Studies in Computational Intelligence,
Volume 71, Advanced Intelligent Paradigms in Computer Games,
Springer 2007, Pages 117-151

Designing and Developing Electronic Market Games

Maria Fasli and Michael Michalakopoulos

University of Essex, Department of Computer Science, Wivenhoe Park,
Colchester CO4 3SQ, United Kingdom
{mfasli,mmichag}@essex.ac.uk

Abstract. As software agents are well-suited for complex, dynamic and constrained environments that electronic markets are, research into automated trading and negotiation has been flourishing. However, developing electronic marketplaces and trading strategies without careful design and experimentation can be costly and carries high risks. This chapter discusses the need for tools to support the design and implementation of electronic market simulations or games to emulate real life complex situations of strategic interdependence among multiple agents. Such games can be used to conduct research on market infrastructure, negotiation protocols and strategic behaviour. After a brief introduction into the field of agents, we present an overview of agent negotiation in general, and auction protocols in particular, which are among the most popular negotiation protocols. We then present the e-Game platform which has been developed to support the design, implementation and execution of market simulation games involving auctions. How the development of market games is facilitated is demonstrated with an example game.

1 Introduction

The vision of future electronic marketplaces is that of being populated by intelligent software entities – agents – representing their users or owners and conducting business on their behalf. Unlike "traditional" software, agents are personalized, semi-autonomous and continuously running entities [20] and these characteristics make them ideal for complex, and unpredictable environments that electronic markets are. The deployment of software agents in electronic commerce would bring about significant advantages: it would eliminate the need for continuous user involvement, reduce the negotiation time and transaction costs and potentially provide for more efficient allocation of goods and resources for all parties involved. As a result, research into agent-based and multi-agent systems for automated trading has intensified over the last few years.

Exchanges (electronic or otherwise) involve broadly speaking three main phases: firstly potential buyers and sellers must find each other or meet in

M. Fasli and M. Michalakopoulos: *Designing and Developing Electronic Market Games*, Studies in Computational Intelligence (SCI) **71**, 117–151 (2007)
www.springerlink.com

a marketplace, secondly they need to negotiate the terms of the transaction and finally they execute the transaction and the goods/monetary resources change hands. Agent technology can be used in all three phases, from identifying potential trading partners, to negotiation and payment systems. Agents can negotiate for goods and services on behalf of their users (individuals or organizations) reflecting their preferences and perhaps even negotiation strategies.

However, despite their increased popularity, the huge potential of agents and multi-agent systems has not been fully realized. The reasons for the slow uptake of agent technology in particular with regards to e-commerce applications are manifold. Firstly, there is a lack of standardization that permeates the field of agents and multi-agent systems. This inevitably raises concerns as individuals and organizations would like to use standard and stable technologies. Secondly, there are inherent issues with trust. Trust becomes very important if an agent's actions can cause its owner financial, or even psychological harm. The prospect of disclosing personal, financial or otherwise sensitive information to an agent and delegating to it the task of conducting business on one's behalf does involve a number of risks. Thirdly, although implementing simple agents to carry out single negotiation tasks is easy, developing flexible agents that can operate in highly dynamic and uncertain environments is nontrivial. In practice, agents may have to negotiate for a bundle of perhaps interrelated goods being traded via different negotiation protocols. Developing agents that can use different negotiation protocols and compete effectively in complex markets with complementary and substitutable goods is a nontrivial challenge. The successful performance of an agent does not only depend on its strategy, but critically on the strategy of the other agents as well. Designing efficient and effective decision-making algorithms and strategies is difficult since real world data about human traders are hard to obtain.

This chapter discusses one way that the last problem can be addressed. In particular, we discuss the need for tools to support the design and implementation of electronic market simulations or games for conducting research on market infrastructure, negotiation protocols as well as strategic behaviour. We present a platform that provides the facilities for developing electronic market games that involve a variety of auction protocols. How the design and implementation of market games is aided is described and demonstrated with an example game. The rest of the chapter is organized as follows. First we provide a brief introduction into agents and multi-agent systems to put the rest of the sections into context. Next we provide an overview of agent negotiations and an introduction into auctions. The following section discusses the need for developing market games. A discussion of the relevant work in the literature with regards to electronic marketplaces and trading agent platforms follows. The architecture and the most significant features of the e-Game auction platform are described next. The following section discusses game development and the facilities provided by e-Game to facilitate this.

The development of a simple game is presented next. The paper ends with a discussion on further work and the conclusions.

2 Agents and Multi-agent Systems

Agents and Multi-agent systems are a relatively new area of research and development in Computer Science. From a historical point of view, the area has its roots in Artificial Intelligence (AI) and Distributed Artificial Intelligence (DAI). It developed as a separate strand of research in the 1990's, as it was recognized that agents and multi-agent systems can play an important role in the development of distributed open systems. Nowadays, the discipline is very much interdisciplinary and encompasses concepts and ideas from other disciplines as diverse as mathematics, sociology, philosophy, computer science and economics – though this is not an exhaustive list.

Although there is no universal agreement as to what exactly constitutes an agent, most researchers and practitioners understand agents to be computational systems which may consist of both hardware and software components which exhibit characteristics such as autonomy, proactiveness, reactiveness and sociality [10]. Autonomy enables an agent to have control over its execution and behaviour, and act without the direct intervention of the user or other entities. Proactiveness enables an agent to actively seek to find ways to satisfy its design objectives or its user's goals. Reactiveness enables an agent to react to the changes that occur in its environment in a timely fashion and which affect its goals and design objectives. Finally, it is very rare that an agent will be useful on its own, therefore sociality enables an agent to do things with other agents by coordinating with them. Coordination can take two forms: cooperation and negotiation. In cooperation, agents usually have to work with each other as they cannot do something on their own, i.e. the task may be too complex for one agent. Agents are usually cooperative, and although they are interested in maximizing their own benefit, they are also interested in the welfare of the group as a whole. In negotiation, self-interested agents compete with each other for resources, goods etc. They are strictly utility maximizers, and they are not interested in the welfare of the other agents or the society.

Multi-agent systems comprise multiple agents who are situated in a common environment and they can interact with each other through communicative or physical actions. Interaction is both necessary and unavoidable in a common environment and agents may interact with one another because they have to or unintentionally. Interaction is shaped through a set of rules, namely an interaction protocol which dictates the messages or actions that are possible in an interaction situation. We can classify interaction protocols into three broad categories. Communication protocols enable agents to communicate with each other. Such protocols dictate the kind of messages that can be exchanged between agents and their meaning. Cooperation protocols enable

agents to work together to achieve a common objective, perform a complex task or solve a difficult problem. Negotiation protocols enable agents to reach agreements when they are competing with each other and have conflicting goals.

Agents and multi-agent systems have found applications in a number of domains ranging from manufacturing control, information management, entertainment to education. More recently, research efforts have been focused on deploying agents in electronic commerce. Such agents represent individuals or organizations and they search for products or services, evaluate different alternatives, negotiate deals with other trading partners and execute transactions. Indeed, the potential for agents in e-commerce is enormous, and some researchers and technology analysts consider e-commerce to be the killer application for agents.

3 Negotiation Protocols

When agents are in competition or have conflicting goals and attempt to resolve these, they do not interact randomly, but their interaction is governed by a set of rules, namely a protocol [10]. A negotiation or market protocol provides a set of rules and behaviours to be followed by agents that will interact in it. *Mechanism design* is the design of protocols governing strategic interaction among self-interested agents. A negotiation situation is characterized by three elements:

1. The negotiation set which represents the space of possible offers or proposals that agents can make.
2. A protocol which defines the rules that the agents need to follow in order to arrive at an outcome and also rules on what constitutes a legal offer or proposal.
3. A collection of strategies that agents can use to participate in the negotiation. The agents' strategies are private, they are not dictated by the protocol itself and may take into consideration the possible strategies of other agents.

As negotiation situations may differ, some protocols may be more suitable than others for particular situations. There are a number of factors that determine the type of mechanism or negotiation protocol that can be designed for or employed in a particular situation:

– *Number of attributes.* Negotiation can take place over one attribute (e.g. price) or many (e.g. price, quantity and delivery day). Protocols that enable multi-attribute negotiation are more complex than those for single-attribute negotiation due to mainly two issues. Firstly, the negotiating agents have to be able to express preferences and construct offers on multiple attributes which complicates the individual agent's reasoning

mechanism. Secondly, determining the winner in such protocols is a difficult problem especially if multiple winners are to be allowed.
- *Number of agents.* The number of agents involved in the negotiation process also affects mechanism design. Negotiation can be on *one-to-one*, *one-to-many*, or *many-to-many*. In the last case, if there are n agents, there are potentially $n(n-1)/2$ concurrent negotiation threads active.
- *Number of units.* The price of multiple units may differ from protocol to protocol.
- *Interrelated goods.* In certain cases, the goods are only worth to agents when they are in combinations, thus agents negotiate over packages or bundles rather than individual goods. Protocols that enable negotiation over interrelated goods are more complex than those for single good negotiations as expressing and evaluating offers on combinations of multiple objects (which may be substitutable or interrelated) are difficult problems.

A number of protocols have been developed to deal with different negotiation situations. Bargaining protocols, for instance, enable agents to reach a deal through a process of making offers and counteroffers until a deal can be agreed on which is acceptable to all negotiating parties. Voting protocols take as input the individual preferences of a society of agents and attempt to aggregate them in a social preference which dictates an outcome to be imposed on the society. Auctions are a family of negotiation protocols whose distinctive characteristic is that the price of the good under negotiation is determined by the marker participants. We will examine auctions in more detail in subsequent sections.

