IADIS International Journal on WWW/Internet
Vol. 6, No. 1, pp. 50-60
ISSN: 1645 – 7641

OPTIMIZING BLACKBOARD IMPLEMENTATION OF AGENT-CONDUCTED AUCTIONS

Katarzyna Wasielewska *Department of Mathematics and Information Sciences, Warsaw University of Technology, Warsaw, Poland*

Maciej Gawinecki, Marcin Paprzycki, Maria Ganzha, Paweł Kobzdej *Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland*

ABSTRACT

Blackboard is a classic abstraction for exchanging information in distributed environments. Once an entity puts a data element on the blackboard, information about this fact is either delivered to subscribers (push) or interested entities check for new data (pull). Another dimension of information exchange is method of access to the blackboard, which can be either direct, or mediated by a "blackboard-manager." This article describes design and implementation of a blackboard used in to inform about status of agent-based price negotiations. Preliminary experimental results are also reported.

KEYWORDS

Blackboard, price negotiations, English auction, software agents, e-commerce.

1. INTRODUCTION

As described in [3], efficient implementation of autonomous price negotiations in agent-based e-commerce systems (such as the system under consideration within the E-commerce Agent Platform project [1]) is an important research problem. Recently, one of more interesting approaches to agent negotiations has been proposed in [4]. There, two roles have been distinguished: a negotiation *participant* (*buyer* or *seller*) and a negotiation *host*. The host is a "mediator/manager" that enforces adherence of the negotiation process to a specific protocol (e.g. a Dutch auction). In many price negotiation scenarios, one of functions of the host is to inform participants about the current status of the negotiation [7,8,9]. According to the considered abstraction [4], status of the negotiation is provided within a common multicast space – a form of a blackboard. For example, for the English auction, the blackboard includes such information as: the current active bid (the highest price), its submission time (and

possibly the id of participant that submitted it – which may or may not be a public information). In this note we consider the problem of how the information from the blackboard can be effectively delivered to negotiation participants.

2. APPROACHES TO BLACKBOARD DESIGN

There are two main aspects of possible organization of the blackboard. First, we have to consider the communication channel. Complying with FIPA [5] standards concerning development and implementation of agent systems requires exchange of ACL messages. Here, (a) the negotiation host can *push* information about changes to subscribed participants (active blackboard), or (b) interested participants can *pull* the information from the blackboard periodically by requesting it from the host (passive blackboard). Note that in the *push* approach, time is spent preparing, serializing and sending messages to *all* registered participants. In the *pull* approach, on the other hand, host may be flooded with auction status requests, thus slowing down its response to actual bids.

Disregarding FIPA standards allows agents to obtain information by directly invoking methods on shared object(s). Here, in the *push* approach, participants subscribe at the negotiation host by registering their listener objects, and the negotiation host uses a callback mechanism to inform them about changes in the negotiation status (content of the blackboard). In the *pull* approach, auction status may be accessed by participating agents by calling a specific method of the Shared Host (e.g. in predefined time intervals). Note that here we not only break FIPA rules of agent interactions (they do not communicate through message exchanges), but since shared objects can be used only in a local environment, negotiation participants have to reside within the same JVM (container) thus affecting overall scalability of the approach.

2.1 Tested Architectures

To investigate performance characteristics of the four possible approaches to blackboard design we have selected the English auction (EA). Note however that most of other standard auctions that require use of blackboard could be also used here (see also [8,9]). In the English auction sale of a a single item (or a collection of items treated as a "unit") is is negotiated by single seller with multiple buyers that bid "against each other." Negotiation starts when buyers are informed about the starting price. Next, they submit increasing bids until no further bidding takes place or is allowed. If the highest bid is larger than sellers' lowest acceptable price (*reservation price*), the highest bidder wins the negotiation. Throughout the auction the highest current bid is "visible" to all participants within the blackboard. There are two standard ways of ending an English auction:

(a) after a period of inactivity – no bid was submitted during a specified time interval,

(b) expire date is met – date after which auction ends.

