# Rule-Based Framework for Automated Negotiation: Initial Implementation

Costin Bădică[1], Adriana Bădiţă[1], Maria Ganzha[2], Alin Iordache[1], Marcin Paprzycki[3]

[1] University of Craiova, Software Eng. Dept. Bvd. Decebal 107, Craiova, 200440, Romania
badica_costin@software.ucv.ro
[2] Elblag University of Humanities and Economy, ul Lotnicza 2, 82-300 Elblag, Poland
ganzha@op.pl
[3] Computer Science Institute, Warsaw School of Social Psychology, 03-815 Warsaw, Poland
Marcin.Paprzycki@swps.edu.pl

**Abstract.** The note reports on the current status of an implementation of a rule-based negotiation mechanism in a model e-commerce multi-agent system. Here, we briefly describe the conceptual architecture of the system and its initial implementation utilizing JADE and JESS. A particular negotiation scenario involving English auctions performed in parallel is also discussed.

## 1 Introduction

Recently, we have started developing, implementing and experimenting with a multi-agent e-commerce system (see [5] and work referenced there). One of the directions of our work is to provide agents with flexibility required for price negotiations [10] governed by mechanisms unknown in advance. In this context, rule-based approaches have been indicated as a very promising technique for parameterizing the negotiation design space ([1, 2, 4, 9, 11, 12]). Proposals have been put forward to use rules for describing either negotiation strategies ([4, 11]), mechanisms ([1]) or both ([6]), while special attention has been paid to auctions, as one of the best understood forms of negotiations ([12]). Note that when designing systems for automated negotiations one should distinguish between *negotiation protocols* (or *mechanisms*) that define "rules of encounter" between participants and *negotiation strategies* that define behaviors aiming at achieving a desired outcome.

In this paper we discuss design and implementation of a rule-based framework for enforcing specific negotiation mechanisms inspired by work presented in [1]. We proceed as follows. In the next section we describe briefly the negotiation framework introduced in [1] and show how it fits into our e-commerce model. In section 3 we outline our design and give some details of the sample implementation using JADE ([7]) and JESS ([8]). In particular we highlight how rules are activated by the negotiation host in response to messages received from the negotiation participants. Furthermore we show how, in our implementation, the rule-based sub-agents of the negotiation host (as described in [1]) share a single JESS rule engine, rather than having separate rule engines within each sub-agent. We follow with description of two experiments: a simple experiment to highlight agent interactions and a more complex experiment with many agents and many parallel negotiations performed to asses the scalability of the implementation.

## 2 Conceptual Architecture

Authors of [1] sketched a complete framework for implementing portable agent negotiations. Their framework comprises: (1) negotiation infrastructure, (2) generic negotiation protocol and (3) taxonomy of declarative rules. The *negotiation infrastructure* defines roles of negotiation participants and of a host. Participants negotiate by exchanging proposals within a negotiation locale managed by the host. Depending on the negotiations type, the host can also play the participant role participant. The *generic negotiation protocol* defines the three phases of a negotiation: admission, exchange of proposals and formation of an agreement, in terms of how and when messages should be exchanged between the host and participants. *Negotiation rules* are used for enforcing the negotiation mechanism. Rules are organized into a taxonomy: rules for participants admission to negotiations, rules for checking the validity of negotiation proposals, rules for protocol enforcement, rules for updating the negotiation status and informing participants, rules for agreement formation and rules for controlling the negotiation termination.

Our goal is to create a model system in which agents perform functions typically observed in e-commerce. In this environment, e-shops and e-buyers are represented by shop and seller, and respectively client and buyer agents. Let us consider a simplified version of this scenario that involves a single shop agent $S$ and $n$ client agents $C_i$, $1 \leq i \leq n$. The shop agent is selling $m$ products $\mathcal{P} = \{1, 2, \ldots, m\}$. 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 (we therefore restrict our attention to the case where all sought products are available through shop agent $S$). Shop agent $S$ is using $m$ 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 in 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 client agent $C_i$, will engage in negotiation with seller $S_k$, to purchase product $k$. This simple scenario is sufficient for the purpose of our paper, i.e. to illustrate how a number of rule-based automated negotiations can be performed concurrently. In this setting, each seller agent $S_j$ plays the role of a negotiation host defined in [1]. Therefore, in our system, we have exactly $m$ instances of the framework described in [1]. Each instance is managing a separate negotiation "locale", 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). See figure 1a for an example.

