Journal of Universal Computer Science, vol. 13, no. 2 (2007), 244-266
submitted: 31/7/06, accepted: 15/1/07, appeared: 28/2/07 © *J.UCS*

Implementing Rule-Based Automated Price Negotiation in an Agent System

Costin Bădică
(University of Craiova, Romania
badica_costin@software.ucv.ro)
Maria Ganzha
(Systems Research Institute, Polish Academy of Science Warsaw, Poland
Elbląg University of Humanities and Economy, Elbląg, Poland
maria.ganzha@ibspan.waw.pl)
Marcin Paprzycki
(Systems Research Institute, Polish Academy of Science Warsaw, Poland
Warsaw School of Social Psychology, Warsaw, Poland
paprzyck@ibspan.waw.pl)

Abstract: The idea of automating e-commerce transactions attracts a lot of interest among researchers and IT practitioners, and multi-agent systems are claimed to be one of promising software technologies for achieving this goal. Since price negotiations are one of crucial aspects of e-commerce transactions, in this paper we present a rule-based implementation of automated price negotiations utilized in a multi-agent system that models an e-commerce environment. We start by summarizing state-of-the-art in rule-based approaches to automated negotiations. We follow with a brief description of the conceptual architecture of our system and a simplified scenario that involves multiple buyer agents participating in multiple English auctions performed in parallel. A detailed discussion of the design and implementation of price negotiations, using JADE and JESS, and presentation of sample experiments complete the paper.

Key Words: multiagent system, electronic commerce transaction, rule-based representation
Category: I.2.11, K.4.4, I.2.4

1 Introduction

Progress in development of the Internet as a global e-commerce network resulted in a shift from simple "business Web presence" to advanced use of e-commerce technologies. As a result extra attention has been devoted to automating e-commerce transactions ([Laudon 2004]). At the same time it is easy to see that in existing e-commerce systems humans make most important decisions in various activities taking place along the lifeline of any e-commerce transaction. However, software agents are claimed to be one of the best technologies for automating e-commerce processes [Kowalczyk and al.2002]. It is expected that intelligent agents will be able to substantially reduce (if not eliminate) need for human involvement in all but most crucial decisions. Following this belief, we have undertaken a project to contribute to the development of such an agent system, with the main goal to develop and implement—using agent tools—a large-scale model of an e-commerce environment [Ganzha et al. 2004].

Commercial transactions can be conceptualized as consisting of four phases (here, we slightly modify results presented in [Wooldridge 2002, Laudon 2004]): (i) *pre-contractual phase* including activities like need identification, product brokerage, merchant brokerage, and matchmaking; (ii) *negotiation* where negotiation participants negotiate according to the rules of a particular price negotiation mechanism and using their private negotiation strategies; (iii) *contract execution* including activities like: order submission, logistics, and payment; and (iv) *post-contractual phase* that includes activities like collecting managerial information and product or service evaluation. In e-commerce information technologies are utilized to various degree to mediate each one of these processes. Focus of this paper is on the *negotiation phase* taking place within a multi-agent e-commerce system. In this context we are particularly interested in utilizing rule-based approaches to represent and execute negotiation mechanisms and present our approach to design and implementation of flexible rule-based price negotiations.

We start our presentation with an overview of the state-of-the-art of rule-based approaches used in automated negotiations. We follow with a brief summary of the conceptual architecture of our agent-based system, together with a sample scenario. In the next section we discuss the design and implementation of price negotiations, emphasizing their following aspects: (i) representation of a particular negotiation mechanism (i.e. an English auction) using a taxonomy of JESS rules; (ii) structure of the negotiation host—a software entity that is responsible for management of the negotiation process; and (iii) behavior and communication of negotiating agents: host and participants. We conclude presentation of our system by describing two experiments: (i) a simple experiment—that highlights agent interactions and rule activations and (ii) a more complex experiment with multiple agents and multiple parallel negotiations performed simultaneously, which allows us to look into scalability of the implementation. In conclusions we point to future research directions.

2 Overview of Automated and Rule-Based Negotiations

Let us start from a definition of negotiations according to [Lomuscio et al. 2002, Jennings et al. 2001]. Negotiation is a process by which a group of agents communicate to try to come to a mutually acceptable agreement on some matter. It is one of important methods for establishing agent cooperation. Understood in this way, negotiation mechanism consists of two parts:

- negotiation protocol—convention under which negotiation operates,

– negotiation strategies—specification of the sequence actions the agent plans to make during negotiation and that are supposed to lead to a desired outcome.

Since we are interested in software agents, let us recall that one of their main characteristics is autonomy. Therefore, to automate negotiation, their complete formalization has to be developed. One of first steps toward such a formalization is classification of negotiation scenarios and one of first attempts at achieving this goal can be found in work of Wurman and colleagues [Wurman et al. 2001]. Here, a mathematical characterization of auction rules designed to parameterize the auction design space was introduced. The proposed parametrization was organized along three axes: i) *bidding rules*—stating when bids may be posted, updated or withdrawn; ii) *clearing policy*—stating how the auction commands resource allocation (including auctioned items and money) between auction participants (this corresponds roughly to agreement making in our approach); iii) *information revealing policy*—stating how and what intermediate auction information is supplied to participating agents. This work has influenced a number of later projects (see below) and was the foundation of the Michigan AuctionBot project (e.g. [Wurman et al. 1998]).

In [Reeves et al. 1999] authors presented an improvement of the declarative language, which has been originally used in the Michigan AuctionBot Project ([Wurman et al. 1998]). This language has been used to create scenarios configuring negotiations. Three important research questions have been addressed in this work: (1) how can one represent information to allow automatic inference of negotiation structures; (2) how can one automate negotiations in a way that will closely drive a realistic automated platform (the Michigan Internet AuctionBot); (3) how can one use auction results to form a final contract? In the language prototype, described in [Reeves et al. 1999], authors have proposed concepts and vocabulary to reason about several aspects of the negotiation process: (1) high-level knowledge about alternative negotiation structures, (2) general-case rules about auction parameters, (3) rules to map the auction parameters to a specific auction platform (e.g. the Michigan Internet AuctionBot), and (4) special-case rules for specific domains, including rules from potential buyers and sellers about capabilities, constraints, and preferences. However, it should be noted that work of Reeves and colleagues was primarily focused on general contract forming, rather then price negotiations.

In another continuation of work originated in [Wurman et al. 1998], experiences gained during the Michigan Internet AuctionBot project were used in design and implementation of a new rule-based scripting language (*AB3D*) for expressing auction mechanisms [Lochner and Wellman 2004]. According to its authors, *AB3D* allows initialization of auction parameters, definition of rules for triggering auction events, declaration of user variables and definition of rules for controlling bid admissibility. The latest version of *AB3D* implementation is available for download at [AB3D]. It has to be noted that, at this stage of development, the proposed language does not seem to have a declarative semantics, has not been standardized, and its only specification is its implementation and the description found in [Lochner and Wellman 2004].