Let us now look in more details into the way that the four proposed approaches to the design of the blackboard can be implemented in the case of an English auction.

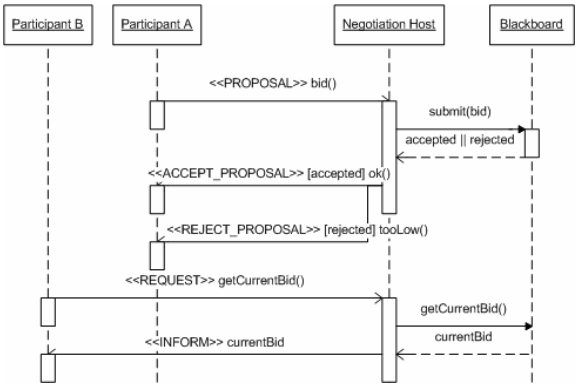

Figure 1. Pull approach sequence diagram

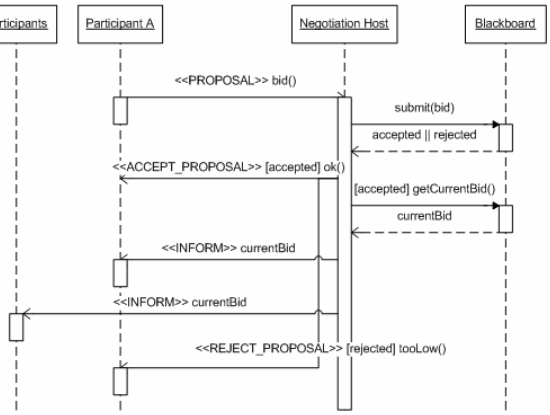

Figure 2. Push approach sequence diagram

2.1.1 *Host Agent* pushing information to participants

In Figures 1 and 2 we present sequence diagrams of message-based push and pull approaches utilized in the English auction. Here, the negotiation host is realized as a *Host Agent* (*HA*) and communicates via ACL messaging with negotiation participants. As an example we can see that in the *push* approach (Figure 1) Participant A sends an ACL_PROPOSE message with a bid to the *Host Agent*. The *HA* submits the bid to the blackboard object, which returns true when the bid becomes active (satisfies condition `bid > current_active_bid + minimal_increment`, as well as other necessary conditions [7,9]). If the bid is accepted, the *Host Agent* sends to Participant A an ACL_ACCEPT_PROPOSAL message, and in the follow-up step informs all participants (including Participant A) about the new (current) active bid (using an ACL_INFORM message). If the bid is rejected, the *HA* sends to Participant A an ACL_REJECT message (containing information about the reason of the rejection).

Let us now see, how this scenario has been translated into the internal architecture of the *Host Agent.* Since the remaining participating agents are relatively simple, we omit their detailed descriptions. In Figure 3 we present the activity diagram of the *Host Agent* in the push approach.