## 3 Design and Implementation

Let us now discuss: (i) how the negotiation host is structured into sub-agents; (ii) how rules are executed by the host in response to messages received from participants and how rule firing control is switched between sub-agents; (iii) how the generic negotiation protocol was implemented using JADE agent behaviors and ACL message exchanges.

**The Negotiation Host.** Host and negotiation participant agents are ordinary JADE agents. 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 has a *handle()* method that is activated whenever the sub-agent must react to check the category of rules it is responsible for. In addition to sub-agents, the host encapsulates two objects: the *Negotiation Locale* stores the *negotiation template* (a structure that defines negotiation parameters; see [1]) and the list of negotiation participants; the *Blackboard* object is a JESS rule engine (class *jess.Rete*) that is initialized with negotiation rules. Whenever category of negotiation rules is checked, the rule engine is activated. The host contains handler methods that are activated by *action()* methods of the agent behaviors. 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.

**Controlling Rule Execution.** Rather then implementing each sub-agent of the negotiation host as a separate rule engine [1], we use a single JESS engine shared by all sub-agents. Rules and facts managed by the rule engine are partitioned into JESS modules. *Blackboard facts* are instances of JESS *deftemplate* statements and they represent: the negotiation template; the active proposal that was validated by the *Proposal Validator* and the *Proposal Enforcer* sub-agents; seller reservation price (not visible to participants); negotiation participants; the negotiation agreement that is eventually generated at the end of a negotiation; the information digest that is visible to the negotiation participants; the maximum time interval for submitting a new bid before the negotiation is declared complete; the value of the current highest bid. Each category of rules for mechanism enforcement is stored in a separate JESS module. This module 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 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 focused module. This way, the JESS facility for partitioning the rule-base into distinct JESS modules proves very useful for controlling 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.

**Generic Negotiation Protocol and Agent Behaviors.** The negotiation process has three phases: (1) admission, (2) proposal submission and (3) agreement formation. Tasks of sending and receiving messages according to the constraints stated by the negotiation protocol are implemented using JADE agent behaviors.

The *admission* phase starts when a new participant requests admission by sending a PROPOSE message to the host. The host grants (or not) the admission of the participant to the negotiation and responds with either an ACCEPT-PROPSAL or a REJECT-PROPOSAL message. Currently, the PROPOSE message is sent by the participant immediately after its initialization stage, just before its *setup()* method returns. The task of receiving the admission proposal and issuing an appropriate response is implemented as a separate host behavior. When a participant is admitted, it receives from the host a template representing auctions parameters: auction type, auctioned product, minimum bid increment termination time window, currently highest bid.

A participant enters the phase of *submitting proposals* immediately after it was admitted (participants join negotiation dynamically). This event is signaled by the recep-

tion of an ACCEPT-PROPOSAL message together with the negotiation template containing currently highest bid. As soon as they obtain the negotiation template and currently highest bid agent sends its first bid. The negotiation protocol states also that a participant will be notified by the host if its proposal was accepted (ACCEPT-PROPOSAL) or rejected (REJECT-PROPOSAL). When a proposal is accepted, the protocol requires that all other participants are notified accordingly with INFORM messages. Strategies of participant agents must be defined in accordance with the generic negotiation protocol. The strategy defines when a negotiation participant submits a proposal and what are proposal parameters. For the time being we utilize a simplistic solution: participant submits first bid immediately after it was admitted and subsequently, whenever it receives notification that another proposal was accepted by the host. Each time the value of the bid is equal to that of the currently highest bid plus an increment (that is private to the participant). Additionally, each participant has its own reservation price and if the value of the new bid exceeds it then the proposal submission is canceled.

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 will be notified by the host with INFORM messages. The agreement formation check is implemented as a timer task (class *java.util.TimerTask*) that is executed in the background thread of a *java.util.Timer* object.
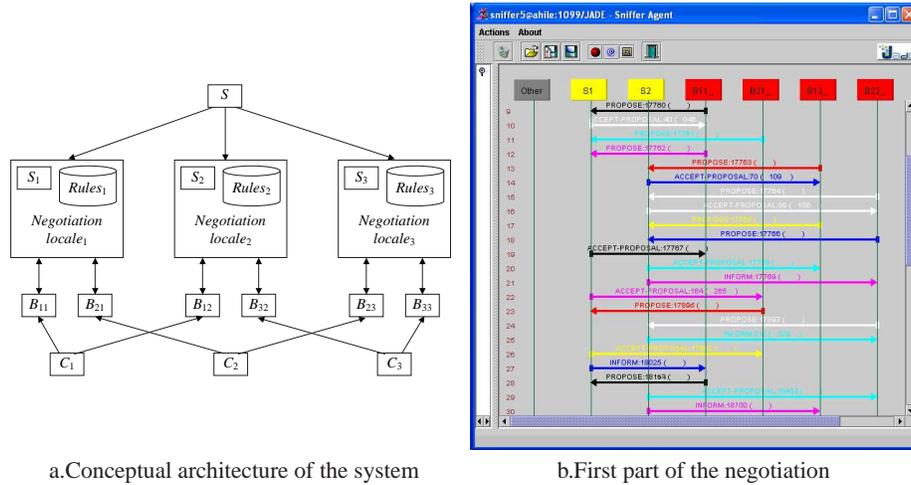


