Implementing rule-based mechanisms for agent-based price negotiations

Costin Bădică
Software Engineering Department
University of Craiova
Bvd.Decebal 107, Craiova, 200440, Romania
badica_costin@software.ucv.ro

Adriana Bădiţă
Software Engineering Department
University of Craiova
Bvd.Decebal 107, Craiova, 200440, Romania
badita_adriana@yahoo.com

Maria Ganzha
Department of Informatics
Elbląg University of Humanities and Economy
ul. Lotnicza 2, 82-300 Elbląg, Poland
ganzha@euh-e.edu.pl

ABSTRACT

This note describes a sample implementation of automated negotiations in an e-commerce modeling multi-agent system. A specific set of rules is used for enforcing negotiation mechanisms. Discussion of system design and implementation using JADE and JESS is provided. Finally, an experiment involving multiple English auctions performed in parallel is discussed.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

automated negotiations, software agents, auctions, rule-based systems

1. INTRODUCTION

The goal of developing and experimenting with a multi-agent e-commerce system was recently specified in ([2, 10]). One of challenges is to illustrate that it is possible to use autonomous agents to completely substitute for humans to automate basic e-commerce activities such as: product brokering, merchant brokering, negotiations, payment etc., in a complete e-commerce scenario, rather than in isolation ([1]). Currently, focus of our work is on efficiently providing agents with flexibility necessary for engaging in automated price negotiations governed by mechanisms unknown in advance ([4]). This paper describes software components used in a rule-based framework for automated negotiation and their implementation using JADE ([12]) and JESS ([13]).

Rule-based approaches have been indicated as a very promising technique for parameterizing the negotiation design space in multi-agent systems ([5, 6, 8, 15, 17, 20, 21, 11]). In addition, proposals have been put forward to use rules for describing both strategies ([8, 17]) and mechanisms ([5, 4]) of automated negotiations. In particular, a special attention has been devoted to auctions, as one of the most popular and best understood forms of automated negotiations ([20]). Here we understand negotiations as a process by which a group of agents communicates with each other to come to a mutually acceptable agreement on a price ([16]). When conceptualizing automated negotiations *negotiation protocols* (or *mechanisms*) and *negotiation strategies* have to be distinguished. The *protocol* defines "rules of encounter" between negotiation participants by specifying the requirements that enable their interaction. The *strategy* defines the behavior of participants aiming at achieving a desired outcome. This behavior must be consistent with the negotiation protocol, and usually aims at maximizing individual "gains" of each of negotiation participants.

In this paper we focus our attention on the design and implementation of a rule-based framework for enforcing specific negotiation mechanisms. For this purpose we follow the conceptual framework described in [5], where agent negotiations were conceptualized as consisting of: (i) negotiation participants and a host where the negotiations take place; (ii) a generic negotiation protocol, and (iii) a taxonomy of rules for enforcing a specific negotiation mechanism (see section 2 and [5] for more details). Here we extend work presented there by providing implementation details using JADE ([12]) and JESS ([13]) and by presenting a scenario involving multiple English auctions performed in parallel. In particular, in our implementation, the rule-based sub-agents of the negotiation host share a single JESS rule engine, rather than having separate rule engines within each sub-agent.

We proceed as follows. In section 2 we summarize the software framework for automated negotiations introduced in [5] and show how it fits into our e-commerce model. We follow with an outline of the system design and provide some details of the proposed implementation. In particular we highlight how rules are activated by the negotiation host in response to messages received from the negotiation participants. The next section presents a simple experiment performed with the sample implementation used to highlight agent interactions.

2. CONCEPTUAL ARCHITECTURE

Authors of [5] analyzed the existing approaches to formalizing negotiations (including the FIPA approach [9]) and argued that they do not provide enough structure for the development of truly portable agent-based e-commerce sys-

SAC'06 April 23-27, 2006, Dijon, France

tems. Consequently, they outlined a framework that is based on an abstract negotiation process comprising: (1) negotiation infrastructure, (2) generic negotiation protocol, and (3) taxonomy of declarative rules. The *negotiation infrastructure* defines roles involved in the negotiation process: participants and a host. Participants negotiate by exchanging proposals within a negotiation locale that is managed by the negotiation host. Depending on the type of negotiations, the host can also play the role of a participant (for example in an iterative bargaining scenario). The *generic negotiation protocol* defines, in terms of messages exchanged between the host and negotiation participants, the three main phases of negotiations: (1) admission, (2) exchange of proposals and (3) formation of an agreement. *Negotiation rules* are needed for enforcing a specific negotiation mechanism. Rules are organized into a taxonomy that contains the following categories: (a) rules for participants admission to negotiations, (b) rules for checking the validity of negotiation proposals, (c) rules for protocol enforcement, (d) rules for updating the negotiation status and informing participants, (e) rules for agreement formation and (f) rules for controlling the negotiation termination.