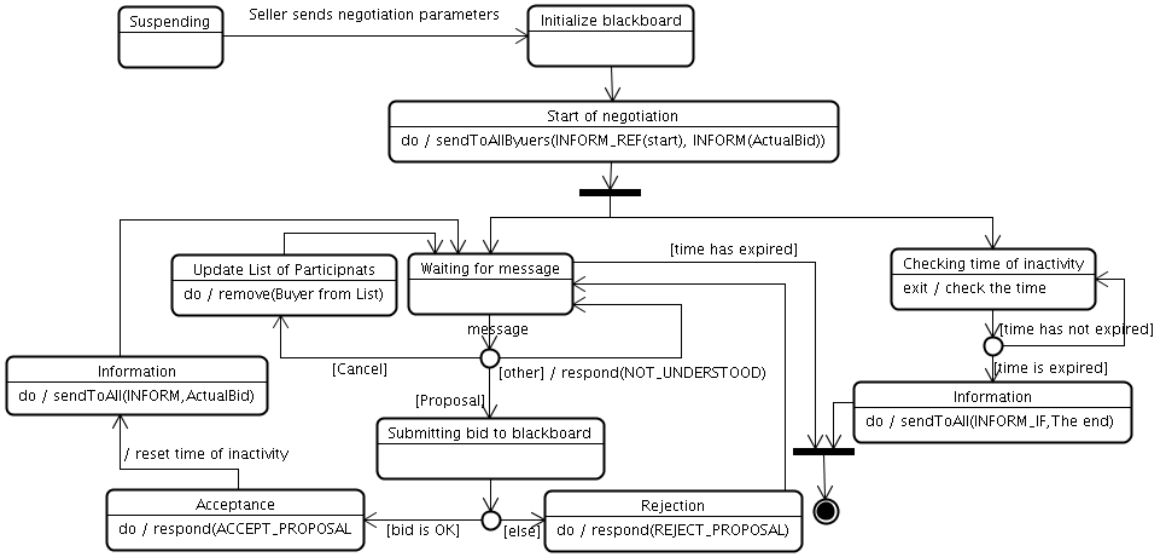

Figure 3. Host activity diagram in push approach

As we can observe, to start the negotiation the *Host Agent*, sends the *begin auction* signal and the first active bid (the Seller starting price) and waits for subsequent bids [9]. Depending on the scenario it runs the expiration date or the period of inactivity timer. *Host Agent* handles messages of two types. If the CANCEL message is being received, the *HA* removes the sender from the list of active auction participants (and will thus not send to it further messages). If the PROPOSAL is received (containing a new bid) it is submitted to the blackboard that checks whether it can be accepted. If accepted, appropriate acceptance messages (including information about the new highest bid) are sent to the bidder and to all buyers still actively participating in the auction.

2.1.2 Participants pulling information from the *Host Agent*

In Figure 4 we present activity diagram of the *Host Agent* in the message-based pull approach. To start, the *Host Agent* sends a signal-message (including starting price) to all buyers. At this point bidding begins, and the *HA* runs the auction and controls the expire date or the time of inactivity functions. Here, the *Host Agent* may receive two types of messages: (1) a REQUEST to which it responds with the currently active bid (an ACL_INFORM message), or (2) a PROPOSAL which contains a bid. If the bid is accepted the *HA* places it on the blackboard and informs buyer with an ACL_ACCEPT_PROPOSAL, otherwise it replies with an ACL_REJECT_PROPOSAL. When the auction ends, *Host Agent* informs participants about the result.

