A Generic Negotiation Model for MAS Using XML*

Philippe Mathieu
Équipe SMAC, LIFL
Université de Lille I
Villeneuve d'Ascq, France
mathieu@lifl.fr

Marie-Hélène Verrons
Équipe SMAC, LIFL
Université de Lille I
Villeneuve d'Ascq, France
verrons@lifl.fr

Abstract – *In this paper, we present a generic negotiation model for multi-agent systems (MAS), 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, multi-agent systems, artificial intelligence, XML, software engineering.

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. As a matter of fact, two problems motivate agent negotiations : the complexity of the decision making and the quantity of the messages required. 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. In this paper, we focus on the former one.

Since several years, negotiation has been studied by many researchers ([6, 4]), 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 Magnet [3] developed by the university of Minnesotta and works done at HP Laboratories [1]. Of course, negotiation can be used in other domains like meeting scheduling or reservation systems, but it seems that these ways have not been really studied. 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 cover 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 [8] in 1980, which is a fundamental of all negotiation works [7].

To conceive our model, three levels are necessary. The internal level which contains 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. 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.

The success of a negotiation depends of course on strategies adapted to the problem processed. We will not discuss

*0-7803-7952-7/03/$17.00

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.

We have identify many criteria to describe a negotiation, where we can find the number of rounds in a negotiation process, the minimum number of agreements needed to confirm the contract, the retraction possibility, 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 propose 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 ... Each agent can also have his own XML file to define the parameters of his 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 will first detail the protocol used (the phases of the protocol, the communication primitives and its properties). Then, we will describe ANTS and the different ways to use it. Finally, we compare our works to others dealing with the same subject.

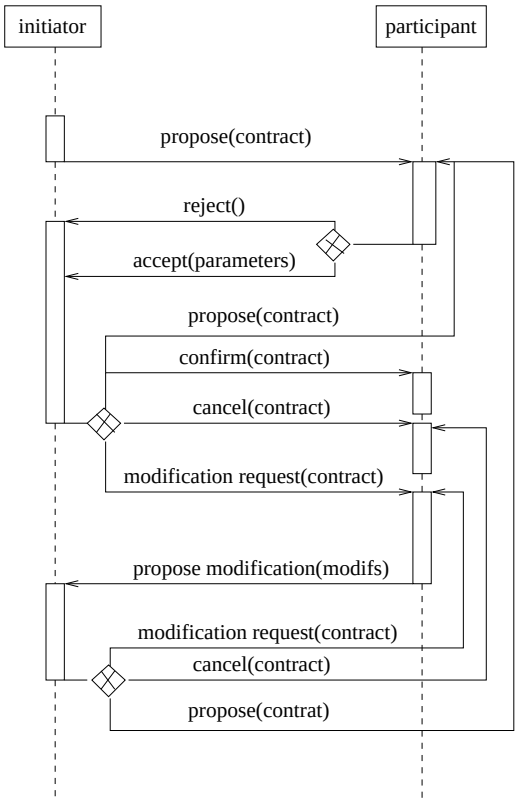

Figure 1: Negotiation protocol of ANTS

2 Proposed protocol

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 in the negotiation) as in the Contract Net Protocol framework [8]. We first describe the phases that compose our negotiation protocol, and then the communication primitives between agents used in this protocol. Finally we give characteristics of our negotiation protocol.

2.1 Protocol phases

We distinguish three phases for a negotiation process : the first one is the proposition phase which begins 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 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 thus 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.

2.2 Communication primitives

For agents to understand each other, they need communication primitives defined before beginning to negotiate. These primitives are specific to the negotiation protocol and they define the progress of the negotiation process. As Figure 1 shows, communication primitives of initiators are different to communication primitives of participants. As a matter of fact, it is the initiator who leads the negotiation process, and participants only have to answer his queries. Let's ex-

amine these communication primitives, beginning with initiators ones.

Initiator primitives The initiator begins and leads his negotiation process. He thus have specific primitives to do so. The initiator can send four communication primitives to a set of participants :

- *propose(contract)* : this is the first message sent by the initiator. He sends a contract proposition to the participants. The contract contains different resources to negotiate.
- *modification request(contract)* : this message indicates to participants that the contract can't be taken like this and it has to be modified. The initiator asks participants to send him one or several possible modifications of the contract in order to propose a new one, better fitting everyone. This can also be a way to refine the contract.
- *confirm(contract)* : this message indicates participants that the contract is confirmed. The negotiation has been a success.
- *cancel(contract)* : this message indicates participants that the contract is cancelled. The negotiation failed.

Participant primitives Messages sent by a participant are only received by the initiator. Other participants don't know about these messages. Moreover, participants don't know the set of participants in the negotiation, they thus cannot form a coalition during negotiation.
Participants have three communication primitives which are answers to the initiator queries.

- *accept(parameters)* : this message replies to a contract proposition from the initiator. By this message, the participant indicates the initiator that he accepts the contract as it is. Parameters can be used in case of a partially instantiated contract. For example, it is the case in Vickrey auctions where participants have to propose a price for the article sold.
- *reject* : this message replies to a contract proposition from the initiator. By this message, the participant indicates the initiator that he refuses the contract.
- *propose modification(modification list)* : this message replies to a modification request from the initiator. The participant sends to the initiator a list of possible modifications for the contract. The number of modifications contained in the list is a negotiation parameter. This list can be empty if there is no possible modification for the contract.

