In *Proceedings of the Second International Conference on Autonomous Agents (Agents-98),*
Minneapolis, MN, USA, May 1998

The Michigan Internet AuctionBot: A Configurable Auction Server for Human and Software Agents

Peter R. Wurman Michael P. Wellman William E. Walsh

University of Michigan
Artificial Intelligence Laboratory
{ pwurman, wellman, wew } @umich.edu

Abstract

Market mechanisms, such as auctions, will likely represent a common interaction medium for agents on the Internet. The Michigan Internet AuctionBot is a flexible, scalable, and robust auction server that supports both software and human agents. The server manages many simultaneous auctions by separating the interface from the core auction procedures. This design provides a responsive interface and tolerates system and network disruptions, but necessitates careful timekeeping procedures to ensure temporal accuracy. The AuctionBot has been used extensively in classroom exercises, and is available to the general Internet population. Its flexible specification of auctions in terms of orthogonal parameters makes it a useful device for agent researchers exploring the design space of auction mechanisms.

1 Introduction

The emergence of electronic commerce is feeding a gold-rush atmosphere in the computer industry. Forecasts predict U.S. electronic commerce revenues will grow from $8 billion in 1997 to $327 billion in the year 2002.[1] The automation of commerce activities is a major step in the evolution of the economy, and we expect that, for some tasks, the processes and conventions employed online will differ from those prevalent in the offline economy.

It is widely observed that agent technology may have a profound effect upon the way goods are bought and sold. For example, shopping agents, such as Bargain Finder[2] and Jango[3] (a commercial product based on the ShopBot [3]), can make online comparison shopping dramatically more efficient, potentially shifting the competitive balance between consumers and retailers. Firefly [12] expands a consumer's awareness by suggesting products—in this case, music CDs—based upon the reported preferences of others with similar tastes.

The agents mentioned above facilitate the connection of potential buyers and sellers. This represents the first stage of an electronic commerce activity [6]. To complete the transaction, the agents must negotiate a mutually acceptable contract, and exchange the goods. Negotiation on the Internet often amounts to one party (typically the seller), presenting a take-it-or-leave-it offer (e.g., a sale price). Auctions represent a more general approach to price determination, admitting a range of negotiation protocols—including fixed price as a special case.

Auctions have rapidly achieved enormous popularity on the Internet. EBay,[4] one of several commercial sites that run user-created auctions, claims to be transacting nearly $2 million a week. Onsale,[5] the first and most prominent of the seller-run online auctions, reported gross revenue for the second quarter of 1997 of $18.6 million—a 50% increase over the previous quarter. The industry has rapidly spawned subindustries, such as newsletters,[6] auction software providers,[7] and specialized search engines.[8]

In addition to their use in online retail, automated auctions are also found at the core of systems for *market-based resource allocation* [2, 13, 16, 17, 18, 20]. To support our research on this topic, as well as electronic commerce, we have developed a general platform for price-based negotiation—the Michigan Internet AuctionBot.[9] The AuctionBot serves as infrastructure for our research efforts, and can be used by others exploring related

[1] Source: Forrester Research, Inc.
[2] `http://bf.cstar.ac.com/bf`
[3] `http://www.jango.com`

[4] `http://www.ebay.com`
[5] `http://www.onsale.com`
[6] For example, AuctionLand (`http://www.neomax.com`), which also provides ratings of over 200 online auction sites.
[7] For example, OpenSite Technologies (`http://www.opensite.com`).
[8] `http://www.bidfind.com`
[9] `http://auction.eecs.umich.edu`

mechanisms. We have also deployed the system over the World-Wide Web as an experiment in Internet commerce, and as an instructional tool to support education in the design and development of market-based multiagent systems.

This paper presents an overview of the AuctionBot system, highlighting its versatility as a tool for exploring the auction design space, and the potential benefits of a standardized agent interface to online auctions. The next section defines auctions and categorizes the messages required to interact with them. Sections 3 and 4 present overviews of the main features and overall design of the AuctionBot. We follow this with a more detailed explanation of the auction parameter space in Section 5, and our treatment of time in Section 6. Section 7 briefly presents the agent-level interface for the AuctionBot. We relate our experience with several classroom exercises in Section 8.