4 Desiderata for Negotiation Protocols

From the multi-agent systems point of view mechanism design is the design of protocols governing multi-agent strategic interactions. Agents have their individual preferences and evaluations of goods and services that are available in the market and they are seeking to maximize their utilities by exchanging goods and services with the other participants. The aim of traditional mechanism design is to design a system in which rational agents interact in such a way so that desirable social outcomes follow. The desired properties of the social outcomes are encapsulated in a social choice function which ultimately describes what is to be achieved through the use of the mechanism [21], i.e. utility maximization across all agents, efficient allocation of goods and resources, etc.

When designing mechanisms for strategic interactions we are particularly interested in those mechanisms that enjoy certain game-theoretic and computational properties [29]:

Pareto efficiency. Pareto efficiency or economic efficiency measures global good. A solution x is Pareto efficient if there is no other solution x' such

that at least one agent is better off in x' than in x and no agent is worse of in x' than in x. Although Pareto efficiency is a useful criterion for comparing the outcomes of different protocols, it has nothing to say about the distribution of welfare across agents.

Social welfare. Social welfare is measured through a welfare function which provides a way of adding together the different agents' utilities. A welfare function provides a way to rank different distributions of utility among agents, and thus provides a way to measure the distribution of wealth across agents. But to be able to measure social welfare, inter-agent utility comparisons are required which are not always straightforward and may involve utility transformations.

Individual rationality. Individual rationality dictates that an agent should not lose out by participating in a mechanism. Put differently, an agent should only participate in a mechanism if its payoff from the negotiated solution is no less than the payoff that the agent would have by not participating. A protocol is individual rational if it is individual rational for each participating agent.

Stability. Protocols can be designed with dominant strategies: strategies that an agent can use which guarantee maximization of payoffs. In particular, we prefer protocols in which truth-telling is the agents' dominant strategy as agents can decide what to do without having to counterspeculate about the others' strategies. A negotiation protocol is stable if it is designed so that it motivates the agents to behave in a desired way.

Budget balance: In strict budget balance the total payment that agents make must be equal to zero, in other words money is not injected into or removed from a mechanism. Alternatively, in weak budget balance the total payment is nonnegative, this ensures that the mechanism does not run at a loss. In *ex ante* budget balance the mechanism is balanced on average, while in *ex post* the mechanism is balanced at all times and for all instances.

Computational efficiency. The negotiation protocol should be computationally efficient. As little computation as possible should be required on behalf of the agents.

Distribution and communication efficiency. Distributed protocols are preferred to protocols with a central control point, since they avoid failure and performance bottleneck. Protocols should be communication efficient by requiring as little communication as possible to reach the desired outcome.

As we are interested in negotiation protocols for electronic markets computation is crucial. Computation in mechanism design can be distinguished along two distinct dimensions [27]:

Agent:

– Valuation complexity. How much computation is required to provide preference information within a mechanism?

– Strategic complexity. Are there dominant strategies? Do agents have to counterspeculate or do any opponent modelling when computing an optimal strategy?

Infrastructure/mechanism:

– Winner-determination complexity. How much computation is required of the mechanism infrastructure in order to compute an outcome given the agents' preferences?
– Communication complexity. How much communication is required between the agents and the mechanism to compute an outcome?

The challenge is to design computationally efficient mechanisms without sacrificing useful game-theoretic properties, such as efficiency and stability. Mechanism design has a fundamental role to play in devising protocols for complex distributed systems that comprise of self-interested interacting agents. In addition, the use of simulated marketplaces where the consequences of using different protocols on agent behaviour as well as strategic interactions can be studied in detail, can play an important role in designing efficient electronic markets.

5 Auctions

One of the most popular ways of negotiating for goods and services is via auctions [3, 17]. Auctions are a method of allocating goods based upon competition among the interested parties [10]. They constitute one of the oldest forms of market and some pinpoint their origin to Babylon in 500 BC. The term auction comes from the Latin root "auctio" which means "increase". Nowadays all sorts of goods and services are being traded in auctions ranging from paintings to spectrum licenses. With the advent of the World Wide Web, auctions have become extremely popular as the negotiation protocol of choice for conducting consumer-to-consumer (C2C) negotiations. Auction sites such as eBay [9] and onSale [26] have been reporting millions of dollars in transactions from auction sales. Governments have used auctions to sell spectrum and TV licenses, rights to drill for oil and for privatizing government owned companies.

There are two main self-interested parties in an auction: the auctioneer and the bidders. The auctioneer is usually a seller who wants to sell goods at the highest possible price (or subcontract out tasks at the lowest possible price) who may be the owner of the good or service, or a representative of the actual seller. An auctioneer can also be a buyer who is looking to buy a good from a number of sellers at the lowest possible price. The bidders can be buyers who want to buy goods at the lowest possible price (or get awarded contracts at the highest possible price), or sellers who compete for sales. Agents that participate in auctions are self-interested and rational seeking to maximize

Table 1. Auction terminology

Term	Meaning
Bid	Bids are offered by bidders to buy or sell the auctioned good.
Reservation price	The maximum (minimum) price a buyer (seller) is willing to pay (accept) for a good. This is usually private information.
Process bid	The auctioneer checks the validity of a submitted bid according to the rules of the auction.
Price quote	Information about the status of the bids currently in the auction.
Bid quote	The amount a seller would have to offer in order to trade.
Ask quote	The amount a buyer would have to offer in order to trade.
Clearance	The process of matching buy and sell bids.
Clearing price	The final transaction price that the buyer pays and the seller receives.

their own profit. They can have different attitudes towards risk: they can be risk prone or risk neutral. The former are likely to raise their bids so that they are more likely to win, whereas the latter tend to bid more conservatively. Some common terminology used in auctions is described in **Table 1**.

The process of an auction usually involves a number of stages. The bidders (buyers and/or sellers) first register with an auction house. Once the auction opens, the bidders offer their bids according to the rules of the particular auction protocol used. A bid indicates a bound on the bidders' willingness to buy or sell a good. Depending on the auction protocol, the auctioneer processes the bids according to the rules of the auction and may generate price quotes accordingly. When the auction closes, buyers and sellers are matched and the transaction price is set. Subsequently, the transactions between buyers and sellers are executed and the buyers pay while sellers ship the goods or provide the service.

Auctions are very versatile protocols and a number of mechanisms have been designed to deal with different situations and requirements.

5.1 Classification of Auctions

Auctions can be characterized along three main dimensions: the *bidding rules*, the *information revelation policy* and the *clearing policy* used.

The bidding rules specify how the auction is to be conducted and what type of bids the participants are allowed to submit:

- Single object or multiple object. In the former, bidders bid on one commodity only, whereas in the latter, bidders are allowed to submit bids for multiple commodities.
- Single attribute or multi-attribute. In single attribute auctions, the negotiation takes place over one dimension or attribute of the good, namely price. In multi-attribute auctions more than one attributes of the good are being negotiated upon, i.e. not only price, but warranty, delivery day etc.
- Single or double. In single auctions, there is a single seller (or buyer) who negotiates with multiple buyers (or sellers), whereas in double auctions there are multiple buyers who negotiate with multiple sellers.
- Open (outcry) or sealed-bid (SB). In open auctions, the bidders bid openly and thus the other participants know theirs bids. In contrast, in sealed-bid auctions the bidders offer their bids in secret, thus not revealing any information to the other participants.
- Ascending or descending price. In the former, the bidders submit increasingly higher bids and the highest bidder gets the item, whereas in the latter the price is lowered by the auctioneer until a buyer decides to bid and obtain the item.
- First-price or second-price (Mth). In first-price auctions, the highest bidder wins and pays the value of her bid, whereas in second-price (Mth) auctions, the highest bidder wins but she only pays the amount designated by the second (Mth) highest bid.

A typology of auctions based on the above distinctions is illustrated in Figure 1. However, this typology is not an exhaustive representation of all different auction formats. In particular, although multi-dimensional auctions such as combinatorial and multi-attribute auctions are included, more

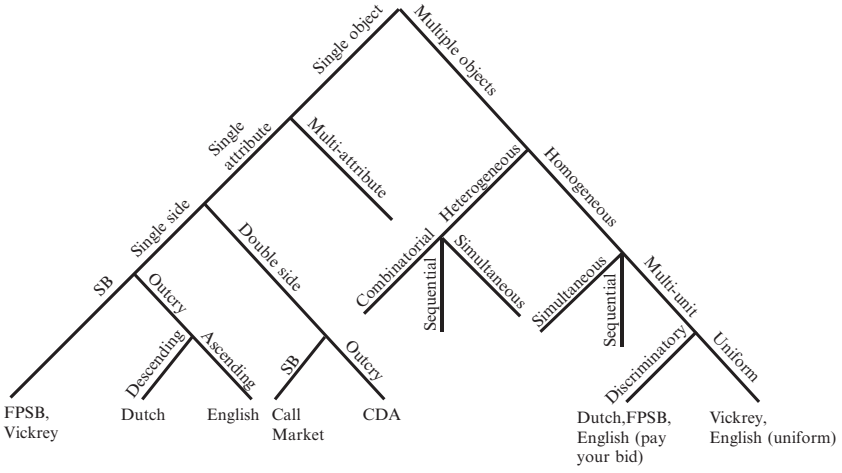

Fig. 1. Auction typology based on bidding rules [10] p. 218

complicated formats such as multi-attribute combinatorial auctions have been left out. Multi-dimensional auctions are in general highly complicated auction formats and are currently the subject of research with somewhat restricted use in practice.