In our work we follow a rule-based framework for enforcing specific negotiation mechanisms introduced in ([Bartolini et al. 2005, Bartolini et al. 2002]). First, we have to point out that in these papers we find a slightly different understanding of the term "negotiation mechanism." There, this term denotes what above has been specified as the negotiation protocol. In their work, authors of [Bartolini et al. 2005, Bartolini et al. 2002], present a complete framework for implementing portable agent negotiations that consists of: (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 exchange proposals within a "negotiation locale" managed by the host. The *generic negotiation protocol* defines three phases of a negotiation: admission, exchange of proposals and formation of an agreement, in terms of how, when and what types of messages should be exchanged between the host and negotiation 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 validity of 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. Unfortunately, the proposed approach was implemented in a very restricted and rigid way and it is difficult to envision this demonstrator system as a starting point of an implementation of a truly generic price negotiations environment. It is the aim of this paper to show an alternative design and implementation of a modified version of this approach.

The proposal for formalizing negotiations introduced in [Tamma et al. 2002] goes beyond the framework of [Bartolini et al. 2005, Bartolini et al. 2002]. Its authors suggest usage of an ontology-based approach to expressing negotiation protocols. Specifically, whenever an agent is admitted to negotiation it is to obtain a specification of the negotiation rules in terms of a shared ontology. In some sense, the negotiation template proposed by [Bartolini et al. 2005, Bartolini et al. 2002] and used by our implementation (see [Bădică et al. 2005d]) is a "simplified" negotiation ontology and the participants must be able to "understand" slots defined in the template. This approach has been exemplified with a sample scenario and taken further in [Tamma et al. 2005] by investigating how the ontology can be used to tune the negotiation strategy of participating agents. However, paper [Tamma et al. 2002] contains neither implementation details, nor experimental results. Furthermore, we were not able to obtain a complete version of the ontology proposed in the paper.

As mentioned above, the negotiation mechanism consists not only of the negotiation protocol, but also of the negotiation strategy. In [Lomuscio et al. 2002] authors presented an interesting overview of issues related to negotiation strategies. The main goal of this discussion was to identify possible parameters that can be used to classify negotiation mechanisms. As a result of such classification, it was possible to provide a conceptual framework within which protocols and strategies for negotiation could be classified and reasoned about.

Another group of approaches to address issues related to defining agent strategies was based on defeasible logic. First, a formal executable approach for defining the strategy of agents participating in negotiations using defeasible logic programs was reported in [Dumas et al 2002] and [Governatori et al. 2001]. An English auction and bargaining with multiple parties were used as an illustration. Here, sets of rules for describing strategies of participating agents were presented. While paper [Governatori et al. 2001] contains no implementation details and no experimental results; both these papers influenced other projects.

In [Skylogiannis et al. 2004] a preliminary implementation of a system of agents (building on the architecture introduced in [Dumas et al 2002]) that negotiate using strategies expressed in defeasible logic was described. The implementation is demonstrated with a bargaining scenario involving one buyer and one seller agent. The buyer strategy is defined by a defeasible logic program. Furthermore, defeasible logic programs expressed courteous logic programs proposed in [Reeves et al. 1999] and yet supported efficient reasoning, which suggest that they might be the appropriate representation formalism of negotiation strategies. Interestingly, the proposed approach used a defeasible logic engine—DR-DEVICE that was implemented by translating defeasible rules into JESS rules. It should be mentioned, that our analysis of available tools also resulted in selecting JESS as the rule engine.

Finally, let us mention two more projects related to utilization of rule-based mechanisms in agent negotiations. The CONSENSUS system that enables agents to engage in combined negotiations was presented in [Benyoucef et al. 2002]. CONSENSUS allows agents to negotiate different complementary items on separate servers on behalf of human users. Each CONSENSUS agent uses a rule base partitioned into: i) *basic rules* that determine the negotiation protocol, ii) *strategy rules* that determine the negotiation strategy, and iii) *coordination rules* that determine the knowledge for assuring that either all of the complementary items or none are purchased. Note that in CONSENSUS the rule-based approach is taken beyond mechanism and strategy representation to capture also coordination knowledge. While this work supports general usefulness of rule-based approach to negotiations, it lacks a rule-based representation of the negotiation mechanism and thus is somewhat limited in applicability.

Another interesting work is the open environment for automated negotiations specifically targeted auctions—*auction reference model* (*ARM*), and its associated *declarative auction specification language* (*DAL*); see, [Rolli et al. 2005, Rolli and Eberhart 2005]. It should be noted that, while not explicitly using rules, a *DAL* specification actually models the flow of an auction using a rule-based approach. *DAL* constructs comprise: views, validations, transitions and agreement generators, where views are analogous to visibility rules, validations are analogous to validity and protocol enforcement rules, transitions are analogous to update rules and agreement generators are analogous to agreement formation and negotiation life-cycle rules. The proposed approach is quite interesting; however, it is also very tightly linked to SQL (it uses SQL statements embedded in the representation). While SQL is declarative in nature, it is tightly linked to a relational implementation, which reduces flexibility of the proposed method.

Overall, while there is a large body of work devoted to automatic price negotiations arising in the context of agent systems, none of them seems to be fully satisfactory. Therefore our aim is to modify and extend work of Bartolini, Jennings and Preist ([Bartolini et al. 2005, Bartolini et al. 2002]) in the direction of flexibility and robustness. Before we present details of our design and implementation of price negotiations, let us start from an overview of our model system.

3 System Architecture Overview

Conceptually, our system models a distributed marketplace in which agents perform functions typically observed in e-commerce ([Ganzha et al. 2004]). There are two "sides" of commerce conceptualized in our design. On the one hand, e-shops are represented by shop and seller agents, while on the other, product buyers are represented by client and buyer agents. In Figure 1 we present Use Case diagram of the complete system. Outside bounds of the system we can see a *User-Client* who attempts at buying product(s) and a *User-Seller* who tries to sell products in her e-store. Let us now briefly summarize the most important agents appearing in the system and their functionalities (for a complete discussion of the system see [Bădică et al. 2005c, Bădică et al. 2005e, Bădică et al. 2005f]). *User-Client* is represented by the *Client Agent* (*CA*). The *CA* is completely autonomous and as soon as the desire to purchase product P is communicated by the *User-Client*, it will work without supervision until either P is purchased or, due to the market circumstances (e.g. prices are to high), purchase is abandoned. The *CA* communicates with the *Client Information Center* (*CIC*) agent which stores complete information which e-shops sell which products. The *CA* utilizes its trust assessment of each e-store to select these that it will interact with [Bădică et al. 2006c]. For each selected store, the *CA* delegates a single *Buyer Agent* (*BA*) with a mission to get involved in price negotiations and if