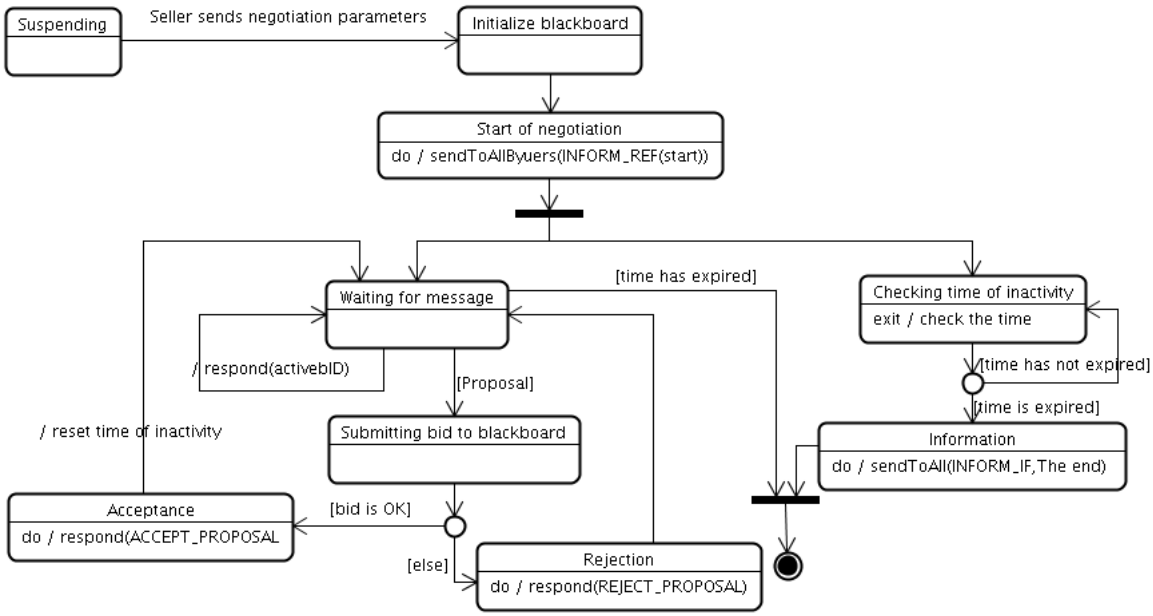

Figure 4. Host agent activity diagram in pull approach

2.1.3 Utilization of a *Shared Host*

Let us now consider solutions based on shared objects. Here, in the push approach we have an updater running on the shared object class that informs other agents whenever auction status has changed. This solution can be implemented using Java listeners and firing events, i.e. whenever new bid is accepted an event with information on the current active bid is fired and when the auction is finished an event with the winning bid is fired. On the listening side, buyer and seller agents implement listeners and handle the above mentioned events. Therefore, informing occurs only when auction status changes.

Shared object approach can also work in the the pull-based approach. Here, buyers (and, in the case of other negotiation mechanisms, e.g. Continuous double auction [8], the seller) regularly check status of the auction by calling methods on the *Shared Host* instance. In fact, this approach represents a form of busy waiting. Access to auction status and parameters is implemented with a static synchronized method called on host shared object. Here, buyers may attempt at checking status close to (known to them) end of inactivity period – to reduce number of method calls. However, this requires more advanced strategy for negotiations and thus is outside of scope of this initial work.

2.2 Assumptions of the Simulation

Several simplifying assumptions have been made while testing efficiency of the above described approaches to blackboard implementation. (1) We focus only on the negotiation process so the preparation and post-negotiation phases are omitted. (2) The simulation starts when the *Host Agent* starts the negotiation process. (3) The simulation ends when negotiation

terminates and its result is known. (4) We consider only one negotiation running at a time with one blackboard object devoted to it. (5) Negotiation host is a separate agent, or a singleton class (shared object). (6) Furthermore:

- auction starting price is always 0,
- buyer agent maximum prices have been generated randomly from the interval [0,1000],
- bid increments are defined to be relatively small in comparison with the maximum acceptable buyer prices so that each negotiation involved a relatively large number of submitted bids,
- buyer always increases price by a minimum bid increment,
- *Negotiation Host* (*Host Agent* or *Shared Host* object) knows the reservation (minimal accepted) price of seller, and therefore agreement is made on termination of auction only when the active bid is equal or greater then the reservation price,
- when value of active bid plus minimum bid increment is higher than buyers' maximum acceptable price then the buyer resigns,
- item to be sold is unimportant (product *P*),
- security related issues are ignored.

All participants (buyers and seller) are represented by software agents and communicate using ACL messages, content of which is defined by a simple auction ontology. Solution was implemented using JADE platform version 3.4 ([6]) and tested on a single machine with an Intel Pentium M processor 740, and 512MB DDR2 RAM. On this computer we have run a single container, where all agents resided. It should be noted that this latter fact is in line with the way agent negotiations have been functionalized in [1,7,8,9]. There, it was assumed that negotiations take place within a single "locale" controlled by the *Negotiation Host.* Obviously, this being the case we omit considerations related to events that can take place when negotiating agents communicate over the network (e.g. when their bids/status queries are delayed, or lost, due to the network traffic). However, we do not consider this to be an important limitation in the context of our current interest – blackboard implementation. Effectiveness of the implementation depends on the way that the blackboard can handle incoming bids and does not depend on the way that these messages have been delivered.