2 Auctions

A negotiation mechanism is essentially a protocol within which agents interact to determine a contract. Auctions constitute a general class of such protocols, as characterized in the standard definition expressed by McAfee and McMillan (1987):

> An auction is a market institution with an explicit set of rules determining resource allocation and prices on the basis of bids from the market participants.

Notice that the stereotyped image—of a fast-talking person with a gavel calling out prices—is but a particular special case of this class (technically, the *English open outcry* auction). Another classic mechanism is the *Dutch* auction, in which the auctioneer begins at a high price and incrementally lowers it until some bidder signals acceptance. *Sealed bid* auctions come in first- and second-price varieties (FPSB and Vickrey [15], respectively), and involve no iteration. In all of these *single-sided* mechanisms, bidders are uniformly of type buyer or uniformly of type seller. The classic single auctions have been the main province of theoretical studies of auctions [7, 9].

Double-sided auctions admit multiple buyers and sellers at once. The *continuous double* auction (CDA) [5]—a general model for commodity and stock markets—initiates trades as soon as matches are detected. Another common mechanism for two-sided markets is the *clearing house* or *call* market [8]. These markets aggregate bids over time and clear at scheduled intervals.

Figure 1 shows one possible taxonomy for a small part of auction space. Auctions are first classified by whether they are single or double sided, and then by

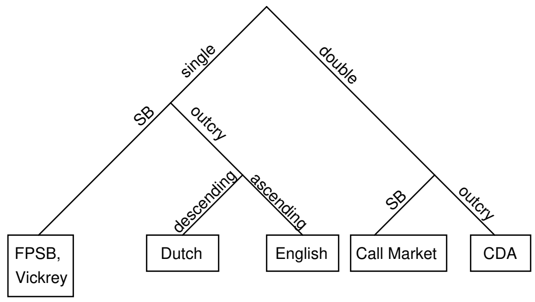

Figure 1: A classification of classic auction types.

whether bids are sealed (SB) or public (outcry). Further distinctions can be made, as we have to differentiate the English and Dutch auctions. However, because there are many distinguishing characteristics—often orthogonal—there are many possible ways to structure such a taxonomy. In Section 5, we present an alternate view of the auction design space based on parameterization.

Formalizing multiagent negotiation in terms of auctions provides a unifying framework and access to a large body of underlying theory. Despite the diversity of these mechanisms, we can clearly define their activities and the messages used to interact with them. This is attractive to designers of multiagent systems because it structures the communication.

We first define the three core auction activities. All auctions must have the first and third behavior, and most have the second as well.

- *Receive Bids:* When a bid is received, the auctioneer must verify that it satisfies the rules of the auction.
- *Supply Intermediate Information:* Auctions commonly supply agents with some form of intermediate results, which we generically term a *price quote*.
- *Clear:* The central purpose of an auction is to determine the resource exchanges and the payments between buyers and sellers.

In our formulation, the auction need not get directly involved in the execution of the transaction, though many auction services on the Internet participate in the transfer of money or goods.

To support these activities, the following message types are required:

- *Bid:* Sent by agent to auction. A bid specifies an agent's offer to buy or sell quantities of a good as a function of the price of the good. It may also include other qualifications on an offer, such as bid expiration conditions and whether the bid may be subdivided.

- *Bid Withdrawal:* Sent by agent to auction.[10] Not all auctions allow an agent to withdraw its bid.
- *Bid Admittance:* Sent by auction to agent when a bid satisfies the rules of the auction.
- *Bid Rejection:* Sent by auction to agent when a bid violates the rules of an auction.
- *Price Quote:* Sent by auction to agent. The precise content varies by auction, but the "standard" definition we employ is the price that the agent would have had to offer in order for its bid to have been one of the bids in the set that would have transacted at the time the quote was issued. This definition is given in past tense because a price quote is necessarily relative to the bidding state at quote time.
- *Transaction Notification:* Sent by auction to agents involved in a transaction. The message specifies the terms of an exchange, including price and quantity, and information necessary to execute the exchange, such as trading partner identity.

