Utilizing Dutch Auction in an Agent-based Model E-commerce System

Costin Bădică, Maria Ganzha, Maciej Gawinecki, Paweł Kobldej, and Marcin Paprzycki

Abstract—Recently, we have presented an initial implementation of a model agent-based e-commerce system, which utilized a simple price negotiation mechanism—English Auction. In this note we discuss how a Dutch Auction involving multiple units of a product can be included in our system. We present UML diagrams of agents involved in price negotiations and briefly discuss rule-based mechanism exemplifying Dutch Auction.

Keywords—e-commerce, rule-based price negotiation mechanism, Dutch Auction, agent system.

I. INTRODUCTION

RECENTLY, we have proposed a model agent-based e-commerce system [2], [3], [4], [9]. While our approach differs from what is typically discussed in the literature (for more details see [1]), here the most important differences are: (1) we consider a sequence of price negotiations (each one of them involves a single unit or a collection of items sold/treated as a single unit), (2) negotiations are "atomic events," as agents cannot join negotiation in progress, (3) each price negotiation can involve a different price negotiation mechanism (e.g. first 243 units may be sold using Vickrey auction, while the next 37 units using fixed price with a deep discount).

It is the latter feature of our system that is our point of departure. In our very early work we have actually implemented a version of the system that involved multiple modularized price negotiation mechanisms dynamically loaded by Buyer Agents (see [11]). Since then we have concentrated our attention on extending functionality of our system ([1]), its formalization ([3]), and on rule-based representation of price negotiation mechanism and its implementation ([4], [6]), which was focused solely on an English Auction —one of the simplest and best understood price negotiation mechanisms.

The aim of this note is to show how our system can accommodate a more complex price negotiation mechanism—Dutch Auction. Here, we consider a robust version of a Dutch Auction, where multiple users can purchase sub-quantities of the inventory put for sale (not only a single unit—as in the case of an English Auction). Though, we still restrict our attention to a single product being sold in a single negotiation (e.g. we are not interested in multi-item auctions involving, for instance, vodka and appetizers).

We proceed as follows. In the next section we define the specific version of Dutch Auction that we are concerned with. We follow with an overview of the system—focused on these features that are pertinent to the proposed form of price negotiations. In the next section we discuss our rule-based approach to representing Dutch Auction. We conclude by specifying subsequent steps in developing our system.

II. DUTCH AUCTION—GENERAL CONSIDERATIONS

While, for all practical purposes, there exists an all agreed on definition of an English Auction, the situation with the Dutch Auction is more complicated. Historically, Dutch Auction has been used in Holland as ways for selling produce and flowers. Today, sometimes the name “Dutch Auction” is used to describe a “short” version of a “true” Dutch Auction. For example, in the Finance Glossary ([16]) one can read: “Named after the Dutch tulip auctions, this form of auction is one where the auctioneer starts with a high asking price, which is then lowered until a bidder accepts the auctioneers price. This is a quick way of auctioning goods, since a sale only requires one bid.” Obviously, it is possible to use Dutch Auction to sell a single unit of a product; however, a “full version” of a Dutch Auction is—most typically—used to sell multiple units (note that in the case of Dutch Auction it is the Seller who is the “driving force” of negotiation, which is a reverse of an English Auction, where Buyers are active and the Seller is “passive”). What makes the situation particularly confusing is the fact that nowadays the name “Dutch Auction” is used also for another type of an auction — uniform second price auction [13], [14] (this happens, for instance, in some eBay auctions).

In our work we have chosen the following conceptualization of a Dutch Auction ([12]): “In a Dutch auction, bidding starts at an extremely high price and is progressively lowered until a buyer claims an item by calling "mine," or by pressing a button that stops an automatic clock. When multiple units are auctioned, normally more takers press the button as price declines. In other words, the first winner takes his prize and pays his price and later winners pay less. When goods are exhausted, the bidding is over.”