The information revelation policy specifies what information, if any, is revealed to the participants of the auction and has three aspects:

- When to reveal information: on the arrival of each bid, periodically, after a period of activity/inactivity or on market clears.
- What information is to be revealed: if the participants are to obtain information that helps them revise their bids, then the ask and bid quotes need to be made available. Auction closure is another piece of information that can be revealed to the agents as it may be useful in formulating their strategy.
- Who obtains the information: participants only, or everyone.

The clearing policy specifies when the market clears, i.e. when buyers and sellers are matched, and consists of decisions to the following:

- When to clear the market: on arrival of each bid, on closure, periodically, after a period of activity/inactivity.
- Who gets what: who is allocated the goods, how are the winners determined.
- At what prices: what price do buyers pay and sellers receive and how is this determined. Do they pay the first price, second price or some other price. When there are multiple units being auctioned, do all buyers pay the same price (uniform) or a different one (discriminatory).

Finally, auctions can be distinguished into private value, common value and correlated value according to the bidders' valuation of the commodity. In private value auctions, the value of the good depends only on the bidders' own preferences and usually the bidder wants to acquire a good for her own use and consumption. In common value auctions, a bidder's value of an item depends entirely on the others' value of it, which in turn by symmetry is identical to the bidder's, i.e. the good is worth essentially the same to every bidder, but the bidders may have different estimations of this value. In such auctions the bidders want to acquire the commodities for resale or commercial use. In correlated value auctions, a bidder's value depends partly on her own preferences and partly on what she thinks the other agents' valuations are.

5.2 Single Auctions

In single auctions (single good, single attribute), agents are negotiating over one good which is available on its own, the negotiation has one dimension – usually price – and there is one seller (or one buyer) and multiple buyers (or sellers). The most well known basic auction formats are the English, Dutch, First-Price Sealed-Bid (FPSB), and Vickrey.

The English auction is an open-outcry and ascending-price auction which begins with the auctioneer announcing the lowest possible price (which can be the reservation price). Bidders are free to raise their bid and the auction proceeds to successively higher bids. When there are no more raises the winner of the auction is the bidder of the highest bid. The distinct characteristic of the English auction is that the bidders gain information by observing what the others bid. This may be invaluable in forming their own strategy and in some cases even revising their valuation of the auctioned item. There are several variations of the English auction basic format. The bidder's strategy is a series of bids as a function of her private value, her prior estimates of the other bidders' valuations, and the past bids of others. A bidder's best strategy (dominant) is to bid a small amount more than the previous highest bid until she reaches her private value and then stop. In this way a bidder may acquire an object for considerably less than her maximum valuation, simply because she need only increase each bid by a small increment. This on the other hand, means that the seller does not necessarily receive the maximum value for the auctioned good.

The Dutch auction is an open and descending-price auction. The auctioneer announces a high opening bid and then lowers the price until a bidder accepts it. Each bidder needs to decide in advance the maximum amount that she will bid. The bidder must decide when to stop the auction, i.e. submit a bid, based upon her own valuation of the commodity and her prior beliefs about the valuations of the other bidders. No relevant information on the valuation of the other bidders is disclosed during the auction until it is too late. Generally speaking, there is no dominant strategy in the Dutch auction.

In the First-Price Sealed-Bid (FPSB) and Vickrey auctions, the bids submitted are sealed. Sealed-bid auctions have two distinctive phases: the bidding phase in which participants submit their bids, and the resolution phase in which the bids are opened and the winner is determined.

In the FPSB auction the highest bidder wins and pays the amount of her bid. An agent's strategy is her bid as a function of her own private value and prior beliefs about the other bidders' valuations. There is no dominant strategy in the FPSB auction. A high bid raises the probability of winning, but lowers the profit if the bidder is actually awarded the item. Assume that an agent bids her true valuation and it turns out that it is the highest bid, b_h, and wins the auction. Now consider the second highest bid offered b_{h-1}. The winner could have offered just a small increment on that price and still be the winner. Thus the difference between b_h and b_{h-1} represents a loss for the winner. Therefore, agents are better off not bidding their true valuations, but a small amount below it.

In the Vickrey auction, also known as uniform second-price sealed-bid auction, the highest bidder wins but only pays the second highest bid. The agent's strategy is a function of her private value and her prior beliefs of the others' valuations, but the best strategy for a bidder, which constitutes a dominant one, is to bid her true valuation. She then accepts all offers that

are below her valuation and none that are above. Truth-telling in the Vickrey auction ensures that globally efficient decisions are being made; bidders do not have to waste time in counterspeculating what the others will do.

In private value auctions, and when the agents are risk-neutral the Dutch and the FPSB auctions are strategically equivalent, i.e. the strategy space is the same, whereas the Vickrey and the English are equivalent, i.e. bidders have the dominant strategy to bid an amount equal to their true valuation. Moreover, assuming valuations are drawn independently, all four mechanisms yield the same revenue on average for the auctioneer (revenue equivalence theorem). If the bidders are risk-prone, then the Dutch and the FPSB auctions yield higher revenue than the English and the Vickrey.

In non-private value auctions, the Dutch is strategically equivalent to the FPSB auction, whereas the Vickrey is not equivalent to the English. The latter is due to the additional information that bidders can obtain about each other's valuations in an open auction. With more than two bidders, the expected revenues are not the same: English>=Vickrey>=Dutch=FPSB (revenue non-equivalence theorem).

5.3 Double Auctions

In double or two side auctions there are multiple sellers and multiple buyers that participate in an auction in order to trade a commodity. These types of auctions are more often used in exchanges and financial markets for trading stocks, bonds, securities, etc. The Continuous Double Auction (CDA) is a general auction mechanism that is used in commodity and stock markets. The general process is as follows:

- Both sellers and bidders submit their bids.
- The bids are then ranked highest to lowest to generate demand and supply profiles.
- From the profiles the maximum quantity exchanged can be determined by matching selling offers with demand bids.
- The transaction price is set and the market clears.

Double auctions may clear either continuously or periodically. In continuous double auctions the buyers and sellers are matched immediately on detection of compatible bids, (stock markets) while in periodic double auctions also known as call markets or clearing houses, bids are collected over specified intervals of time and then the market clears at the expiration of the bidding interval.

One of the issues in double auctions is determining the transaction or clearing price. Consider a set of single unit bids L. M of these bids are sell offers and the remaining $N = L - M$ are buy offers. The Mth price rule sets the clearing price at the Mth highest price among all L bids [35]. The $(M+1)$st price rule sets the clearing price at the $(M+1)$st highest price among all L bids. Determining the bids that are going to transact then, the transaction set, proceeds

as follows: while the highest remaining buy bid is greater than or equal to the lowest sell bid, remove these from the set of outstanding bids (breaking the ties arbitrarily) and add them to the set of matched bids (transaction set). For instance, let M={14,13,11,10,5,4,3} and N={12,9,8,7,6,3}. The ordered list of all bids L={14,13,12,11,10,9,8,7,6,5,4,3,3}. The Mth price is therefore 8 and the $(M+1)$st price is 7. The transaction set consists of the bids {(12,3), (9,4), (8,5)}. For instance, take the first set of bids in the transaction set (12,3). If the Mth price is used, then the buyer will pay 8 and the seller will receive 8.

The Mth price is undefined if there are no sellers and the $(M+1)$st price is undefined if there are no buyers. In double auctions and as there are multiple buyers and sellers there are mechanisms that allow the agents to obtain information about the current state of the auction in terms of the submitted buy and sell bids. The price quote reveals information to the agents as to whether their bids would be in the transaction set. The bid quote is the price that a seller must offer in order to trade, that is the $(M+1)$st price. The ask quote is the price that the buyer must offer in order to trade, that is the Mth price.

The Mth and $(M+1)$st price rules are generic rules that apply not only in double auctions, but in single auctions as well. In the English auction there is one seller and multiple buyers, thus M=1. The seller (auctioneer) may submit a reservation price which can be 0, and according to the rules of the auction the bidders will start submitting successively higher bids. The Mth price in the English auction is going to be the highest bid submitted which determines the winner. In the Dutch auction the seller (auctioneer) starts by announcing a high price (bid) and then successively lowers it until a bidder accepts to pay that price. In this respect, as the auctioneer successively lowers the price, he essentially withdraws the previous bid and submits a new one. M is 1, and this is the Mth price which is the clearing price that the bidder is willing to pay when she announces that she accepts the auctioneer's bid, it is as if the bidder submits a buy bid which is equal to that of the auctioneer. In the first-price sealed-bid auction the Mth price works in exactly the same way as the English auction, the auctioneer may have a reservation price which can be 0, and the Mth price is the highest bid in the auction. In the Vickrey auction, the clearing price is defined as the $(M+1)$st bid among all bids. Again the auctioneer can be perceived as having a bid which is the reservation price or 0, and the bidders submit bids. The winner is the bidder with the highest bid (Mth bid) but she only pays the $(M+1)$st highest bid among all bids.

5.4 Multi-Dimensional Auctions

Buyers and sellers are often interested in other attributes of a good apart from its price. For instance, when one negotiates the purchase of a car with a dealer, she can negotiate not only on the price, but other attributes too, such as for example the warranty, or the extras such a CD player and leather seats or free insurance cover for a year. Auctions that allow bidders to submit bids

on more than one attributes or dimensions of a good are called multi-attribute auctions [5]. Such auctions are common in procurement situations [30]. The attributes under negotiation are usually defined in advance, and bidders can compete either in an open-cry or sealed-bid auction on multiple attributes. Mechanism designers for multi-attribute auctions are faced with a number of problems in determining the winner for such auctions, also known as the *winner determination problem*. Multi-attribute auctions enable the application of auction protocols to situations where multiple qualitative attributes need to be taken into account beyond the single dimension of price. This process allows more degrees of freedom for bidders in specifying their bids, while at the same time it allows for an efficient information exchange among the market participants. Such auction formats are currently the subject of extended research and experimentation.