An agent can have only one active bid at a time—a new bid received by the auction supersedes any old bids. This requirement is not a restriction per se; the general definition of a bid allows an agent to specify any arbitrary offer function. An agent can "edit" its bid simply by submitting a new one, to the extent allowed by auction rules.

Within this interaction protocol, the perceptions of the agent are restricted to observing the price quote. In many situations the agent can infer from the price quote its current allocation. If the agent has a buy bid greater than (or a sell bid less than) the price quote, then it is currently winning. However, if the agent's bid equals the price quote, then it cannot, in general, know whether its current bid is in the transaction set. For situations when it is necessary that agents correctly deduce their participation there are two approaches. The first approach augments the price quote with the agents' allocation. The second approach generates a disambiguating price quote customized for particular agents. For example, in some auctions it would be sufficient to send losing agents a price quote that is δ above (or below) the actual winning price. In both cases the content of the price quote message is agent dependent.

As an example of the communication process, consider the messages sent in the running of an English auction. The first message is a price quote[11] sent by the auctioneer to each agent. Agents see the price quote and determine whether to bid. When the auctioneer receives a new bid that beats the current price quote, it sends an acknowledgment to the bidder, and generates a new price quote, at some δ over the new high bid, for all non-winning agents. This continues until some period of inactivity elapses, at which time the auctioneer closes the auction and sends transaction messages to the seller and the winning bidder.

The messages described in this section are sufficient to describe any of the auction mechanisms mentioned in this paper. Other messages would be necessary if an auction revealed more intermediate information than the simple price quote. In addition, implementing auction mechanisms in a larger context will require tools to aid agents in finding appropriate auctions, inform them about auction rules, and perform many other market facilitation functions. Such issues are being addressed in several experimental automated market systems [1, 4, 11, 14].

3 AuctionBot Features

The AuctionBot manages a large number of simultaneous auctions. In order to participate in any of these auctions, a user must register. Human users can inspect their accounts via a web page presenting an organized view of their bids, auctions they initiated, and past transactions. This has proven especially useful to students participating in complex economies as classroom exercises (discussed in Section 8). The account view allows a student to track her bids in multiple auctions and provides an automatic accounting of her final transactions.

When they do not wish to babysit their bids, users can choose to be notified of price quotes and clears via e-mail.

The AuctionBot organizes active auctions in a hierarchical catalog. The user initiating an auction can position it anywhere within the existing catalog, or extend the catalog to create an appropriate subcategory. The user also has the choice not to list the auction in the public catalog, choosing, instead, to limit visibility to a private group. We chose to give AuctionBot users complete control over the catalog structure for several reasons. Most importantly, it allows us to avoid exercising editorial control over the subject of AuctionBot negotiations. It also minimizes maintenance on our part. The result is undoubtedly more flexible than anything we would come up with, albeit probably less consistent and coherent.

The AuctionBot is designed to support several languages for expressing bids. For discrete goods, the base language allows a bidder to specify a set of price-quantity pairs, which effectively define a step-wise demand function. The stereotypical bid—expressing a

[10] Technically, this message could be implemented by submitting a replacement bid with zero demand.

[11] The initial price quoted is the reserve price of the seller. In human auctions this value is presumably communicated to the auctioneer outside of the mechanism. In the AuctionBot, the selling agent must place a sell bid.

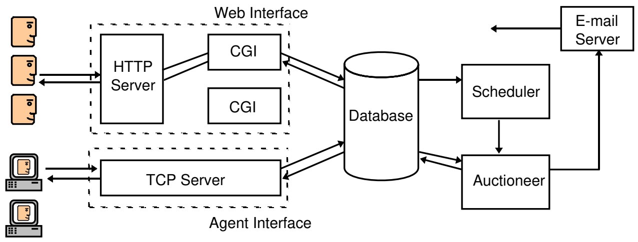

Figure 2: The AuctionBot architecture.

price for some individual object—is encoded by a single point. We are currently working on languages that support functional representations of continuous offer curves for divisible goods, and offers for bundles of interdependent goods of different types.