Based on the categories of rules, in [5] it is suggested to partition the negotiation host into a number of corresponding sub-agents: *Gatekeeper*, *Proposal Validator*, *Protocol Enforcer*, *Information Updater*, *Negotiation Terminator* and *Agreement Maker*. Each sub-agent is responsible for enforcing a specific category of rules. Sub-agents interact with each-other via a blackboard and with negotiation participants by direct messaging.

The proposed e-commerce environment [10] acts as a distributed marketplace by hosting e-shops and allowing e-buyers to visit them and purchase sought-after products ([1, 2, 10]). In the multi-agent environment, e-shops and e-buyers are represented by *shop* and *seller*, and respectively *client* and *buyer* agents. Here, let us consider a simplified version of this scenario that involves a single shop agent S and n client agents C_i, $1 \leq i \leq n$ (a scenario with multiple shops involves a simple generalization but makes notation somewhat more messy). The shop agent is selling m products $\mathcal{P} = \{1, 2, \ldots, m\}$. Note that we identify a product by its index j, $1 \leq j \leq m$. We assume that each client agent C_i, $1 \leq i \leq n$, is seeking a set $\mathcal{P}_i \subseteq \mathcal{P}$ of products (i.e. we restrict our attention to the case where all sought products are available through the shop agent S). Note that, while interesting on its own, we do not consider here the problem of dependencies between sought-after products (see [6] and papers cited there for work in that direction). Shop agent S is utilizing m seller agents S_j, $1 \leq j \leq m$ and each seller agent S_j is responsible for selling single product j. Each client agent C_i is using buyer agents B_{ik} to purchase products belonging to the set $\mathcal{P}_i$. Each buyer agent B_{ik} is responsible with negotiating and buying exactly one product $k \in \mathcal{P}_i$, $1 \leq i \leq n$. To attempt purchase buyer agents B_{ik} migrate to the shop agent S and engage in negotiations; a buyer agent B_{ik}, that was spawned by the client agent C_i, will engage in negotiation with the seller S_k for purchasing product k.

This simple scenario is sufficient for the purpose of our paper, i.e. to illustrate how multiple rule-based automated negotiations can be performed in parallel. In this setting, each *seller agent* S_j plays exactly the role of a *negotiation host* defined in [5]. Therefore, in our system, we shall have exactly m instances of the framework described in [5]. Each instance is responsible for managing a separate negotiation "locale" (where a single product is sold), while all instances are linked to the shop agent S. For each instance we shall have one separate set of rules that describes the negotiation mechanism implemented by that host (seller agent). Note that each seller may use a different negotiation mechanism (different form of an auction, or an auction characterized by different parameters).

3. DESIGN AND IMPLEMENTATION

In the design and implementation of the system we focus our attention on negotiation agents (host/seller and participant) and their associated infrastructure. Here, we show: (i) how the negotiation host agent is structured into sub-agents; (ii) how rules are executed by the negotiation host in response to various messages received from negotiation participants and how rule firing control is switched between various sub-agents of the negotiation host, and (iii) how the generic negotiation protocol was implemented using JADE agent behaviors and ACL message exchanges.

3.1 The Negotiation Host / Seller

Host and negotiation participant agents are implemented as ordinary JADE agents and thus they extend the *jade.core.Agent* class. The host agent encapsulates a number of sub-agents that are implemented as ordinary Java classes: *Gatekeeper*, *Proposal Validator*, *Protocol Enforcer*, *Information Updater*, *Negotiation Terminator* and *Agreement Maker*. Each sub-agent defines a *handle()* method that is activated whenever the sub-agent must act to check the category of rules it is responsible for. Note, again, that these sub-agents are not full-blown JADE agents, but classes within the host agent (we continue to use the name sub-agent to stay in agreement with terminology firstly proposed in [5]).

The seller agent encapsulates also two objects representing the negotiation locale and the blackboard: *Negotiation Locale* and *Blackboard* "boxes". The *Negotiation Locale* object stores the *negotiation template* (a structure that defines negotiation parameters; see [5]) and the list of participants that were admitted to a given negotiation. The *Blackboard* object is a JESS rule engine (class *jess.Rete*) that is initialized with negotiation rules.