Observe that this conceptualization is in accordance with the following FIPA Dutch Auction Interaction Protocol [15]: “First, the good may be split: for example the auctioneer may be selling five boxes of tulips at price X, and a buyer may purchase only three of the boxes. The auction then continues, with a price at the next increment below X, until the rest..."
of the good is sold or the reserve price met. Such partial sales of goods are only present in some markets; in others the purchaser must bid to buy the entire good. Secondly, the flower market mechanism is set up to ensure that there is no contention amongst buyers by preventing any other bids once a single bid has been made for a good. Offers and bids are binding, so there is no protocol for accepting or rejecting a bid. In the agent case, it is not possible to assume, and too restrictive to require, that such conditions apply. Thus it is quite possible that two or more bids are received by the auctioneer for the same good. The protocol (...) thus allows for a bid to be rejected. This is intended only to be used in the case of multiple, competing and simultaneous bids. It is outside the scope of this specification to pre-specify any particular mechanism for resolving this conflict. In the general case, the agents should make no assumptions beyond "first come, first served." In any given domain, other rules may apply."

Since we consider Dutch Auction within an agent system, the above considerations can be translated into the following scenario. Let us assume that within a Negotiation Host (a place where the negotiations take place), we have a single Seller Agent (SeA) and multiple Buyer Agents (BA)s. The SeA attempts at selling $N$ units of product $P$. It starts bidding (by posting an initial bid on a blackboard) and subsequently reduces the price. At a certain moment one of BAs (e.g. $BA_{25}$) responds with an ACL message informing that it is ready to purchase $M$ units of $P$ (where $M \leq N$) at a price equal to the current bid. At this stage, the number of available units is reduced by $M$ and $BA_{25}$ is removed from negotiations to process its reservation (see below). If $N - M > 0$, process continues until either all $N$ units are sold, or the minimal reservation price of the SeA is reached without further takers. While, in general, it is possible for a given $BA$ to bid for multiple quantities of a product; e.g. it could buy 23 items at $25$ and 17 items at $23$, we have decided to simplify the system and assume that each $BA$ is allowed to submit at most one successful bid, which results in it being removed from further negotiations.

Let us also note that (1) when implementing our negotiation sub-system, rather than straightforwardly encoding the FIPA Dutch Auction Interaction Protocol into agents behaviors (as in [11]), we have decided to follow the more general approach proposed in [8]. Thus, to gain flexibility and modularity, we combine a generic negotiation protocol with a rule-based declarative representation of the negotiation mechanism (this solution was already applied to implement English Auctions [4]). (2) Furthermore, since we implement our system in an actual agent environment (JADE) we avoid the above described theoretical situation where two messages arrive at exactly the same time. It will be the JADE runtime that will prevent such an occurrence—one of messages will actually arrive earlier.

Our system acts as a distributed marketplace where products available in e-shops are sold by Shop and Seller agents, while users utilize Client and Buyer agents to localize products that satisfy their needs. In Figure 1 we present Use Case diagram of the complete system. Let us note first that outside of bounds of the system we have depicted User-Agent who attempts at buying products and User-Seller who tries to sell products through her e-store.

Let us now briefly summarize functionalities of agents appearing in the system. User-Agent is represented by the Client Agent (CA). The CA is assumed to be completely autonomous. As soon as the decision to purchase $M$ units of product $P$ is communicated by the User-Agent, it will work until either purchase is completed or has to be abandoned (e.g. due to the current market-prices being higher than the user-specified maximum). The CA communicates with the Client Information Center (CIC) agent which possess a complete information which e-shops sell which products (for more information about information management in the system see [10]). For each store that sells the desired product, the CA delegates a single Buyer Agent (BA) to participate in price negotiations and, if successful, possibly attempt at making purchase. In our system, successful price negotiations result in a product reservation (for a specific time period). When the reservation expires without an actual purchase, item is returned to the pool of products available for sale. Since multiple BAs representing the same CA can win price negotiations and report to the CA, the CA has to decide if either of available offers is good enough to make a purchase. Buyer Agents either migrate to the negotiation host or are created locally [5]. They can participate in negotiations only if the Gatekeeper Agent (GA) allows this. The GA utilizes trust information to evaluate if a given BA should be admitted (e.g. BAs that win price negotiations but do not purchase products may be barred from subsequent negotiations—see [7] for more details). The GA is one of agents that represent an e-store and is created by the Shop Agent (SA). Its role is to admit (or not) incoming BAs and to organize the negotiation process. In this latter capacity it creates and manages a pool of generic Seller Agents (SeAs) that negotiate price with incoming BAs. The SA is the central manager of the e-shop. In its role of facilitator of product sales, the SA utilizes the GA, as well as a Warehouse Agent (WA) that is responsible for inventory and reservation management.