Mechanism designers will find the most interesting aspect of the AuctionBot is its support of a wide variety of auction types.[12] It is the combination of the variety of auctions and the agent API that makes the AuctionBot a useful platform for both commerce and research.

4 AuctionBot Architecture

The AuctionBot's basic design is shown in Figure 2. The interface consists of two distinct portions—the web interface for humans and the TCP/IP interface for software agents. The right side of the diagram shows the auctioneer processes and scheduler. The interface and the auctioneer programs update information in a common database.

The following example illustrates the interaction between the basic AuctionBot program elements. A human user specifies a bid via a sequence of Web forms. The final form hands off the data to a bid-submission program which inserts the bid into the database and marks it as `unprocessed`. The bid-submission program then returns to the user a confirmation that the bid has been submitted. To keep the interface responsive, the submission program does only cursory verification of the bid. Full verification—which may involve examining all of the other bids in the auction—is done by the auctioneer program.

The scheduler is a daemon process that continually monitors the database for auctions that have events to process or bids to verify. When it finds such an auction, it forks the appropriate auctioneer program. The auctioneer loads the auction parameters and the set of current bids from the database. The auctioneer validates bids as necessary, and may do one clear and/or one price quote each time it is run. The current configuration does not allow the auctioneer to perform more than one clear or quote in a single run due to the overhead of managing the bids and maintaining consistency with the database. We developed this scheme based on our expectation that the majority of auctions would be used by humans and have activity levels measured in seconds or minutes. To provide more responsive auctions at the higher levels of activity that software agents could generate, we plan to extend the design to keep high activity auctions running between events.

Note that the relationship between the interface and the scheduler is asynchronous. If the scheduler goes offline or falls behind in its tasks, the interface continues to operate, and vice versa. All data is timestamped to ensure that when an auction event occurs, it does so with the set of bids that were active at the time the event was supposed to happen. This is discussed further in Section 6.

5 Auction Parameters

The AuctionBot supports the widest range of auction types of any auction service we know of. We achieve this flexibility by decomposing the auction design space into a set of orthogonal parameters. With these parameters, we can implement many of the classic auction types, and quite a few that have not been studied before.[13]

We broadly categorize these parameters by whether they define acceptable bids, control the schedule of clear and quote events, determine the information made available, or specify the matching and price setting algorithm.

5.1 Bidding Restrictions

Participation: For the purposes of specifying the mechanisms, we are interested only in whether the auction allows one buyer or many buyers, and one seller or many sellers. The three combinations of interest are {1:many}, {many:1}, and {many:many}. A restriction to "1" essentially means that the auction is one-sided, and the sole buyer or seller must be designated. Within the AuctionBot, we assume this agent is the auction initiator, and simply reject bids of the corresponding type from other agents.

Discrete Goods: To enforce the discrete good rule an auction simply rejects bids for non-integer quantities.

Bid rules: The AuctionBot supports several other restrictions on allowable bids. One such rule requires that a user's new bid must dominate its previous bid. Note that this would also prohibit bid withdrawal. A

[12] The AuctionBot currently implements every auction mentioned in this paper except the Dutch. Such an extension would be easy, but superfluous, as the Dutch auction is strategically equivalent to the FPSB [7].

[13] An earlier version of this parameterization appears in [10]. The FM96.5 (FishMarket) testbed [11] is based on a detailed parameterization of the space of Dutch auctions.

related rule requires that bidders beat the current price quote.

5.2 Auction Events

Clearing Schedule: Running the matching algorithm and producing an allocation constitutes a market *clear* event. The frequency and number of clears is one of the most important parameters defining an auction. Clears can be triggered at scheduled times (as in sealed bid auctions), at random times, by bidder activity (as in Dutch and CDAs), or by bidder inactivity (as in English auctions).

Closing Conditions: The closing conditions are logical conditions that determine whether a clear should be the final clear. Auctions can close at a scheduled time, at a random time, after a period of inactivity, or when designated bids are matched.

Quote Schedule: Like clears, price quotes can vary in number and frequency. The same options are available: schedule, random, activity, or inactivity.