The auction formats described so far allow bidders to bid on individual goods. However, there are situations in which a bidder may not be interested in a single good, but in a variety of distinct, but complementary and interrelated or substitutable commodities. It is thus often the case that goods are only worth to bidders in combination and not when sold separately. So buyers and sellers may have preferences not only for a particular commodity, but for sets or bundles of commodities. By allowing bidders to bid on combinations of different commodities, the efficiency of the auction can be further enhanced. Auctions in which bidders are allowed to bid on bundles of goods are called combinatorial or combinational auctions [8]. Combinatorial auctions are the focus of intense research [8, 23, 24, 25, 28], however, they are currently rare in practice. This has partly to do with the problems regarding efficient and computationally tractable mechanism design for such auctions and partly with the fact that such auctions are cognitively complex and therefore difficult to comprehend by participants.

6 The Need for Market Games

The shift from traditional markets to fully automated electronic ones where various entities such as individuals and organizations are represented by software agents conducting business on their behalf, presents us with a number of challenges. The first challenge that needs to be addressed is that of the infrastructure that is required in place in order for such electronic markets to be fully functional and enable participants first to find and then to interact with each other. The second challenge is with regard to the interactions in such marketplaces. What negotiation protocols or mechanisms are needed to enable participants to interact with each other and reach desirable outcomes. It goes without saying that no unique protocol will suit the needs of all negotiation situations. Each domain has its own characteristics and therefore it is impossible to impose a unique protocol as the panacea for all negotiation situations. The third challenge, is how to build efficient and effective trading

agents and endow them with strategies that will enable them to participate and compete in highly complex and dynamic environments that marketplaces are.

Undoubtedly, designing and implementing electronic markets is a complex and intricate process [22]. Agents within such markets are self-interested and are trying to maximize their own utility without necessarily caring about the welfare of others or the society as a whole. As agents are strategic reasoners, if they can gain from lying they will do so, irrespective if this leads to inefficient outcomes in the market. Mechanism design explores such interactions among rational self-interested agents with the view of designing protocols such that when agents use them according to some stability solution concept (dominant strategy equilibrium, Nash equilibrium, mixed strategies Nash equilibrium), then desirable social outcomes follow. But, traditional mechanism design is underpinned by a set of assumptions that may not be realistic for computational agents in complex environments. The most fundamental assumption is that agents are rational and therefore can compute their complete preferences with respect to all possible outcomes. Agents are also assumed to know and understand the protocols and abide by them. In addition, the society of agents participating in the mechanism is static, i.e. the set of agents does not change. Most attention in mechanism design has been focused on centralized mechanisms: agents reveal their preferences to a central mechanism which then computes the optimal solution given these preferences. Furthermore, communication costs are irrelevant in traditional mechanism design and the communication channels are assumed to be faultless. These assumptions are problematic, in particular in the context of electronic markets being populated by software agents, as [7]:

- Agents do not have unlimited memory and computational power.
- Electronic marketplaces are open and unpredictable environments in which agents may cease to operate for a number of reasons.
- In such open systems with heterogeneous agents, a machine-understandable specification of all the associated interaction protocols and associated rules cannot be guaranteed.
- Centralized mechanisms may be unable to compute the outcome because the problem might simply be intractable.
- Communication does not come for free and may not be faultless in a computational setting. Moreover, given the inherent heterogeneity, semantic interoperation between all agents cannot be taken for granted.

Therefore, although traditional mechanism design offers us insights into designing protocols for agent negotiation, its applicability to complex scenarios is inherently limited. The complexity of electronic markets may place them beyond the analytical power of mechanism design. But, even in apparently simple environments, using simple mechanisms or protocols without prior careful design and experimentation, can unfortunately lead to expensive failures or inefficient outcomes, as the conduct of the 3G mobile phone license

auctions in some countries in Europe has amply demonstrated [16]. Consequently, developing markets without careful consideration and experimentation can be costly and carries high risks. Moreover, testing trading agents and strategies in real life complex markets is difficult, impractical and also carries high risks. Individuals, businesses and organizations need inexpensive and safe ways to evaluate the appropriateness and applicability of protocols in particular situations as well as the effectiveness and robustness of strategies.

One approach to address these problems is to implement and experiment with market simulations. The intrinsic value of simulations in other domains such as medicine, military applications and education is well-established and accepted. Market simulations offer a number of advantages. Firstly, they allow various parties to test market mechanisms and strategies in a safe environment. For instance, an organization which considers creating an electronic marketplace and using a particular auction as the negotiation protocol of choice can test its appropriateness in this particular domain by experimenting through simulation. The behaviour of participants and their strategies in the marketplace can also be simulated. The advantages and vulnerabilities of strategies can be assessed as well as the impact of using different strategies in the marketplace as a whole. The effectiveness of strategies in different market settings can also be explored. Secondly, using market simulations is also a relative inexpensive means of testing protocols and strategies. When one considers the potential cost of using inappropriate strategies in a marketplace or a negotiation protocol that may be easy to manipulate, experimenting and testing using simulations can significantly reduce the risk of expensive failures. Thirdly, employing market simulations enables the testing of different market settings and configurations and the testing of even extreme conditions and how these affect both the marketplace as a whole as well as individual participants. Extreme even malicious behaviours can also be simulated. In essence, simulations offer a powerful tool which together with formal analysis can provide us with valuable insights into the workings of market infrastructures, the use of protocols as well as the effectiveness of strategies.

Simulated markets may offer the only way to actually approach these problems in a systematic way. Guided first by theory and following a detailed analysis of a given domain, one can proceed to choose appropriate negotiation protocols and then design and implement a market simulation, and finally verify the appropriateness of the protocols chosen through experimentation. The same applies to negotiation strategies.

However, there is one inherent limitation in developing and experimenting with simulations on one's own. The results of such experiments conducted by a single person or organization may lack the touch of realism, since the bidding strategies explored may not necessarily reflect diversity in reality. This is the central idea and major motivation behind the International Trading Agent Competition [30] which features artificial trading agents competing against each other in market-based scenarios. Currently two scenarios are available

and run as part of the competition: the Supply Chain Management game and the Market Design game. Ideally, we would like open-source platforms where market-based scenarios could be implemented and researchers have the opportunity to build and test their trading agents against those of others. Such platforms or tools that would allow researchers and developers to design and implement their own marketplaces are currently lacking.

7 Multi-Agent Trading Platforms

One of the first electronic marketplace systems developed was Kasbah [6] in which users were able to create buyer and seller agents that could trade goods on their behalf. When creating a new buying (selling) agent in Kasbah, the user had to define a set of parameters that guided the agent's behaviour. These parameters included:

- desired date to buy (sell) the good by;
- desired price, i.e. the price that the user would like to buy (sell) for the good;
- highest (lowest) acceptable price, which represents the highest (lowest) price that the user is willing to pay (accept) for the good.

The user controlled the agent's behaviour by specifying its "strategy" which was described by the type of price-raise or price-decay function for buyer and seller agents respectively. A price-raise (price-decay) function specified how the agent would raise (lower) its price over time and there were three functions available: linear, quadratic and cubic. Once agents were created, they posted offers in the marketplace and then they waited for other interested parties to contact them. The role of the marketplace was to facilitate connections among agents. For instance, once an agent posted an advert for selling a good, the marketplace returned a list of buyer agents that were interested in the same good as well as notified the respective buyers that a new seller had entered the market. Subsequently, interested parties communicated and negotiated for deals directly. Users had the final say in authorizing the deals that their agents had reached after negotiation. Although users could create new buyer and seller agents, the functionality and strategy of these were completely determined by Kasbah, and users were not able to implement their own agents and plug them into the marketplace.

Another prototype electronic marketplace was MAGMA [32]. The idea behind MAGMA was that of providing the necessary infrastructure and services for building virtual marketplaces. Such services included banking and secure payments, mechanisms for advertising goods as well as transportation of goods. The MAGMA marketplace consisted of the following: trader agents, an advertising server, a relay server and a bank. The trader agents' role was to engage in negotiations to buy and sell goods and services. The advertising server provided advertising and retrieval services while the relay server

facilitated communication between the different components in the system. The bank provided basic banking services. The negotiation mechanism used in MAGMA was the Vickrey auction. Unlike Kasbah, agents could be written in potentially different languages as long as they complied with the MAGMA API.

The AuctionBot system was implemented at the University of Michigan as a tool to support research into auctions and mechanism design [34]. AuctionBot was a flexible, scalable, robust auction server capable of supporting both human and software agents. The AuctionBot system had two interfaces: a Web and a TCP/IP interface. The Web interface allowed human agents to register, create, monitor and participate in auctions via web forms. The TCP/IP interface allowed software agents to connect and participate in auctions. The database was used to store the bids submitted by both human and artificial agents. A scheduler continually monitored the database for auctions that had events to process or bids to verify, operating in essence as a daemon process. An auctioneer process would load the auction parameters and the set of current bids from the database and would then validate bids as necessary and could clear the auction or provide price quotes.

