Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Simulation Modelling Practice and Theory 13 (2005) 371–388
SIMULATION MODELLING PRACTICE AND THEORY
www.elsevier.com/locate/simpat

An agent-based simulation testbed for evaluating internet-based matching mechanisms

Marijn Janssen *, Alexander Verbraeck

Faculty of Technology, Policy and Management, Delft University of Technology, Jaffalaan 5, 2628 BX Delft, The Netherlands

Received 28 November 2003; received in revised form 29 November 2004; accepted 13 December 2004
Available online 1 January 2005

Abstract

The aim of the research presented in this paper is to develop a simulation testbed to support businesses in their decision-making about the potential use of electronic matching mechanisms. Matching mechanisms are used to match supply and demand of independent selling and buying organizations, each having their own goals, requirements and interests. The autonomous characteristics of agent-based systems and process orientation of discrete-event simulation are combined in our agent-based simulation testbed. In this way both the autonomous trading behavior of organizations and their business processes can be simulated. Several predefined components containing the behavior of various kinds of matching mechanism have been developed. These components can be further customized to model an empirical situation more closely. The approach is illustrated with a case study in the computer market.

Keywords: Agent-based simulation; Matching mechanisms; Trading

* Corresponding author. Tel.: +31 15 27 81140/83805; fax: +31 15 27 83741.
E-mail addresses: marijnj@tbm.tudelft.nl (M. Janssen), alexandv@tbm.tudelft.nl (A. Verbraeck).

1569-190X/$ - see front matter
doi:10.1016/j.simpat.2004.12.002

1. Introduction

Traders, which can be buyers, sellers and middlemen, want to buy or sell their products and services more and more over the Internet to increase the efficiency and effectiveness of trading. Electronic trade exchanges use all kinds of mechanisms to match supply and demand. The variety of matching mechanisms that can be chosen from, and the difficulty to predict behavior makes it difficult for traders to decide which mechanism would be preferable to use over the Internet for their particular situation. Traders do not want to experiment with new matching mechanisms in practice, as they are afraid to lose turnover and want to avoid the risk of becoming alienated from customers. Simulation seems to be an excellent tool to evaluate matching mechanisms for traders in compressed time while reducing costs and risk.

Negotiation protocols are the protocols that specify the kinds of deals buyers and sellers can make, as well as the sequence of offers and counter-offers that are allowed [41]. Buyers and sellers have a *trading strategy* to choose the appropriate mechanisms and the protocols. A strategy is the way buyers or sellers behave in an interaction [41]. Matching mechanisms are the problem solving approach of the traders having a trading strategy and willing to trade with each other using the negotiation protocols. A trading situation consists of heterogeneous types of actors, having their own goals, interests, and requirements. Actors can be small or large organizations; they can have various levels of automation, and a strategy to offer or to demand the lowest price or the best quality. Buyers and sellers can have different and even opposing requirements for matching mechanisms, as for example one party might want to minimize trading time, while the other might want to maximize the reliability of a chosen trading partner. The most conspicuous opposing requirement is that buyers want to have the lowest price at the best possible trading conditions while sellers want to have the highest possible price to maximize revenue. As such, it is essential for buyers and sellers to understand and evaluate matching mechanisms that can be used, as these should satisfy both customers and suppliers needs.

Trading situations are complex by nature, and analytic methods can only be applied in a limited way. Although analytical approaches contribute to insight into and design of matching mechanisms, they do not help traders to evaluate different matching mechanisms for the impact on the effectiveness and efficiency of their business processes. In practice all kinds of matching mechanisms can be used in accordance with traders' interests [5]. The English auction is often used to sell art, the Dutch auction is commonly used to sell perishables such as flowers or fish, and second-price sealed bid or Vickrey auctions are most often used in procurement situations. The English and Dutch auctions require intensive interactions to decrease or increase bids, while the Vickrey mechanism requires a more cautious bid, as bidding can only be done once and the second highest (or second lowest) bid is accepted. As a result it is argued that mechanisms need to be evaluated based on the accommodation of the special needs for a certain environment [15]. As such it is essential to capture the characteristics of the trading environment in the modeling efforts.

Traders are interested in the use of matching mechanisms to reduce trading time and costs, and to reduce possible negative effect of strategic behavior of buyers and

sellers, such as the effect of coalition formation to manipulate the market behavior, or the time delay between the start of the negotiation and the purchase or delivery of a product when there is an immediate need. The effect on the business processes of individual traders *and* the effect on the market behavior consisting of many traders interacting with each other are of importance for evaluating matching mechanisms.

In this research the autonomous characteristics of agent-based systems and the process orientation of discrete-event simulation are combined to develop an agent-based simulation testbed for evaluating matching mechanisms. This testbed aims at helping traders to understand the nature and consequences of the introduction of Internet-based matching mechanisms for their trading situation. In the following section we discuss the integration of the agent-based and the process oriented discrete-event simulation approaches. In section three we develop an agent-based architecture and in section four we define various kinds of matching mechanisms. These two elements make up our agent-based simulation testbed. We use the testbed for evaluating matching mechanisms in the computer market in section five, and we draw conclusions and make recommendations for future research in the final section.

2. Discrete-event simulation and multi-agent systems

The philosophy behind simulation is to develop a dynamic model of a system where we have identified an issue or a problem, experiment with this model, and experiment with alternative models for the system [31]. Simulation is performed prior to implementation and can be used to evaluate process design options. One of the advantages of simulation is that what-if analysis can be carried out without changing reality at lower costs [1].

Discrete-event simulation means that the time aspects of a sequence of discrete-events are modeled [25]. This is however, not enough in our case, as we have to find the 'right' models to describe not only the organizational processes for buyers and sellers, but also their trading strategies, and the possible matching mechanisms. On the one hand the most powerful models are the ones that minimize the semantic gap between the units of analysis that are intuitively used to conceptualize the problem and the constructs present in the modeling approach. On the other hand a key issue in modeling is reducing the complexity by eliminating the details that do not influence its relevant behavior [4].

