifications could be arguments.

4 ANTS STRATEGIC LEVEL

Our model separates the strategic level in order to leave to the user the possibility to write his own negotiation strategy for a specific application.

4.1 *Presentation*

There are two kinds of strategies : the strategy for an initiator for him to decide either to confirm or to cancel the contract, and to choose among the different modification propositions. The other is the strategy used by a participant for him to decide whether to accept a proposition or not, and to choose a modification to propose in case of a modification request. Two interfaces have thus been defined, corresponding to the decisions that have to be taken either for the initiator role, or for the participant role. The success of a negotiation depends of course on strategies adapted to the problem processed. We will not discuss here about strategies, which, to be optimal, must be different according to the kind of negotiation done. This is an important field which goes out of this paper. Therefore, we propose simple but generic strategies, which work for all kinds of problems, and that the user can easily refine.

TOOLS FOR STRATEGIES

In order to give basis to develop strategies, two priority lists are defined in ANTS. Each person defines a priority list for resources and a priority list for persons. Thus, each person will be able to give a priority to a contract according to priorities of resources included in the contract, and according to the initiator's priority. For example, if I took an appointment with a colleague and my boss asks me for an appointment at the same time, I will take the appointment with my boss (who has a greater priority) and I will move the appointment with my colleague.
These lists can also be used in case that I am initiator of a contract and I requested modifications from participants, I can weight their answer according to the priority I gave them.

ANTS also provides rates of success or retraction of negotiations that have been done in the past, given a participant and a set of resources. It is thus possible to know if a participant globally accepts propositions he receives, and if he keeps his engagements.

4.2 *Applications*

Several applications have been achieved with ANTS, among which we can cite an auction application, a Dutch auction application, an appointment taking system and a time-table planning system. For each application, strategies have been implemented, fitting better to the application. These applications can be found at `http://www.lifl.fr/SMAC/projects/ants`.

5 ANTS FEATURES

ANTS is a Java API for negotiation between agents. It is aimed to provide a generic software architecture for contract-based negotiations to applications developers in order to facilitate their work. We discuss here about the different ways to use ANTS, and its major features.

5.1 *ANTS major features*

ANTS major features are its conception in three levels, its negotiation cardinality, the management of deadlocks and the XML parameterisation.

NEGOTIATION CARDINALITY

Negotiation cardinality is an important feature for MAS. Its purpose is to know how many agents negotiate together. Different kinds of negotiation cardinality exist (Guttman and Maes 1998) , from one-to-one to many-to-many. Kasbah is an example of one-to-one negotiation : one buyer negotiates an article with one seller at a time. This form of negotiation is useful when only two persons are involved in the negotiation. But when a negotiation involves many participants with an initiator, it is a one-to-many negotiation. Our protocol enables contract-based negotiation between one initiator and several participants. Our implementation of this protocol in ANTS allows several negotiations to take place simultaneously, thus finally negotiations take place between several initiators and several participants, that is to say many-to-many negotiation. The advantage provided by many-to-many negotiation is that it enables one-to-many and one-to-one negotiation.

DEADLOCKS

Deadlocks are an important problem in negotiation applications. It can cause many damages if it is not resolved. Deadlocks can appear when two agents propose a contract on the same resource one to the other, and when they choose to negotiate sequentially contracts on same resources. Both are then waiting for the other's answer and the deadlock appears. Deadlocks are avoided in ANTS thanks to our mechanism of answer delay. As a matter of fact, each initiator defines the delay that participants have to answer. If a participant doesn't answer before this delay, the initiator takes into account a default answer for him and so, negotiation is not blocked.

XML PARAMETERISATION

```
<?xml version="1.0"?>
<!DOCTYPE ants SYSTEM "ants.dtd" >
<ants>
<negotiation-name>rdv</negotiation-name>
<resources-list>
<resource>8h-9h</resource>
<resource>9h-10h</resource>
<resource>10h-11h</resource>
<resource>11h-12h</resource>
<resource>14h-15h</resource>
<resource>15h-16h</resource>
<resource>16h-17h</resource>
<resource>17h-18h</resource>
</resources-list>
<agents-list>
<agent><name>Paul</name>
       <address>localhost</address>
</agent>
<agent><name>Peter</name>
       <address>localhost</address>
</agent>
<agent><name>John</name>
       <address>localhost</address>
</agent>
</agents-list>
<default-communicator>
fr.lifl.ants.magique.MagiqueCommunicator
</default-communicator>
<default-initiator-strategy>
rdv.RdvInitiatorStrategy
</default-initiator-strategy>
<default-participant-strategy>
rdv.RdvParticipantStrategy
</default-participant-strategy>
<nbRounds>20</nbRounds>
<nbRenegotiations>3</nbRenegotiations>
<minAgreements>100%</minAgreements>
<answer-delay>10</answer-delay>
<default-answer value="refuse"/>
<simultaneity value="deferred"/>
<retraction-allowed value="true"/>
<nb-modifications-by-round>5
</nb-modifications-by-round>
</ants>
```