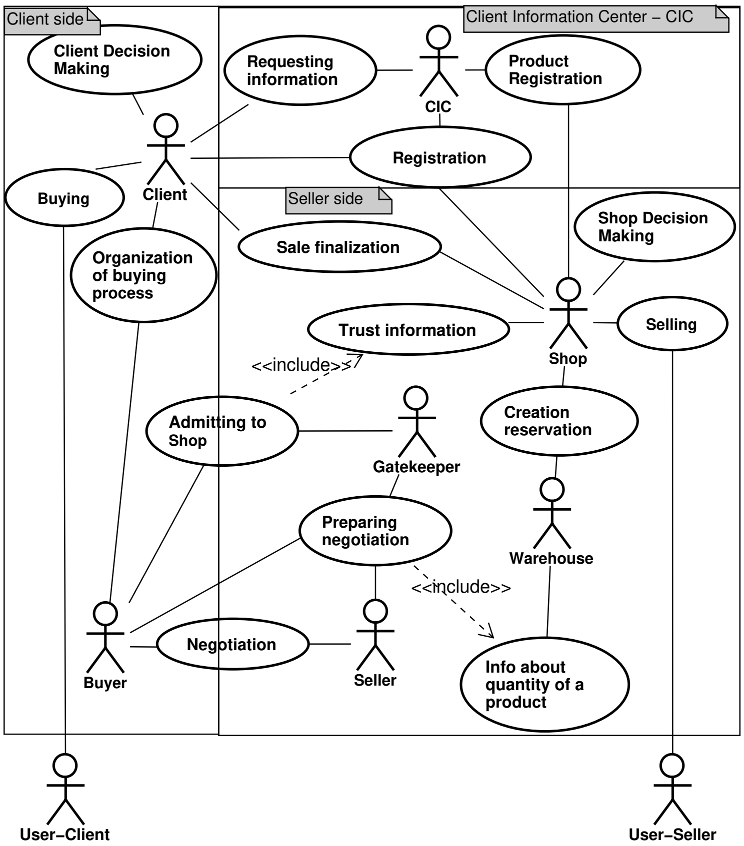

Figure 1: Use Case diagram of the proposed agent-based e-commerce system

successful, possibly attempt at making a purchase—in our system successful price negotiations result in a product reservation for a specific time period; after which products that have not been actually purchased are returned to the pool of products available for sale. Since multiple *BA*s representing the same *CA* can win price negotiations (at separate e-stores) and report to the *CA* conditions of possible transaction, it is the *CA* that decides if either of available offers is good enough to make a purchase. In the system we have designed, *Buyer Agents* can either migrate to the negotiation host or be created locally on the request of the *CA* (for more information about agent mobility see [Bădică et al. 2005f]). Overall, *BA*s can participate in price negotiations only if the *Gatekeeper Agent*

(GA) allows this. This time it is the GA that utilizes *trust information* to evaluate if a given BA should be admitted (e.g. BAs that win price negotiations but do not make a purchase may be barred from subsequent negotiations; for more information about trust management see [Bădică et al. 2006c]). The GA is one of agents that represent the e-store and is created by the *Shop Agent* (SA). The SA is the "central manager" of the e-shop. Facilitating the selling process, the SA utilizes the GA, as well as a *Warehouse Agent* (WA) that is responsible for inventory and reservation management; and a number of *Seller Agents* (SeA) that negotiate price with admitted BAs.

In our experiments, reported in this paper, we consider a simplified scenario that involves a single *Shop Agent* and n *Client Agent*s CA_i, $1 \leq i \leq n$. The SA is selling m products $\mathcal{P} = \{1, 2, \ldots, m\}$. We assume that each client agent CA_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 the SA). The SA is using m *Seller Agents* SeA_j, $1 \leq j \leq m$ and each SeA_j is responsible for selling a single product j. Each CA_i is using buyer agents BA_{ik} to purchase products from the set $\mathcal{P}_i$. Each BA_{ik} is responsible for negotiating and buying exactly one product $k \in \mathcal{P}_i$, $1 \leq i \leq n$. To attempt at making a purchase *Buyer Agent*s BA_{ik} migrate to the SA and engage in negotiations (here we omit trust related issues—all agents are admitted; furthermore, only agent mobility based approach is used); a BA_{ik}, that was spawned by the *Client Agent* CA_i, will engage in negotiation with seller SeA_k, to purchase product k. Since, in the context of this paper, we are interested only in price negotiations, we omit here all issues related to post-negotiation actions of the system (these have been reported in [Bădică et al. 2005c, Bădică et al. 2005e, Bădică et al. 2005f]). Note that this simple scenario is sufficient for the purpose of our paper, i.e. to discuss details of design and implementation of our rule-based system and to show how a number of rule-based automated negotiations can be performed concurrently. In the proposed setting, each *Seller Agent* SeA_j plays the role of a negotiation host defined in [Bartolini et al. 2005, Bartolini et al. 2002] (see below). As a result, in our system, we have exactly m instances of the framework described there. Each instance is managing a separate "negotiation locale", while all instances are linked to a single *Shop Agent*. For each instance we have one separate set of rules together with a negotiation template that describes the negotiation mechanism implemented by that host. Note that each seller may use a different negotiation mechanism (e.g. a different form of an auction, or an auction characterized by different parameters, such as the starting price or the bidding increment).

4 Design and Implementation

Let us now discuss in detail the design and the implementation of the system. As noted above, the starting point of our work is the rule-based framework for automated negotiation proposed in [Bartolini et al. 2005, Bartolini et al. 2002] (and briefly described in section 2, above). In this paper we focus our attention on the following aspects of price negotiations: (i) how rules can be utilized to represent a negotiation mechanism (an English auction in particular); (ii) how the negotiation host agent is structured into sub-agents; (iii) how rules are executed by the negotiation host in response to various messages received from negotiation participants, (iv) how rule firing control is switched between various sub-agents of the negotiation host, and (v) how the generic negotiation protocol and participant agent strategies were implemented using JADE agent behaviors and ACL message exchanges between the host and participants.

4.1 Rule-based Representation of English Auctions

Let us start our discussion by showing how rules can be utilized for representing a particular negotiation mechanism—English auction. Technically, English auction is a single-item, first-price, open-cry, ascending auction ([Laudon 2004, Wooldridge 2002]). In an English auction there is a single item (or a collection of items treated as a single item) sold by a single seller, and multiple buyers bidding against each other for buying that item. Usually, (1) there is an established initial price, (2) a seller reservation price that must be met by the winning bid for the item to be sold, and (3) a minimum value of the bid increment (a new bid must be higher than the currently highest bid plus that minimal bid increment in order to be accepted), and (4) all bids are visible to all auction participants. There are two possible mechanisms for ending the bidding phase of the negotiation. First, there is a total time limit for the auction; e.g. auction will end at 13:47. This mechanism is often used in Internet-based auctions; e.g. in eBay or Allegro. Second, there is a certain period of inactivity; e.g. in art auctions at Sotheby's, when the auctioneer says: "going once, going twice, sold," and any new bid causes the "inactivity clock" to be restarted. Since the first scenario involves so called "sniping" (and makes the whole process somewhat more complicated), for the time being we have decided to follow the second approach and defined the "time window" as the time of inactivity that causes auction to end.