Multi-agent systems (MAS) can exhibit characteristics of organizations, and sometimes of intentional organization design [3,14,27]. In analogy to MAS we take an agent-based approach to represent the decentralized nature of the problem, the multiple loci of control, multiple perspectives, and competing interests. Multi-agent systems approaches have already been applied in various disciplines [8,14,19,22,27].

During the last few years, there has been a considerable growth of interest in the potential of multi-agent technology in the context of software engineering. This has led to several architectures for multi-agent systems, including the ZEUS [24], FIPA-OS [7], RETSINA [33], LARKS [32], and dMARS [11] architectures. LARKS [32]

focuses on matchmaking in multi-agent systems and consists of service providers, services requesters and middle agents witch can advertise and request. The matchmaking is based on a combination of syntactic and semantic matching [32]. LARKS aims at developing a common language to deal with the complexity of heterogeneous agents that are incapable of understanding each other.

Our aim differs from the aim of many MAS, our agent-based simulation testbed is primarily aimed at supporting decisions-makers to evaluate matching mechanism prior to implementation in an efficient and fast way. In spite of the fact that agents form the basis of both agent-based simulations and multi-agent systems, there are several differences. In agent-based simulations, the agents are interacting in a simulated environment, where modeling reductions have been applied to the behavior of the environment. In that way, the behavior of a group of agents can be tested under various circumstances. Furthermore, the simulator provides an artificial time mechanism that allows the agent–agent and agent–environment interactions to take place (much) faster or slower than reality [40]. A fast mode is used to carry out experiments over time where different agent strategies are compared, while a slow mode enables humans to study the effects of different agent mechanisms in detail, using the information provided by visualization mechanisms within the agents. By placing the agents in an agent-based simulation, it becomes possible to study the design choices and the negotiation protocols and trading strategies, both in detail and over a prolonged period of time, where the experimental conditions can be easily changed.

None of the discussed multi-agent system architectures combines the time-based, discrete-event simulation of business processes, the trading behavior, and the autonomous behavior, to apply it to the evaluation of Internet-based matching mechanisms.

Wigand et al. [37] divide performance indicators for evaluating matching mechanisms into three categories; *market* efficiency, meaning the use of synergies through coordinated appearance on sales and supplier markets, *process* efficiency, meaning the optimization of the total process, and *resource* efficiency, meaning the avoidance of unused capacities and uneconomical allocation of scarce resources. The market efficiency is determined by the interactions between traders, the process efficiency is determined by the tasks making up the organizational business processes, and the resource efficiency is determined by the utilization of the resources of organizations. This means that in order to evaluate matching mechanisms in practice, the interactions between independent organizations, the business processes within organizations, and the utilization of resources need to be modeled.

3. An agent-based simulation architecture

Multi-agent systems are focused on building an information system instead of evaluating the impact prior to implementation; nevertheless a logical start for developing a simulation testbed is to investigate similar systems. Multi-agent systems are information system consisting of interacting software agents. Many different definitions of software agents can be found in the literature. e.g. [2,18,23]. The overuse of

the word 'agent' has tended to mask the fact that, in reality, there is a truly heterogeneous body of research being carried out using the term 'agents' [23]. Inherent in the concept of agents are the notions of problem solving capabilities, delegation, and autonomy. Each agent is usually assigned a separate problem or part of a problem. A larger problem is thus decomposed, and each sub-problem is assigned to a specific agent having corresponding competences. Multi-agent systems try to solve the entire problem. To coordinate a real-life situation, agents can also serve as wrappers for real entities [30].

Applying agent-based simulation for testing matching mechanisms means that all independent actors are viewed as agents. Many different definitions of software agents can be found in literature, but the common characteristics for an entity to be called a software agent are that it should possess at *least* the attributes autonomy, goal directedness, ability to communicate, and ability to react on the environment or to deliberately start actions [1,6,23]. *Agents* are thus autonomous, goal driven entities that are able to communicate with other agents and whose behavior is the consequence of their (1) observations, their (2) knowledge and their (3) interactions with other agents. An agent's behavior is often viewed as a manifest of 'intelligence' [29]. In our situation *behavior* refers to the trading strategies of organizations that are based on (1) observations, such as actual market price and offered product quality, (2) knowledge of prior transactions, such as reliability of delivery and outcomes of negotiations processes, and (3) a world model for reasoning about possible actions that can be taken. This last element will in this case contain trading strategies for the making and receiving of bids and offers which are often classified into a soft, moderate, or tough conceder strategy [26]. A soft conceder strategy lowers or increases an attribute very quickly towards the lowest or highest acceptable price, a tough conceder strategy lowers an attribute very slowly and a moderate conceder strategy is in between the soft and tough conceder strategy.

