A Platform for Electronic Commerce with Adaptive Agents

Henrique Lopes Cardoso, Eugénio Oliveira

Faculdade de Engenharia, Universidade do Porto, NIAD&R-LIACC
Rua dos Bragas
4050-123 Porto Codex, Portugal
Phone: +351-22-2041849 Fax: +351-22-2074225
hlc@ipb.pt eco@fe.up.pt

Abstract. Market research suggests that organisations, in general, have a differentiation strategy when approaching Electronic Commerce. Thus, in order to be useful, agent technology must take into account this market characteristic. When extending its application to the negotiation stage of the shopping experience, one should consider a multi-issue approach, from which both buyers and sellers can profit. We here present SMACE, a layered platform for agent-mediated Electronic Commerce, supporting multilateral and multi-issue negotiations. In this system, the negotiation infrastructure through which the software agents interact is independent from their negotiation strategies. Taking advantage of this concept, the system assists agent construction, allowing users to focus in the development of their negotiation strategies. In particular, and according to experiments here reported, we have implemented a type of agent that is capable of increasing the performance with its own experience, by adapting to the market conditions, using a specific kind of Reinforcement Learning technique.

1 Introduction

Agent technology has been applied to the Electronic Commerce domain, giving birth to what is known as agent-mediated Electronic Commerce. Many of these online implementations refer only to the first stages of the *Consumer Buying Behaviour* model [9], those of discovering what particular product a shopper should buy (*product brokering*) and finding it through the online merchants (*merchant brokering*). These automated search engines help the user on finding the best merchant offer, classifying those offers according to the price that they state. Examples of such systems include BargainFinder [2] and Jango [10]. In some extent, this is the only presently viable implementation of agent technology to online shopping, since many of the merchant sites consist of catalogues, including product and service descriptions and a so called *take-it-or-leave-it* way of doing business, which consists of presenting the demanded price of the good.

Most of the commercial online sites where it is possible to negotiate over the terms of a transaction consist of auctions, mostly based on the traditional *English auction* protocol. In this kind of interaction, it is common to have the shoppers bidding on the

price they are willing to pay for a given good, with all the remaining product and transaction characteristics being fixed. This is what makes these auctions *single-sided* mechanisms, in which shoppers compete for limited resources. On the other hand, *double-sided* auctions, like the *continuous double auction*, admit multiple buyers and sellers at the same time, both parts being able to bid. This possibility enables a way to implement a more realistic agent-based approach to today's online shopping experiences. In fact, the Internet has changed radically the "rules of the game". Shoppers have today a virtually world wide marketplace, allowing them to compare merchant's offers in a global scale. Thus, the competition among merchants has increased, and they are obliged to find out new ways of capturing shoppers' attention.

Some examples of agent technology applied to auctions are the academic projects AuctionBot [1, 19], which is an auction server where software agents can be created to participate in several types of auctions, and Fishmarket [8, 14], an electronic auction house for the *Dutch auction*.

According to market research [15], from the three well-known generic business strategies with which organisations may align Electronic Commerce – cost leadership, differentiation, and focus –, differentiation is the preferred one. This fact poses questions about the advantages of using the currently available information seeking agent-based systems applied to the Electronic Commerce domain. In order to be helpful, this technology should take into account that merchants want to differentiate themselves, and that shoppers can and want to benefit from that.

In order to build agent-based tools that respond to these requirements, merchants must provide in their sites multi-issue product and service descriptions that can be treated in an automatic way. This would enable software agents to search a set of merchants for the best product offering, while considering their differentiation intent. This kind of service, besides being an advantage to merchants, can also be seen as a powerful tool to shoppers as well, since they may compare the merchants' offers according to their preferences. The most notable work on defining a common language that makes the web accessible to software agents is the XML specification [3], developed by the World Wide Web Consortium (W3C) [18]. XML is a mark-up language that follows HTML (a presentation format language), making it possible to represent the semantic content of a document.

One important breakthrough in this area is the project Tête-à-Tête [17]. It provides a means for the shopper to compare, given its preferences, the multi-featured offers made by several merchants that sell a particular product. However, this system is not merely a search engine for shopper assistance. It uses software agents, on both sides of the bargaining process, which interact in order to encounter the shopper's desire.