Finally, the negotiation host contains handler methods that are activated by *action*() methods of agent behaviors (see sub-section 3.4). Each handler method delegates the call to the responsible sub-agent. Finally, the sub-agent activates the rule engine via a member object that points to the parent host agent.

3.2 Controlling Rule Execution

Rather then implementing each sub-agent of the negotiation host as a separate rule engine, we are using a single JESS rule engine that is shared by all sub-agents. This rule engine is implemented using class *jess.Rete*. The advantage is that we now have a single rule engine per negotiation host rather than 6 engines as suggested in [5]. Furthermore, this means that in the case of m products sold, we will utilize m instances of the JESS rule engine, instead of $6m$ instances necessary in [5].

Rules and facts managed by the rule engine are partitioned into JESS modules. Currently we are using one JESS module for storing the blackboard facts and a separate JESS

module for storing rules used by each sub-agent.

Blackboard facts are instances of JESS *deftemplate* statements and they can represent: (1) the negotiation template; (2) the active proposal that was validated by the *Proposal Validator* and the *Proposal Enforcer* sub-agents; (3) a withdrawn proposal, if the negotiation mechanism supports proposals withdrawal; (4) seller reservation price (not visible to participants); (5) negotiation participants; (6) the negotiation agreement that is eventually generated at the end of a negotiation; (7) the information digest that is visible to the negotiation participants; (8) the maximum time interval for submitting a new bid before the negotiation is declared complete; or (9) the value of the current highest bid. Note that these facts have been currently adapted to represent English auctions (and will be appropriately modified to represent other price negotiation mechanisms).

Each category of rules for mechanism enforcement is stored in a separate JESS module that is controlled by the corresponding sub-agent of the negotiation host. Whenever the sub-agent handles a message it activates the rules for enforcing the negotiation mechanism. Taking into account that all rules pertinent to a given host are stored internally in a single JESS rule-base (attached to a single JESS rule engine), the JESS *focus* statement is used to control the firing of rules located only in the focus module. This way, the JESS facility for partitioning the rule-base into disjoint JESS modules proves very useful to efficiently control the separate activation of each category of rules.

Note that JADE agent behaviors are scheduled for execution in a non-preemptive way and this implies that firings of rule categories are correctly serialized and thus they do not cause any synchronization problems. This fact also supports our decision to utilize a single rule engine for each host.

3.3 English Auctions

In this paper we consider a particular negotiation scenario involving English auctions. We consider English auctions because they are one of the most common and easiest to understand auction mechanism and they became so popular because of the establishment of many online auction houses like `eBay.com`.

Technically, English auctions are single-item, first-price, open-cry, ascending auctions ([14],[18]). In an English auction there is a single item sold by a single seller and many buyers bidding against each other for buying the item until the auction terminates. Usually, there is a time limit for ending the auction, a seller reservation price that must be met by the winning bid for the item to be sold and a minimum value of the bid increment. A new bid must be higher than the currently highest bid plus at least the minimal bid increment in order to be accepted. All the bids are visible to all the auction participants and cannot be withdrawn.

3.4 Generic Negotiation Protocol and Agent Behaviors

The generic negotiation protocol specifies a minimal set of constraints on sequences of messages exchanged between the host and participants. In [5], the allowed message exchanges between the host and participant agents, were described. Let us start from the realization that the negotiation process has three phases: (1) admission, (2) proposal submission and (3) agreement formation. The admission phase starts when a new participant requires admission to the negotiation, by sending an ACL PROPOSE message to the negotiation host. The host grants (or not) the admission of the participant to the negotiation and responds accordingly with either an ACL ACCEPT-PROPSAL or an ACL REJECT-PROPOSAL message (currently admission is granted by default). In the way that the system is currently implemented, the PROPOSE message is sent by the participant agent immediately after its initialization stage, just before its *setup()* method returns. The task of receiving the admission proposal and issuing the response is implemented as a separate behavior of the negotiation host.

When a participant is accepted to the negotiation, it also receives from the host the negotiation template. In our implementation the negotiation template was adapted to represent parameters of auctions: auction type, auctioned product, minimum bid increment, termination time window, currently highest bid.