The agent-based perspective should enable us to model the interactions between autonomous entities and the reasoning logic incorporating trading knowledge to react on bids and offers. The business process perspective should enable us to model the organizational processes and the consumption of resources necessary for executing business processes and interactions. Several business processes need to be simulated as organizations need processes to make, submit, receive, evaluate and submit counter bids, and for dealing with the financial and logistic settlement. For example organizations might want to assess the impact of matching mechanisms on the throughput time of a negotiation process, on the resources consumed for negotiation or on the administrative burden needed to settle the transaction financially and logistically. Simulation of processes means that the time dependent sequence of activities is modeled, and that the internal business logic is reduced extensively [17]. We combine agent-based modeling and discrete-event simulation of business processes. The agent-based perspective depicted in Fig. 1 shows the autonomous aspects for interacting with other agents (interaction view) and reasoning logic containing an organization's trading strategies (reasoning view), and the business process aspect focuses on modeling the time dependent sequence of activities (process view). The

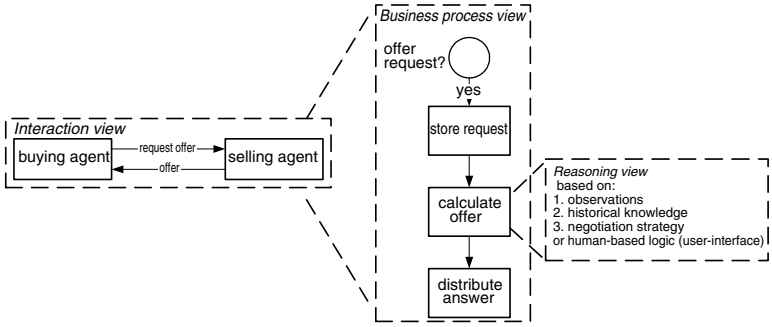

Fig. 1. Relationship between views.

reasoning view does not necessarily contain programming code, reasoning can also be done in an interactive simulation by human beings interacting with the agent using a user-interface.

We follow the ideas of agent architectures developed for MAS [14] and develop an agent architecture using object orientation. Objects already possess the encapsulation for attributes and methods that make it an autonomous entity. Pure object oriented environments also focus on communication between objects, where even retrieving the value of an attribute is implemented as a 'get' method, which can be granted or denied by the object. Implementing an agent means that we have to extend objects to enable an agent to comply with the common characteristics that make up an agent, including autonomy, goal driven direction, communication, and behavior to react on the environment or to deliberately start actions. An agent architecture is a means to make sure that all modeled entities meet the characteristics that make up an autonomous agent, which can be done by incorporating the interaction, process and reasoning views discussed in the preceding section. An agent architecture should help in breaking up an application into logical parts that can be reused, however, the architecture should not pose a limitation to the extensibility of an agent, as different organizations with different strategies should be modeled and new kinds of matching mechanisms might have to be added.

An agent architecture can be developed by looking at the characteristics of agents given the underlying organizational context. The context helps to define the nature of agents, the behavior and the interactions of agents. Agents are situated in a particular environment, receive inputs related to the states of the environment, and they act on the environment. In Internet-based trading situations organizations communicate using messages. Therefore an agent should be able to receive incoming messages and send outgoing messages. Organizations are independent, have behavior and are purposefully trying to achieve particular objectives. Hence, agents should be clearly identifiable, autonomous entities with well-defined boundaries and interfaces. An agent behavior should define which purposeful behavior an agent is capable of and which types of messages it can send and receive. Many systems emphasize the need for explicit and separate representation of *control* and *behavior* [10,13]. We adopt the distinction between control and behavior, as an autonomous agent's

control can be relatively generic and its behavior should be relatively specific. Note that this distinction is similar to the distinction between the interaction view and the combined business process and reasoning logic view. The business processes and reasoning logic containing an organization's trading strategies determine an organizations behavior. In our research the behavior is specific for our domain, the evaluation of Internet-based matching mechanisms. The separation of control logic and behavior algorithms from the data allows us to write reusable objects. *Control* is responsible for dealing with incoming and outgoing messages and deciding what to provide to the behavior models and when to provide it. *Behavior* models trading strategies and matching mechanisms.

Apart from this distinction between control and behavior we make a further distinction to decouple the agent's behavior from the *visualization* of its behavior. This distinction is useful, as not all agents will always be visualized. When there are hundreds or even thousands of agents in the system it might not be useful, or even impossible to visualize them all. The three elements make up the simulated agent's architecture are depicted in Fig. 2. A simulated agent is an aggregate of *Control*, *Behavior* and *Visualization* classes. The *Control* and *Visualization* classes are also aggregates of a number of classes. The *Behavior* class can be extended into one or more specific classes describing the desired behavior, in this way enabling reuse.

The *Control* class is an aggregate of *Resource Scheduling*, *Reactive* and *Deliberative* classes. This aggregate of classes is responsible for handling the interactions with other agents by receiving and sending messages, for scheduling resources and for allocating tasks to the limited resources of an agent. At least the *Reactive* class is necessary to construct an autonomous agent.

Resources are necessary to model the trading processes of buyers and sellers performed by humans and computers. Resources could be persons working in a buying and selling organization, computers negotiating on behalf of a buyer or seller and computers matching supply and demand. A *Resource Scheduling* class is able to schedule the number of resources available. In this way it can be used to model for example the working time of employees, illness of people and the availability of computing power. The workload of resources and the time of activities can be measured for the purpose of performance evaluation.

The *Reactive* class receives all messages. It can decide to reject messages or it can schedule tasks and resources to process the message. The *Deliberative* class can

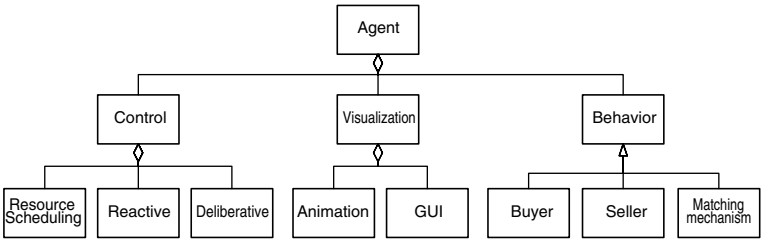

Fig. 2. The simulated agent architecture.

initiate new tasks. This aggregate of classes provides the autonomy, reactive and deliberative behavior, and communication properties of agents.