A number of auctions were implemented and the system was capable of running many auctions simultaneously. These auctions could be parameterized offering great flexibility. Participation in auctions for trading discrete goods could be on the following basis: {1 seller: many buyers}, {many sellers: 1 buyer}, {many buyers: many sellers}. The system revealed information by providing bid and ask quotes during auctions, and information about past transactions and auctions could also be made available. The auctions' closing time and the timing of clear and/or quote events were among the parameters that could be determined. The Mth and $(M + 1)$st price rules were used as the basis for all auctions. In particular the implementation of the Mth and $(M + 1)$st rules was based on the 4-heap algorithm [33]. Agents could be written in different programming languages.

The Trading Agent Competition (TAC) [31] is a non-profitable organization whose aim is to promote research into trading agents and electronic markets. The purpose of the competition is to stimulate research in trading agents and market mechanisms by providing a platform for agents competing in well-defined market games with an emphasis on developing successful strategies for maximizing profit in constrained environments. The competition has been running since 2000 and for the first three years it was based on the AuctionBot infrastructure. Since 2003 the underlying infrastructure for the games and the competition has been provided by a team of researchers at the Swedish Institute of Computer Science. The first TAC game was the Travel Agent Game. The market-based scenario featured in TAC travel was that of a travel agent that had to assemble travel packages for a number of clients. Although constructing an agent to take part in an auction for a single good is relatively simple, developing an agent to participate in simultaneous auctions offering complementary and substitutable goods is a complex task. This was

the form of the problem that agents had to face in the TAC Travel game. This game was discontinued after 2006. An alternative implementation of the TAC Travel Agent game was provided within the Agentcities initiative which uses a distributed peer-to-peer approach based on FIPA standard infrastructure components, protocols and languages [36].

Currently, TAC operates two games: the supply chain management game (SCM) and the more recent, introduced in 2007, Market Design Game. The first game simulates a supply chain management environment. Traditional SCM deals with the activities within an organization that range from procuring raw materials, manufacturing, to negotiating with customers and acquiring orders from them, and delivering finished products. In today's highly interconnected and networked world, more and more businesses and organizations choose to do business online. This is a dynamic environment where manufacturers may negotiate with suppliers on the one hand, while at the same time compete for customer orders and have to arrange their production schedule and delivery so that customer orders are delivered on time. The ability to respond to changes as they occur and adapt to variations in customer demand and the restrictions as imposed by procurement is of paramount importance. This is the kind of environment that agent technology is best suited for. The TAC SCM game was designed to capture many of the dynamics of such an environment. In the second game, agents are in essence brokers whose goal is to attract potential buyers and sellers as customers and then match these. The interested reader can find more details about TAC and the market games at [31]. Although to some extent the TAC servers that implement the various games are configurable, this is limited to changing certain parameters and there are no provisions for developing games in a systematic way.

One of the first experimental agent-based financial market frameworks was the Santa Fe Artificial Stock Market Model [1, 18].

An alternative approach to developing e-commerce simulation games which builds on and extends the Zeus framework was proposed in [14].

8 e-Game

The **e**lectronic **G**eneric **a**uction **m**arketplac**e** (e-Game) is a configurable auction server that enables both human and artificial agents to participate in electronic auctions [12]. The platform can be extended to support many types of auctions. e-Game has also been designed to support market simulations analogous to those of TAC which can be designed and developed by third parties.

The platform has been developed in JAVA in order to gain the benefits of platform independence and transparency in networking communication and object handling. e-Game is based on a modular architecture separating the interface layer from the data processing modules and the database (Figure 2). The database is used to model the auction space (auction type, startup, closing

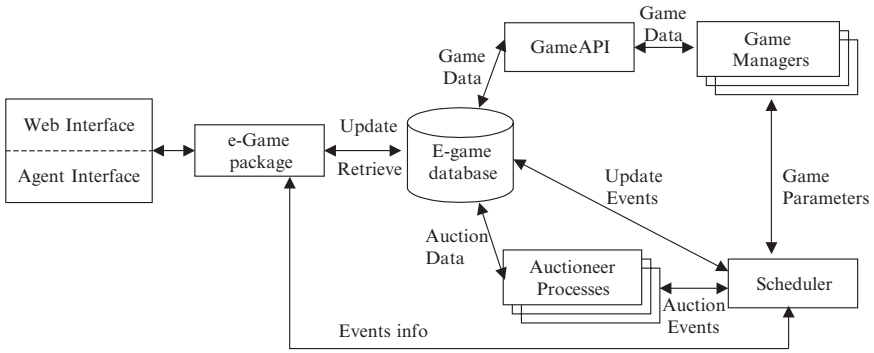

Fig. 2. The architecture of e-Game

time, clearing rules), as well as the user data (registration, user bids, auctions the user has created). It also stores information regarding the market simulations (which game started which auctions, ids of players, scores). It provides fault tolerance, in that the events queue can be reconstructed if there is a problem with the *Auctioneers* or the *Scheduler*. Its modularity and expandability are further strengthened by adopting the object-oriented paradigm for its components.

8.1 Interfaces

e-Game is designed to enable both web users and software agents to participate in electronic auctions. The interface to the outside world consists of two components: the *Web* and the *Agent* interfaces, similarly to [34].

The *Web* interface enables human users to interact with e-Game and provides a series of functions such as registration, creation of new auctions using a variety of parameters (auction type, information revelation, clearing and closing mechanism), bid submission, search facilities according to a series of attributes (auctioned item, status of auction, user's participation in an auction) and viewing the user's profile (change personal information, view previous auction information).

The *Agent* interface enables software agents to interact with the platform and provides agent developers the same functionality with the *Web* interface with respect to auctions plus an additional set of commands for participating in market games. Agents connect to the *Agent* Interface using the TCP protocol and submit their commands using the FIPA ACL language [13]. A subset of the FIPA ACL performatives have been implemented in e-Game to make requests (*request*: when an agent submits a bid), communicate the results of actions (*inform*: when e-Game informs an agent about the result of its bid submission) and refuse requests (*refuse*: when an invalid action has been requested). The actual content of the message follows the FIPA Semantics

Language, which is formally defined in [13]. Nevertheless, a different content-language can be used, for example XML, by developing the appropriate parser and plugging it in, in the existing system. Adopting the FIPA ACL language contributes towards common standards among different agent platforms – if agent technology is to become widely adopted, it is necessary for agents (their developers) to adopt a common communication protocol. For every command an agent submits, e-Game returns an appropriate result together with a command status value that indicates the outcome of the command.

As the majority of the functions that the web users and agents perform are identical, they are implemented in a single package which is shared between the two interfaces. This contributes towards extensibility and also makes the maintenance of the system easier.

8.2 Scheduler

The *Scheduler* runs in parallel with the *Web* and the *Agent* interfaces and is responsible for starting up auctions, games and passing bids to the appropriate *Auctioneer* processes. The auctions can be scheduled by web users or by a specific *GameManage*r handler according to a market game's rules, whereas games are scheduled by web users. The *Scheduler* provides a fault-tolerant behaviour, since if it is brought off-line for some reason, when it is restarted it can reconstruct its lists of events by looking into the database for pending events.

8.3 Auction Support

Currently e-Game supports a wide range of auction types (**Table 2**). These basic types of auctions can be further refined by allowing the user to define additional parameters that include [35]:

- Price quote calculation: Upon arrival of new bids, at fixed periods or after a certain period of buy/sell inactivity.
- Auction closure: At a designated date and time or after a period of buy/sell inactivity.

Table 2. Auctions supported by e-Game

Auction type	Sellers	Buyers	Units
English	1	Many	1
Vickrey	1	Many	1
FPSB	1	Many	1
Dutch	1	Many	1
Continuous Single Seller	1	Many	Many
Double	Many	Many	Many

– Intermediate clearing: Upon arrival of new bids, at fixed periods or after a certain period of buy/sell inactivity.
– Information revelation: Whether the price quotes and the closing time are revealed or not.

Such a parameterization can have an effect on the users' or agents' behaviour, since they modify the basic functionality of an auction as well as change the amount of information revealed regarding bids and their status.

Auctions are controlled by the corresponding *Auctioneer* classes. Each *Auctioneer* is started by the *Scheduler* and implements a specific auction protocol. Therefore, there are as many *Auctioneer* classes, as the number of auction protocols supported by e-Game. *Auctioneers* inherit the basic behaviour and state from a *BasicAuctioneer* class. Once an auction starts, the *BasicAuctioneer* class is responsible for generating the auction's events based on the choices the user made during the setup phase. There may be a number of similar or different classes of active *Auctioneers* within the e-Game platform, each one handling an auction with a unique id at any one time.

The implementation of all auctions is based on the Mth and $(M+1)$st price clearing rules. The clearing rules for the auctions are just special cases of the application of the Mth and $(M+1)$st clearing rules as described in section 5.3. However, much depends on the initial set up performed by the user. For instance, to achieve a chronological order clearing in an auction one can set up the intermediate clearing periods to take place as soon as a bid is received and thus the parameterized auctioneer will attempt to perform intermediate clearings upon the arrival of a new bid. As in classical chronological matching, if a portion of the bid cannot transact, then it remains as a standing offer to buy (or sell). Hence, another important aspect of auction implementation has to do with the processing of bids and in particular the states that a bid can be in. The transition diagram of Figure 3 illustrates bid state information in e-Game. The implementation of the *Mth* and $(M+1)$st clearing rules uses ordered lists.

8.4 Game Management

Game management is facilitated through the *GameManager* class which provides the functionality for the different installed games. There are as many different *GameManagers* as the number of installed games. In the majority of the cases a *GameManager* needs to provide the functionality for the following phases of a game:

1 The initialization phase in which initial resources and future auction events are generated. The resources are allocated to the agents via the TCPServer and the auction events are placed in the *Scheduler*'s events list.
2 The control phase allows one to control certain aspects of the game in an agent-like manner. This means that the developer can implement certain behaviour in their *GameManager* serving the scenario's purposes.