4.1.1 Negotiation template

Bartolini, Jennings and Preist [Bartolini et al. 2005, Bartolini et al. 2002] define the *negotiation template* as a set of parameters and constraints of a given negotiation. An example of such a constraint in the case of an English auction is the minimum value of the bid increment (a constant during a given English auction). In Table 1 we summarize *all* values that parameterize our approach to

Table 1: English auction parameters

Parameter	Visibility
Initial price	Public
Minimum value of the bid increment	Public
Period of inactivity	Public
Seller reservation price	Private to *Seller*

the English auction as well as their visibility (for instance the *Seller* reservation price is visible only to the *Seller Agent*).

In the process of preparing negotiations (see Figure 1) the *BA* obtains from the *GA* the template that contains all necessary information. Such a (buyer) template, represented in XML, could look like this:

```
<?xml version="1.0" encoding="ISO-8859-1" ?> <MyTemplate>
    <NegotiationType>SingleItemEnglishAuction</NegotiationType>
    <BidIncrement>5.0</BidIncrement>
    <CurrentlyHighestBid>0.0</CurrentlyHighestBid>
    <TerminationWindow>300</TerminationWindow>
</MyTemplate>
```

It specifies that the negotiation type is: *SingleItemEnglishAuction*, bid increment is 5.0 units, starting bid is going to be 0.0 units and the window of inactivity that will terminate an auction is 300 seconds. The template that is sent by the *GA* to the *SeA*, when commanding it to start negotiations, has the same form, contains the same information as that sent to the *BA*, plus the *Seller* reservation price.

4.1.2 Negotiation rules

Negotiation rules are organized into a taxonomy and are used for enforcing a particular negotiation mechanism. We have rules for checking the validity of proposals, rules for protocol enforcement, rules for updating negotiation status and informing participants, rules for agreement formation and rules for controlling the negotiation termination ([Bartolini et al. 2005, Bartolini et al. 2002]). Whenever a new proposal is submitted by a participant, the proposal goes through a processing flow that first, checks its syntactic compliance with the negotiation rules and then utilizes its semantics in the negotiation process.

Let us illustrate the approach by presenting two sample rules in a pseudocode notation that is independent of any implementation-level language.

POSTING-BUYER rule specifies that a *buyer* participant can post a proposal whenever there is an offer already posted by a *seller* participant. A proposal is called *valid* if it is syntactically well-formed and semantically compliant with the negotiation template. A proposal is called *posted* if it can be posted depending on the type of proposals that were previously posted by other participants (i.e. the negotiation reached a state that allows the proposal to be posted).

POSTING-BUYER
IF
There is a valid proposal Pr of a participant with role *buyer* $\wedge$
There is an active proposal of a participant with role *seller*
THEN
Proposal Pr is posted

IMPROVEMENT-BUYER rule specifies that a *buyer* participant must post a proposal with a price that overbids the currently highest bid by at least a given increment (that is a parameter of the auction and specified in the template). Note that a proposal that passed the improvement tests is called *active*.

IMPROVEMENT-BUYER
IF
Bid increment is Inc $\wedge$
Currently highest bid is B $\wedge$
Proposal Pr with price P was posted by a participant with role *buyer* $\wedge$
$P > B + Inc$
THEN
Proposal Pr is active

In our approach, constraints describing English auctions were encoded as a modularized set of JESS rules (see below). These rules were then used to initialize rule inference engines encapsulated by the negotiation host. Let us now discuss in more details how price negotiations were actually implemented.

4.2 The Negotiation Host—*Seller* agent

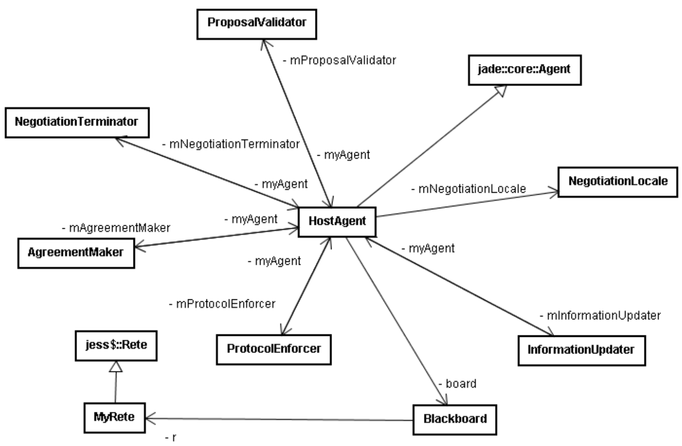

Figure 2: The class diagram showing the structure of the *Seller* agent

Note, first, that in our system the *negotiation host* originally defined in [Bartolini et al. 2005, Bartolini et al. 2002] became a *Seller* agent. It means that the *Seller* plays two roles, that of a negotiator and that of the negotiation manager. This decision may seem inappropriate in the context of e-commerce—to avoid any suspicion of inappropriate behavior, the host should be independent. However, this is not a problem since we are developing a model system only. Moreover, the proposed solution is much more resource conscious (e.g. messages do not have to be send to a separate host) and we believe it fits better with our e-commerce model—as the negotiation manager is part of the store infrastructure, it is natural for it to have a tight relation with the seller-negotiator. Finally, we will try to distinguish the two roles by referring to the *host* when considering management of price negotiations and *Seller* agent, when talking about specific actions of the *Seller*.

All agents in the system have been implemented as ordinary JADE agents (i.e. they extend the base class for all JADE agents – *jade.core.Agent*). The *host* encapsulates the negotiation controlling sub-agents that are implemented as ordinary Java classes (in what follows, we shall call them *host components*; see Figure 2): *Proposal Validator*, *Protocol Enforcer*, *Information Updater*, *Negotiation Terminator* and *Agreement Maker*. Therefore, each host component is an ordinary member object within the *host*, defining a *handle()* method that is activated whenever a given component must react to check the compliance of a submitted proposal with the category of rules it is responsible for.

Enforcement of a negotiation mechanism involves a complex activation pattern of negotiation rules. For the case of the English auction, the activation pattern is: i) the *Proposal Validator* is activated when a new proposal is submitted (see below); if the proposal is not valid then it is rejected; ii) if the proposal is valid then the *Proposal Enforcer* is activated (by the *host* – see below); if protocol enforcement rules are violated then the proposal is rejected; iii) if the proposal is compliant with protocol enforcement rules then it becomes *accepted* and the *Information Updater* is activated — its role is to update negotiation status and to inform negotiation participants. Separately, iv) the *Negotiation Terminator* is activated when a timing event occurs – e.g. if there was no activity for time longer than that specified by the "period of inactivity" parameter; furthermore, if negotiation termination is detected then the *Agreement Maker* is triggered— its role is to check if the agreement can be formed (e.g. if the *Seller* reservation price was satisfied), or if no actual agreement was reached and to inform interested parties about the situation.