Figure 2: System XML file for appointment taking application

The novelty in ANTS is that the parameters that are needed to configure a negotiation application are set up in XML files, thus avoiding recompilations at each change of a parameter value, and facilitating the writing of a new application. Two kinds of files are defined : one for the system parameterisation, one for each agent (which is optional). The system file contains common characteristics for all users of the negotiation system. We define them in a DTD file called `ants.dtd` available at `http://www.lifl.fr/SMAC/projects/ants`. In this file, we find, among others, common resources, agents initially present in the system, retraction ability, plus default values for users parameters. An example of a system file for appointment taking is shown in Figure 2. Each agent can have its own file to set up its individual resources, its communicator, its strategies and negotiation parameters like default answer and answer delay.

5.2 *ANTS use modes*

ANTS can be used in different modes, which gives its generality. Among these ways to use it, we find the kind of resources negotiated, simultaneous management, automatic renegotiation and agents use modes.

RESOURCES

Resources that will be negotiated can be common to all agents or individual. If we take the example of meeting scheduling, each agent has the same agenda, and so the same time slots. Thus, resources (time slots) are common to all agents and any of them can make a proposition on the time slots they want. On the contrary, auctions applications are typically those where we find individual resources. Agents wishing to sell articles will sell only their own articles, and not the one of its neighbours. So, for this kind of applications, resources are individual, visible to all agents but only the agents that possess them can make a contract proposition. Resources are described in XML files. If they are common to all agents, they are set up in the system file, but if they are individual, they are set up in the agent file.

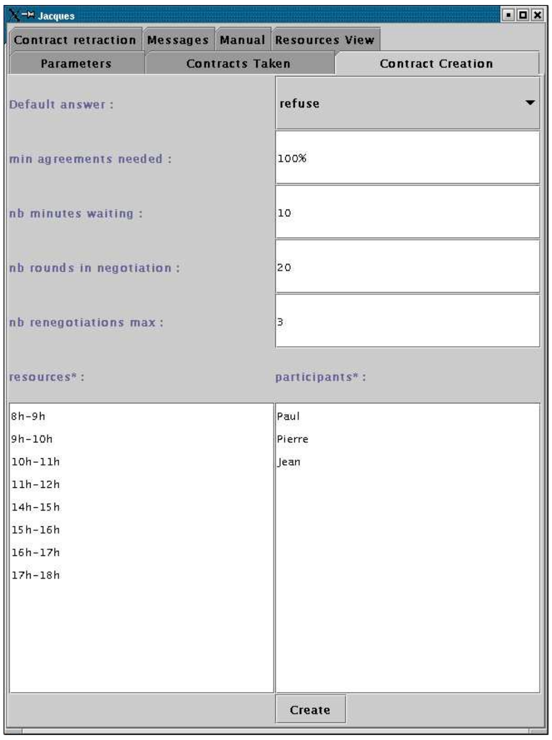
SIMULTANEOUS MANAGEMENT

The management of negotiations is an important criterion in a negotiation application. Negotiations can be processed sequentially, or in parallel, depending on the constraints of the application. Two managements are possible in ANTS, immediately or deferred simultaneous management. The user opts for the one he prefers. When he chooses to negotiate immediately all contracts, no restriction is made on the resources, they can already being negotiated for another contract. But if the user chooses to negotiate in a deferred way, the only negotiations that will take place simultaneously are the ones which involves disjoint sets of resources. The other negotiations will wait for their turn.