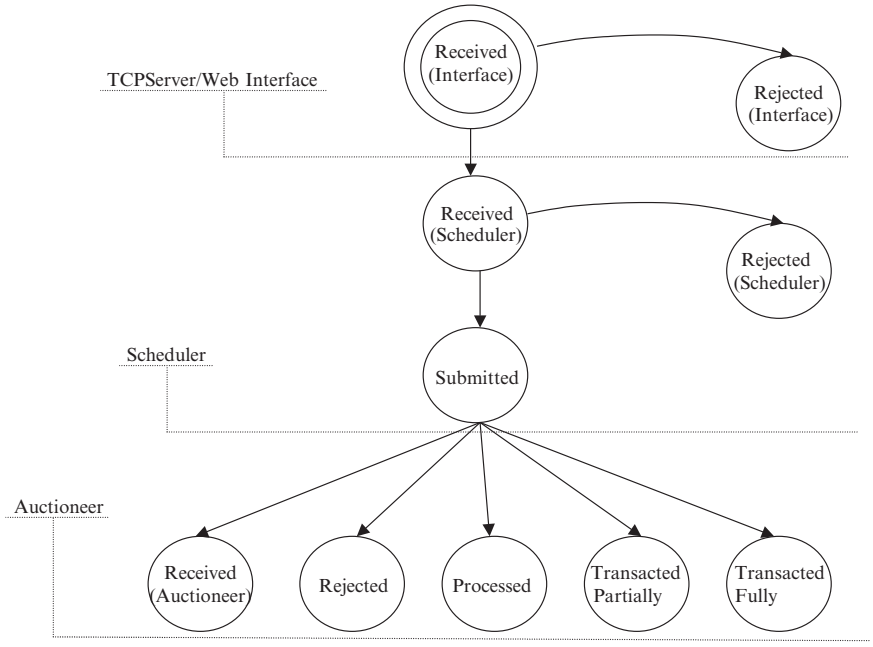

Fig. 3. Bid state transition diagram

For example, it may be appropriate to ensure a ceiling in the prices of some auctions, or buy certain goods for which agents show no preference or trigger events when certain conditions arise in a game.

3 The closing phase, during which the *GameManager* calculates the scores of the participating agents and stores the results in the database for future reference.

9 Developing Trading Games

Apart from the support for agents and web users regarding auction operations, e-Game's main feature is that it supports the design, development and execution of different market simulations or games that involve auctions analogous to those of the Trading Agent Competition.

Market simulations can be distinguished into three categories [14]:

1 Primitive markets in which buyers and sellers interact directly with each other. There are no intermediaries in such markets.
2 Facilitated markets in which there are middle agents such as auctioneers, banks etc. present in the market.
3 Regulated and embedded markets in which there are regulator agents that attempt to control the various interactions among agents having in mind

the social welfare and not only the welfare of individual agents. There may be agents that vary some variables such as availability of goods in the market and interest rates such that they affect the behaviour of others.

We are more interested in the latter two types of market simulations as these can be used to emulate real life complex situations of strategic interdependence among multiple agents.

9.1 Desirable Features

Trading games as other types of games need to be designed carefully in order to be successful, engaging and provide a challenge to the participating players. In particular, when designing a new market game one should have in mind the following key points:

Realism. The more realistic the market simulation is, the more comprehensible it will be to the people who wish to develop agents to participate in it. Ideally, the auction-based market scenario should describe a realistic situation, which may be encountered as part of everyday life (holiday planning, buying interrelated goods, scheduling of resources).

Strategic challenge. The game should present difficult strategic challenges that artificial trading agents may encounter and which require complex decision making. It is up to the agent developers to incorporate and experiment with computational intelligence techniques in their agents, but it is up to the game developer to provide challenges as part of the game.

Efficient use of e-Game. Ideally, the game should not be based on a single type of auction: there is range of auctions to choose from which can be further parameterized. Using different types of auctions provides more challenges for the participants who will have to lay out different strategies depending on the auction type and the amount of information that is revealed at run time.

Fairness with regards to the game specification. In a market game different agents with most probably different strategies will try to accomplish a certain goal. Though the goal will be similar in concept for every agent (for example, assemble one computer by combining different resources), the achievement of individual goals will give different utilities to each agent. One should consider specifications that give everyone the opportunity to come up with the maximum utility.

Fairness with regards to agent design. Agents with a more appropriate bidding strategy should manage to achieve a better outcome, than agents with "naïve" and "simplistic" strategies. When "naïve" strategies actually represent corresponding actions in real life (for which people can actually reason), this prerequisite does not hold. However, the purpose of introducing this key point is that there should not be any flaws in the market game that would allow non-rational strategies to outperform rational ones.

Computational efficiency. The *GameManager* of a market game should have the ability to operate in a real-time manner. Normally, there should not

be any algorithms that take a long time to run, especially at critical points in the game. It is acceptable to have such algorithms at the end of the game when an optimal allocation may be required and scores have to be calculated.

Communication efficiency. As little communication as possible should be required in order to reach a desirable outcome.

It is the developer's responsibility to ensure that a game is realistic, fair and non-trivial. Transparency in e-Game is enforced by publishing the logfiles and outcomes of auctions.

Since the e-Game platform allows for web users as well as agents to participate in auctions, it is feasible for everyone to participate in a market game as well. This may sometimes be desirable when comparisons between humans and software agents need to be drawn with respect to laying out a strategy to maximize utility. Moreover, simulations can be run with human agents in order to test selected predications of economic and game theory and psychology and the effects of particular mechanisms and conditions in the market on participants and their strategies.

9.2 Game Design and Development

When designing a market game one often starts with some basic idea of a strategic situation, perhaps inspired by real life and then the details of the game are added in and clarified progressively. The design of the game may have to go through several stages of refinement until all the details are fleshed out.

One of the first issues that needs to be clarified is the objective of the game. In other words, what is it that participants are trying to achieve in the market game and how success is measured. The next issue to be considered concerns the goods, services or other resources that are available in the market and which of these are initially owned by the agents and which are those that can be obtained or traded. The availability of these goods or resources in the market has to be limited so as to encourage competition among agents. This may have to be considered in relation to the number of players participating in a game. The mechanisms that enable the exchange of goods and services in the game need to be decided next. This aspect of game design needs particular attention and obviously needs to be underpinned by theoretical results in mechanism design. For instance, in this stage, decisions regarding what types of auctions will be used and how much information will be revealed to the participants and at which point in the game need to be taken. Deciding if auction closure will be revealed to the participants is an important issue, as in electronic auctions this may result in sniping, i.e. bidders may delay the submission of their bids until the auction is about to close. Bidders may resort to sniping in order to avoid revealing their preferences or their identities early in the auction or avoid driving the prices high quickly, in the hope that ultimately they will get the good at a low price. Furthermore, it is important to consider how competition is induced through the use of different auction

protocols and how these affect the participants' behaviour in the game. The game duration needs to be given consideration as well. Relatively simple games may last a few minutes, whereas more complex market games can last longer. In any case, a complex game should not last longer than 60–70 minutes so as to enable participants to play a substantial number of games against various other players to allow them to draw conclusions on their approaches and strategies. Complex games may simulate the passing of time in terms of days and during each day a sequence of activities may have to be carried out by the agents. For instance, information on the availability of goods may be posted in the beginning of the day, while negotiations may follow, and transactions may have to go through by the end of the simulated day.

In terms of implementation, *GenericGame* needs to be extended by the game developers who will have to implement or override a series of abstract methods which are related to generating the each time parameters for the agents that participate in the game, implement the rules of the game, monitor its progress and finally calculate and generate scores. In general, a game involves the following phases:

1 Set up. In this initial phase of the game the developer sets up the game environment and specifies the auctions that are to be run and their parameters.
2 Generation of parameters. These may include client preferences, initial endowments, other information regarding the state of the market, scheduled auctions, availability of goods and other resources and any other data the game developer wants to communicate to each agent in the beginning of the game. The agents are responsible for retrieving this information and using it.
3 Execution. e-Game runs the actual auctions that are part of the game and may also simulate other entities in the market according to the *GameManager* controlling the game. Agents are allowed to submit bids according to the rules of the game. Information is revealed according to the parameters specified in the setup. The agents are responsible for retrieving information on the status of their bids and using this in their decision-making process.
4 Score calculation. Success is measured in terms of a utility function and although this depends on the strategy that an agent is using, the utility function itself needs to be provided by the game developer.

Each game may involve a number of players/agents. The developer may also consider providing a sample agent that the participants can use as the basis to build on. This can also act as a "dummy" to fill in one or more of the available player slots when a game runs and not enough agents have registered for playing. Agent developers will want to be able to follow the progress of their agents in a game. Therefore, a game developer should consider providing the means to facilitate this. Along with the respective *GameManager*, a game developer may also provide an applet to be loaded at runtime which

graphically represents progress during the game, so that participants and also others can follow it.

In order to integrate a new game into the e-Game platform one needs to provide the class file, together with an XML file that describes general properties of the game such as the name of the game, the name of the implemented classes (game, applet), a short description for the game, its duration in seconds, the number of allowed participants and the resources (items) that this game uses. This information is then parsed by the *GameImporter* application and is stored in the database. Web users can then browse the web site and following the link *Agent Games* they can view all the installed types of games, together with appropriate links to schedule new instances, watch a current game and view previous scores. At the end of a game, participants can view scores and resources obtained by each agent. When a new game is scheduled, the *Scheduler* receives the corresponding event and at the appropriate time loads the user defined *GameManager* class.