To achieve its functions the *host* encapsulates also two member objects representing the *Negotiation locale* and the *Blackboard* (see Figure 2): *Negotiation Locale* and *Blackboard* "boxes". The *Negotiation Locale* object holds the *negotiation template* and the list of participant agents that were admitted to a given negotiation (obtained from the *Gatekeeper* agent—see above) and involves Java-based parts of the host. The *Blackboard* object encapsulates a JESS rule engine (class *jess.Rete*) that is initialized with negotiation rules. Whenever the category of negotiation rules is checked by one of the host components, the rule engine is activated (i.e. its *executeCommand()* method with the *(run)* command line given as an input parameter, is activated).

Handler methods of host components are activated by invoking the *action()* method that is present in the implementation of *host* behaviors. Each handler method delegates the call to the responsible component. In other words, the message received from the *BA* is stored in a local variable and then passed as a parameter among host components, as needed. Finally, that component activates the rule engine via the *myAgent* member object that points to the parent host agent; as described in the previous paragraph (see also Figure 2).

Note that each category of negotiation rules is part of a separate JESS module. For simplicity, we have decided to name JESS modules comprising a given category of JESS rules similarly to the host component that is responsible for enforcing them (e.g. rules POSTING-BUYER and IMPROVEMENT-BUYER are part of the *Protocol Enforcer* JESS module). As a result, rules of a given module are fired by activating *handle()* methods of appropriate components of the negotiation host.

Note also that for representing valid, posted and active proposals we utilize JESS templates; however, their definitions are not included below, only sample JESS rules that use them are shown.

Let us now show how the above presented rules POSTING-BUYER and IMPROVEMENT-BUYER look like when actually implemented in JESS.

```
(defrule Protocol-Enforcer::POSTING-BUYER
    ?fact <- (Protocol-Enforcer::valid-proposal
        (proposal-id ?id)
        (submitter ?s)
        (role Buyer)
        (auctionGoods ?a)
        (price ?p)
        (proposal-time ?t)
    )
    (Blackboard::active-proposal
        (proposal-id ?id2)
        (submitter ?s2)
        (role Seller)
        (auctionGoods ?a)
        (price ?p2)
        (proposal-time ?t2)
    )
=>
    (assert
        (Protocol-Enforcer::posted-proposal
            (proposal-id ?id)
            (submitter ?s)
            (role Buyer)
            (auctionGoods ?a)
            (price ?p)
            (proposal-time ?t)
        )
```

```
    )
    (retract ?fact)
)

(defrule Protocol-Enforcer::IMPROVEMENT-BUYER
    (Blackboard::negotiation
        (negotiation-id ?)
        (negotiation-type ?)
        (auctionGoods ?a)
        (seller-proposal ?)
        (bid-increment ?bid-inc)
        (termination-window ?)
        (currently-highest-bid ?h)
        (buyer ?)
    )
    ?fact <- (Protocol-Enforcer::posted-proposal
        (proposal-id ?id)
        (submitter ?s)
        (role ?r)
        (auctionGoods ?a)
        (price ?p)
        (proposal-time ?t)
    )
    (test (>= ?p (+ ?h ?bid-inc)))
=>
    (assert
        (Blackboard::active-proposal
            (proposal-id ?id)
            (submitter ?s)
            (role ?r)
            (auctionGoods ?a)
            (price ?p)
            (proposal-time ?t)
        )
    )
    (retract ?fact)
)
```

4.3 Controlling Rule Execution

Within each *host* we use a single JESS rule engine that is shared by all of its components, rather then implementing each host component as a separate rule engine. The advantage is that we now have a single rule engine per negotiation host rather than 6 engines as suggested in [Bartolini et al. 2005, Bartolini et al. 2002]. Furthermore, this means that in the case of m price negotiations taking place concurrently, we will utilize m instances of the JESS rule engine, instead of $6m$ instances necessary in [Bartolini et al. 2005, Bartolini et al. 2002].

While the solution of sharing a single JESS engine by all host components had the advantage of optimizing the implementation, it also posed some extra problems in controlling how rules are executed. We therefore decided to utilize JESS modules for partitioning rules and facts managed by the rule engine. There is one JESS module for storing the blackboard facts and a separate JESS module for storing rules used by each host component (see previous section for example of JESS rules residing in specific JESS module).

Blackboard facts are instances of JESS *deftemplate* statements and they represent: (1) the negotiation template; (2) the active proposal that was validated by the *Proposal Validator* and the *Proposal Enforcer* components; (3) Seller reservation price (not visible to other participants); (4) negotiation participants; (5) the negotiation agreement that is eventually generated at the end of a negotiation; (6) the information digest that is visible to negotiation participants; (7) the period of inactivity; and (8) 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 JESS module is controlled by the corresponding host component. Whenever the component handles a message, it activates the rules for enforcing an appropriate 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 in a specific focus module. In this way JESS facility for partitioning the rule-base into disjoint JESS modules proves very useful to efficiently control separate activation of each category of negotiation rules.

Note also that JADE 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 when accessing JESS facts stored by the shared JESS engine. This fact also supports our decision to utilize a single rule engine for each host.

4.4 Generic Negotiation Protocol and Agent Behaviors

The *generic negotiation protocol* states a minimal set of constraints on the sequences of messages exchanged between the host and participants during a negotiation. This generic protocol is further specialized using the negotiation rules, so we can conceptually consider that a negotiation mechanism is defined by the following "equation:" *negotiation mechanism* = *generic negotiation protocol* + *negotiation rules*.

According to [Bartolini et al. 2005, Bartolini et al. 2002], the negotiation host coordinates the interaction of negotiation participants by managing a negotiation process that consists of three phases: (1) admission, (2) proposal submission and (3) agreement formation. As noted above, in our system the admission phase has been removed from the negotiation process itself. Instead of a sub-agent (component) of a host that dealt with admission, we created a full-blown *Gatekeeper* agent that is responsible for this and other functions [Bădică et al. 2005c, Bădică et al. 2005e]). When a *Buyer* agent is accepted to the negotiation, it receives from the *Gatekeeper agent* the negotiation protocol and the template and, after receiving strategy from its *Client* agent, awaits start of negotiations. *Buyer* agents enter the phase of submitting proposals after they were dispatched to the negotiation. Specifically, *Buyer* agents that were granted admission are "simultaneously" released by the *Seller* that sends them a start message and they—possibly immediately—start submitting bids according to their private strategies [Ganzha et al. 2004].

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 negotiation state is updated by the host (for example, in an English auction, the currently highest bid is appropriately increased) and the remaining participants are notified accordingly with ACL INFORM messages about the new state of the negotiation.

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

Let us now illustrate the above processes by a sample message exchange that occurs during a negotiation when a *Buyer* agent submits a proposal that is accepted by the *Seller* agent.

