Introducing interaction-based auctions into a model agent-based e-commerce system—preliminary considerations

Maciej Gawinecki, Paweł Kobzdej
System Research Institute Polish Academy of Sciences
(maciej.gawinecki,pawel.kobzdej)@ibspan.waw.pl

Maria Ganzha
Department of Administration, Elbląg University of Humanities and Economy
ganzha@euh-e.edu.pl

Marcin Paprzycki
Computer Science Institute, SWPS and Systems Research Institute, Polish Academy of Science
mpaprzycki@swps.edu.pl

Abstract

In our work we have proposed an agent-based model e-commerce system. In this system buyer agents negotiate prices with seller agents. Thus far our attention was devoted "one sided" price negotiation mechanisms, i.e. mechanisms in which one side is active, while the other side is passive (possibly other than initiating the process). Examples of such negotiations are, among others, English and Dutch Auctions, as well as a number of sealed-bid auctions. Here, we focus our attention on auctions in which both sides take active part. We discuss how these auctions can be incorporated into our system and present their UML formalizations.

1. Introduction

Currently, we are developing a complete model agent-based e-commerce system ([2]). In this system Buyer Agents, representing User-Clients participate in price negotiations with Seller Agents representing User-Merchants. While agents representing User-Clients attempt at selecting the best offer, Seller Agents attempts at maximizing profits resulting from product sales. Within the project we implemented a price negotiation subsystem based on the framework proposed in [7]. However, our work on price negotiations has been pursued separately from the design of the system itself. The main reason was that the system has been completely re-designed after the initial implementation and thus work on its core remained behind other workstreams. It is only now, when a number of specific decisions related to the system itself have been made, when we can re-evaluate our approach to price negotiations.

Let us note that we follow a classic understanding of price negotiations as a process by which agents come to a mutually acceptable agreement on a price ([14]). Furthermore, we distinguish *negotiation mechanisms* that define "rules of encounter" between negotiation participants; and *negotiation strategies* that specify behavior of participants aiming at achieving a desired outcome (typically, to maximize their "gains").

In this context, *auctions* are one of the best-understood forms of automated negotiations ([19]) and, recently, parameterizations of the auction design space with the goal of facilitating negotiations in multi-agent systems have been observed ([7, 8, 9, 13, 19, 14]). Among them, the Foundation for Intelligent Physical Agents—FIPA ([11]), defined a set of standard specifications of negotiation protocols including English and Dutch auctions. Authors of [7] analyzed the existing approaches to formalizing negotiations (including FIPA protocols) and argued that they do not provide enough structure for the development of truly portable systems, and outlined a complete framework comprising: (1) negotiation infrastructure: defining roles of negotiation participants and of a host, (2) a generic negotiation protocol: defining the 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, and (3) set of declarative rules used for enforcing the negotiation mechanism. Specifically, rules have been organized into a taxonomy matching the identified phases of negotiations: rules for 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. Finally, they introduced a *negotiation template* that contains parameters specific to a given form of price negotiations.

With a large number of existing results concerning agent systems appearing in autonomous price negotiations, let us delineate what makes our approach unique (in the context of this paper; for more details see [2]).

1. In most, if not all, papers only a "single price negotiation" is considered. Specifically, negotiation of a single item (e.g. using an English Auction) or a single collection of items (e.g. using a multi-item Dutch Auction) is contemplated. Once the negotiation is over, agents (agent system) that participated in it complete(s) its(their) work. We are interested in a different (more realistic) scenario when a number of items of a given product are placed for sale one after another. Taking into account that this situation closely resembles what happens in any Internet store, it is interesting that it is practically omitted from research considerations.

2. Since a sequence of items is sold we decided to organize price negotiations as a "discrete process." Here, except of a specific case of fixed prices, buyer agents are "collected" and released together in a group to participate in a price negotiation. While the negotiation takes place buyer(s) communicate only with seller(s) (they can be envisioned as being placed in a closed negotiation room). At the same time the next group of buyer agents is collected (as they arrive) and will participate in the next negotiation.

3. Since multiple subsequent auctions (of the same product) take place, we can go beyond one more "limitation" of known to us agent systems. While negotiations may involve complicated mechanisms, e.g. mixed auctions ([17]), since only a single item is sold, only a single mechanism is utilized. In our case, price negotiation mechanisms can be dynamically adjusted. For instance, first 12 items may be sold using a Dutch Auction, while the next 52 use second-price sealed bid auction.