5.3 Information Revelation

Price Quotes: For divisible-good auctions, the standard price quote is a single price that balances supply and demand. For discrete-good auctions, the general form of the price quote is a range. The *bid quote* is the price an agent would have to bid under in order to place a winning sell bid. The *ask quote* is the price an agent would have to bid over to place a winning buy bid. The ask quote is always greater than or equal to the bid quote.

In the literature, bid-ask spreads are typically used in auctions that clear continuously (and therefore the buyers and sellers do not overlap). However, under the quote interpretation described in Section 2, the bid-ask quote is meaningful even when the buyers and sellers overlap. For example consider a simple case where there is one agent with an offer to buy at $10 and another with an offer to sell at $5. The buyer and seller overlap, so if the auction cleared at this moment, they would transact. A bid-ask quote, in this case, would report that a new buyer would need to outbid $10 and a seller underbid $5 in order to transact in the hypothetical clear.

Transaction History: Auctions may publicize selected information about past transactions. Such information may include the prices, quantities, or even the identities of the transacting agents.

Schedule Information: Auctions may or may not reveal the timing of upcoming clear and quote events. In particular, an auction can used randomized or aperiodic scheduled events and a hidden schedule to discourage attempts at price quote manipulation.

Agent 1	sell 1 unit at $2
Agent 2	buy 1 unit at $3
Agent 3	buy 1 unit at $4

Table 1: A sequence of three bids.

5.4 Allocation Policies

The last, and perhaps most important, parameter is that determining the allocation policy. The allocation policy dictates which agents transact, and at what price(s), as a function of the bids. The AuctionBot currently supports three different policies, all of which assume that goods are measured in integer quantities. They are also *uniform-price* mechanisms, meaning that all transactions determined by a particular clear occur at the same price. Several more allocation policies are in the process of implementation (including a Walrasian auction for divisible goods), or on the drawing board.

The Mth- and $(M+1)$st-price policies are generalizations of the classic first- and second-price mechanisms to multiple units. In both algorithms, M refers the number of units offered for sale. Bids are sorted by price, and the auction counts down M (or $M+1$) units. Roughly speaking, agents with sell bids at or below this mark transact with agents with buy bids at or above this mark. We have analyzed these mechanisms in detail elsewhere [19].

The *chronological match* policy implements the sequential trade effect of the CDA. When the auction processes a new bid, it determines whether the bid can be satisfied by any standing offer. The portion of the new bid that is satisfied transacts. What remains of the bid, if anything, is added to the list of standing offers.

The following example illustrates some differences among these policies. Table 1 shows three bids in the order they are received by the auction. In this example, the Mth-price algorithm would form the transaction `<Agent 1 sells one unit to agent 3 for $4>`. The $(M+1)$st-price algorithm would form the transaction `<Agent 1 sells one unit to agent 3 for $3>`. The *surplus* (difference between reported willingness to pay by buyers and to accept by sellers) in both cases is $2, but it is allocated differently between the agents.

The chronological-match policy takes bids in the order they are received and forms transactions at the price of the earlier bid. Thus, it would form the transaction `<agent 1 sells one unit to agent 2 for $2>`. The surplus in this case is only $1.

This example also illustrates an important point in mechanism design. Seemingly minor differences in parameter settings can affect overall economic efficiency and lead to drastically different agent strategies. For example, in the Mth-price policy, Agent 3 can garner more of the surplus by bidding $\$3+\epsilon$ rather than $4. In the $(M+1)$st-price case, agent 3 can derive no benefit from changing its bid, but agent 1 can.[14]

Tie-breaking rules can also influence the outcome of an auction. The AuctionBot can either break ties arbitrarily, or in favor of the earlier bids. Note that if a new bid ties some existing bids, re-evaluating the earliest-bid rule will not change the allocation, whereas re-evaluating the arbitrary rule may. Thus, when using arbitrary tie breaking, it adds no information to signal an agent how many units it is winning.