Let us assume that *Buyer* agent B_{11} is bidding for product 1 and *Seller* agent S_1 is selling product 1. Note that in our implementation, name of the specific product does not need to be a part of the bidding process and *BA*s are released to a specific negotiation that involves a specific product (it is the *GA* that is responsible for assuring that only *BA*s interested in product 1 participate in negotiation that involves product 1). If *Buyer* agent B_{11} wants to post a bid of 20.0 units, its message will have the following form:

```
(PROPOSE
  :sender  ( agent-identifier
    :name B11_@Toshiba:1099/JADE
    :addresses (sequence http://Toshiba:7778/acc ) )
  :receiver  (set ( agent-identifier :name S1@Toshiba:1099/JADE ) )
  :content  "B11_@Toshiba:1099/JADE%Buyer%20.0"
  :language  PlainText  :ontology  Proposal )
```

The *negotiation host* first checks if this bid is valid and compliant with protocol enforcement rules (see above, section 4.1.2). Validity assumes checking the contents of the message—the `:content` field. In this particular case the contents indicates the name of the submitter—B_{11}, its role—*Buyer* and the bid value—20.0. Assuming that the value of the currently highest bid is 11.0 and that the minimum bid increment is 5.0, the bid is and accepted. The *Seller* agent S_1 response will look as follows:

```
(ACCEPT-PROPOSAL
  :sender  ( agent-identifier
    :name S1@Toshiba:1099/JADE
    :addresses (sequence http://Toshiba:7778/acc ) )
```

```
  :receiver  (set ( agent-identifier :name B11_@Toshiba:1099/JADE ) )
  :content  "S1%SingleItemEnglishAuction%1%5.0%20.0%90000%S1%B11_@Toshiba:1099/JADE"
  :language  PlainText  :ontology  AcceptUpdatedTemplate )
```

As a result, bid submitted by the *Buyer* agent B_{11} becomes the current highest bid. This information is posted within the *Blackboard* and send to all remaining *Buyer* agents (to B_{21} in this particular case) as an ACL INFORM message that looks like this:

```
(INFORM
  :sender  ( agent-identifier
    :name S1@Toshiba:1099/JADE
    :addresses (sequence http://Toshiba:7778/acc ))
  :receiver  (set ( agent-identifier :name B21_@Toshiba:1099/JADE ) )
  :content  "S1%SingleItemEnglishAuction%1%5.0%20.0%90000%S1%B11_@Toshiba:1099/JADE"
  :language  PlainText  :ontology  InformUpdatedTemplate )
```

4.5 Participants Strategy

Strategies of participant agents are defined in accordance with the negotiation mechanism (i.e. English auctions in this particular setting). Basically, the strategy defines if and when a participant should submit a proposal and what are the values of the proposal parameters depending on various factors including: current status of the negotiation, values of the negotiation parameters, participant agent goals, etc. Since our main current goal is to implement a complete agent system, with all defined functionalities, for the time being we opted for a very simple strategy: each *Buyer Agent* submits a first bid immediately after it is released to the negotiation and subsequently, whenever it gets a notification that another participant issued a proposal that was accepted by the *Seller*. The value of the bid is equal the sum of the currently highest bid and an increment value that is private to the participant (and higher than the minimal increment). Each participant has its own valuation of the negotiated product. If the value of the new considered bid exceeds this value then the proposal submission is canceled (given product became "too expensive" for a given *BA*). Note that in the case of an English auction there is no particular strategy for the *Seller Agent* as it plays only a passive role.

At present, agent strategies were implemented in Java as participant agent behaviors ([JADE]). In the future we will design the system in such a way that strategies will be flexible and dynamically loadable (possible in the rule-based form similar to, for instance, [Dumas et al 2002, Governatori et al. 2001], or as dynamically loadable modules, as suggested in [Ganzha et al. 2004]). This will provide us with the required flexibility needed to easily add and tune multiple strategies. Obviously, in practice, this form of strategy representation is going to be required only for more involved forms of price negotiations (where utilization of complicated strategies makes sense in the first place).

5 Experiments and Discussion

In this section we discuss two experiments performed with the above described sample implementation. First, we consider a simple experiment involving only a few agents with the goal of highlighting agent interactions. Second, we consider a simple "scalability" testing experiment involving multiple agents and multiple negotiations that run in parallel.

5.1 Experiment 1

In the first experiment we consider an e-shop that is selling 2 products, both products have a Seller reservation price of 50 and during an English auction 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 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}, while client C_2 is using buyers B_{21} and B_{22}. Some of the messages exchanged between agents in this experiment are shown in Figures 3 and 4 (note that only sellers and buyers are shown on that figure—other agents are not shown, as they do not play any active role in price negotiations). While figures 3 and 4 show messages exchanged between agents during negotiation, their content is not visible. Therefore we provide an explanation of message exchanges in table 2. The table header contains buyer names together with their reservation prices and bid increments.

In table 2 we can find some interesting details. First, when buyer B_{21} is about to make its first bid, buyer B_{11} had already submitted its first bid and that bid was accepted. 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 being 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 was not aware that the other participant B_{12} also posted a bid and this bid was accepted.

5.2 Experiment 2

In this experiment we considered $m = 10$ products and $n = 12$ clients seeking all of them, i.e. $\mathcal{P}_i = \mathcal{P}$ for all $1 \leq i \leq 10$. The auction parameters were the

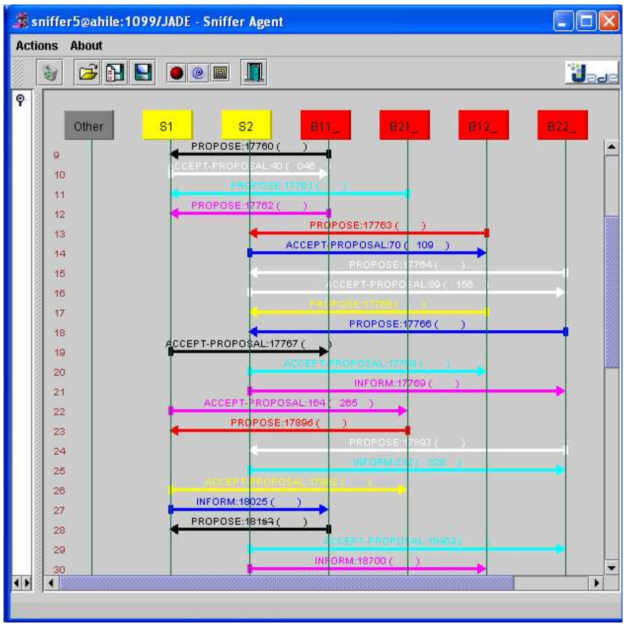

Figure 3: Negotiation stage – part 1

Table 2: Explanation of message exchanges shown in figures 3 and 4

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

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 this experiment 143 agents were created: 1 shop SA, 10 sellers SeA_i, $1 \leq i \leq 10$, 12 clients CA_i, $1 \leq i \leq 12$, and 120 buyers BA_{ik}, $1 \leq i \leq 12$, $1 \leq k \leq 10$, and 10 English auctions were run concurrently. One separate JESS rule engine was also created for each English auction (therefore a total