The *Behavior* class contains the behavior of an agent. Behavior determines buying and selling strategies and the behavior of the matching mechanisms. The behavior can be modeled in terms of the tasks that need to be accomplished given its position [39]. Behavior is dependent on the circumstances, as a result modeled organizations should be able to have various types of behavior. As a result the *Behavior* class of an agent can be extended e.g. the *Behavior* class can extended into a Buyer, Seller or Matching Mechanism class. The *Behavior* class encapsulates data and methods and can be activated by *Control* classes.

Child classes of the *Behavior* class can contain emulated trading behavior and simulated tasks. Matching mechanisms and the buying and selling strategies will be *emulated*, i.e. actual software is written to execute matching mechanisms and trading strategies of buyers and sellers. We choose to also incorporate time-based *simulation* of processes, as we want to evaluate the effect of matching mechanisms on time-based performance such as delivery time, utilization of resources, and waiting time.

The *Matching* class has generic methods to periodically or continually match bids and offers. This class needs to inherit the mechanisms from the classes that will be discussed in the proceeding section. This class contains also the methods to interact with the *Control* class to receive bids and offers, and to submit the matching results (e.g. accepted, rejected, new bid, new offer) to a buyer or seller agent. It does not contain any matching mechanisms itself. The *Buyer* and *Seller* classes have methods to receive, evaluate, make and submit offers or bids. The buyer and seller classes also interact with the *Control* class to receive and submit orders. The trading strategies include soft, moderate, or tough conceder strategies.

The *Visualization* class is responsible for showing a window containing an animation of the time-ordered dynamics, a static background, an overview of performance indicators and a user-interface. This class can be constructed from animation components and user interaction components. With the help of a user-interface, traders can be given the opportunity to gain experience with a potential situation by participating in the simulated trading process.

The construction of the agent is shown in Fig. 3 in order to explain the instantiation of the agent-architecture. On the right side the event handler is shown which includes a list of scheduled events such as 'determine bid', 'match supply and de-

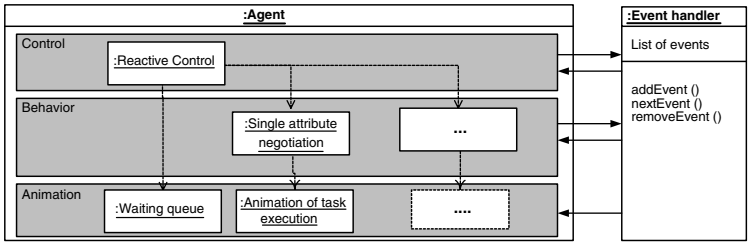

Fig. 3. Instantiation of the simulated agent architecture.

mand', used to determine the right sequence of events. On the left side a simple agent is shown consisting of a reactive control class, a single attribute negotiation class as behavior, a simple task, a horizontal queue, and a simulated task for the animation of the tasks. Note that we do not animate the incoming queue of the control layer and that the dotted tasks are added to stress that an agent usually executes more than one task. The control class determines which tasks of the agent will be executed, and a corresponding task for this agent will be added to the event list. The Event handler determines which event from which agent should be executed next. In the case of a software agent the simulated time to determine a bid can be very small or almost immediate, in the case of a human this time can be a couple of minutes or even days.

4. Matching behavior

Matching behavior classes are aimed at modeling a trading situation using Internet-based matching mechanisms quickly, but also to give traders a global overview of possible mechanisms that might be used. A matching agent can be constructed by extending the *Behavior* class into a matching class as shown in Fig. 2. The matching class contains the programming code to emulate a matching mechanism.

There is a vast field of economic literature on matching mechanisms design. A detailed characterization of matching mechanisms can be found in [20,21,28,36,38]. Matching mechanisms are often categorized as distributive or integrative [9,28]. In a distributive or competitive matching the outcome space is unidimensional; one organization's gain is another organization's loss. Win–win solutions exist in integrative or cooperative matching. One particular outcome can be better for both parties. The degree of competition or cooperation falls within a continuum between distributive and integrative negotiation. Teich et al. [34] provide a 'market framework' for categorizing types of matching mechanisms; this framework is shown in Fig. 4. In the following part, we first discuss mechanisms that can be used to execute (reverse) auctions, one buyer with many sellers or vice versa, and thereafter we discuss mechanisms that can be used for many-to-many trading situations.

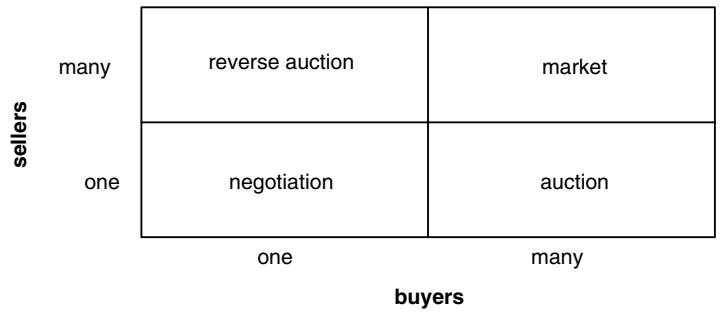

Fig. 4. Market framework.

Vickrey [36], and McAfee and McMillan [21] provide an introduction and comprehensive review on auctions. We limit ourselves to mentioning the various auction mechanisms. An auction is a market institution with an explicit set of rules determining resource allocation and prices on the basis of bids from the market participants [21]. An *English* auction consists of the seller, or auctioneer, continuously raising the price until only one-bidder remains. In a *Dutch* auction the seller or auctioneer starts with a high price and continuously lowers the price until a seller is found. The *continuous double auction* is a combination of the English and Dutch auction. First the price is raised and sellers bid to determine the minimum price, thereafter the price is lowered, maximally till the minimum price. This kind of mechanism can be highly iterative with a great deal of communication overhead in an environment with many participating organizations.