5.5 Benefits of Parameterization

We can construct many of the classic auctions from these primitive rules. The English Open Outcry and Vickrey auctions both use the single seller restriction. The English auction is implemented with the Mth-price policy, price quotes based on activity, a single clear based on inactivity, and a rule that bids must beat the price quote by δ. The Vickrey auction has a single clear, no price quotes, and uses the $(M+1)$st-price policy. The FPSB auction can be similarly constructed.

The CDA allows multiple buyers and sellers, uses the chronological match policy, with clears and price quotes based on bidder activity. A variety of different call markets can be constructed from the Mth- and $(M+1)$st-price policies.

Other combinations of these parameters yield some new, possibly interesting auctions. For instance, the Mth- and $(M+1)$st-price policies can be used with auctions that clear with each new bid. Or, the chronological match algorithm can be used with periodic, or inactivity based clears.

From a software engineering perspective, the advantage of the parameterization scheme is that we can implement many different specialized auctions with only three auctioneer programs. Each auctioneer is written for the general case and supports all of the restrictions imposed by the parameter settings through common code modules.

6 Ensuring Temporal Accuracy

The asynchrony between the scheduler and the interface creates some interesting bookkeeping challenges. The general problem is to ensure that only the correct bids are in the active state at a given auction event. For example, Figure 3 depicts a situation in which an agent bids before the time of a scheduled clear and then places a revised bid after the scheduled clear. The scheduler is slightly delayed and does not invoke the auction until

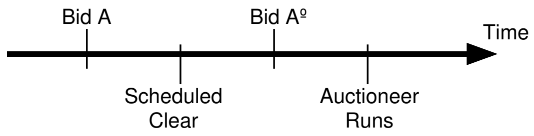

Figure 3: Bids before and after a scheduled event.

after the second bid has been placed. This would not be an issue if the auction were running synchronously. However, within the asynchronous AuctionBot architecture we need to ensure that the clear executes with bid A and not A' active.

The dependency between the validation of bids and scheduled clear events goes both ways. Consider Figure 3 again, but this time assume it illustrates an auction in which the scheduled clear is based on inactivity, and that the period of inactivity is greater than the time interval between A and A'. The scheduler has been delayed, so the auctioneer has not validated bid A — the scheduled clear time indicated in the diagram was calculated from a previous bid. When the auctioneer runs and verifies A, it must also recalculate the time of the clear event. The correct behavior is to include A' in the next clear.

To address these issues we need to keep careful track of the state of a bid. Figure 4 diagrams the transitions possible in the life of a bid. The auctioneer uses only active bids, indicated by grey boxes, when making clear or price quote calculations. The interface inserts a new bid into the database with a timestamp and a state of `unprocessed`. The next time it runs, the auctioneer first determines when its next clear or quote event should occur. It then loads all the bids that were submitted before the time of the next event. If the bid state is `unprocessed`, the auctioneer verifies whether the bid satisfies the auction's rules, and transitions the bid to either `valid` or `rejected`. The auctioneer must also check whether any active bids have expired or been replaced, and transition them to the appropriate closed state. Five distinct types of closed states are tracked for auditing purposes.

Bid withdrawals, like bid submissions, arrive asynchronously, and so can occur out of sequence with clear events. To track a user's request to withdraw a bid, we have two extra states. When a user withdraws a bid, indicated by the dashed arrows, the interface transitions it to the appropriate `withdraw-requested` state and marks it with a timestamp. If the withdraw request occurred before the auction event, the auctioneer transitions the bid to the `withdrawn` state. Otherwise the auctioneer considers the bid active at the time of the auction event.[15]

[14] This property, formally called *incentive compatibility*, means that an agent's dominant strategy is to bid its true valuation for a good. In a one-shot $(M+1)$-st price auction, it holds for buyers who desire a single unit, whereas in a one-shot Mth price auction it holds for sellers offering a single unit.