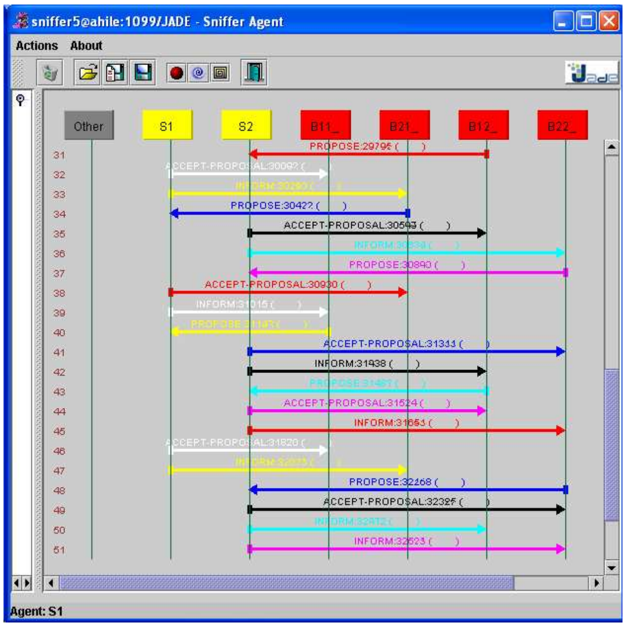

Figure 4: Negotiation stage – part 2

of 10 JESS rule engines were run in parallel on a single computer). The average number of messages exchanged per negotiation was approximately 100 and all auctions finished successfully. This means that a total of more than 1000 messages was exchanged during negotiations. While the total number of agents and messages is still small (for instance in comparison with these reported in [Chmiel et al. 2004]), this experiments indicates that the proposed approach has good potential for supporting experiments on large-scale.

Figure 5 shows messages exchanged between the seller SeA_1 and buyers BA_{i1}, $1 \leq i \leq 12$ that were captured with the help of the JADE sniffer agent.

6 Conclusions and Future Work

In this paper we discussed rule-based representations of mechanisms for automated negotiation in a model multi-agent e-commerce system. Our discussion was supplemented by providing design details and some initial experimental results obtained using our own implementation of a rule-based price automated negotiation framework. The results support the claim that rules are a feasible and scalable technology for approaching flexible automated negotiation in e-commerce.

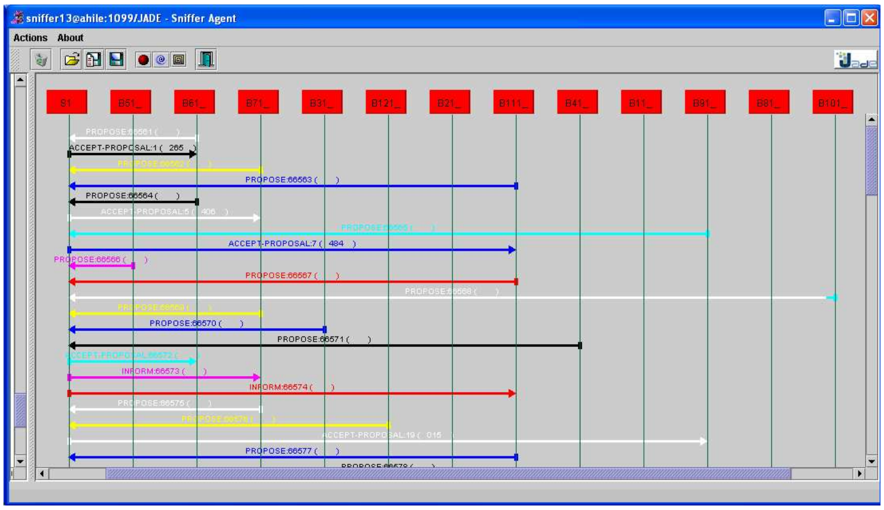

Figure 5: Negotiation of a seller with 12 buyers in an English auction

As future work we plan: (i) to complete re-integration of the rule-based framework into our re-designed agent-based model e-commerce system; (ii) to asses the generality of our implementation by extending it to include other price negotiation mechanisms; (iii) to conceptualize representation and ways of efficient implementation of multiple strategy modules; (iv) to investigate the applicability of rule-markup languages ([RuleML]) for devising an open rule-representation of negotiation mechanisms. We will report on our progress in subsequent publications.

Acknowledgement

Work of Maria Ganzha and Marcin Paprzycki has been partially sponsored by the Maria Curie IRG grant (project E–CAP).
Work of Costin Bădică has been partially supported by the CNCSIS 94/2005 grant (project HiperProc).

References

[AB3D] AB3D Home Page. See `http://ai.eecs.umich.edu/AB3D/`

[Bădică et al. 2006a] Bădică, C., Ganzha, M., Paprzycki, M.: "Rule-Based Automated Price Negotiation: an Overview and an Experiment"; Proc. of Int. Conf. on Artif. Intel. and Soft Comp., ICAISC, Zakopane, Poland. Lect. Notes in Artif. Intel. 4029, Springer, Berlin, (2006) 1050-1059.

[Bădică et al. 2006b] Bădică, C., Bădiţă, A., Ganzha, M., Iordache, A., Paprzycki, M: "Implementing rule-based mechanisms for agent-based price negotiations"; Proceedings of the 21^{st} Annual ACM Symposium on Applied Computing, SAC, Dijon, France. ACM Press, New York, NY, (April 2006) 96-100.

[Bădică et al. 2006c] Bădică, C., Ganzha, M., Gawinecki, M., Kobzdej, P., Paprzycki, M.: "Toward Trust Management in an Agent-Based E–Commerce System—Initial Considerations"; Proceedings of the MISSI 2006 Conference, Wroclaw University of Technlogy Press, Wroclaw, Poland, (2006) 225-236

[Bădică et al. 2005a] Bădică, C., Ganzha, M., Paprzycki, M., Pîrvănescu, A.: "Combining Rule-Based and Plug-in Components in Agents for Flexible Dynamic Negotiations"; M. Pěchouček, P. Petta, L.Z. Varga (eds.): Proc. of CEEMAS'05, Budapest, Hungary; Lect. Notes in Artif. Intel. 3690, Springer, Berlin (September 2005) 555-558.

[Bădică et al. 2005b] Bădică, C., Ganzha, M., Paprzycki, M., Pîrvănescu, A.: "Experimenting With a Multi-Agent E-Commerce Environment"; Proc, of PaCT'2005, Krasnoyarsk, Russia. Lect. Notes in Comp. Sci. 3606, Springer, Berlin (2005) 393-401.