The remaining part of this paper is devoted to the effects that these features of our system have on price negotiations. There is also an important difference between this and our earlier work, where we have been conceptualizing only "one-sided" auctions; i.e. auction in which one side is active, while the other is passive (except possibly initializing of the process). Note that, for instance, in an English Auction, the Seller Agent remains passive while Buyer Agents submit a sequence of bids. This time we focus our attention on auctions that involve active Buyer and Seller agents. To provide the context, we start with a brief description of agent system under development. We follow with conceptualization of two types of one-to-one auctions: Iterative Bargaining, and Double Dutch Auction, as well as two variants of a Double Auction: the Two-Round version and the Continuous Double Auction. Each of these auctions is formalized with the help of an UML Activity Diagram.

2. System Description

Our system acts as a distributed marketplace in which e-shops are represented to the outside world by Gatekeeper and Seller agents, while e-buyers are represented by Client and Buyer agents. In Figure 1 we present Use Case diagram of the complete system. Outside of its bounds we can see a *User-Client* who will attempt at buying products and a *User-Seller* who tries to sell products in her e-store.

Let us now very briefly summarize the most important agents appearing in the system and their functionalities (for a complete discussion of the system see [2, 4, 5, 6, 10]). *User-Client* is represented by the *Client Agent* (*CA*). The *CA* is assumed to be fully autonomous and as soon as the decision to purchase product *P* is communicated by the *User-Client*, it will work until either *P* is purchased or purchase is abandoned (e.g. because prices are too high). The *CA* communicates with the *Client Information Center* (*CIC*) agent which contains complete information which e-stores sell which products. For each store that sells the requested product, the *CA* delegates a single *Buyer Agent* (*BA*) to participate in price negotiations and if successful, possibly attempt at making a purchase (successful price negotiations result in a product reservation for a specific time period). Since multiple *BA*s representing the same *CA* can win price negotiations and report to the *CA*, it is the *CA* that makes the decision if either of available offers is good enough to make a purchase. *Buyer Agents* can participate in negotiations only if the *Gatekeeper Agent* (*GA*) admits them (if they are trusted; e.g. *BA* that wins multiple price negotiations but does not make purchase may be barred from subsequent negotiations). The *GA* represents a

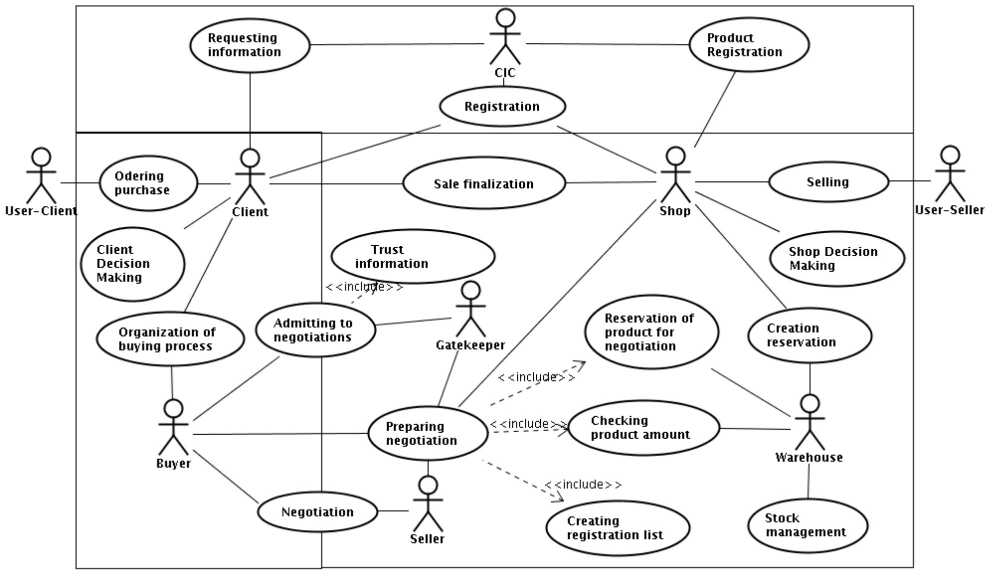

Figure 1. Use Case diagram of the proposed system

given 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 set of *Seller Agents* (*SeA*) that negotiate price with incoming *BA*s.