Automatic renegotiation

Many times, during negotiations, some contracts can't be met any longer and has to be negotiated again. It is the case when appointments are negotiated. For this purpose, we propose to renegotiate automatically contracts that have to be moved. But you can't always question a contract that has been taken. For example in auctions, when an article is sold, it is definitely sold, you can't retract yourself. That's why we define a parameter called retraction allowed, used to know whether it is possible or not to retract yourself from a contract previously taken. This is a common parameter to all agents which is defined in the system XML file. If retraction is allowed, when an agent retracts itself, the initiator of the contract can automatically renegotiate the contract, and a number of renegotiations is defined by the initiator (in the agent XML file) to know how many times a contract can be negotiated again.

Agents use modes



Default answer :	refuse
min agreements needed :	100%
nb minutes waiting :	10
nb rounds in negotiation :	20
nb renegotiations max :	3

resources* :	participants* :
8h-9h	Paul
9h-10h	Pierre
10h-11h	Jean
11h-12h	
14h-15h	
15h-16h	
16h-17h	
17h-18h	

Figure 3: A tab of the GUI of a negotiating agent in ANTS

As we mentioned before, a human user has several ways to use its agent. He can use it with a graphical interface to interact with it, in this case, the agent is a help decision tool for the user. The agent manages the negotiations and it is the user who answers contract proposition, and creates contract to negotiate. Through the interface (Figure 3) , the user views messages received and sent, contracts taken and being negotiated, and he can create a new contract, cancel a contract he has previously taken and reply to a contract proposition.

Another way to use the agent is the automatic way, in this case, the agent manages the whole negotiation and replies itself to propositions, the graphical interface is not used, and the agent runs like a background task.

In the next section, we compare our work to others in the same field.

6 COMPARISON WITH OTHER WORKS

We are obviously not the only ones who are interested in negotiation between agents and in proposing a generic architecture to accomplish it. Let's cite the works achieved at HP Laboratories by Claudio Bartolini et al. (Bartolini and Preist 2001; Bartolini et al. 2002b; Bartolini et al. 2002a) who want to create a general framework for automated negotiation dedicated to market mechanisms. In this paper, they define two roles : participant and negotiation host. A participant is an agent who wants to reach an agreement, while the negotiation host is responsible for enforcing the protocol and rules of negotiation. Rules of negotiation include posting rule, visibility rule, termination rule . . . It is the negotiation host who is responsible for making agreements. This framework proposes a general negotiation protocol parameterised with rules to implement a variety of negotiation mechanisms. It has common properties with our, like enabling one-to-one, one-to-many and many-to-many negotiations, or like parameterisation.

Another formal work we can cite is the one done by Morad Benyoussef et al. (Benyoucef et al. 2000) who want to create a Generic Negotiation Platform for marketplaces. These two works are close to our, but they are more directed to electronic commerce whereas our model aims to fit also other types of automated negotiations.

Let's now examine two platforms for negotiation : magnet and zeus. **Multi AGent NEgotiation Testbed** (Collins et al. 1998) is a testbed for multi-agent negotiation, implemented as a generalised market architecture and developed at the university of Minnesota. It provides a support for a variety of types of transaction, from simple buying and selling of goods to complex multi-agent contract negotiation. A session mechanism enables a customer to issue a call-for-bids and conduct other business. The negotiation protocol for planning by contracting consists of three phases : a call-for-bids, bidding and bid acceptance. In contrast, our protocol enables the initiator of the call-for-bids to make counter-propositions until an agreement is reached. In MAGNET, there is an explicit intermediary into the negotiation process and agents interact with each other through it, whereas all agents directly interact with each other in our negotiation process.

ZEUS (Nwana et al.) is a generic Java API achieved by British Telecom in order to easily conceive cost-based negotiation applications between autonomous agents. Zeus proposes a negotiation protocol between two agents (an initiator and a participant) and on a single resource per contract. The protocol consists of a call-for-bids, and no mechanism of counter-proposition is provided. Moreover, it is possible to negotiate simultaneously different contracts on the same resource, that we don't allow. Another difference with our protocol is that retraction is not possible with Zeus. Once a contract is taken you can't retract yourself. Moreover, Zeus provides only cost-based strategies, and so is less generic than our protocol which is not dedicated to cost-based contracts. Although it is possible to add an interaction protocol in Zeus, it is a difficult thing to do, as says S. Thompson in the mailing list of Zeus in April 2002. On the other hand, ANTS negotiation protocol is parameterisable via XML files, which simplifies modifications.