a.Conceptual architecture of the system          b.First part of the negotiation

**Fig. 1.**

## 4   Experiments

In the first experiment we consider that shop is selling 2 products, both products have a reservation price of 50 and require a minimum bid increment of 5. There are 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

**Table 1.** Explanation of message exchanged during negotiation in experiment 1

| $B_{11}$   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 |
| | | | win 60 |

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 buyers $B_{11}$ and $B_{12}$, and similarly client $C_2$ is using buyers $B_{21}$ and $B_{22}$. Some of the messages exchanged between agents in this experiment are shown in figure 1b (note that only sellers and buyers are shown on that figure (clients are not shown, as they only play the role of creating buyers and sending them to negotiations). While Figure 1b show messages exchanged between agents during negotiation, their content is not visible. Therefore we provide an explanation of message exchanges in Table 1. The table header contains buyer names together with their reservation prices and bid increments.

There are some 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.

In the second experiment we considered 10 products and 12 clients seeking all of them. The auction parameters were the same for all auctions: reservation price 50 and minimum bid increment 5. Clients reservation prices were randomly selected from the interval [50,72] and their bid increments were randomly selected from the interval [7,17]. In the experiment 143 agents were created: 1 shop, 10 sellers, 12 clients and 120 buyers and 10 English auctions were run in parallel. The average number of messages exchanged per negotiation was about 100 and all the auctions finished successfully. While the total number of agents is still small (as compared to [3]) this experiment indicates that the proposed approach has good potential for scalability.

## 5   Concluding remarks

In this note we have discussed a multi-agent system that utilizes a rule-based approach to implement flexible automated negotiations. This system is being implemented using JADE and JESS and its simplified version works for the case of English auctions. As future work we plan to: i) complete the integration of the rule-based framework into our e-commerce model; ii) asses the generality of our implementation by extending it to include other price negotiations; iii) allow the logical specification of the rules in order to asses their correctness; iv) investigate the effectiveness of describing and/or publishing negotiation rules using rule markup languages. We will report on our progress in subsequent papers.

**Acknowledgement**. We would like to thank authors of [1] for providing us with their sample implementation. This was a valuable input for producing the implementation and experimental results described here.

## References

1. Bartolini, C., Preist, C., Jennings, N.R.: A Software Framework for Automated Negotiation. In: *Proc. of SELMAS'2004*, LNCS 3390, Springer Verlag (2005) 213–235.
2. 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.): *Proc. of DCW'2002*, LNCS 2468, Springer verlag (2002) 165–181.
3. Chmiel, K., Tomiak, D., Gawinecki, M., Karczmarek, P., Szymczak, Paprzycki, M.: Testing the Efficiency of JADE Agent Platform. In: *Proc. of the 3$^{rd}$ International Symposium on Parallel and Distributed Computing*, Cork, Ireland. IEEE Computer Society Press, Los Alamitos, CA, USA, (2004), 49–57.
4. Dumas, M., Governatori, G., ter Hofstede, A.H.M., Oaks, P. (2002): A Formal Approach to Negotiating Agents Development. In: *Electronic Commerce Research and Applications*, Vol.1, Issue 2 Summer, Elsevier Science, (2002) 193–207.
5. 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)
6. 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, *Proc. of the 8th International Conference on Artificial Intelligence and Law*, IAAIL, ACM Press, (2001) 168–177.
7. JADE: Java Agent Development Framework. See `http://jade.cselt.it`.
8. JESS: Java Expert System Shell. See `http://herzberg.ca.sandia.gov/jess/`.
9. Lochner, K.M., Wellman, M.P.: Rule-Based Specification of Auction Mechanisms. In: *Proc. AAMAS'04*, ACM Press, New York, USA, (2004).
10. 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.
11. 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): *Proc. of RuleML'04*, Hiroshima, Japan. LNCS 3323 Springer-Verlag (2004) 205–213.
12. 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.