A communication primitive is common to initiators and participants :

- *retract(contract)* : the contract has been confirmed but a participant or the initiator can't meet it anymore. The agent then decides to retract himself from the initiator.

2.3 Applications achievable with this protocol

In this subsection, we present the type of applications achievable 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 no one 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, when negotiating to buy a car, you can first negotiate the colour, and then the price ... This protocol is not adapted to combined negotiations, where contracts need to be linked. For example, you can't create two contracts and say both 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 argumentationis not included in ANTS. The protocol could be adapted since the parameters of acceptation or modifications could be arguments.

In this section, we have presented the negotiation protocol used in ANTS, let's now see the different use modes of ANTS.

3 ANTS

ANTS is a Java API for negotiation between agents. It is aimed to provide a generic software architecture for contract-based negotiations to applications developpers in order to fa-

cilitate their work. The internal objects needed to the implementation of ANTS are described in [5]. We discuss here about the different ways to use ANTS, and its major features.

3.1 ANTS features

ANTS major features are its negotiation cardinality (many-to-many), the management of deadlocks, its conception in three levels and the XML parameterisation. We detail here only the two last features because of space limitations.

Conception in three levels The first feature of ANTS is its conception in three levels, in order to separate the implementation of communications between agents, the implementation of negotiations management and the implementation of negotiations strategies. 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. For example, communications between distributed agents can be done via e-mail or a Multi-Agent System (MAS) platform, while communications between centralised agents can be done in a round-robin way. It is easy to define which communicator or which strategy an agent will use as it is set up in an XML file. This separation of these three levels is a difficult software engineering problem, and from our knowledge, no other platform than ANTS separates them.

XML parameterisation 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*. Common resources, agents initially present in the system, retraction ability are found in it, plus default values for users parameters. 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. Figure 2 shows the system XML file for an appointment taking application.

3.2 ANTS use modes

ANTS can be used in different modes, which gives its genericity. Among these ways to use it, we find the kind of resources negotiated, automatic renegotiation, tools for strategies 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

```
<?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

common to all agents and any of them can make a proposition on the time slots he wants. 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.

Automatic renegotiation Many times, during negotiations, some contracts can't be met any longer and have 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.

Tools for strategies 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. 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.

Agents use modes 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 , 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.

ANTS features and use modes have been applied to several negotiation applications like appointment taking, Dutch and English auctions and timetable creation. These applications can be downloaded at *http://www.lifl.fr/SMAC/projects/ants*.

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

4 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. [1] 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. [2] who want to create a Generic Negotiation Platform for marketplaces. They "have identified a number of operations that are common to different negotiation processes", like "defining attributes and default values for the formalized concepts, setting up the end conditions for rounds, phases and the whole negotiation and defining the information to be displayed to or hidden from the players". As us, they think that there is a need to separate the process of negotiation from the other parts of the software, and that the rules governing the negotiation should not be hardcoded. Whereas both are issues for them, only the second one is still an issue for us.

A third work is the SilkRoad project [9]. This project aims to facilitate the design and implementation of negotiation support systems for specific application domains. SilkRoad facilitates multi-attribute negotiations in e-business scenarios through a specific design methodology and a generic sys-

tem architecture with reusable negotiation support components. A negotiation support system built on the basis of the SilkRoad architecture model acts as an intermediary between the actual negotiating agents (which might be software agents or humans) and thereby provides rule-driven communication and decision support. This project has common points with our, like the possibility to have human agents and the genericity of the system. These three 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 a platform for negotiation called magnet.**Multi AGent NEgotiation Testbed** [3] 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.

These previous works, like our, are based on the general **Contract Net Protocol** model [8] which works on bids invitation between a Manager agent and Contractor agents. From all these works, Magnet is probably the one which is closest 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.

5 Conclusion

In this paper, we have presented a generic protocol for contract-based negotiation and a Java API called ANTS, 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 recompilations. These works are a part of software engineering and distributed artificial intelligence works. Many implementation perspectives of these works on different software supports are possible (distributed, centralised, WEB) and strategic level enhancement for different specific problems is considered. We intend to extend our protocol in order to achieve multi-level and combine negotiations, and to define the protocol to be used to negotiate in a separate file, in order to improve the genericity of the system and to provide a library of negotiation protocols. This API will now be applied to different problems like distance teaching, network games, workfbw systems.

References

[1] Claudio Bartolini, Chris Preist, and Nicholas R. Jennings. A generic software framework for automated negotiation. Technical Report HPL-2002-2, HP Laboratories Bristol, 2002.

[2] Morad Benyoucef, Rudolf K. Keller, Sophie Lamouroux, Jacques Robert, and Vincent Trussart. Towards a Generic E-Negotiation Platform. In *Proceedings of the Sixth International Conference on Re-Technologies for Information Systems*, pages 95–109, Zurich, Switzerland, 2000.

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

[4] Sarit Kraus. *Strategic Negotiation in Multiagent Environments*. MIT Press, 2001.

[5] Philippe Mathieu and Marie-H´elène Verrons. A generic model for contract negotiation. In *Proceedings of the AISB'02 Convention*, London, UK, 3-5 April 2002.

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

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

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

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