[Bădică et al. 2005c] Bădică, C., Ganzha, M., Paprzycki, M.: "Mobile Agents in a Multi-Agent E-Commerce System"; Proc. of the 7^{th} SYNASC Conference, Timişoara, Romania. IEEE Computer Society Press, Los Alamitos, CA (2005), 207-214.

[Bădică et al. 2005d] Bădică, C., Bădiţă, A., Ganzha, M., Iordache, A., Paprzycki, M.: "Rule-Based Framework for Automated Negotiation: Initial Implementation"; A. Adi, S. Stoutenburg, S. Tabet (eds.): Proc. RuleML, Galway, Ireland. Lect. Notes in Comp. Sci. 3791, Springer Verlag (November 2005) 193-198.

[Bădică et al. 2005e] Bădică, C., Ganzha, M., Paprzycki, M.: "UML Models of Agents in a Multi-Agent E-Commerce System"; Proc. ICEBE, Beijing, China. IEEE Computer Society Press, Los Alamitos, CA, (2005) 56-61.

[Bădică et al. 2005f] Bădică, C., Ganzha, M., Paprzycki, M.: "Two Approaches to Code Mobility in an Agent-based E-commerce System"; C. Ardil (ed.): Enformatika, 7 (2005) 101-107.

[Bartolini et al. 2005] Bartolini, C., Preist, C., Jennings, N.R.: "A Software Framework for Automated Negotiation"; Proc. of SELMAS. Lect. Notes in Comp. Sci. 3390, Springer, Berlin (2005) 213-235.

[Bartolini et al. 2002] Bartolini, C., Preist, C., Jennings, N.R.: "Architecting for Reuse: A Software Framework for Automated Negotiation"; Proc. of AOSE: Int. Workshop on Agent-Oriented Software Engineering, Bologna, Italy. Lect. Notes in Comp. Sci. 2585, Springer, Berlin, (2002) 88-100.

[Benyoucef et al. 2002] Benyoucef, M., Alj, H., Levy, K., Keller, R.K.: "A Rule-Driven Approach for Defining the Behaviour of Negotiating Software Agents"; J.Plaice et al. (eds.): Proc. of DCW. Lect. Notes in Comp. Sci. 2468, Springer, Berlin (2002) 165-181.

[Chmiel et al. 2004] Chmiel, K., Tomiak, D., Gawinecki, M., Karczmarek, P., Szymczak, Paprzycki, M.: "Testing the Efficiency of JADE Agent Platform"; 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.

[Dumas et al 2002] Dumas, M., Governatori, G., ter Hofstede, A.H.M., Oaks, P.: "A Formal Approach to Negotiating Agents Development"; Electronic Commerce Research and Applications, 1, 2 (Summer), Elsevier Science (2002) 193-207.

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

[Ganzha et al. 2004] Ganzha, M., Paprzycki, M., Pîrvănescu, A., Bădică, C., Abraham, A.: "JADE-based Multi-Agent E-commerce Environment: Initial Implementation"; Analele Universităţii din Timişoara, Seria Matematică-Informatică, XLII (Fasc. special) (2004) 79-100.

[Governatori et al. 2001] Governatori, G., Dumas, M., ter Hofstede, A.H.M., and Oaks, P.: "A formal approach to protocols and strategies for (legal) negotiation"; Henry Prakken (ed.): Proc. of the 8^{th} Inter. Conf. on Artif. Intel. and Law, IAAIL, ACM Press, (2001) 168-177.

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

[Jennings et al. 2001] Jennings N. R. , Faratin P., Lomuscio A. R. , Parsons S., Sierra C. and Wooldridge M.: Automated Negotiation: Prospects, Methods and Challenges. In: *Int Journal of Group Decision and Negotiation*, 2001, Keynote Paper.

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

[Kowalczyk and al.2002] Kowalczyk, R., Ulieru, M., Unland, R.: Integrating Mobile and Intelligent Agents in Advanced E-commerce: A Survey. In: *Agent Technologies, Infrastructures, Tools, and Applications for E-Services, Proceedings NODe'2002 Agent-Related Workshops*, Erfurt, Germany. LNAI 2592, Springer-Verlag, pp.295-313, 2002.

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

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

[Lomuscio et al. 2002] Lomuscio, A.R., Wooldridge, M., Jennings, N.R.: "A classification scheme for negotiation in electronic commerce"; F. Dignum, C. Sierra (Eds.): Agent Mediated Electronic Commerce: The European AgentLink Perspective, Lect. Notes in Comp. Sci. 1991, Springer, Berlin (2002) 19-33.

[Reeves et al. 1999] Reeves, D.M., Grosof, B.N., Wellman, M.P., and Chan, H.Y.: "Toward a declarative language for negotiating executable contract"; AAAI-99 Workshop on Artif. Intel. in Electr. Comm. AIEC, (1999).

[Rolli and Eberhart 2005] Rolli, D., Eberhart, A.: "A Descriptive Auction Language. Electronic Markets"; Electronic Markets – The International Journal, (2005).

[Rolli et al. 2005] Rolli, D., Luckner, S., Gimpel, A.: "An Auction Reference Model for Describing and Running Auctions"; 7 Internationale Tagung Wirtschaftsinformatik, Bamberg, Germany, (2005).

[RuleML] RuleML Initiative. See `http://www.ruleml.org`.

[Skylogiannis et al. 2004] Skylogiannis, T., Antoniou, G., Bassiliades, N.: "A System for Automated Agent Negotiation with Defeasible Logic-Based Strategies – Preliminary Report"; Boley, H., Antoniou, G. (eds): Proc. RuleML'04, Hiroshima, Japan. Lect. Notes in Comp. Sci. 3323 Springer, Berlin (2004) 205-213.

[Tamma et al. 2005] Tamma, V., Phelps, S., Dickinson, I., Wooldridge, M.: "Ontologies for Supporting Negotiation in E-Commerce"; Engineering Applications of Artificial Intelligence, 18, Elsevier (2005) 223-238.

[Tamma et al. 2002] Tamma, V., Wooldridge, M., Dickinson, I: "An Ontology Based Approach to Automated Negotiation"; Proc. AMEC'02: Agent Mediated Electronic Commerce, Lect. Notes in Artif. Intel. 2531, Springer, Berlin, (2002) 219-237.

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

[Wurman et al. 1998] Wurman, P.R., Wellman, M.P., Walsh, W.E.: "The Michigan Internet AcutionBot: A Configuarable Auction Server for Human and Software Agents"; Proc. of the Second Int. Conf. on Autonomous Agents. Agents'98, Minneapolis, USA. ACM Press, New York, USA, (1998) 301-308.

[Wurman et al. 2001] Wurman, P.R., Wellman, M.P., Walsh, W.E.: "A Parameterization of the Auction Design Space"; Games and Economic Behavior, 35, 1/2, (2001) 271-303.

[Wurman et al. 2002] Wurman, P.R., Wellman, M.P., Walsh, W.E.: "Specifying Rules for Electronic Auctions"; AI Magazine, 23, 3, (2002) 15-23.