In order to make it possible to automate the negotiation step of the shopping experience through the use of autonomous software agents, it is necessary to define both a common ontology to represent product and transaction related knowledge and a language for agent interaction. A protocol will then define what messages in that language may be exchanged in what conditions. The greatest effort in defining a general language for knowledge and information exchange between agents is KQML [7], developed by the ARPA Knowledge Sharing Effort.

Automated negotiation has been addressed in some relevant research. The Kasbah [5] marketplace allows the creation of predefined agents with simple time-dependent negotiation strategies, but considers only the price of a product when negotiating over

it. In [6], negotiation is described as a process where the parties move towards agreement through concession making. That negotiation is modelled as multilateral and multi-issued. Agent negotiation strategies are defined as combinations of tactics, including time-, resource- and behaviour-dependent ones.

Learning in negotiation is an even more recent topic of research. In [12], learning consists of finding the appropriate tactic parameters and combination in the overall negotiation strategy. Using genetic algorithms pursues this. In [20] the Bazaar system is described, a sequential negotiation model that is capable of learning, which is done by updating the agent's belief model in a bayesian way, that is, with probabilistic updates, according to its knowledge of the domain.

This paper presents a multi-layered platform, SMACE, which provides a means for testing different negotiation strategies. SMACE is a Multi-Agent System for Electronic Commerce that supports and assists the creation of customised software agents to be used in Electronic Commerce negotiations. The system has been used for testing automated negotiation protocols, mainly those based on the *continuous double auction*, supporting also a *multi-issue* approach in the bargaining process. On-line learning in Electronic Commerce negotiations is another subject also addressed within SMACE. The system includes two main types of agents: those that consider a strategy as a weighted combination of several tactics, and those that learn to combine these tactics in a dynamic way, by means of Reinforcement Learning techniques.

The rest of the paper is organised as follows. Section 2 addresses SMACE, a multi-agent platform for Electronic Commerce, describing its architecture and negotiation model employed. Section 3 describes the strategies implemented in the predefined agents available in the system. In section 4 we present experiments conducted in order to test the performance of agents enhanced with learning capabilities. Finally, in section 5, we finalise with some conclusions and we focus on some topics of our future work.

2 SMACE: a platform for agent-mediated Electronic Commerce

In this paper we present SMACE, a multi-agent system for Electronic Commerce, where users can create buyer and seller agents that negotiate autonomously, in order to reach agreements about product transactions.

2.1 Negotiation model

At a particular point in time, each agent has an objective that specifies its intention to buy or sell a specific product. That objective has to be achieved in a certain amount of time, specified by a deadline. Negotiation stops when this deadline is reached.

The negotiation model that we adopted is based on the one in [6]. So, we concern about multilateral negotiations over a set of issues. Multilateral refers to the ability of buyer and seller agents to manage multiple simultaneous bilateral negotiations with other seller or buyer agents. In auction terms, it relates to a *sealed-bid continuous double auction*, where both buyers and sellers submit bids (proposals) simultaneously and trading does not stop as each auction is concluded (as each deal is made).

Negotiation is realised by the exchange of proposals between agents. The negotiation can be made over a set of issues, instead of the single-issue price found in most auctions. A proposal consists of a value for each of those issues and is autonomously generated by the agent's strategy.

The proposal evaluation is based on *Multi-Attribute Utility Theory (MAUT)*. In order to do that, an agent i must take into account the preferences defined by its creator for each issue $j \in \{1, \ldots, n\}$ under negotiation:

- a range of acceptable values $[min^i_j, max^i_j]$, which must be satisfied in order to accept a proposal;
- a scoring function V^i_j: $[min^i_j, max^i_j] \rightarrow [0, 1]$, which calculates a normalised score that agent i assigns to a value for issue j inside the range of acceptable values (the higher the score, the better the agent's utility);
- a weight w^i_j, which translates the relative importance of the issue j in the overall negotiation.

Assuming normalised weights ($\sum_j w^i_j = 1$), the agent's utility function for a given proposal $x = (x_1, \ldots, x_n)$ combines the scores of the different issues in the multidimensional space defined by the issues' value ranges: $V^i(x) = \sum_j w^i_j V^i_j(x_j)$. After generating a proposal, an agent will decide on sending it upon comparing its utility to the one associated with the previously received proposal. The one with highest utility will prevail.