In a first-price *sealed bid auction* model each buyer submits a sealed bid known only to the seller (or the auctioneer). The buyer with the highest bid is awarded the item at the bid price. The advantage of this model is that it is a one-step procedure and it does not require much communication, so there will be no problems caused by a low communication speed or a slow reaction. The disadvantage is that it requires strategic reasoning about the beliefs of other participants and how they will bid to determine the maximum (minimum) bid. The *Vickrey mechanism* is similar to the sealed-bid mechanism; the difference is that the settlement price for the transaction will be the second highest (lowest) bid [36]. Under this mechanism buyers are motivated to tell the truth about their reservation prices. A *reservation price* is the maximum (minimum) amount somebody is willing to pay. With other mechanisms it may be beneficial for agents to lie about their reservation price to receive more payoff.

Guttman and Maes [9] mention two main techniques for many-to-many matching as found in practice; multi-attribute utility and distributed constraint satisfaction technique. *Multi-attribute Utility Theories* (MAUT) are theories for quantitatively analyzing decisions involving multiple, interdependent objectives [16]. A *utility* is a function that can be used to map a state of an attribute, such as a high price, low quality, long delivery time onto a real number, which describes the associated degree of utility ('happiness'). A basic question is how preferences can be mapped into a coherent utility function. When there are conflicting goals and only some goals can be achieved, for example trade off between attributes like price, delivery time and quality the utility function should specify the appropriate trade-off. A *weight-value structure* is built on top to represent dependencies between attributes. Individual attributes can have a different impact on the outcome by assigning different weights to individual factors. Individual attributes do not all have the same impact. When a product need to be delivered quickly, the delivery time will get a relative high weight. Weights are typically subjective and can differ from person to person. Multi-attribute matching is much more difficult to analyze analytically than auctions based on one attribute.

Distributed Constraint Satisfaction Problems (DCSP) analyze decision problems using constraints. In a DCSP the states are defined by the values of a set of variables and goals that specify a set of constraints that these values must obey [35]. Variables

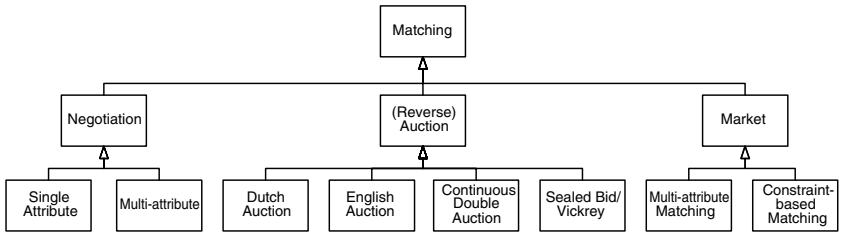

Fig. 5. Matching behavior.

and constraints are distributed among two or more actors. In this way actors can have their own set of constraints, which they have to communicate to jointly solve a problem. Constraints come in several varieties. Unary constraints concern the value of a single variable, binary constraints relate pairs of variables, and higher-order constraints involve three or more variables. Constraints can be absolute—violation rules out a potential solution—or a preference that states which solutions are preferred. Each variable, V_i in a DCSP has a domain D_i, which is the set of possible values that the variable can take. The domain can be discrete or continuous. Once a decision is formulated in this way a solution can be found, if any exists [35].

The mechanisms described before are translated into matching behaviors that fit into our agent architecture. In Fig. 5 the behavior class hierarchy is shown. Based on the framework of Teich et al. [34] we define the classes *Negotiation*, *Auction* and *Market* that inherit the methods form the *Matching* class. We have only three classes, because the *Auction* class also contains the methods for executing a reverse auction, as they are similar. These classes override the methods of the parent class *Matching* to replace the functionality in such a way that the algorithms for one-to-one negotiations, one-to-many auctions and many-to-many markets can be executed.

The *Negotiation*, *Auction* and *Market* classes have child classes containing the emulated matching mechanisms. These classes override the methods of their parent class. Different actions to match demand and supply can be performed depending on parameters. New kind of matching mechanism can easily be created for the simulation model by combining methods or creating new classes.

5. Application: the computer market

In this section we describe an implementation of the agent-based simulation model for the Dutch business-to-business computer systems market [12]. A computer assembler was interested in making using of innovative matching mechanisms over the Internet and wanted to experiment without having the risk to loose customers and to avoid potential channel conflicts. The computer market is continuously under pressure to lower prices due to the intensive competition. The computer assembler feared competition based on price, because the information provided by an open network, such as the Internet, makes a fast comparison of offers possible.

In the business-to-business computer systems market, a deal often includes a number of computer systems and other products like network equipment, back-up and recovery hardware, software, and services such as the installation of software on computers, installation of the computers at the office and after-sales services, e.g., maintenance and repair. The computer market consists of computer suppliers, such as computer manufacturers and assemblers, and customers as shown in Fig. 6. The activities of the computer assembler under study are shown in more detail in this figure. The computer assembler uses four sales channels, namely dealers, dedicated dealers, Internet-selling, and call centers. The sales departments take care of the logistics and financial settlement. The bold lines represent the physical activities of the computer assembler including warehousing of components, assembling of computer systems and installation of software, transportation to customers and installation of the computer systems at the customers' offices. The computer assembler assembles computers from components based on customer orders and has no own computer inventory.