3. EXPERIMENTAL RESULTS

The aim of the simulations was initialize work toward establishing efficiency and scalability of above discussed approaches to blackboard implementation. The first series of experiments was devoted to comparison between message-based push and pull approaches (no shared objects) and the criterion was a number of messages sent during the negotiation. In Table 1 we depict what happened in a sample run involving the 10 buyers. In this experiment, the seller's reservation price was 100, the minimum bid increment was 5, the time of inactivity was 1 minute and the expiration date was "distant enough" for the auction to end because of the elapsed time of inactivity. In the table, we report the total number of messages generated by each participant. The value *Total,* represents sum of messages send by *all* buyers and the host.

Finally, the *Maximum price* reported for each buyer denotes the value of the active bid when a given participant (buyer) withdrew from further bidding.

Let us note that in our experiments an extremely simplistic negotiation strategies were in place. In the push approach every buyer, after receiving information about the active bid (message from the *Host* Agent), waited 5 seconds and then immediately attempted at submitting a new bid (if submitting another bid was still possible – if the current highest bid was below its maximum price), or immediately informed the *HA* that it resigns from the negotiation (if the current price was already too high). In the pull approach, each active participant was checking the current highest price after 5 seconds from receiving information about the bid status and if there was a new active bid it immediately submitted one of its own (if it was still possible), or resigned from the negotiation. If the current active bid was still the same, it continued checking the status of negotiation until the active bid changed, or until the negotiation ended. While admitting that this specific approach is overly simplistic, we have decided to utilize it in the initial assessment of performance characteristics of possible blackboard implementations. However, to see more clearly what is happening in the pull approach, in Table 1, we include two values: (a) all messages generated until the auction was over (due to the time of inactivity), and (b) messages generated until the last active bid was submitted.

Table 1. Comparison of number of messages in (message-based) pull and push approaches to blackboard implementation

AGENT	Pull approach	Push approach	Pull approach *	Maximum price
Seller	1	1	1	
Buyer1	389	146	389	730.98
Buyer 2	72469	143	416	831.44
Buyer3	113	49	113	240.54
Buyer4	316	122	316	606.35
Buyer5	332	125	332	637.42
Buyer6	148	62	148	309.05
Buyer7	274	111	274	550.44
Buyer8	56	24	56	117.01
Buyer9	313	62	313	597.55
Buyer10	407	80	407	781.53
Host	74829	2023	2757	
Total	**149647**	**2948**	**5522**	
Total number of submitted bids	**157**	**157**		

* - messages are counted up to the moment when last active bid was submitted

A number of observations can be made. It is clear (see the last column) that Buyer8 had the lowest maximum price and thus pulled from the negotiations first – it send only 24 bids in the push approach (we can assume that each message except the last one was a bid, while the last message was its resignation from negotiations). Since both pull and push experiments have been run for the same distribution of maximum prices and bid increments (we have generated the distribution once and then run experiments with all approaches to assure fairness of comparison) Buyer8 had also the smallest maximum price in the pull approach and this is represented by the fact that it generated only 56 messages (asking about the status of the blackboard and submitting bids of its own).

Particularly large number of messages in the pull approach was generated by the *Host Agent* and buyer that remained in the auction until the end (in the above example it was Buyer2). This is related to the fact that the negotiation strategy was set to be: bid immediately if the price is in your acceptable range. Therefore, all buyers except the one who "had the most money" finished their bidding "as fast as they could." At this stage, only Buyer2 remained interested in the status of negotiation and kept constantly checking the state of the blackboard (not knowing that no other agent is going to post a bid) until the end of the time of inactivity. Here one could assume, for instance that the Buyer, instead of constantly asking about the status of the blackboard, would ask only near the end of the known period of inactivity. Such strategy would clearly result in: (a) possible congestion at times near the end of time of inactivity, but also in (b) a smaller number of messages generated in the pull approach. Therefore, we decided to view messages send by Buyer2 in the final stages of the auction as a special effect of our experimental setup and report also situation when these messages are not taken into account. This being the case, when comparing the push and the pull approaches represented in Table 1, we should concentrate our attention on the number of messages send by agents other than Buyer2. When such comparison is made, number of messages send in the pull approach is still substantially larger than in the push approach. The difference is between approximately 2 times as many messages (e.g. Buyer6 and Buyer8), up to almost 5 times as many (e.g. Buyer10).