3. Negotiation organization—general considerations

In our work we continue to follow the proposal put forward in [7]. However, the way that our system and processes that take place in it are structured allows us to modify and simplify their proposals. In [7] three negotiation roles have been proposed: *buyer*(s), *seller*(s) and the *negotiation host* (named *host* thereafter). While in the description and in the way that the negotiations were actually coded, as well as in our earlier work the distinction between the *host* and the *seller* was somewhat fuzzy, here we make it clear. The *host* provides the *infrastructure* where the negotiations take place and are managed. The *host* can be facilitated locally by the e-store, or can be located within a certified authority (to assure that the negotiation process is not being tainted by the *host*—for instance, in the case of a multi-item Dutch Auction it is enough to "pretend" that two messages arrived in a reverse order to favor a certain participant over another). It is the latter setup that is used in the case of many-to-many versions of the Double Auction. Furthermore, we assume that the *host* is a generic infrastructure that can handle any form of price negotiations. To start the negotiations process the *SeA* sends to the *host* the list of participants as well as the negotiation template, containing all necessary parameters. This information is used to initialize an *instance of a host*, to handle a specific negotiation. Upon completion of the initialization process the *host* informs the *SeA* that it is ready to support negotiations. In this way, in turn, the *SeA* knows that the *host* initialization process has been successfully completed. As a result, in all auctions it will be the *SeA* that will make the "first move." Interestingly, this turns out to be a purely technical decision that does not affect on the negotiation process.

Since *BA*s are admitted to negotiations in groups, we can assume that the product to be auctioned is established beforehand. It is the *GA* that learns which product a given *BA* would like to buy and makes sure that only these *BA*s that are interested in that product are released to the same "negotiation room." This means that it is not necessary to check the "product name" during the negotiation process. It could be possible to assume that it is not necessary to check if a given agent was allowed to send a proposal to a given negotiation. Observe that negotiations are handled by one of the pool of *SeA*s (unknown in advance) and unknown in advance instance of the *host*. This being the case, only *BA*s that are admitted to a given negotiation (and the *SeA*) know where to send their bids to. However, in the case of some negotiations (e.g. multi-item Dutch Auction), winning *BA*s are not

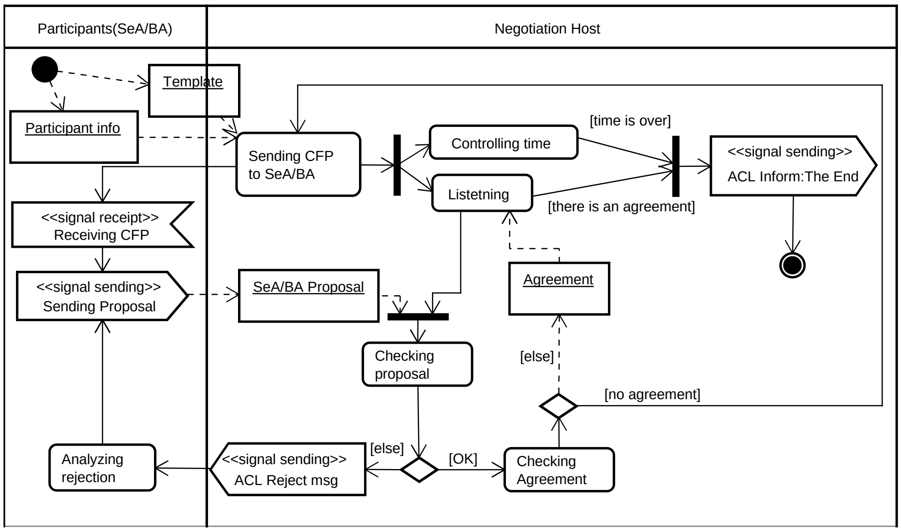

Figure 2. Activity diagram of an Iterative Bargaining Auction

allowed to bid again. This being the case, for the sake of uniformity, we have decided that validity of sending a proposal to a given negotiation by a given participant has to be checked in all auctions. Unfortunately, due to the lack of space we had to omit further details related to rule-based mechanisms and focus our attention on the UML-introduced structure of negotiations.