9.3 The Computer Market Game

To provide the reader with an idea of the type of market simulations that can be created using e-Game we will briefly discuss one such market simulation[1]. In the Computer Market Game (CMG), which lasts nine minutes, each of the six agents participating is a supplier whose task is to assemble PCs for its five clients.

There are only three types of parts that are necessary to make up a properly working PC: a motherboard, a case and a monitor. There are three types of motherboards, each one bundled with CPUs of 1.0 GHz (MB1), 1.5 GHz (MB2) and 2.0 GHz (MB3). There are two types of cases, one with a DVD player (C1) and the other with a DVD/RW drive (C2). There is only one type of monitor.

In the beginning of the game the agents receive their clients' preferences in terms of a bonus value for upgrading to a better motherboard (MB2 or MB3) and a better case (C2). An example set of preferences for an agent A is given in **Table 3.** Hence, client 2 of agent A offers 115 monetary units for motherboard MB2 and 175 for the MB3 one. For obtaining the better case (C2) the client offers 290 monetary units. e-Game generates values in the following ranges for these goods: MB2 = [100...150], MB3=[150...200], C2=[200...300].

Components are available in certain quantities and are traded in different auctions (**Table 4**). Figure 4 illustrates how the auctions for the various items are scheduled during the 9-minute period of the game. The dotted part of the lines indicates that the auctions may close anytime during that period, but the exact closing time is not revealed to the agents.

[1] Note: this is a simple market simulation which has been developed for teaching and not research purposes.

Table 3. An example of client preferences

Agent	Client	MB2	MB3	C2
A	1	120	165	234
A	2	115	175	290
A	3	140	190	219
A	4	145	187	270
A	5	135	164	245

Table 4. Availability of goods and auctions in the CMG game

Component	Quantity	Auction
MB1	17	Mth Price
MB2	8	Mth Price
MB3	5	Mth Price
C1	20	Mth Price
C2	10	Mth Price
M	30	Continuous Single Seller

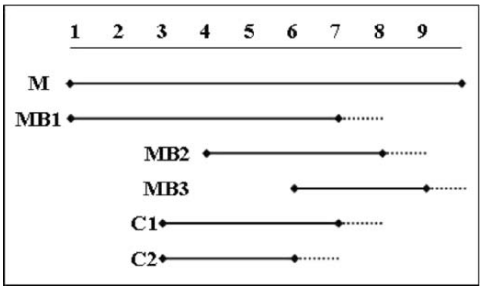

Fig. 4. The auctions' schedule in the CMG game

An agent's success in this game depends on the satisfaction of its clients. An individual client i's utility (CU_i) is:

$$CU_i = 1000 + MotherboardBonus + CaseBonus \tag{1}$$

For each client that has been allocated a fully-assembled PC, the agent gets 1000 monetary units plus any bonus for upgrading to a better motherboard or case. If no fully-assembled PC is allocated to a client, then this utility is 0. For every extra component purchased which exceeds the quantity needed to satisfy its clients the agent has to pay a penalty which is perceived as a storage cost and is determined at the beginning of the game as a random amount between 150 and 300 units. This has been added as a disincentive to buy extra items surplus to requirements. An agent's utility function is the

sum of all the individual client utilities minus the expenses (*Expenses*) and the penalties (*Penalties*) incurred:

$$AU = \Sigma(CU_i) - Expenses - Penalties \tag{2}$$

The agent's strategy should be focused on providing a fully-assembled PC to each one of its clients or to as many as possible, while at the same time trying to minimize costs. There are obvious interdependencies between goods, as a fully-assembled PC requires three components. In creating a strategy for this game, one has to take into account that the availability of goods is limited, prices in auctions may vary, auctions close at different times and therefore one may have to switch to a different auction if they fail to acquire a certain good, and customers give different bonuses for upgrading to a better specification. Moreover, an agent's success does not only depend on its own strategy, but crucially that of the other agents too. Nevertheless, an agent does not have any means of knowing the identity of the other players or monitoring their bids directly.

The original version of the CMG game did not include any penalties. In the process of using the game with a group of students who had to develop agents as part of their coursework, we discovered that a significant number of agents were buying more than they needed in an effort to deprive the market of various goods and therefore lower the utility of the other participants. To this end, we extended the agent's utility function to account for penalties in the form of storage costs for extra components. This is an example where experimentation has revealed a flaw in the design of the market game and which had to be rectified.

In order to implement the above scenario the developer would have to provide an XML file describing the general properties of the game (names of game and applet classes, number of players, duration of the game, auctioned resources), together with the appropriate classes. The specific *GameManager* would schedule the auctions at the beginning of the game by implementing the method *startupGame* and using the *scheduleAuction* methods. The next step would be to generate the parameters that each agent receives by implementing the method *generateParameters*. The utility of each agent would be calculated at the end of the game by examining the winning bids of each auction. Finally, any language resources that the developer used would be freed-up in the *closeGame* method. In addition, the developer may choose to write an applet so that web users can view the progress of their agent. Such an applet for this simple game is illustrated in Figure 5.

A number of other games have also been developed based on the e-Game infrastructure by us but also by students. All these games are publicly accessible from e-Game's website[2].

[2] http://csres43:8080/egame/index.jsp and http://sh718:8080/egame/index.jsp

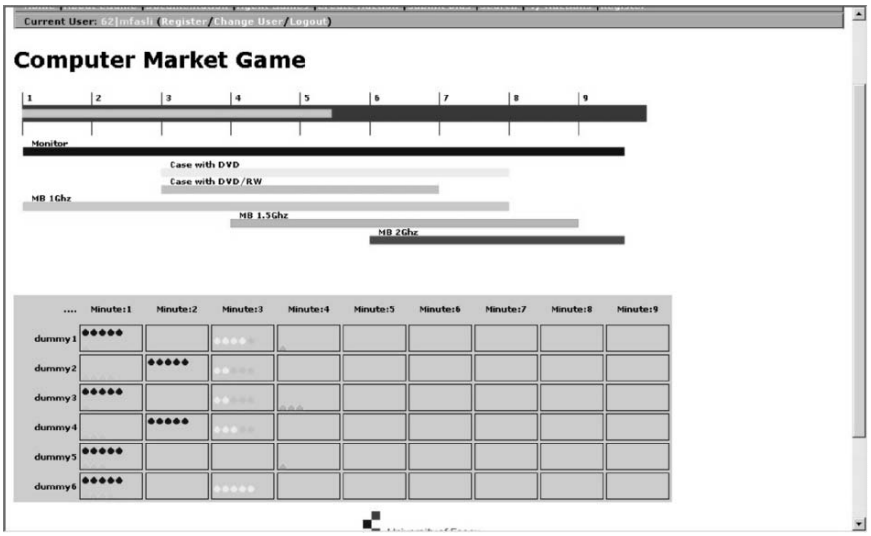

Fig. 5. The CMG applet

10 Further Work

e-Game can be further extended in a number of ways. A possible direction is with regards to multi-attribute auctions. In a multi-attribute auction, the auctioneer lists a number of negotiable attributes and bidders are asked to submit multi-dimensional bids. Multi-attribute auctions are often used in selecting suppliers for various services in the public sector. The heterogeneity of the negotiable attributes often calls for more consideration during the bidding phase, since bidders do not only compete on price, but also on other features such as quality, additional features on the requested item, availability, extended warranty, provision of additional resources and more. The inclusion of such a feature increases the complexity of the implemented marketplaces, since evaluating heterogeneous bids is a complex problem. There have been works that suggest ways to maximize the auctioneer's turnover [5]. This extension also contributes to increasing the complexity on the bidder's side (game participants) that now need to consider a number of attributes when submitting their bids and therefore provides more opportunities to agent developers to include computational intelligence techniques in their agents. The inclusion of such a feature is non-trivial, but it will provide insight into the topic of multi-attribute auctions.

In a fully automated marketplace, agents are not only concerned with the process of negotiation, but also proactively seek opportunities to trade. To enable agents to search for scheduled auctions or even set-up auctions, the *Agent* interface can be extended in order to provide the same functionality to software agents that the *Web* interface provides to humans. We are also

considering allowing agents to initiate auctions during a game and act as sellers of goods that they own, or have previously acquired through trading, but no longer desire or they are surplus to requirements.

Although e-Game's components were developed using Java, a web services interface can also be built that would allow game developers to have access to e-Game's API using the programming language of their choice.

Finally, we are considering the possibility of introducing a rule-based specification such as [19] in order to describe complex market-based scenarios.

11 Conclusions

Although agents and multi-agent systems are regarded by many practitioners, researchers and technologist analysts as the technology that has the potential to revolutionize electronic commerce, so far their immense potential has not been fully realized. This chapter discussed how some of the obstacles of deploying agent technology in e-commerce applications can be overcome by developing electronic market games, that can be used to conduct research on market infrastructure, negotiation protocols and strategies.

e-Game is a generic auction platform that allows web users as well as agents to participate in auctions and auction-based market games. The aim of the e-Game project is to offer the infrastructure for running complex market simulations and conducting experiments with different bidding strategies and enable agent developers to incorporate and experiment with computational intelligence techniques. The most important feature of e-Game and what distinguishes it as a platform from other efforts such as [32, 6, 34, 31, 15] is that it provides independent developers the facilities to design, implement and run auction-based market simulations. Unlike JASA [15] for instance, e-Game does not simply provide the facilities for scheduling, running or conducting experiments on the use of strategies in standalone auctions. Individual users/developers can write their own *GameManager* and define complete market scenarios together with accompanying applets and sample agents that can be inserted and run on top of the e-Game infrastructure. The development of these modules is completely separate from the rest of the system. By making use of existing classes and implementing simple interfaces, users can develop new games quickly. The progress of integrating new games with e-Game is fully automated: since the user developed classes implement certain (simple) interfaces and inherit from existing classes, e-Game can access these methods without needing to know the rest of the user-defined methods that implement the game's rules.