Following [6], the sequence of proposals and counter-proposals in a two-party negotiation is referred to as a *negotiation thread*. That thread will remain active until one of the parties accepts the received proposal or withdraws from the negotiation. Because of the multilateral nature of de negotiation model (many sealed bilateral negotiations per agent), after an acceptation the agent will wait for a deal confirmation from its opponent.

2.2 Architecture

SMACE works as an open environment where buyer and seller agents meet in the marketplace, as shown in figure 1. This entity facilitates agent meeting and matching, besides supporting the negotiation process.

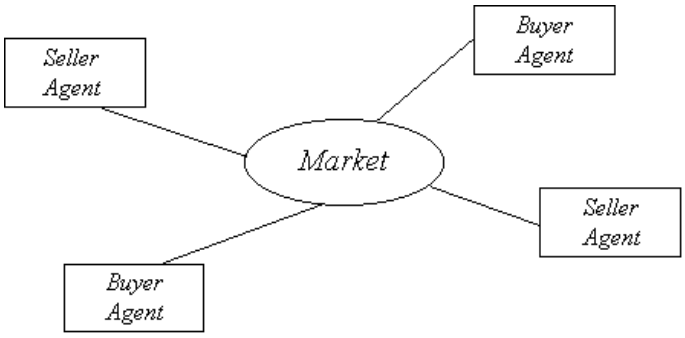

Fig. 1. The marketplace

SMACE allows users, through its web user interface, to create buyer and seller agents that negotiate under the model described in the previous section. The system was designed in layers, in order to separate the infrastructure components – that provide the communication and negotiation protocols – from those associated with the agents' negotiation strategies. This provides both openness and easy expandability. As a supporting platform, JATLite [11] was used to provide the communication infrastructure needed. Figure 2 gives an overview of how such architecture was implemented.

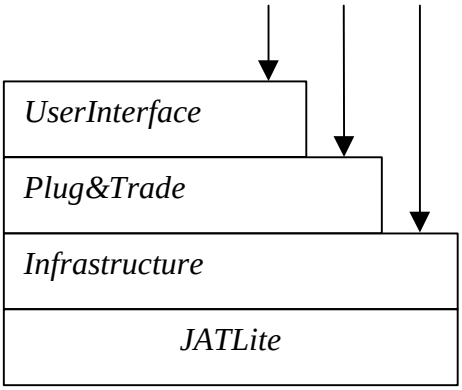

Fig. 2. SMACE architecture

SMACE consists of an API, fully implemented in Java (JDK1.1.4 API), with three layers built on top of the JATLite packages:

- *Infrastructure* – this layer contains two fundamental parts:
  - *MarketPlace*: the application that represents the marketplace, as an environment where the agents meet and trade. It includes message routing and agent brokering facilities.
  - *MarketAgent*: a template for the creation of market agents, which has already implemented the negotiation model. Building an agent with this template requires only assigning it a negotiation strategy.
- *Plug&Trade* – this layer includes ready-to-use predefined market agents, that were built using the *MarketAgent* template:
  - *MultipleTacticAgent (MTA)*: a market agent that considers a negotiation strategy as a combination of several tactics, as described in the next section.
  - *AdaptiveBehaviourAgent (ABA)*: a market agent that is able to weight several tactics in an adaptive way, using Reinforcement Learning techniques, as described in the next section.
- *UserInterface* – this layer consists of an application that provides both an HTML user interface for creating and monitoring the operation of *Plug&Trade* market agents and their persistence.

The agents communicate with each other in the *MarketPlace*, which is an enhanced JATLite *router*, facilitating the message routing between the agents and working as an information centre for the agents to announce themselves and search for contacts.

While accepting agents from anywhere to enter the marketplace and trade (provided that they use the same negotiation protocol), SMACE allows the user to launch predefined agents (both of *MTA* and *ABA* types) by adjusting their parameters. In order to do so, one can use the SMACE user interface. Through this interface, the user can monitor the agents' activities and change their settings. Taking another path, the user may create his own agent, with his own strategy, in any programming language or platform he wishes. The SMACE API Infrastructure package assists agent building in Java. This package allows the user not to worry about communication and negotiation protocol details, spending his efforts on building his own negotiation strategy, that is to say, the agent's deliberative knowledge.