4. Negotiation organization—one-to-one negotiations

Let us start our considerations from two cases of one-to-one interactive negotiations. First, we focus our attention on Iterative Bargaining. While for this, as for most other auctions, there exist multiple variants, we have decided to utilize our personal experience in the market. After being initialized the *host* send *Call For Proposals* to *Seller Agent* (see above for the argument why the *SeA* is the first to submit a bid). After the *SeA* submits the first proposal, it is checked by the *host* (see [5], where we have described the way in which the *host* is organized and works). If the proposal has been successfully validated, the *host* posts it on the *Blackboard* (the place where information available to participants is posted [5]) and sends a CFP to the *BA*. This process continues until (1) time is over—no agreement was reached, or (2) *SeA* and *BA* proposals "meet" (one of participants issues a bid that matches the information posted on the Blackboard). Note that our version of iterative bargaining requires that both sides submit their bids in turns. Thus it is impossible for the *BA* to submit a series of bids to the "silent *SeA*." While this assumption may seem somewhat limiting, we believe that the general spirit of Iterative Bargaining is kept. In Figure 2 we present the UML Action Diagram of our version of iterative bargaining. Note that roles of *Buyer* and *Seller* are symmetric and thus they can be subsumed under a single role (*Participants*).

The second one-to-one negotiation that we have decided to model is the, so called, Double Dutch Auction, which has been proposed in [15]. This auction is relatively counterintuitive and in its basic version works like this (based on [1]): a *buyer* price clock starts ticking at a very high price and continues downward. At some point the *buyer* stops the clock and bids on the unit at a favorable price. At this point a *seller* clock starts upward from a very low price and continues to ascend until stopped by a *seller* (who offers product at that price). Then the *buyer* clock resumes in a downward direction, followed by the *seller* clock moving upward. Trading is over when the two prices cross (purchase is made at the crossover point). In figure 3 we present the UML Activity Diagram of the Double Dutch Auction.

As previously, the *SeA* sends information to the *host*, which becomes initialized. Interestingly, as in the case of Iterative Bargaining, actions of both participants are symmetric and they can be represented as a single entity (*Participants*). For the same reasons as above, it will be the *SeA* that will receive the first CFP and watch its clock move. Let us note that the negotiation ends in a

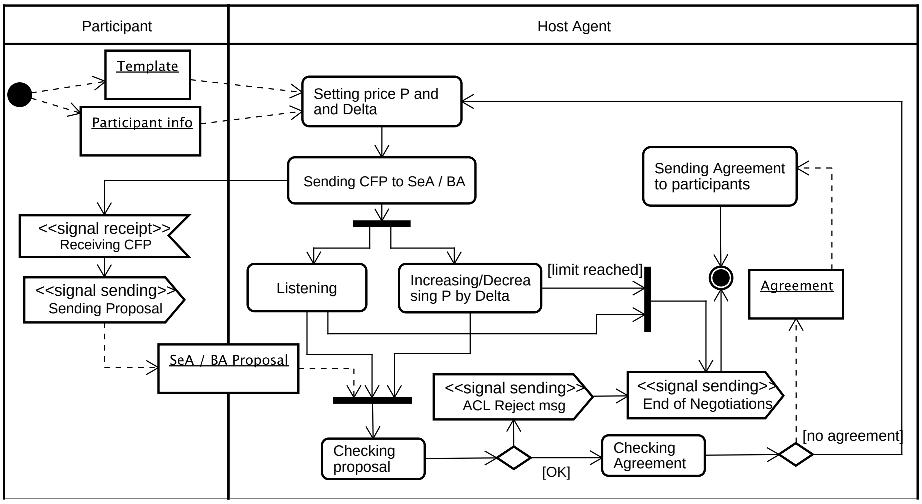

Figure 3. Activity diagram of a Double Dutch Auction

failure if in the first round of bidding the acceptable sale price is above the acceptable purchase price. Since the "price clock" moves only in one direction, this situation cannot be reversed and the negotiation ends. The only other way that the negotiation can fail is if either participant issues as invalid bid. We have decided that the price negotiation mechanism is so delicate, that there is no reasonable way to "remedy" such situation and the best way is to end the process. In all remaining cases negotiation ends with a success when sale and purchase prices "cross" due to the action of the "price clock." This situation is recognized by the *host* that accordingly informs both participants.