A participant will enter the phase of submitting proposals immediately after it was accepted to the negotiation. This event is signaled by the reception of an ACL ACCEPT-PROPOSAL message together with the negotiation template. The negotiation template contains also the currently highest bid issued by one of the remaining participants. Note that participants are allowed to join negotiation dynamically. As soon as they obtain the negotiation template (and consequently the currently highest bid) they can start bidding according to their strategy. Here, the current proposal differs from that put forward in [10], where agents were admitted to the negotiations in groups.

The generic negotiation protocol states also that a participant will be notified by the negotiation host if its proposal was either accepted (with an ACL ACCEPT-PROPOSAL) or rejected (with an ACL REJECT-PROPOSAL). In the case when a proposal was accepted, the protocol requires that all the other participants will be notified accordingly with ACL INFORM messages.

Strategies of participant agents must be defined in accordance with the constraints stated by the generic negotiation protocol. Basically, the strategy defines when a negotiation participant will submit a proposal and what are the values of the proposal parameters. In our system, for the time being, we opted for an extremely simple solution: the participant will submit a first bid immediately after it was granted admission to the negotiation and subsequently, whenever it gets a notification that another participant issued a proposal that was accepted by the host. The value of the bid is equal to the sum of the currently highest bid and an increment value that is private to the participant. Additionally, each participant has its own valuation of the negotiated product in terms of a reservation price. If the value of the new bid exceeds this reservation price then the proposal submission is canceled. The implementation of the participant agent defines two JADE agent behaviors for dealing with situations stated above. Obviously, as the system matures, we plan to develop, implement and experiment with a number of negotiation strategies that can be found in the literature.

Finally, the agreement formation phase can be triggered at any time. When the agreement formation rules signal that an agreement was reached, the protocol states that all the participants involved in the agreement will be notified by the host with ACL INFORM messages. The agreement formation check is implemented as a timer task (class *java.util.TimerTask*) that is executed in the background

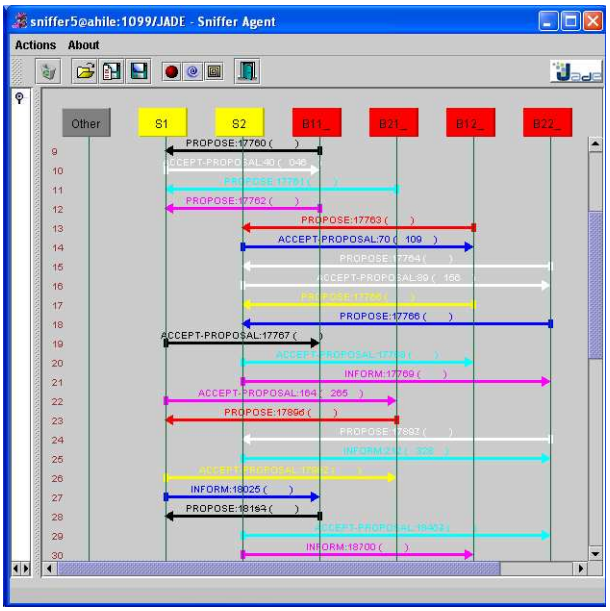

Figure 1: Negotiaton stage – part 1

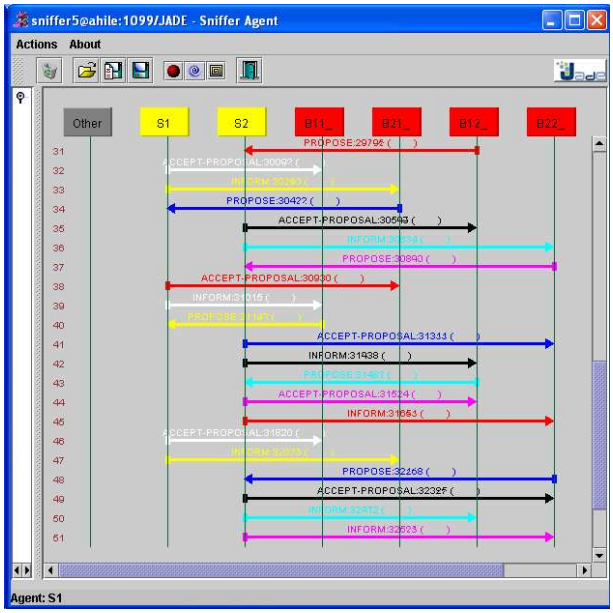

Figure 2: Negotiaton stage – part 2

thread of a *java.util.Timer* object.