[15] This may involve verifying the bid and transitioning it from

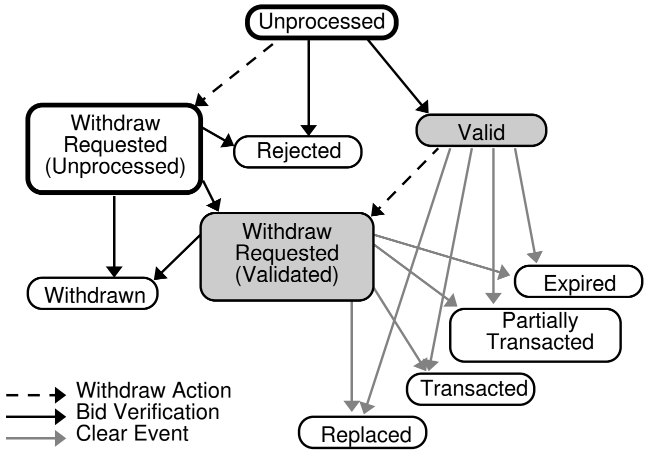

Figure 4: State transitions of bids.

7 Agent interface

The agent interface is a TCP/IP-level message protocol that allows agents to access all of the features of the AuctionBot present in the web interface. Agents can place bids, create auctions, request auction information, or review their accounts. The notification feature, equivalent to the e-mail notification available in the web interface, is not yet available in the agent API.

To the best of our knowledge, the AuctionBot is the only online auction site with explicit support for user-written software agents. The client code is publicly available, and we plan to enhance the libraries with support for common high-level strategies and market agent functions.

In fact, the API is complete enough that developers can use it to build their own front end to the AuctionBot. This flexibility is particularly useful for researchers who wish to use the functionality of the AuctionBot but would rather provide an alternate interface.

8 Uses of the AuctionBot

The AuctionBot has been open for public use at the University of Michigan since September 1996, and to the entire Internet since January 1997. We have used the AuctionBot to create online markets for used textbooks, and several individuals have successfully sold various objects. The volume of public retail activity has been small to date, we believe in large part because commercial sites like EBay are already serving that sector well. Network externalities make it difficult for other sites to reach the critical mass of participation needed to sustain an active marketplace.

The AuctionBot's most demanding use has come during several classroom exercises run in computer science courses at the University of Michigan. In the largest of these exercises, we assigned each of 90 students one of three roles: consumer, component manufacturer, or widget assembler. Consumers are endowed with raw materials which they sell to the component manufacturers. The manufacturers convert the raw materials into one of four components and sell them to the assemblers. Assemblers can combine different components to create one of three widget types. The assemblers sell the widgets to the consumers, who derive utility from owning widgets. Each agent has additional budget or production constraints that make its task challenging.

Perhaps the most interesting qualitative observation about the exercise is how students react to the reality that the outcomes of their actions depend on the unpredictable behaviors of 89 other students. Although they were invited to try to collude, such efforts proved fruitless at this scale. Despite some incentive mismatches in our design of the exercise, the aggregate results were consistent with an efficient mix of widgets produced, and attentive students were able to discern structure in the movements of prices and draw appropriate implications for their agent strategies.

We have used the AuctionBot for other classroom exercises, including a task allocation problem that is also a subject of our multiagent systems research [16]. We welcome efforts by researchers interested in setting up their own experiments on this platform.

Our primary current use of the AuctionBot is as part of a comprehensive testbed in market mechanism design. We have used the system to prototype protocols for market-based scheduling [17], and the simple web-based demonstration we have deployed may be a good introduction for those interested in experimenting with the AuctionBot.[16]

9 Conclusion

The Michigan Internet AuctionBot is a versatile, robust online auction server that supports both human and software agents. It is potentially attractive to those wishing to develop or prototype online marketplaces because it is free and it implements the widest variety of auctions. It has demonstrated usefulness as an instructional tool by supporting large-scale, complex classroom exercises that involve many interacting markets. As a research platform, the large parameter space and open agent API make it particularly well suited for running experiments in computational market mechanisms and agent strategies.

`withdraw-requested(unprocessed)` to `withdraw-requested(validated)`.