3 Negotiation strategies

The goal of negotiation is maximising the utility gained in a transaction, and in order to do so the focus is on how to prepare appropriate proposals as well as counter-proposals. The negotiation strategy will define the way to do so. There are no restrictions on the negotiation strategies that can be implemented in the market agents. However, as discussed in the previous section, SMACE assists the activation of two kinds of predefined agents.

3.1 Combinations of tactics

The predefined agents already implemented in the SMACE system (*MTA* and *ABA*) use combinations of tactics as the underlying philosophy of implementing their strategy. A tactic is a function used to generate a proposal value, for a given issue, based on a given criterion. Tactics can be combined using different weights, representing the relative importance of each criterion in the overall strategy. The values that will be part of the proposal will be calculated by weighting accordingly the values proposed by each one of the tactics used. The tactics implemented were adopted from [6]:

- Time-dependent tactics: agents vary their proposals as the deadline approaches. These tactics use a function depending on time that can be parameterised.
- Resource-dependent tactics: agents vary their proposals based on the quantity of available resources. These tactics are similar to the time-dependent ones, except that the domain of the function used is the quantity of a resource other than time.
- Behaviour-dependent tactics: agents try to imitate the behaviour of their opponents in some degree. Different types of imitation can be performed, based on the opponent's proposal variations.

Other kinds of tactics can be considered or other variants of the tactics mentioned. Whereas time-dependent tactics depend on a predictable factor, it is difficult to foresee the results of applying resource- or behaviour-dependent ones, since they depend on "run-time variations" of factors.

3.2 Adaptive behaviour through Reinforcement Learning

The *MTA* predefined market agents are somewhat fixed, in the sense that they will use the same tactic combination, no matter what the results obtained are, unless the user specifies otherwise. However, in repeated negotiations, agents should be capable of taking advantage of their own experience. This consideration led us to the development of an agent that, enhanced with learning capabilities, can increase its performance as it experiences more and more negotiations – the *AdaptiveBehaviourAgent (ABA)*. Tactics provide a way of adaptation, in some degree, to different situations, considering certain criteria. But it is not clear what tactics should be used in what situations. The ultimate goal of our adaptive agent is to learn just that.

The idea is to define a strategy as the way in which the agent changes the tactic combination over time. In order to do that, we used a kind of automated learning that can take place online, from the interaction with the environment: Reinforcement Learning [16]. It is also the most appropriate learning paradigm to dynamic environments, such as the one we are addressing.

By applying this kind of learning in the adaptive agents, we aimed at enhancing those agents with the ability of winning deals in the presence of competitors and increasing the utility of those deals. We intended to check if the agents adapt to a given market environment, associated with the transaction of a given type of product.

In dynamic environments, such as an electronic market, actions are *non-deterministic*, in the sense that they do not always lead to the same results, when executed in the same state. For this reason, we implemented a specific kind of Reinforcement Learning – *Q*-learning – that estimates the value of executing each action in each state (also known as the quality *Q*). In our environment, actions are weighted combinations of tactics that will be used in the proposal generation process. The characterisation of the states is a major factor to the success of the algorithm implementation, and will determine the relevance of the results obtained. In our case, we considered two main variables: the *number of negotiating agents* and the *time available for negotiation*, that is, the time left till the agent's deadline.

Updating the *Q* values associated with each action-state pair – *Q(s,a)* – consists of rewarding those actions that leaded to good results while penalising the ones that failed to achieve the agent's goal. The general *Q*-learning update formula is the following:

$$Q(s,a) = Q(s,a) + \alpha\,[r + \gamma max_{a'} Q(s',a') - Q(s,a)] \quad \textbf{(1)}$$

where α is the learning rate, representing the impact of the update in the current value; *r* is the reward obtained by executing action *a* in state *s*; γ is the discount factor, meaning the importance of future *Q* values (in future states) to the *Q* currently being updated; $max_{a'} Q(s',a')$ is the maximum *Q* value for the actions in the next state.

For the *ABA* agents, actions leading to deals are rewarded with a function depending on the deal values' utility and on the average utility obtained so far. This allows us to distinguish, from the deals obtained, those that correspond to greater utilities. Considering the average utility takes into account, when classifying the goodness of a deal, the characteristics of the environment (the difficulties) that the agent is repeatedly facing; the same deal in harder conditions should have a greater

reward because it is closer to the best possible deal. Goal failure imposes penalisation (negative reward) to the last action used.