In Table 2 we represent total number of messages send in the system for an increasing number of buyer agents. Results presented there are an average from multiple runs with seller's reserved price 100, minimum bid increment 50, time of inactivity 1 min, and expiration time set to make the auction end because of the time of inactivity. In addition to comparing message-based pull and push approaches, we consider what happens when in the push approach all participating agents want to be informed about auction result. In this case, after the negotiation is over, the *Host Agent* sends message with auction result to all agents that participated in the from the start.

Table 2. Comparison of number of messages for different number of buyers

No of buyers	Pull approach	Push approach	Push approach with send result to all
10	129700	325	336
20	135445	740	767
30	136782	955	1175
40	132557	1258	1342
50	138862	1578	1790
60	105289	1949	2202

We see that the difference in number of messages sent in both versions of the push approach is negligible. This is because additional messages are sent only when the negotiation ended and their number is directly proportional to the number of auction participants. Therefore, taking into account the total number of messages sent in the push approach, this "message overhead" is not significant (about 5% of the total number of messages).

We may observe that the number of messages sent in the push approach in both versions increases linearly with number of buyers. In the pull approach, on the other hand, it remains independent of the number of agents, as most messages are exchanged when, for all practical purposes, the negotiation is actually over (messages are being exchanged only by the winning buyer and the host – a while other buyers do not participate any longer).

In the next series of experiments we have compared execution times of the two approaches based on shared objects. Note that here the message count was of the same order as in the case of message-based approaches (the only change was in the way that the information was propagated). Surprisingly we have found that times of both approaches are practically equal the difference was less than 2%). Therefore we have decided to compare execution times of message and shared-object- based approaches to blackboard implementation, using auction completion time as the performance measure. Obtained results are depicted in Figure 7. We can see the average time of executing an auction using the message-based pull (2 version) and push approaches and well as the shared-object push approach. To avoid clogging the picture we have decided to skip the timing results obtained with the shared-object-pull approach.