We first constructed an agent-based simulation model of the existing situation. We modeled the repeating trading activities and left out incidental activities and non-trading activities such as repairing damaged computer systems, identification of customer needs, training, and so on. The products are modeled using different products attributes and contract terms, namely prices, quality of computer systems and delivery times. The interviewees indicate that customer often search for the lowest price. The brand is an indicator for the perceived quality of the systems

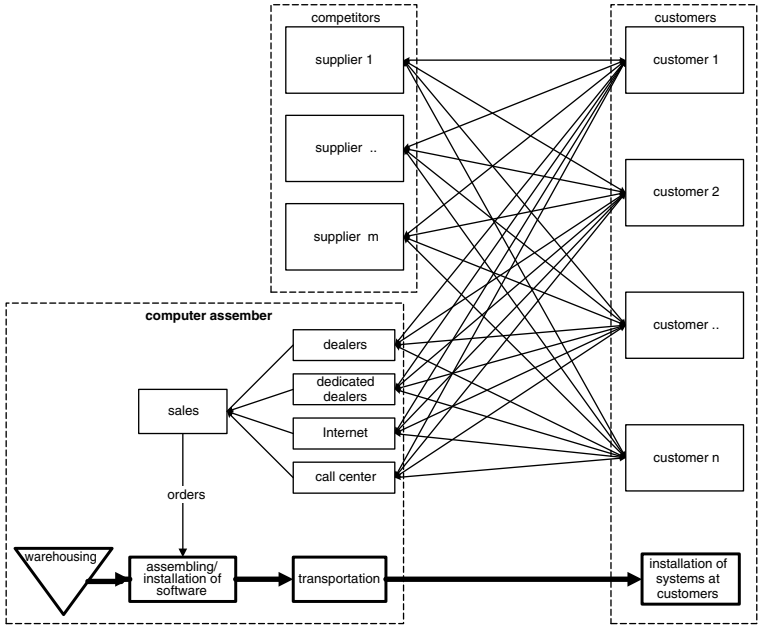

Fig. 6. Overview of computer market.

and influence and customer are prepared to pay higher prices for premium brands. The offer should meet customer's time constraints. Order concern often many systems and additional equipment, which might not be on stock or needs configuration of hardware and installation of software which can be a time consuming process. Consequently products can often not delivered and installed within the desired time frame.

We designed two alternative models to provide insight into the possible matching mechanisms. Session participants from both computer assembler and dealers were invited to interact with the models. The first model is based on the idea to auction superfluous products and the second model is based on one-to-one negotiation using multi-attributes. The session participants involved generated the idea for the second model, based on their experiences with the first model.

The risk of products becoming obsolete is high, therefore, the computer assembler wanted to know if the introduction of a new distribution channel consisting of an auction for selling obsolete products could decrease this risk. An English auction was chosen as the characteristics seem to fit the needs, the auction can last several days and bidders can increase their price in time. We modeled this auction using the *English Auction* class discussed in the preceding section. A user-interface, shown in Fig. 7, was designed to enable customers to interact with the auction. At the top of the figure potential products that need to be ordered (new order) to fulfill the

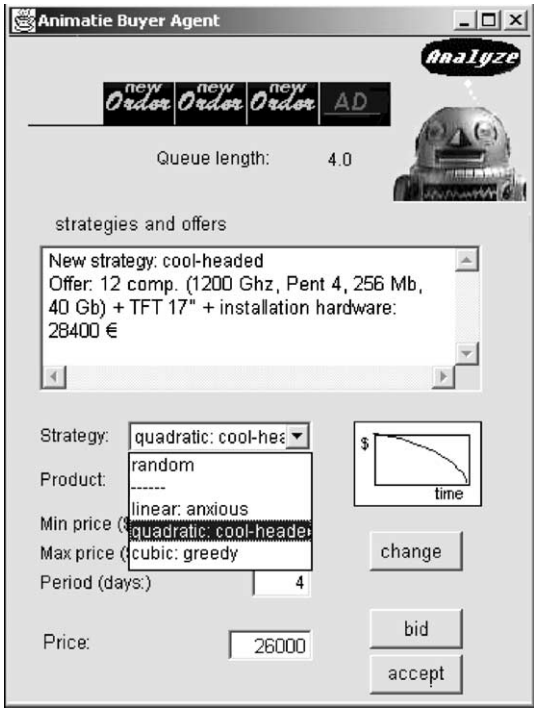

Fig. 7. Simple user-interface for buyers.

computer need, and offers (AD) from computer suppliers are listed in a waiting queue. Customers have the opportunity to bid automatically on products by choosing a strategy (anxious, greedy and cool-headed strategy which are similar to soft, moderate, and tough conceder strategies), a minimum price to start the bidding with, a maximum price they were willing to pay and the number of days to use the strategy to come to a higher price. The robot at the top is the resource for automatically making a counter-offer or accepting offers. The text box 'strategies and offers' includes feedback information whenever a trading strategy is set or changed and includes bids and counter offers for manual trading. Strategies can be set of altered by setting the parameters and pressing the change button. An offer or a counter offer can be accepted using the accept button or a counter-offer can be made by filling in the price and pressing the bid button.

While trading with the system and looking at the impact on business processes, it became clear to the session participants that much effort would be required to realize an English auction in comparison to the benefits. The number of superfluous computer systems is relatively low and the computer assembler estimated that the selling of these systems using a new auction channel would not be worth the setting up of business processes and accompanying infrastructure to sell and deliver the products. Even the use of a third party auctioneer who takes care of the settlement was rejected due to the necessary time-efforts needed of the computer assembler. The session participants also found that the added value for attracting customers with a mechanism primarily based on the attribute price was limited; therefore, a multi-attribute negotiation idea was suggested.

In the second model a one-to-one, multi-attribute negotiation was introduced. The basic idea behind this model is to provide customers with a customized offer, where the price is dependent on a number of attributes, and the customers can react by providing a counter offer. We modeled this auction using the *Multi-attribute* class using the following attributes to make an offer:

- *Customer profile*: installed number of computer systems of the computer brand;
- *Order characteristics*: required amount;
- *Required delivery time*: the later the better, because it enables better planning;
- *Market price*: the price of computer systems offered by competitors is gathered daily by a price-signaling agent.