4. EXPERIMENT

Let us consider that shop is selling 2 products using English auctions, both products have a reservation price of 50 and require a minimum bid increment of 5. We also consider 2 clients C_1 and C_2, each seeking both products. Client C_1 has a reservation price of 52 for product 1, a reservation price of 61 for product 2 and a bid increment of 9. Client C_2 has a reservation price of 54 for product 1, a reservation price of 63 for product 2 and a bid increment of 11. Client C_1 is using 2 buyers B_{11} and B_{12}, and client C_2 is using 2 buyers B_{21} and B_{22}. Messages exchanged between agents are shown in figures 1 and 2. Note that only shop (here labeled as *Shop*), sellers and buyers are shown in these figures (clients are not shown, as they only play the role of creating and sending buyers to negotiations).

Figures 1 and 2 show messages exchanged between agents during price negotiation. Unfortunately, in those figures, only message types are shown, while message content is invisible. An explanation of message exchanges is provided separately in table 1.

There are few interesting facts to note in table 1. First, when buyer B_{21} is granted admission to the negotiation, buyer B_{11} had already submitted a bid and that bid was accepted. Therefore B_{21} will get a value of 9 in the negotiation template for the currently highest bid; note that this is an example of a participant that dynamically joins negotiation in progress. Second, the negotiation between S_1 and agents B_{11} and B_{21} ended without a winner. The highest accepted bid was 49 from B_{11} but this value is lower than the reservation price 50 of S_1. According to their strategies, none of the participants B_{11} and B_{21} is able to issue a higher bid that is still lower than their own reservation prices. Third, negotiation between S_2 and agents B_{21} and B_{22} ended with agent B_{22} becoming a winner and the highest bid 60. Finally, note that bid 11 of buyer B_{22} was rejected because at the time this bid was submitted there was already a highest bid of 9 accepted, and thus, the rule saying that the minimum value of the bid increment is 5 was violated. However, by the time B_{22} submitted its bid, it wasn't aware that the other participant B_{12} also posted a bid and got it accepted.

5. CONCLUSIONS

In this paper we presented a multi-agent system that utilizes rule-based approach to implement flexible automated negotiations. System described here is under development using JADE and JESS toolkits. The state of its current implementation was described in some detail and experimental results of running the prototype were discussed. As future work we plan to: to complete the integration of the rule-based framework into our e-commerce model; to asses the generality of our implementation by extending it to include other price negotiation mechanisms; to allow the logical specification of the rules in order to asses their correctness; to investigate the effectiveness of describing and/or publishing negotiation rules using rule markup languages; to conceptualize representation and ways to efficiently implement multiple strategy modules. We will report on our progress in subsequent papers.

6. ACKNOWLEDGMENTS

We would like to use this opportunity to thank our colleagues authors of [5] for providing us their prototype implementation. This was a valuable input for producing the implementation and experiments described in our paper.

7. ADDITIONAL AUTHORS

Alin Iordache (Software Engineering Department, University of Craiova, Bvd.Decebal 107, Craiova, 200440, Romania, email: `alin_eestec@yahoo.com`) and Marcin Paprzycki (Computer Science Institute, SWPS, 03-815 Warsaw, Poland, email: `marcin.paprzycki@swps.edu.pl`).

Table 1: Explanation of message exchanges shown in figures 1 and 2

B_{11} 52 9	B_{21} 54 11	B_{12} 61 9	B_{22} 63 11
request admission	request admission	request admission	request admission
admission granted 0	admission granted 9	admission granted 0	admission granted 0
bid 9	bid 20	bid 9	bid 11
accept bid 9	accept bid 20	accept bid 9	inform 9
inform 20	inform 29	inform 20	bid 20
bid 29	bid 40	bid 29	reject bid 11
accept bid 29	accept bid 40	accept bid 29	accept bid 20
inform 40	inform 49	inform 40	inform 29
bid 49		bid 49	bid 40
accept bid 49		inform 60	accept bid 40
			inform 49
			bid 60
			accept bid 60
			win 60

8. REFERENCES

[1] Bădică, C., Ganzha, M., Paprzycki, M., Pîrvănescu, A.: Combining Rule-Based and Plug-in Components in Agents for Flexible Dynamic Negotiations. In: *Proceedings of CEEMAS'05*, Budapest, Hungary. LNAI, Springer Verlag (2005) 555–558.

[2] Bădică, C., Ganzha, M., Paprzycki, M., Pîrvănescu, A.: Experimenting With a Multi-Agent E-Commerce Environment. In: *Proceedings of PaCT'2005*, Krasnoyarsk, Russia. LNCS 3606 Springer Verlag, (2005) 393–401.