Action selection is another important aspect of the Reinforcement Learning paradigm. The simplest rule would be to select the action with the biggest *Q* value. Yet, this rule does not consider that there may exist non-executed actions that may perform better. Furthermore, in dynamic environments and therefore *non-deterministic*, actions do not always lead to the same results. In fact, to obtain a continued reward of great value, the agent should prefer actions that were considered good in the past, but in order to find them it must try actions that were never selected before. This dilemma leads us to the need of a compromise between *exploitation* (to take advantage of good quality actions) and *exploration* (to explore unknown actions or those with less quality). To satisfy this compromise, according to the Reinforcement Learning literature, two policies are possible, and the adaptive agents developed support them both: the ε*-greedy* approach selects uniformly, with a probability ε, a non-greedy action; the *Softmax* policy uses a degree of exploration τ (the temperature) for choosing between all possible actions, while considering their ranking.

In order to make it possible for the agent to increase the utility obtained in the deals made, it is necessary that the agent does not prefer the first action leading to a deal. Before the agent tries enough actions, it has got an incomplete knowledge of the environment, that is, it might know what action to use to likely get a deal (because unsuccessful actions are penalised), but not what the best actions are (those that result in higher utilities). To enforce the agent to try all the actions available before preferring the best ones, we implemented a Reinforcement Learning technique called *optimistic initial values*. This means that all the *Q* values associated with the actions are initialised to a value greater than the expected reward for them. This measure increases, independently of the action selection parameters chosen, the initial action exploration, since the *Q* values will then be updated to more realistic lower values.

4 Experiments

The implementation of the learning algorithm intended to enhance the agents with the capacity of gaining "know how" about the market mechanics in respect to the transaction of certain kinds of products. This sensibility refers not only to the usual pattern of appearance of new agents (the market dynamics), but also to the way buyer and seller agents in a specific environment normally relax their bids. The agent should improve its performance as it experiences more negotiation episodes.

To represent these market-specific features, which work as a reference to the adaptation process, we designed four different scenarios, over which we conducted some experiences. These scenarios did not intend to simulate real-world Electronic Commerce negotiations, but to illustrate situations where it was possible to observe the results of applying the learning skills on the adaptive agents.

4.1 Scenarios' description

The four basic scenarios are illustrated in figure 3. The negotiation was made over the single-issue price, since this option does not affect the results; the strategies implemented do not take advantage of the multi-issue support (each tactic generates a value for an issue). All agents were configured with time-dependent tactics. The *MTA* agents had single time-dependent linear tactics. The *ABA* agents had time-dependent tactics that allowed them to vary their behaviour from an anxious (early concession) to a greedy (late concession) extremes. In the third scenario, a resource-dependent tactic, depending on the number of opponents, was added to the adaptive agent, since one of the opponents was activated only after a period of negotiation time. The adaptive agent in the fourth scenario had also a resource- and a behaviour-dependent tactics. Both kinds of agents have a set of other specific parameters whose values are not described here.

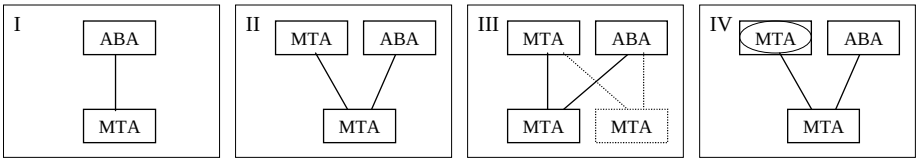

Fig. 3. Experimental scenarios

In scenario I, we intended to check if the adaptive agent was able to increase its utility, after a number of negotiation episodes in the same exact environment configuration. This would allow us to test the learning algorithm. Scenario II expanded this test to the agent's ability to win deals over its competitor, and from those deals, to increase the utility to the best possible value, which was limited by the opponent's fixed strategy. Scenario III was configured in a way that it was preferable to the adaptive agent to achieve deals with the late arriving agent. So, the *ABA* should learn to wait, instead of hurrying on making a deal before its competitor. Finally, scenario IV provided a way of testing the re-adaptation time of the adaptive agent, since its competitor was modified twice after a significant number of negotiation episodes.