5. Negotiation organization—many-to-many negotiations

In the previous section we have described the Double Dutch Auction. This form of price negotiations belongs to a family of Double Auctions. Let us now describe two more members of this family: the Two-Round Auction and the Continuous Auction. Let us note that for the first time we will deal with many-to-many auctions and this fact has important consequences for our system. While double auctions are a well known price negotiation mechanism [1] their introduction into our system requires further considerations. Let us observe that all auctions that involve a representative of a single store can be organized within this store. As the clients arrive that store organizes price negotiations for them. The situation changes when representatives of multiple stores are to interact with multiple buyers. Here, it is unreasonable to expect that a given store would host such negotiations (why to invite competition into "their own" price negotiations?). What helps us is the assumption of a complete separation of the *host* from *participants*. This allows us to envision that there would exist "auction e-houses" that would provide infrastructure for price negotiations. From the technical standpoint an *auction house* would consist of a *Gatekeeper* responsible for admitting *buyers* and *sellers* and organizing the pre-negotiation processes and a *host* agent that will be instantiated to manage individual negotiations. Since such a situation occurs often in real-life (consider eBay as an example of an e-auctioning infrastructure), this assumption seems quite reasonable.

However, we have also to consider the question of trust management (see [4]). Thus far in our system it was assumed that each store and each client utilizes its own knowledge to deal with incoming clients and stores, respectively (we have used only notion of trust, while omitting the concept of reputation). The question that arises is: how will trust management be handled if a neutral *auction house* facilitates price negotiations. Observe that in such a situation incoming agents have zero knowledge who they are negotiating with (agents cannot be rejected due to their past behaviors). There are multiple ways of dealing with such a situation. One of them would be that the *auction house* would also manage trust. In a similar way that the eBay / Allegro (a Polish e-auctioning platform) clients can rank each-other, we could envision that the *auction house* would provide

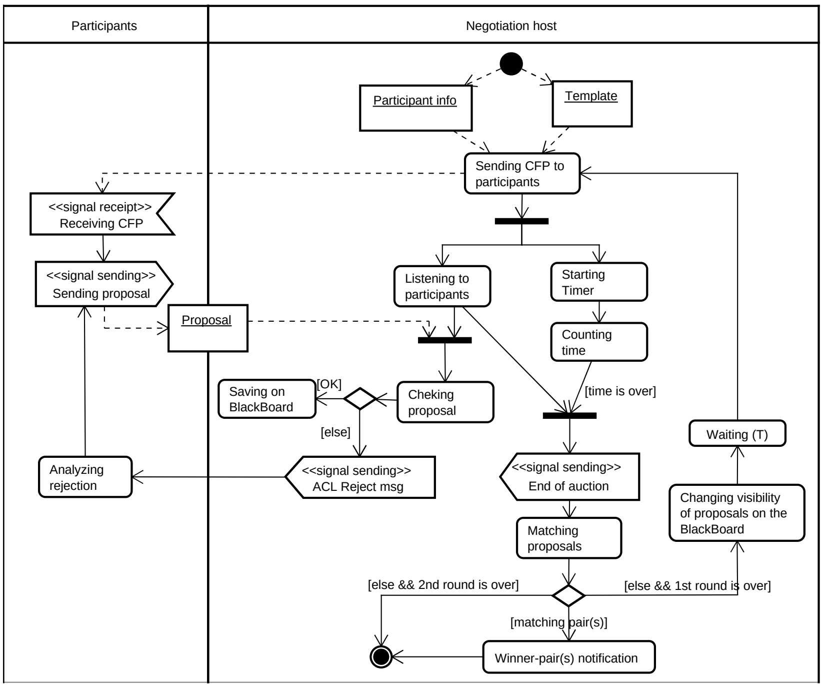

Figure 4. Activity Diagram of a Double Auction

a similar service (based on reputation). However, we have decided, for the time being, to reject this solution. First, this would mean that, for all practical purposes, we would have to abandon the framework proposed in [7] and this is not a step that we would like to make. Second, while trust / reputation management mechanisms can be build into the auction house, making such a move would not introduce anything particularly novel over and above the material already discussed in [4]. Instead we have decided to simplify the proposed solution. The *auction house* will not take part in any form of trust management. Its only role will be to fairly / neutrally facilitate price negotiations. Specifically, it will gather incoming *buyers* and *sellers* and release them into price negotiations. After negotiations are over, it will announce to the winning pair(s) that they have been matched. It is then up to the matched pairs to decide if they are interested in consummating the deal with the partner. Specifically, after being informed that they have been matched with the specific agent, both the *SeA* and the *BA* will utilize their trust information to evaluate “the other.” The *BA* will use the trust information to see if the shop that it is to make a purchase from is trustworthy, while the *SeA* will access the trust information to establish if a given *BA* is trustworthy enough to deal with it, and if this is the case, then what should be the length of reservation (see [4] for more details).