[3] Bădică, C., Ganzha, M., Paprzycki, M.: Mobile Agents in a Multi-Agent E-Commerce System. In: Proceedings of the SYNASC 2005 Conference (2005) (to appear)

[4] Bădică, C., Bădiţă, A., Ganzha, M., Iordache, A., Paprzycki, M.: Rule-Based Framework for Automated Negotiation: Initial Implementation. Accepted for presentation at RuleML'2005, Galway, Ireland. LNCS, Springer Verlag (2005) 193–198.

[5] Bartolini, C., Preist, C., Jennings, N.R.: A Software Framework for Automated Negotiation. In: *Proceedings of SELMAS'2004*, LNCS 3390, Springer Verlag (2005) 213–235.

[6] Benyoucef, M., Alj, H., Levy, K., Keller, R.K.: A Rule-Driven Approach for Defining the Behaviour of Negotiating Software Agents. In: J.Plaice et al. (eds.): *Proceedings of DCW'2002*, LNCS 2468, Springer verlag (2002) 165–181.

[7] Chmiel, K., Tomiak, D., Gawinecki, M., Karczmarek, P., Szymczak, Paprzycki, M.: Testing the Efficiency of JADE Agent Platform. In: *Proceedings of the* 3^{rd} *International Symposium on Parallel and Distributed Computing*, Cork, Ireland. IEEE Computer Society Press, Los Alamitos, CA, USA, (2004), 49–57.

[8] Dumas, M., Governatori, G., ter Hofstede, A.H.M., Oaks, P.: A Formal Approach to Negotiating Agents Development. In: *Electronic Commerce Research and Applications*, Vol.1, Issue 2 Summer, Elsevier Science, (2002) 193–207.

[9] FIPA: Foundation for Physical Agents. See `http://www.fipa.org`.

[10] Ganzha, M., Paprzycki, M., Pîrvănescu, A., Bădică, C., Abraham, A.: JADE-based Multi-Agent E-commerce Environment: Initial Implementation, In: *Analele Universităţii din Timişoara, Seria Matematică-Informatică* (2005) (to appear)

[11] Governatori, G., Dumas, M., ter Hofstede, A.H.M., and Oaks, P.: A formal approach to protocols and strategies for (legal) negotiation. In: Henry Prakken, editor, *Procedings of the 8th International Conference on Artificial Intelligence and Law*, IAAIL, ACM Press, (2001) 168–177.

[12] JADE: Java Agent Development Framework. See `http://jade.cselt.it`.

[13] JESS: Java Expert System Shell. See `http://herzberg.ca.sandia.gov/jess/`.

[14] Laudon, K.C., Traver, C.G.: *E-commerce. business. technology. society* (2^{nd} ed.). Pearson Addison-Wesley, (2004).

[15] Lochner, K.M., Wellman, M.P.: Rule-Based Specification of Auction Mechanisms. In: *Proc. AAMAS'04*, ACM Press, New York, USA, (2004).

[16] Lomuscio, A.R., Wooldridge, M., Jennings, N.R.: A classification scheme for negotiation in electronic commerce. In: F. Dignum, C. Sierra (Eds.): *Agent Mediated Electronic Commerce: The European AgentLink Perspective*, LNCS 1991, Springer Verlag (2002) 19–33.

[17] Skylogiannis, T., Antoniou, G., Bassiliades, N.: A System for Automated Agent Negotiation with Defeasible Logic-Based Strategies - Preliminary Report. In: Boley, H., Antoniou, G. (eds): *Proceedings RuleML'04*, Hiroshima, Japan. LNCS 3323 Springer-Verlag (2004) 205–213.

[18] Wooldridge, M.: *An Introduction to MultiAgent Systems*, John Wiley & Sons, (2002).

[19] Wurman, P.R., Wellman, M.P., Walsh, W.E.: The Michigan Internet AcutionBot: A Configuarable Auction Server for Human and Software Agents. In: *Proceedings of the second international conference on Autonomous agents. Agents'98*, Minneapolis, USA. ACM Press, New York, USA, (1998), 301-308.

[20] Wurman, P.R., Wellman, M.P., Walsh, W.E.: A Parameterization of the Auction Design Space. In: *Games and Economic Behavior*, 35, Vol. 1/2 (2001), 271–303.

[21] Wurman, P.R., Wellman, M.P., Walsh, W.E.: Specifying Rules for Electronic Auctions. In: *AI Magazine* (2002), 23(3), 15–23.