[16] `http://auction.eecs.umich.edu/demos/factory.html`

Acknowledgments

This work was supported in part by grants from NSF, AFOSR, and DARPA, and an equipment donation from IBM. Many University of Michigan students have contributed: Stewart Blacklock, Brian Joh, Yee-Wah Lee, Jonathan Mayer, Tracy Mullen, Ryan Papa, Chris Wong, and Itai Zohar. We are grateful to them, as well as to all of the students who have participated in AuctionBot experiments. Jeff Kopmanis deserves credit for keeping the AuctionBot running smoothly, and Kevin O'Malley for helping us develop the next generation of AuctionBot functions.

References

[1] Anthony Chavez and Pattie Maes. Kasbah: An agent marketplace for buying and selling goods. In *First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, London, 1996.

[2] Scott Clearwater. *Market-Based Control: A Paradigm for Distributed Resource Allocation*. World Scientific, 1995.

[3] Robert B. Doorenbos, Oren Etzioni, and Daniel S. Weld. A scalable comparison-shopping agent for the world-wide web. In *First International Conference on Autonomous Agents*, pages 39–48, 1997.

[4] E. H. Durfee, D. L. Kiskis, and W. P. Birmingham. The agent architecture of the University of Michigan Digital Library. In *IEEE Proceedings – Software Engineering*, volume 144, pages 61–71, 1996.

[5] Daniel Friedman and John Rust, editors. *The Double Auction Market: Institutions, Theories and Evidence*. Addison-Wesley Publishing, Reading, MA, 1993.

[6] Robert H. Guttman, Alexandros G. Moukas, and Pattie Maes. Agent-mediated electronic commerce: A survey. *Knowledge Engineering Review*, to appear.

[7] R. Preston McAfee and John McMillan. Auctions and bidding. *Journal of Economic Literature*, 25:699–738, 1987.

[8] Kevin A. McCabe, Stephen J. Rassenti, and Vernon L. Smith. Auction institutional design: Theory and behavior of simultaneous multiple-unit generalizations of the Dutch and English auctions. *American Economic Review*, 80(5):1276–1283, 1990.

[9] Paul Milgrom. Auctions and bidding: A primer. *Journal of Economic Perspectives*, 3(3):3–22, 1989.

[10] Tracy Mullen and Michael P. Wellman. Market-based negotiation for digital library services. In *Second USENIX Workshop on Electronic Commerce*, pages 259–269, Oakland, CA, 1996.

[11] Juan A. Rodríguez-Aguilar, Francisco J. Martín, Pablo Noriega, Pere Garcia, and Carles Sierra. Competitive scenarios for heterogeneous trading agents. In *Second International Conference on Autonomous Agents*, 1998.

[12] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating 'word of mouth'. In *Proceedings of the Computer-Human Interaction Conference (CHI-95)*, 1995.

[13] Michael Stonebraker, Robert Devine, Marcel Kornacker, Witold Litwin, Avi Pfeffer, Adam Sah, and Carl Staelin. An economic paradigm for query processing and data migration in Mariposa. In *Third International Conference on Parallel and Distributed Information Systems*, Las Vegas, NV, 1994.

[14] Maksim Tsvetovatyy, Maria Gini, Bamshad Mobsher, and Zbigniew Wieckowski. MAGMA: An agent based virtual market for electronic commerce. *International Journal of Applied Artificial Intelligence*, September 1997.

[15] William Vickrey. Counterspeculation, auctions, and sealed tenders. *Journal of Finance*, 16:8–37, 1961.

[16] William E. Walsh and Michael P. Wellman. A market protocol for distributed task allocation. In *Third International Conference on Multiagent Systems*, Paris, 1998.

[17] William E. Walsh, Michael P. Wellman, Peter R. Wurman, and Jeffrey K. MacKie-Mason. Some economics of market-based distributed scheduling. In *18th International Conference on Distributed Computing Systems*, Amsterdam, 1998.

[18] Michael P. Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research*, 1:1–23, 1993.

[19] Peter R. Wurman, William E. Walsh, and Michael P. Wellman. Flexible double auctions for electronic commerce: Theory and implementation. Submitted for publication, 1998.

[20] Fredrik Ygge and Hans Akkermans. Power load management as a computational market. In *Second International Conference on Multiagent Systems*, pages 393–400, Kyoto, 1996.