Let us now define the Two-Round version of the Double Auction. Here we have several *SeA*s and several *BA*s that submit their proposals (sealed), to the *host*. They can do it during some specific time. After the time is over, the *host* checks submitted bids and notifies participants whose bids match and the auction completes. Otherwise (if there were no matching pairs) the *host*—changes the status of invisible proposals to visible and starts second round. In this way all participants can analyze available data and adjust their bids. Note that bids that were posted reflect the vision of the market-value of sold commodity as it is represented by participants of negotiations. This approach is based on results presented in [18], where it was shown that the second round is characterized by a substantially higher number of matches. Note that this mechanism can be used not only to facilitate single-item, but also multiple-item auctions. For a single-item auction, a bid indicates a desire to buy or to sell at a specific price. In a multi-item auction, bid specifies the number of items and the price per item. We assume that if *BA* submits a proposal where it specifies 3 items of some product at $5 per item and there is a *SeA* that submitted a bid for 6 items

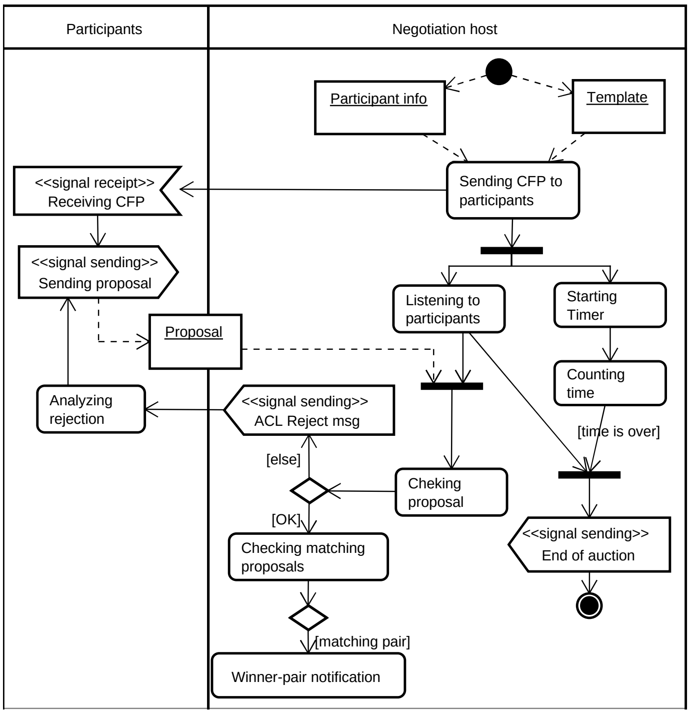

Figure 5. Activity Diagram of a Continuous Double Auction

of the same product at $5 per item, these proposals are matching and three items will be sold. Taking all of this into consideration, in Figure 4 we present the UML Action Diagram of the Two-Round Double Auction.

Process of making visible, within the Blackboard, proposals from the first round (the *Change Visibility Rules* box, is followed by some delay (T)) before the second round of proposals is invited. If the second round, no matches occurred the auction is over. Let us stress, however, that proposed formalization allows for any number of rounds to take place. In some versions of Double Auction it is assumed that when no exact matches were found, offers that are "close enough" will be matched and the price differential will be used to pay for the operation of the auction house (see [16, 15]). Obviously, our formalization is capable of supporting such an approach. This will be taken care off by the *host* that will act on the basis of the received template.

Let us now move to the next version of the Double Auction, the Continuous Double Auction (CDA), which is actually used by stock exchanges, e.g. the NASDAQ and the NYSE (see [1]). The main difference between the Two-Round Double Auction and its Continuous variant is that bids are being matched as soon as they are submitted. Here, *BA*s and *SeA*s submit their bids over a certain period of time. When there is a match, winners are notified and the information posted on the Blackboard. We have decided to utilize the version of the CDA proposed in [20]. Here each new bid from a given agent replaces the previous bid. In this way if a bid was unsuccessful after some time the participant may send an updated one hoping for a match. In Figure 5 we present an Action Diagram of Continuous Double Auction.