Moreover, e-Game provides the facility of having humans participate in complex market simulations as well as humans playing against software agents. This may be useful in designing strategies and examining strategy efficiency and complexity. One could study the benefits from using agents in complex markets in a systematic way. For example, in a simple game, a human may

do better than an agent, but it would be interesting to pit humans against software agents and study what happens when the scenario gets more complicated (penalty points, allocation problems etc). A suitable version of the CMG game for human users has already been developed and a number of laboratory-based experiments have been conducted.

Apart from using the platform for designing and running complex market games, it can also be used for teaching negotiation protocols, auctions and strategies. Students can have a hands-on experience with a number of negotiation protocols and put into practice the principles taught.

Although the work presented here goes some way towards addressing some of the issues of deploying agents in e-commerce applications, we should not forget one other fundamental issue: users need to trust software agents in order to be able to confidently delegate to them the tasks of negotiating and reaching deals on their behalf and they also need to be assured that any legal issues relating to agents trading electronically are fully covered, just as they are in traditional trading practices [11].

12 Resources

Key books

- Baba, N. and Jain, L.C., Computational Intelligence in Games, Springer, 2001.
- Fasli, M. Agent Technology for E-commerce. John Wiley and Sons, 2007.

Key papers and other resources

- Agorics. Auctions. http://www.agorics.com/Library/auctions.html
- Griss, M. and Letsinger, R. Games at work-agent-mediated e-commerce simulation. In Proceedings of the 4th international conference on Autonomous agents (Agents 00), Barcelona, Spain, 2000.
- Sadeh, N. M., Arunachalam, R., Eriksson, J., Finne, N., and Janson, S., TAC'03: A Supply Chain Trading Competition, AI Magazine, 24(1):92-94, 2003.
- Wellman, M. P., Wurman, P.R., O'Malley, K., Bangera, R., Lin, S.-D., Reeves, D. M., and Walsh, W.E., Designing the Market Game for a Trading Agent Competition, IEEE Internet Computing, 5(2):43-51, 2001

Organizations, Societies, Special Interest Groups, Projects

- IEEE Specialist Interest Group in Computational Intelligence in Games, http://ieee-cs.org/games/
- Trading Agent Competition (TAC) Organization. http://tac.eecs.umich.edu/association.html
- Agentcities Project, http://www.agentcities.org/

Key International Conferences/Workshops

- Agent-Mediated Electronic Commerce Workshop (AMEC)
- Autonomous Agents and Multi-agent Systems Conference (AAMAS)
- IEEE International Conference on Electronic Commerce (IEEE ICEC)
- IEEE Symposium on Computational Intelligence in Games (IEEE CIG)
- Trading Agent Design and Analysis Workshop (TADA)

Software

- electronic Generic Auction Marketplace (e-Game) servers: http://csres43:8080/egame/index.jsp and http://sh718:8080/egame/index.jsp
- Trading Agent Competition Servers: http://www.sics.se/tac/
- TAGA and Agentcities, http://taga.sourceforge.net/doc/agentcities.html

Acknowledgements

The introduction to negotiation protocols and auctions is based on Fasli, M. Agent Technology for e-Commerce, © John Wiley and Sons, 2007. Reproduced with permission.

References

1. Agorics. Auctions. http://www.agorics.com/Library/auctions.html.
2. Arthur, W. B., Holland, J., LeBaron, B., Palmer, R. and Tayler, P. Asset pricing under endogenous expectations in an artificial stock market. In Arthur, W. B., Durlauf, S. and Lane, D., editors, The Economy as an Evolving Complex System II, Addison-Wesley Longman, Reading, MA, pages 15–44, 1997.
3. Bakos, Y. The emerging role of electronic marketplaces on the internet. Communications of the ACM, 41(8):35–42, 1998.
4. Bichler, M. A roadmap to auction-based negotiation protocols for electronic commerce. In Hawaii International Conference on Systems Sciences (HICSS-33), 2000.
5. Bichler, M. and Kalagnanam, J. Configurable offers and winner determination in multi-attribute auctions. European Journal of Operational Research, 160(2):380–394, 2005.
6. Chavez, A. and Maes, P. Kasbah: An agent marketplace for buying and selling goods. In First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'96), pages 75–90, 1996.
7. Dash, R. K. and Jennings, N. R. and Parkes, D. C. Computational-mechanism design: A call to arms. IEEE Intelligent Systems, 18(6):40–47, 2003.
8. de Vries, S. and Vohra, R. Combinatorial auctions: A survey. INFORMS Journal on Computing, 15(3):284–309, 2003.
9. Ebay. http://www.ebay.com/.

10. Fasli, M. Agent Technology for e-Commerce. John Wiley and Sons, Chichester, 2007.
11. Fasli, M. On Agent Technology for E-commerce: Trust Security and Legal Issues. Knowledge Engineering Review (in press), 2007.
12. Fasli, M. and Michalakopoulos, M. e-Game: A generic auction platform supporting customizable market games. In Proceedings of the IEEE/WIC/ACM Intelligent Agent Technology Conference (IAT 2004), pages 190–196, Beijing, China, 2004.
13. FIPA Communicative Act Library Specification. http://www.fipa.org/specs/fipa00037/, 2002.
14. Griss, M. and Letsinger, R. Games at work-agent-mediated e-commerce simulation. In Proceedings of the 4th international conference on Autonomous agents (Agents 00), Barcelona, Spain, 2000.
15. JASA: Java Auction Simulator API. http://www.csc.liv.ac.uk/∼sphelps/jasa/.
16. Klemperer, P. How (not) to run auctions: The European 3G Telecom Auctions. European Economic Review, 46(4–5):829–845, 2002.
17. Kumar, M. and Feldman, S. Internet auctions. In Proceedings of the 3rd USENIX Workshop on Electronic Commerce, pages 49–60, 1998.
18. LeBaron, B., Arthur, W. B. and Palmer, R. Time series properties of an artificial stock market. Journal of Economic Dynamics and Control, 23:(1487–1516), 1999.
19. Lochner, K. M. and Wellman, M. P. Rule-based specifications of auction mechanisms. In Third International Joint Conference on Autonomous Agents and Multi-agent Systems Conference (AAMAS), pages 818–825, 2004.
20. Maes, P., Guttman, R., and Moukas, A. Agents that buy and sell: Transforming commerce as we know it. Communications of the ACM, 42(3):81–91, 1999.
21. Mas-Colell, A., Whinston, M. D., and Green, J. R. Microeconomic Theory. Oxford University Press, Oxford, 1995.
22. Neumann, D. and Weinhardt, C. Domain-independent enegotiation design: Prospects, methods, and challenges. In 13th International Workshop on Database and Expert Systems Applications (DEXA'02), pages 680–686, 2002.
23. Nisan, N. Algorithms for selfish agents. In Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science, (STACS'99), LNCS Volume 1563, pages 1–15. Springer, Berlin, 1999.
24. Nisan, N. Bidding and allocation in combinatorial auctions. In Proceedings of the 2nd ACM Conference on Electronic Commerce (EC-00), pages 1–12, Minneapolis, MN, 2000.
25. Nisan, N. and Ronen, A. Algorithmic mechanism design. Games and Economic Behavior, 35:166–196, 2001.
26. onSale. http://www.onsale.com/, 2005.
27. Parkes, D. C. Iterative Combinatorial Auctions: Achieving Economic and Computational Efficiency. PhD thesis, University of Pennsylvania, 2001.
28. Pekec, A. and Rothkopf, M. H. Combinatorial auction design. Management Science, 49(11):1485–1503, 2003.
29. Sandholm, T. Distributed rational decision making. In G. Weiss, editor, Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, pages 201–258. The MIT Press, Cambridge, MA, 1999.
30. Sun, J. and Sadeh, N. M. Coordinating multi-attribute procurement auctions subject to finite capacity considerations. Technical Report CMU-ISRI-03-105,

e-Supply Chain Management Laboratory, Institute for Software Research International, School of Computer Science, Carnegie Mellon University, 2004.
31. Trading Agent Competition. http://www.sics.se/tac/.
32. Tsvetovatyy, M., Gini, M., Mobasher, B., and Wieckowski, Z. Magma: An agent-based virtual market for electronic commerce. Journal of Applied Artificial Intelligence, 6, 1997.
33. Wurman, P., Walsh, W. E., and Wellman, M. P. Flexible double auctions for electronic commerce: theory and implementation. Decision Support Systems, 24:17–27, 1998.
34. Wurman, P., Wellman, M. P., and Walsh, W. The Michigan Internet AuctionBot: A configurable auction server for human and software agents. In Proceedings of the Autonomous Agents Conference, pages 301–308, 1998.
35. Wurman, P. R., Wellman, M. P., and Walsh, W. E. A parameterization of the auction design space. Games and Economic Behavior, 35(1):304–338, 2001.
36. Zou, Y., Finin, T., Ding, L., Chen, H., and Pan R. TAGA: Trading agent competition in Agentcities. In Workshop on Trading Agent Design and Analysis held in conjunction with the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI), 2003.