The most important learning parameters of the *ABA* agents, which have an impact on their performance, are the *learning rate* (that influences the *Q* value updating), and the *degree of exploration* (which controls the action selection when using the *Softmax* approach). In highly dynamic environments, agents should use high learning rates, allowing for fast adaptation to new environment conditions. On the other hand, a high degree of exploration will force the agent to select many different actions, which may also be important in dynamic environments. However, as a consequence of that, the adaptation process slows down. The *ABA* agents were configured with a learning rate of 0.5, a middle value that allows for quick adaptations (notice that scenarios I and II are fixed, and so this parameter becomes more important in scenarios III and IV). The degree of exploration was set to a low value of 0.2, since that the initial exploration was already assured by the use of optimistic initial values (see subsection 3.2). However, exploration is still needed after the initial adaptation period (namely in

scenarios III and IV). The next subsection presents the results obtained by using such values for these parameters.

4.2 Results

In general, the results obtained were satisfactory. That is, the adaptive agent performed well but, for its current implementation, in some cases it took too long to achieve a good result. The scenarios described were run over 2000 negotiation episodes.

In all scenarios illustrated, the adaptive agent tended to achieve the predicted results. Figure 4 shows the utility evolution of the adaptive agent in each one of the scenarios. In scenario I, the agent was able to continuously increase the utility obtained in the deals, by waiting for its opponent to concede. Scenario II was more limited in terms of utility increasing, but the *ABA* could, besides winning the majority of deals over its competitor, increase the average utility of those deals very close to the highest possible in that situation. Results in scenario III allowed us to conclude that the adaptive agent learned to wait and prefer dealing with the late arriving opponent, which enabled it to achieve higher utilities. In scenario IV, we observed that despite the considerable adaptation of the adaptive agent to an initial situation, after changing the agent's competitor it readapted relatively quickly to the new environment conditions.

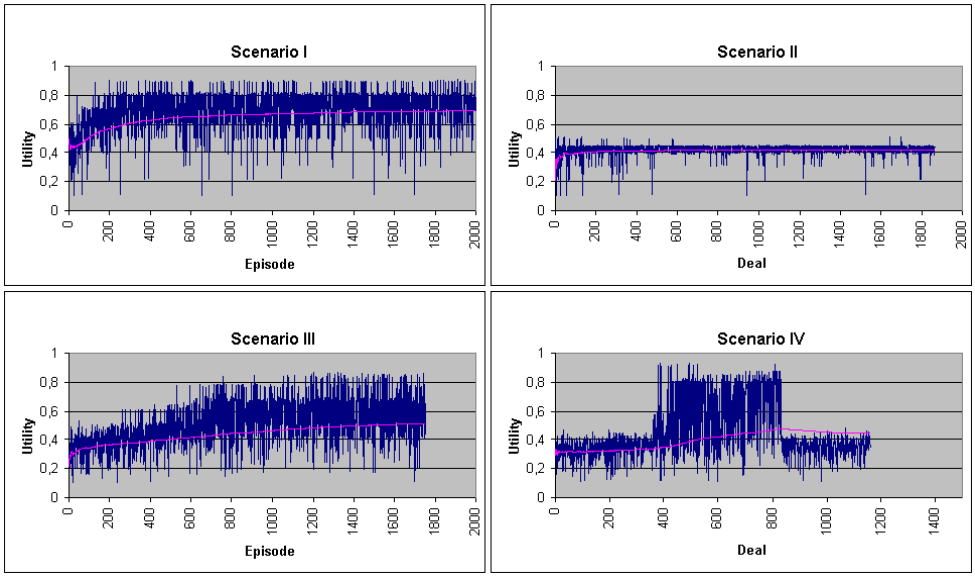

Fig. 4. Utility results

These results show us that, under some circumstances, it is possible to endow software agents with capabilities that allow them to improve their performance with their own experience. The task that now raises is to adapt this mechanism to situations

closer to real Electronic Commerce transactions, where the real negotiating parties (the agents' creators) can benefit from negotiation "know how" stored in their software agents.

5 Conclusions and Future Work

Software agents can help users to automate many tasks. In our case, we focus on automating Electronic Commerce activities, namely those of buying and selling products. There exist several applications of information seeking agents applied to this domain that help users on finding the best price for a given product. As explained above, in order to be helpful, such tools should take into account the multi-issue trend of doing online business today.

The automation of the negotiation process is more critical, since it implies the usage of negotiation strategies that will determine the wins and loses of delegating shopping tasks in autonomous software agents. According to [4], the intelligence or sophistication level that buying or selling software agents may possess is not restricted by Artificial Intelligence limitations, but by user trust considerations.