Considering our earlier work, there are two agents that are directly affected by incorporation of a Dutch Auction into the system. The first one of them is the Gatekeeper agent. In Figure 2 we can see, affected by the changes in the system, fragment of the statechart diagram of the Gatekeeper Agent. It depicts its operations in management of, and interaction with, Seller Agents that take place in the context of a Dutch Auction.

III. SYSTEM ARCHITECTURE—AN OVERVIEW

Let us now put the above defined Dutch Auction in the context of our system. We focus our attention on these of its features that are pertinent to the subject of this note. A complete description of all of its functionalities can be found in [1], [2], [3], [4], [5], [6], [9].

\footnote{Here, we have to note that in the past it was the SA that was responsible for creation of Seller Agents and we used one SeA for each product in the store. Since we have changed the way that products are represented—different features of a given product result in its being represented as a separate product (see [10])—creation of a single SeA for each product defined in such a way made no sense. We have thus decided to utilize a pool of generic SeAs that can represent the store in any negotiation (of any product and using any mechanism). Here, we will use the same approach based on dynamically loadable modules as described in [6], [11].}
In Figure 2 we present a situation when, as the first step, an “initial pool” of Seller Agents is created. Then the GA is awaiting for an event to service. Here, we recognize three possible events: (1) a new price negotiation, that is to be serviced by one of SeAs is being recognized by another “part” of the GA and materializes as an event in the “SeA management subsystem”—in this case, if there is an available SeA, it will be appropriately equipped with a negotiation template (and possibly a specific negotiation strategy) and assigned that task; in the case when a SeA is not available then either one will be created or the task will be buffered (depending on the total number of existing SeAs); (2) an ACL message from the SeA informing that a given BA is a winner of a negotiation arrives—in this case, appropriate information is passed to the SA (following basic rules of development of agent systems it is assumed that the SA does not know SeAs that are created and managed by the GA); (3) SeA informs that negotiation that it was responsible for has ended and it is available to take up the next one—in this case, if there is a task waiting in the buffer, it will be assigned to that SeA; if there are no tasks in the buffer, given SeA may be left idle or may be eliminated (if the GA decides that it has too many active SeAs). Observe that the distinction between situation (2) and (3) would have not occur in the case of an English Auction. Successful end of negotiations in an English Auction means that the SeA has completed its job (it ends an English Auction). However, in the case of a Dutch Auction, the fact that one of BAs placed an accepted bid does not mean that the auction is over as that bid could have been placed for only a part of an inventory that is being auctioned. Note, however, that the solution depicted in Figure 2 is capable of handling both English and Dutch auctions. In the case of an English Auction, the SeA simply informs about the result of the negotiation and that it is free. Therefore, a single design of the GA can handle multiple forms of price negotiations, which was one of overarching goals of our system [1].

In Figure 3 we present a simple statechart diagram of the Seller Agent that is involved in a Dutch Auction. Here, the colored box denoted as “NEGOTIATIONS” encapsulates Dutch Auction. Since the BA is relatively passive (it only waits and places a single bid) it has been omitted from discussions.
IV. RULE-BASED REPRESENTATION OF A DUTCH AUCTION

Focus of this section is to briefly describe our solution to rule-based representation of a Dutch Auction. However, before starting this discussion, it has to be stressed that in our work we assume the negotiation model and the infrastructure proposed by [8]; that was previously used to implement English Auctions, see [4], [6]. To make the presentation self-contained, we start from an overview of that model and then discuss how it was adapted to fit our needs.

Authors of [8] sketched a general framework for implementing agent and non-agent negotiations. Their framework is based on an abstract negotiation process that comprises: a negotiation infrastructure, a generic negotiation protocol and a taxonomy of negotiation rules. The negotiation infrastructure defines roles involved in the negotiation process: Participants and a Negotiation Host. Participants are usually Buyers and Sellers that negotiate by exchanging proposals. The exchange is mediated by the Negotiation Host (i.e., conceptually, there is no direct exchange of messages between the participants, but only via the host) and governed by a generic negotiation protocol that defines how and when messages should be exchanged between the Negotiation Host and Participants.