The final issue that has to be resolved is: who will organize processes that take place before and after negotiations. As noted above, it will be the *auction house GA* that will manage flow of incoming *participants*. At this stage, we will omit the question, how will the *auction house* decide which products are worthy trading. The *auction house* simply registers with the *CIC* products that it would like to auction. This offer is included in the list that the *CA* receives from the *CIC* as well as distributed to all pertinent *SA*s that are registered with the *CIC*. Upon arrival, *BA*s and *SeA*s meet the *auction house GA* and from there the process resembles that described above and in [2].

6. Concluding remarks

In this paper we have considered the question, how interactive auctions (involving active participation of *Buyers* and *Sellers*) can be introduced into our model agent-based e-commerce system. We have specified a number of features of our system that influence the way the negotiations have to be organized. We have followed by UML based formalizations of four forms of negotiations. Two one-to-one auctions: Iterative Bargaining and Double Dutch Auction, and two many-to-many auctions: Two Round Double Auction and Continuous Double Auction. In the near future, all four price negotiation mechanisms will be included in our system.

Acknowledgment

Work sponsored by the Marie Curie IRG grant. Project E-CAP.

References

[1] Agorics, Inc., `http://www.agorics.com/Library/auctions.html`

[2] Bădică, C., Bădită, A., Ganzha, M., Paprzycki M.: Developing a Model Agent-based E-commerce System, in: Jie Lu et. al. (eds.) E-Service Intelligence – Methodologies, Technologies and Applications, Springer, (2006) to appear

[3] Bădică, C., Ganzha, M., Gawinecki, M., Kobzdej, P., Paprzycki, M.: Utilizing Dutch Auction in an Agent-based Model E-commerce System, in: Proceedings of the 14th International Enformatika Conference, World Enformatika Society (2006) 7–12

[4] Bădică, C., Ganzha, M., Gawinecki, M., Kobzdej, P., Paprzycki, M.: Towards Trust Management in an Agent-based E-commerce System – Initial Considerations, in: A. Zgrzywa (ed.) Proceedings of the MISSI 2006 Conference, Wroclaw University of Technlogy Press, Wroclaw, Poland (2006) 225-236

[5] Bădică, C., Bădiţă, A., Ganzha, M., Iordache, A., Paprzycki, M.: Rule-Based Framework for Automated Negotiation: Initial Implementation. In: A. Adi, et. al. (eds.): *Proceedings RuleML'2005*, LNCS 3791, Springer (2005) 193–198

[6] Bădică, C., Ganzha, M., Paprzycki, M.: Two Approaches to Code Mobility in an Agent-based E-commerce System, in: Proceedings of the 7th International Enformatika Conference, World Enformatika Society (2005) 101–107

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

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

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

[10] Gawinecki, M., Ganzha, M., Kobzdej, P., Paprzycki, M., Bădică, C., Scafes, M., Popa G., Managing Information and Time Flow in an Agent-based E-commerce System, in: D. Petcu et. al. (eds.), Proceedings of the 5th International Symposiom on Parallel and Distributed Computing, IEEE Press, Los Alamitos, (2006) 352–359

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

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

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

[14] 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

[15] K. A. McCabe, S. J. Rassenti, V. L. Smith, Designing Call Auction Institutions: Is Double-Dutch the Best?, Economic Journal, 102 , 1992, 9-23

[16] Rajarshi Das, James E. Hanson, Jeffrey O. Kephart and Gerald Tesauro, Agent-Human Interactions in the Continuous Double Auction, in The Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI), Seattle, USA (August, 2001).

[17] Rolli, D., Eberhart, A.: An Auction Reference Model for Describing and Running Auctions. In: *Electronic Markets – The International Journal* (2005)

[18] M. Shubik, A double auction market: teaching, experiment and theory, Cowles Foundation Discussion Paper no. 1443, Yale University, October 2003

[19] 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

[20] Wurman, P.R., Wellman, M.P., Walsh, W.E.: Flexible Double Auction for Electronic Commere:Theory and Implementation. In: *Decision Support System*,1998

[21] J. Cao, D. J. Kerbyson, G. R. Nudd, Performance evaluation of an agent-based resource management infrastructure for grid computing, in: Proceedings of the First IEEE/ACM International Symposium on Cluster Computing and the Grid, 2001, 311-318

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