We have developed SMACE, a platform that includes an infrastructure framework over which it is possible to build agents with different negotiation strategies. The mass development of agent-mediated negotiations in Electronic Commerce will depend on the adoption of standards in this domain, related both to the ontologies used to represent semantically the objects of negotiation and to the software agent's interaction.

As negotiation strategy examples, we implemented two kinds of agents, with the assistance provided by the lower layer of the SMACE system, and made some experiments that involved interactions between these agents. Our results claim that it is possible to build negotiation strategies that can outperform others in some environments.

Directions of our future work include implementing strategies that take effective advantage of multi-issue negotiations, by correlating those issues. In respect to the agents' adaptation capabilities, we intend to refine our learning algorithm implementation. In particular, considering a new definition of a strategy (different from a combination of tactics) might help enhancing the results obtained through the learning experience. We also intend to compare our learning mechanism with other learning approaches, and to continue our research on the practical applications and effective benefits of learning processes.

In which concerns increasing the open nature of our system, we consider adopting some of the emerging standards in the Electronic Commerce domain, namely the XML specification for agent's communication content. The negotiation model that we are using can also be optimised to include support to interactions that may be beneficial, following what is described in [13] as a *critique*.

References

1 AuctionBot. URL: http://auction.eecs.umich.edu/
2 BargainFinder. URL: http://bf.cstar.ac.com/bf
3 Bosak, J. (1997), "XML, Java, and the future of the Web", Sun Microsystems.
4 Chavez, A., D. Dreilinger, R. Guttman and P. Maes (1997), "A Real-Life Experiment in Creating an Agent Marketplace", in *Proceedings of the Second International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'97)*.
5 Chavez, A, and P. Maes (1996), "Kasbah: An Agent MarketPlace for Buying and Selling Goods", in *Proceedings of The First International Conference on The Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'96)*, pp. 75-90.
6 Faratin, P., C. Sierra and N.R. Jennings (1998), "Negotiation Decision Functions for Autonomous Agents", *International Journal of Robotics and Autonomous Systems*, 24 (3-4), pp. 159-182.
7 Finin, T., Y. Labrou and J. Mayfield (1997), "KQML as an agent communication language", in *Software Agents*, J. M. Bradshaw (editor), MIT Press.
8 Fishmarket. URL: http://www.iiia.csic.es/Projects/fishmarket/
9 Guttman, R.H., A.G. Moukas and P. Maes (1998), "Agent-mediated Electronic Commerce: A Survey", *Knowledge Engineering Review*.
10 Jango. URL: http://www.jango.com/
11 JATLite. URL: http://java.stanford.edu
12 Matos, N., C. Sierra and N.R. Jennings (1998), "Determining Successful Negotiation Strategies: An Evolutionary Approach", in *Proceedings, Third International Conference on Multi-Agent Systems (ICMAS-98)*, pp. 182-189, IEEE Computer Society.
13 Parsons S., C. Sierra and N.R. Jennings (1998), "Agents that reason and negotiate by arguing", in *Journal of Logic and Computation*, 8 (3), pp. 261-292.
14 Rodríguez-Aguilar, J.A., P. Noriega, C. Sierra and J. Padget (1997), "FM96.5 A Java-based Electronic Auction House", in *Proceedings of the Second International Conference on The Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'97)*.
15 Romm, C.T. and F. Sudweeks (1998), *Doing Business Electronically*, London: Springer-Verlag.
16 Sutton, R.S. and A.G. Barto (1998), *Reinforcement Learning: An Introduction*, Cambridge: MIT Press.
17 Tête-a-Tête. URL: http://ecommerce.media.mit.edu/Tete-a-Tete/
18 World Wide Web Consortium. URL: http://www.w3.org
19 Wurman, P.R., M.P. Wellman and W.E. Walsh (1998), "The Michigan Internet AuctionBot: A Configurable Auction Server for Human and Software Agents", in *Proceedings of the Second International Conference on Autonomous Agents (Agents'98)*, K.P. Sycara and M. Wooldridge (editors), pp. 301-308, ACM Press.
20 Zeng, D. and K. Sycara (1996), "How Can an Agent Learn to Negotiate?", in *Intelligent Agents III*, J. P. Muller et al. (editors), pp. 233-244, Springer-Verlag.