The offered price of the required computer systems is determined based on (1) if the customer has already an installed base of computer systems of the particular computer assembler, regular customers benefit in this way, (2) the number of computer systems required, large orders are relatively cheaper, (3) the delivery time required, long delivery times are more attractive as it enables a better assembly planning and got a higher price reduction and (4) the actual market price. All price directions for computer systems can be found on the websites of computer systems brands and prices might vary daily. By signaling price movements of competitors a competitive price can be calculated. Prices can be lowered immediately whenever a competitor decreases its price.

The model was presented and the session participants could interact with this model using a user-interface. Account managers of the computer assembler indicated that most of the projects that need computer systems are already known months before the systems are actually needed; however, posting of the order is done only weeks or even days before. Incentives may help to convince customers to order products as soon as their demand becomes available. In this model the incentive is a discount in price as this enables a better assembly planning. The impact of earlier ordering on the trading processes of the dealer and computer assembler is limited, as the making of an offer can be automated and is not time-intensive. Employees representing the computer assembler and dealers were very satisfied with this alternative and are planning to implement it.

As all models were implemented in Java, we were able to support the decision-making of the computer assembler about the introduction of matching mechanisms over the Internet. The agent architecture and matching behavior proved to be suitable for modeling the trading situation. The agent-based simulation testbed helped to evaluate the added value of matching mechanisms prior to implementation. Both the modeling of business processes of the computer assembler and dealers, and the trading behavior consisting of many traders interacting with each other using trading strategies were necessary for evaluating the matching mechanism in this case. Business process modeling was necessary to come to the conclusion that auctioning of superfluous products requires too much effort. Trading strategy modeling was useful for migrating from a one-to-many matching mechanism based on the English auction to a one-to-one matching mechanism based on multiple attributes.

The agent-based architecture helps in breaking up an application into logical parts that can be reused and ensures the behavior of agents is autonomous and includes an interaction, process and reasoning view. In this respect we want to draw an analogy to the adoption of object-oriented simulation languages. The quantitative advantages of object-oriented simulation languages in comparison with process-oriented simulation languages cannot easily be proven. The justification of object-oriented approaches was and is primarily based on the characteristics of inheritance and encapsulation, which enable clear modeling and reuse. The *goal* of modeling is to abstract from reality, but also to keep a high degree of correspondence with reality. Our agent-based simulation testbed ensures a high degree of correspondence with reality. The implementation using an agent-based simulation environment is more closely related to the conceptualization of humans and is closer to their natural way of thinking.

6. Conclusions

In this paper we described an agent-based simulation testbed aimed at evaluating Internet-based matching mechanisms. Market, process and resource based criteria are of importance for evaluating Internet-based matching mechanisms. The market efficiency is determined by the interactions between traders, the process efficiency is determined by the tasks making up the organizational business processes, and the resource efficiency is determined by the utilization of the resources of organizations.

In our simulation testbed each buying and selling organization is modeled using an autonomous agent. Our autonomous agents incorporate an interaction view to enable communication with other agents, a reasoning view containing an organization's trading strategy and a business process view for simulating the trading processes of organizations. Trading behavior is modeled by programming code containing the trading strategies or by providing a user-interface to enable humans to enter their trading strategies. Business processes are modeled by simulating the time dependent sequence of activities. The developed agents are based on a generic agent architecture consisting of *Control*, *Visualization* and *Behavior* classes. The *Behavior* class can implement pre-defined distributive and integrative matching mechanisms and enables the reuse of matching mechanisms. The *Control* class handles the communication with other agents. The *Visualization* class visualizes the trading events and the organizational processes within one agent and the communications between agents. In this way insight can be gained by organizations into the functioning of matching mechanisms within their existing situation. We developed a relative simple architecture for modeling agents. Future research can focus on more advanced architectures containing beliefs, desires and intentions, characteristics commonly included in multi-agent systems. With further research we can also try to bridge the gap between agent-based simulation and multi-agent systems, as it should be possible to translate an agent-based simulation model into a multi-agent information systems as both are based on the concepts of agents.

Our case study showed that the testbed was suitable for evaluating Internet-based matching mechanisms, moreover, we were able to support the decision-making of the computer assembler about the introduction of matching mechanisms over the Internet. The testbed helped traders to evaluate and select matching mechanisms better fulfilling the requirements of the organizations and adopt their business processes to new trading situations prior to the introduction of new matching mechanisms. By explicitly representing organizations by agents, and by taking goals into account it suddenly became very easy to discuss alternative matching mechanisms and to come to mechanisms acceptable by the organizations involved in the design process. The agent-based architecture helps in breaking up a simulation model into logical parts that can be reused and ensures the behavior of agents is autonomous and includes an interaction, process and reasoning view. In this way the architecture ensures that a model can be built quickly, has a high degree of correspondence with reality and we can conclude that agent-based simulation is especially useful to describe trading situations more naturally, as agents can represent the decentralized nature of a trading situation, the multiple loci of control, the multiple perspectives, and the competing interests. Finally, agents have behavior to autonomously decide to enter or leave a simulated trading place.

References

[1] J. Banks (Ed.), Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice, J. Wiley & Sons, New York, 1998.
[2] J.M. Bradshaw, Software Agents, AAAI Press/The MIT Press, Cambridge, MA, 1997.