These previous works, like our, are based on the general **Contract Net Protocol** model (Smith 1980) which works on bids invitation between a Manager agent and Contractor agents. Zeus is dedicated to auctions whereas Magnet is more opened, thus Magnet is probably the closest work to what we present. Nevertheless, none of them takes into account at the same time generic aspects, automatic renegotiations and a mechanism to manage conflicts between simultaneous negotiations, that we propose in ANTS. Moreover, ANTS is the only platform which separates the communication level, the negotiation level and the strategic level.

7 CONCLUSIONS AND PERSPECTIVES

In this paper, we have presented ANTS, a Java API for contract-based negotiation, which enables many-to-many negotiations, simultaneous negotiation of several contracts, and the management of deadlocks in conversation. Three distinct levels were defined : the knowledge representation level allowing the agent viewing the advancement of his/her negotiations, the communication level which we achieved with a multi-agent platform allowing physical distribution, and the strategic level for which we propose generic strategies adaptable to any kind of problem. Each level can be easily extended by the developer as he wants to map with his application, which is a feature that only ANTS proposes. Moreover, XML files are used to set up parameters and define an application, which facilitates the end-user work, and avoid useless re-compilations. Many implementations of these works on different software supports are already possible (distributed, centralised, WEB) and strategic level enhancement for different specific problems is considered. These works are a part of software engineering and distributed artificial intelligence works. The next challenge of ANTS is to integrate the negotiation protocol into the XML configuration file in order to support other protocols that the end-user could choose. This API will now be applied to different problems like distance teaching, network games and workflow systems.

REFERENCES

http://www.lifl.fr/smac/projects/magique.

http://www.madkit.org.

Aknine, S. (2002, July). New Multi-Agent Protocols for M-N-P Negotiations in Electronic Commerce. In *National Conference on Artificial Intelligence, AAAI, Agent-Based Technologies for B2B Workshop*, Edmonton, Canada.

Bartolini, C. and C. Preist (2001). A framework for automated negotiation. Technical Report HPL-2001-90, HP Laboratories Bristol.

Bartolini, C., C. Preist, and N. R. Jennings (2002a). Architecting for reuse: A software framework for automated negotiation. In *Proc. 3rd Int Workshop on Agent-Oriented Software Engineering*, Bologna, Italy, pp. 87–98.

Bartolini, C., C. Preist, and N. R. Jennings (2002b). A generic software framework for automated negotiation. Technical Report HPL-2002-2, HP Laboratories Bristol.

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

Collins, J., M. Tsvetovatyy, B. Mobasher, and M. Gini (1998, August). MAGNET : A Multi-Agent Cntracting System for Plan Execution. In *Workshop on Artificial Intelligence and Manufacturing: State of the Art and State of Practice*, Albuquerque, NM, pp. 63–68. AAAI Press.

Guttman, R. H. and P. Maes (1998, July). Cooperative vs. Competitive Multi-Agent Negotiations in Retail Electronic Commerce. In *Proceedings of the Second International Workshop on Cooperative Information Agents (CIA'98)*, Paris, France.

Kraus, S. (2001). *Strategic Negotiation in Multiagent Environments*. MIT Press.

Mathieu, P. and M.-H. Verrons (2002). A generic model for contract negotiation. In *Proceedings of the AISB'02 Convention*, London, UK.

Nwana, H., L. L. D.T. Ndumu, and J. Collis. ZEUS : A Toolkit for Building Distributed Multi-Agent Systems.

Parsons, S., C. Sierra, and N. R. Jennings (1998). Agents that reason and negotiate by arguing. *Journal of Logic and Computation 8*(3), 261–292.

Rosenschein, J. and G. Zlotkin (1994). *Rules of encounter : designing conventions for automated negotiation among computers*. Cambridge, Mass.: MIT Press.

Sandholm, T. and V. Lesser (1995). Issues in automated negotiation and electronic commerce: Extending the contract net framework. In *First International Conference on Multiagent Systems (ICMAS-95)*, San Fransisco, pp. 328–335.

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

Sykara, K. (1989). Multiagent compromise via negociation. In L. Gasser and M. Huhns (Eds.), *Distributed Artificial Intelligence*, Volume 2, Los Altos, CA, pp. 119–137. Morgan Kaufmann Publishers.