This approach has the advantage that all knowledge required to facilitate negotiations is stored in a central place—the Negotiation Host. Therefore, the Negotiation Host has the power to control and coordinate negotiations. In particular, it is proposed that this knowledge should be captured as a set of negotiation rules organized taxonomically into rule categories for: (1) checking the validity of negotiation proposals, (2) protocol enforcement, (3) updating negotiation status and informing participants, (4) agreement formation and (5) controlling the negotiation termination [8]. Note also that, in the proposed framework, details of the particular negotiation (specific values for parameters specified in generic negotiation

---

**Fig. 2** Gatekeeper agent, functions related to interactions with and management of Seller Agent(s)

**Fig. 3** State Chart diagram of the Seller Agent

Obviously, Dutch Auction is an example of an price negotiations which continues when one BA completes its negotiations, and ends when the inventory is sold or the reservation price of the SeA is reached. However, this situation can be viewed as a generalization of an English Auction. In an English Auction when a Buyer Agent wins negotiations, SeA informs GA about winner (send(GA,NegRes,busy)) and then checks whether the negotiation can be continued—condition \( q\_\neg \neq 0 \). Of course it cannot and thus the SeA informs the GA that it is available (send(GA,info,available)). In this way also the SeA depicted here can handle both English and Dutch auctions.
rules) are provided to its participants in a form of a negotiation template.

While, according to this general framework, roles of the Seller and the Negotiation Host are conceptually different, in the present work, for efficiency reasons, we have decided to merge them together within the same agent—the Seller Agent. This basically means that the SeA will have two responsibilities: (i) to regulate the negotiation by checking bids and bid exchange compliance against the negotiation rules that govern Dutch Auction (see below), and (ii) to post bids (shout prices), as required by the Dutch Auction.

As specified in Section II, we consider multi-unit single-product Dutch Auction, as conceptualized in the FIPA Dutch Auction Interaction Protocol. It can be easily seen that the FIPA Dutch Auction Interaction Protocol is under-specified (a similar observation concerning the FIPA English Auction Interaction Protocol can be found in [8]), as it does not specify important issues like: (a) how often is the SeA allowed to shout the price? for example, shouting prices “very fast” may not leave enough time for the BAs to “think” if to bid or not; (b) how much the SeA must decrement the price in the next bid? for example, an auction might require a minimum value for this decrement (note that, similarly, an English Auction typically imposes a minimum value by which BAs should increment the price of their next bid, [2]); (c) how much time without any activity is allowed before terminating the auction? note that it might happen that neither BAs bid nor SeA wants to decrement the price, and in such a case the auction must be terminated (even if SeA didn’t sell all the inventory).

Taking this into account, we have decided to add the following parameters to the negotiation mechanism (to be later specified in the negotiation template, [8]):

- Minimum value by which the Seller must decrement the price at each announcement: \( H > 0 \); i.e. if the last shouted price was \( X \) then the next shouted price must be at most \( X - H \);
- Minimum time limit that the Seller must wait before issuing the next announcement: \( T_m > 0 \); i.e. if the last price was shouted at time \( T \) and Buyers didn’t bid then the next price must be shouted not earlier than \( T + T_m \); however, if happened that a buyer submitted a bid, \( T_m \) is ignored.
- Maximum time window of inactivity in the auction that, when observed, will terminate the auction: \( T_w > 0 \); obviously, the following must hold: \( T_m < T_w \).

Additionally, number of units \( N \) of the product to be sold is also placed in the negotiation template. This value is decremented with the number items “sold” (actually reserved, according to our model) whenever a successful bid is received from a Buyer.

While parameters of the negotiation mechanism together with negotiation rules are public, i.e. known to all negotiation participants, each negotiation participant may also use a private strategy that dictates how it should actually act during the negotiation. In a Dutch Auction Buyer strategy dictates when exactly the Buyer should accept price shouted by the SeA. It could be extremely simple and require acceptance of shouted price that falls below a given threshold. It could also be rather complex and involve passage of time, number of still available items and/or speed with which items are being purchased by other Buyers. On the other hand, Seller strategy could contain following parameters:

- initial price \( X_0 \), i.e. price that is shouted first.
- timing \( R_i \) of reducing price at each new shout \( i \) (variable at each shout); the following condition must hold: \( T_m \leq R_i \leq T_w \) for all \( i \) to assure that rules are not violated and negotiation does not terminate.
- Reservation price \( X_{res} \); Seller will stop bidding if the price reaches a value below this limit.

Let us now look into the rule-based representation of a Dutch Auction. We define: \( P_r \) – proposal (or bid); \( X(P_r) \) – price, \( T(P_r) \) – time when the proposal has reached the Negotiation Host, and \( M(P_r) \) – number of units “accepted” for purchase. Let us present sample rules that are a part of our representation of the Dutch Auction.