[3] K.M. Carley, L. Gasser, Computational organization theory, in: G. Weiss (Ed.), Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, The MIT Press, Cambridge, MA, 1999, pp. 299–330.
[4] B. Curtis, M.I. Kellner, J. Over, Process modeling, Communications of the ACM 35 (9) (1992) 75–90.
[5] Q. Dai, R.J. Kaufmann, Business models for internet-based B2B electronic markets, International Journal of Electronic Commerce 6 (4) (2002) 41–72.
[6] O. Etzioni, D.S. Weld, Intelligent agents of the internet: Fact, fiction, and forecast, IEEE Expert 10 (4) (1995) 44–49.
[7] FIPA-OS, Specifications and Documentation, Available from <http://www.fipa.org/>, Foundation for Intelligent Physical Agents, Last accessed October 2004.
[8] Z. Guessoum, A multi-agent simulation framework, Transaction of the SCS International 17 (1) (2000) 2–11.
[9] R.H. Guttman, P. Maes, Cooperative vs. competitive multi-agent negotiations in retail electronic commerce, in: Proc. CIA'98, Springer, Berlin, 1998, pp. 135–147.
[10] G. Habchi, C. Berchet, A model for manufacturing systems simulation with a control dimension, Simulation Modelling Practice and Theory 11 (2003) 21–44.
[11] M. D'Inverno, M. Luck, M. Georgeff, D. Kinny, M. Wooldridge, The dMARS architecture: a specification of the distributed multi-agent reasoning system, Autonomous Agents and Multi-Agent Systems 9 (1-2) (2004) 5–53.
[12] M. Janssen, Designing electronic intermediaries, Doctoral Dissertation, Faculty of Technology, Policy and Management, Delft University of Technology, 2001.
[13] N.R. Jennings, On agent-based software engineering, Artificial Intelligence 117 (2) (2000) 277–296.
[14] N.R. Jennings, An agent-based approach for building complex software systems, Why agent-oriented approaches are well suited for developing complex, distributed systems, Communications of the ACM 44 (4) (2002) 35–41.
[15] J.L. Jones, G.J. Koehler, Combinatorial auctions using rule-based bids, Decision Support Systems 34 (1) (2002) 59–74.
[16] R.L. Keeney, H. Raiffa, R.F. Meyer, Decisions with multiple objectives: Preference and value tradeoffs, Wiley, New York, 1976.
[17] A.M. Law, D.W. Kelton, Simulation Modeling and Analysis, McGraw-Hill, New York, 1991.
[18] P. Maes, Agents that reduce work and information overload, Communications of the ACM 37 (7) (1994) 30–40.
[19] P. Maes, R.H. Guttman, A.G. Moukas, Agent-mediated electronic commerce: an MIT media laboratory perspective, Communications of the ACM 42 (3) (1999) 81–91.
[20] A. Mas-Colell, M.D. Whinston, J.R. Green, Microeconomic Theory, Oxford University Press, New York, 1995.
[21] R.P. McAfee, J. Mcmillan, Auctions and bidding, Journal of Economic Literature 25 (2) (1987) 699–738.
[22] H. Mizuta, Y. Yamagata, Agent-based simulation and gaming system for international emissions trading, in: A. Namatame et al. (Eds.), Agent-Based Approaches in Economic and Social Complex Systems, IOS Press, Amsterdam, 2002, pp. 69–78.
[23] H.S. Nwana, Software agents: an overview, Knowledge Engineering Review 11 (3) (1996) 205–244.
[24] H.S. Nwana, D. Ndumu, L. Lee, J. Collis, ZE, ZEUS: a tool-kit for building distributed multi-agent systems, Applied Artifical Intelligence Journal 13 (1) (1999) 129–186.
[25] C.D. Pegden, R.E. Shannon, R.P. Sadowski, Introduction to Simulation Using SIMAN, McGraw-Hill, New York, 1994.
[26] D. Pruitt, Negotiation Behavior, Academic Press, London, 1981.
[27] E. Ramat, P. Preux, Virtual laboratory environment (VLE): a software environment oriented agent and object for modeling and simulation of complex systems, Simulation Modelling Practice and Theory 11 (2003) 25–55.
[28] E. Rasmussen, Games and Information: An Introduction to Game Theory, Basil Blackwell, Cambridge, 1989.

[29] S.J. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, Englewood Cliffs, NJ, 1995.
[30] N.M. Sadeh, D.W. Hildum, D. Kjenstad, Agent-based e-supply chain decision support, Journal of Organizational Computing and Electronic Commerce 13 (3) (2003) 225–287.
[31] R.E. Shannon, Systems Simulation: The Art and Science, Prentice Hall, Englewood Cliffs, NJ, 1975.
[32] K. Sycara, S. Widoff, M. Klusch, J. Lu, LARKS: dynamic matchmaking among heterogeneous software agents in cyberspace, Autonomous Agents and Multi-Agent Systems 5 (2002) 173–203.
[33] K. Sycara, M. Paolucci, M. van Velsen, J. Giampapa, The RETSINA MAS infrastructure, Journal of Autonomous Agents and Multi-agent Systems 7 (1) (2003) 29–48.
[34] J. Teich, H. Wallenius, J. Wallenius, Multiple-issue auction and market algorithms for the world wide web, Decision Support Systems 26 (1) (1999) 49–66.
[35] E. Tsang, Foundations of Constraint Satisfaction, Academic Press, London, 1993.
[36] W. Vickrey, Counterspeculation auctions and competitive sealed tenders, Journal of Finance 16 (1961) 8–37.
[37] R.T. Wigand, A. Picot, R. Reichwald, Information, Organization and Management. Expanding Markets and Corporate Boundaries, John Wiley & Sons, Chichester, 1997.
[38] P.R. Wurman, M.P. Wellman, W.E. Welsh, A parametrization of the auction design space, Games and Economic Behavior 35 (2001) 304–338.
[39] F. Zambonelli, N.R. Jennings, M. Wooldridge, Organizational rules as an abstraction for the analysis and design of multi-agent systems, International Journal of Software Engineering and Knowledge Engineering 11 (4) (2001) 303–328.
[40] B.P. Zeigler, T.G. Kim, H. Praehofer, Theory of Modeling and Simulation, Academic Press, Orlando, FL, 2000.
[41] G. Zlotkin, J.S. Rosenschein, Mechanisms for automated negotiation in state oriented domains, Journal of Artificial Intelligence Research (5) (1996) 163–238.