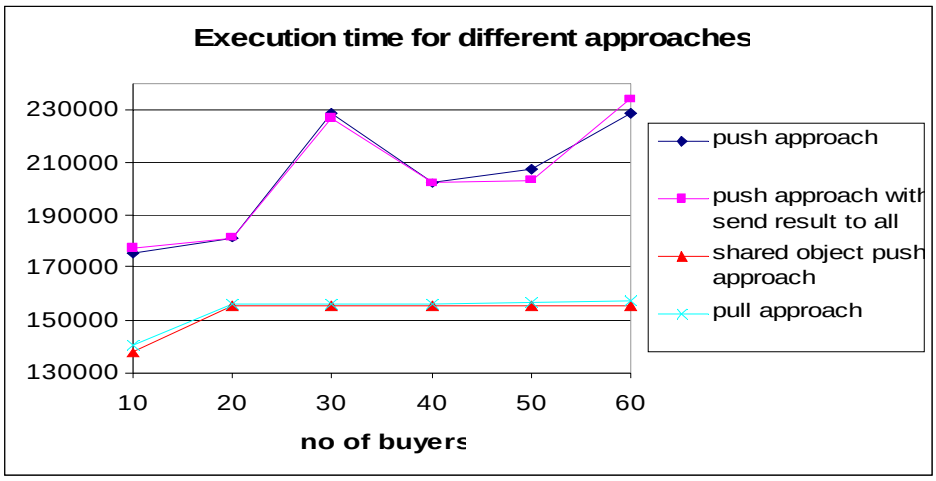

Figure 7. Comparison of execution time for different approaches

It can be observed that in terms of the execution time both versions of the message-based push are very similar. Furthermore, the timing function is linear and slowly increasing. However, we do not have an explanation why a sudden jump observed at 30 buyers. Surprisingly, even though the pull approach generates substantially more messages, the total time for completing a negotiation is shorter. We have to look into this effect in more detail. In terms of execution time the most efficient solution is the shared object approach. It is considerable faster than both versions of push with ACL messaging. Furthermore, for up to 60 buyers, the total time for completing an auction remains flat. Interestingly enough, its performance is extremely similar to that of the message-based pull approach. We do not have an immediate explanation of this fact and have to investigate it further.

3. CONCLUDING REMARKS

In this note we have developed and experimented with four different approaches of the blackboard component used as multicast space in agent-based electronic negotiations. The blackboard here is responsible for delivering (in either passive or active way) information about current status of the auction. We have also considered the way in which the message is delivered to participants (through ACL messaging and through sharing objects and directly invoking messages on them). While come of the results are somewhat surprising and need further investigation we have found out that the shared object approach is more efficient than messaging. Unfortunately, while the shared object approach is not only more efficient and easier to implement, it is not consistent with FIPA standards. In the near future we plan to investigate further the above mentioned surprising results as well as focus our attention of more involved strategies that can be utilized in the pull approach. For instance, it is possible to modify the frequency of information pulling in such a way to still keep the buyer competitive, while reducing the total number of send messages. Note, however, that this may result in the

auction taking longer time as bidding may occur, for instance, only near the end of the inactivity period, rather than almost immediately (as in the approach discussed above). Finally, more research must be conveyed to analyze the aspect of fairness of the blackboard and its utilization.

REFERENCES

Bădică, C., Ganzha, and Paprzycki, M., 2006. Rule-Based Automated Price Negotiation: an Overview and an Experiment. In: L. Rutkowski, et. al. (eds.), *Proceedings of the ICAISC Conference.* LNCS 4029, Springer, Berlin, 2006, pp. 1050-1059.

Guttman, R.H., Moukas, A.G. and Maes, P., 1998. Agent-Mediated Electronic Commerce: A Survey. In *The Knowledge Eng. Rev.,* vol. 13, no. 2, pp. 147-159.

Klaune S., Kurbel K. and Loutchko, I., 2001. Automated Negotiation on Agent-Based e-Marketplaces: An Overview. In *Proceedings of the 14th Bled Electronic Commerce Conference*. Bled, Slovenia, pp. 508-519.

Bartolini, C., Priest C. and Jennings N. R., 2002. Architecting for reuse: A software framework for automated negotiation. In *Proceedings of the Third International Workshop on Agent-Oriented Software Engineering*, Bologna, Italy, pp. 87-98.

FIPA Specifications Grouped by Subject, http://www.fipa.org/repository/bysubject.html

Jade Agent DEvelopment Framework, http://jade.tilab.com

Bădică,C., Ganzha, M., and Paprzycki, M., Implementing Rule-Based Automated Price Negotiation in an Agent System. *Journal of Universal Computer Science*, Vol. 13, No. 2, 2007, pp. 244-266

Gawinecki, M., Kobzdej, P., Ganzha, M., Paprzycki, M., Introducing interaction-based auctions into a model agent-based e-commerce system — preliminary considerations. In: R. do Nascimento et.al. (eds.) Proceedings of the EATIS Conference, ACM Digital Library, ACM Press, New York, NY, 2007, 8 pages

Paprzycki, M., Ganzha, M., Adapting Price Negotiations to an E-commerce System Scenario. In: K. Saeed et.al. (eds.), Proceedings of the CISIM Conference, IEEE CS Press, Los Alamitos, CA, 2007, pp. 380-386