The Seller is allowed to shout a new price when one of the following conditions holds: (i) it is the first shout; (ii) one of Buyers submitted a bid that was successful; (iii) no successful bids were received and at least \( T_m \) time units elapsed since the last shout. Rule 1 checks the third of these conditions. Note that when the Seller offer passes all checks it becomes posted.

**Rule 1 POSTING-SELLER**

\[
\text{IF} \quad \text{No participant with role ‘Buyer’ submitted a successful bid since the last shout} \\
\text{THEN} \quad \text{Proposal } P_r \text{ is posted}
\]

A posted offer that is not the first offer must also satisfy the improvement tests in order to become active. This is realized by rule 2 that asks for the shouted price to be at most equal to the last shouted price minus a template-specified decrement. When this condition holds, the seller offer becomes active.

**Rule 2 IMPROVEMENT-SELLER**

\[
\text{IF} \quad \text{Participant with role ‘Seller’ has posted proposal } P_r \text{ and} \\
\text{Value of last offer posted by ‘Seller’ is } B \text{ and} \\
\text{Minimum decrement is } H \text{ and} \\
X(P_r) \leq B - H \\
\text{THEN} \quad \text{Proposal } P_r \text{ is active}
\]

Now let us see when a bid posted by a Buyer can be accepted, i.e. it becomes active. Informally, a Buyer is allowed to submit a bid only when there is already an active offer shouted by the Seller. Additionally, the Buyer bid is accepted (i.e. becomes active) only if its price is equal to the price shouted most recently by the Seller and the required quantity is at most equal to the quantity available in the Seller offer. These
conditions are checked by rules 3 and 4. Note that multiple Buyer bids are allowed in-between two consecutive Seller bids. The first-come first-served mechanism is used to resolve situation when Buyer bids exceed the available inventory.

**Rule 3 POSTING-BUYER**

IF

Proposal \( P_r \) submitted by a participant with role 'Buyer' is valid and There is a seller active seller offer

THEN

Proposal \( P_r \) is posted

**Rule 4 ACCEPTANCE-BUYER**

IF

Proposal \( P_r \) was posted by a participant with role 'Buyer' and the seller active offer has value \( B \) and \( X(Pr) = B \) and \( M(Pr) \leq N \)

THEN

Proposal \( P_r \) is active

When a Buyer becomes active, it will result in: (i) triggering rules for informing participants (specifically, Buyers are notified how many units are still available for sale); (ii) triggering rules for agreement formation and negotiation termination. An agreement formation rule 5 simply looks for an active Seller offer and an active Buyer bid, in which case it generates a deal by matching those two proposals and updates number of units available for sale.

**Rule 5 AGREEMENT FORMATION**

IF

There is an active bid submitted by a participant \( Par_1 \) with role 'Buyer' and there is an active offer shouted by a participant \( Par_2 \) with role 'Seller'

THEN

An agreement between \( Par_1 \) and \( Par_2 \) to transact according to \( X(Par_1) \) is formed and Available quantity \( N \) is updated

Since the agreement formation rule updates number of items still available for sale, a negotiation termination rule has to be activated. It checks if there are any items left and if they are none the negotiation is terminated.

Some effects of agreement formation and negotiation termination rules have been depicted in Figure 3. It is important to recognize that, operations performed by the SeA (e.g. sending message to the GA) and their conditions represent a different level of description than discussed above rule-based mechanisms within the SeA that trigger them.

V. CONCLUDING REMARKS

In this note we have considered how a Dutch Auction mechanism can be incorporated into our model agent-based e-commerce system. To this effect we have: (1) defined the specific form of Dutch Auction; (2) discussed, and illustrated in the form of UML diagrams, these parts of our system that are affected by adding a new price negotiation mechanism; (3) shown that the proposed modifications are general enough that they will be able to handle not only Dutch Auction, but also accommodate earlier considered English Auction; (4) specified, and illustrated by selected sample-rules how rule-based mechanisms will be used to represent Dutch Auction.

The next step will be to adapt the complete set of rules for the JESS rule-engine. We also will investigate which rules are common to both English, Dutch and other auctions and which are negotiation-specific. This may allow us to form a set of core (reusable) rules applicable to a wide class of price negotiations. We will report on our progress in subsequent papers.

ACKNOWLEDGMENT

Work of M. Ganzha, M. Gawinecki, P. Kobzdej and M. Paprzycki has been partially sponsored by the M. Curie IRG grant, project E-CAP.

REFERENCES


