# MAGNET: A Multi-Agent Contracting System for Plan Execution

**John Collins, Maksim Tsvetovatyy, Bamshad Mobasher, and Maria Gini**
Department of Computer Science and Engineering, University of Minnesota
200 Union St SE, Minneapolis, MN 55455
{jcollins, tsvetova, mobasher, gini}@cs.umn.edu

### Abstract

We present a system for multi-agent contract negotiation, implemented as a generalized market architecture called MAGNET. MAGNET provides support for a variety of types of transactions, from simple buying and selling of goods and services to complex multi-agent contract negotiations. In the latter case, MAGNET is designed to negotiate contracts based on temporal and precedence constraints, and includes facilities for dealing with time-based contingencies. In contrast with other approaches to multi-agent negotiation, we introduce an explicit intermediary into the negotiation process, which helps in controlling fraud and discouraging counterspeculation.

## Introduction

Automated contracting protocols generally assume direct agent-to-agent negotiation. For example, Smith (Smith 1980) pioneered research in communication among cooperating distributed agents with the Contract Net protocol. The Contract Net has been extended by Sandholm and Lesser (Sandholm & Lesser 1995) to self-interested agents. In these systems, agents communicate and negotiate directly with each other. In Sandholm's TRACONET system, for example, both the bidding and contract execution mechanisms are complicated by the need to operate in an environment where agents cannot trust each other. They do not assume or take advantage of an independent market infrastructure that can act as a trusted intermediary and affect the timing and/or functionality of the protocol elements.

We have designed and implemented a generalized multi-agent market infrastructure that can provide explicit and integrated support for complex agent interactions, such as in automated contracting, as well as other types of negotiation protocols, including sealed-bid auctions and open-bid or advertised-price buying and selling. We call the resulting system MAGNET.

The use of an independent market infrastructure adds value and practicality to automated contracting protocols. By independently verifying the identities of participants, by tracking the state of the negotiations and any commitments that result, and by enforcing the rules of the protocol, fraud and misrepresentation are curtailed, and unproductive counterspeculation is minimized.

We are particularly interested in supporting Plan Execution by Contracting, an activity in which a contractor agent, in order to fulfill its goals, must contract with other self-interested supplier agents for all or part of the necessary tasks. These agents are self-interested and exhibit limited rationality. Agents providing resources or services will attempt to gain the greatest possible benefit, and agents requesting resources or services will attempt to pay the lowest price.

The MAGNET system incorporates a simple three step, leveled commitment protocol, with a contractor agent issuing a call-for-bids, suppliers replying with bids, and the contractor accepting the bids it chooses. We avoid the need for open-ended negotiation by means of bid break-downs and time-based decommitment penalties (Collins *et al.* 1997). Once the contractor agent receives the bids from supplier agents, it must evaluate the bids based on cost and time constraints, and select the optimal set of bids (or parts thereof) which can satisfy its goals. The resulting *task assignment* forms the basis of an initial schedule for the execution of the tasks.

This paper is organized as follows: in Section 2 we describe the interactions of agents with the MAGNET infrastructure. Section 3 describes the general architecture. Section 4 discusses our implementation, with an example. Section 5 covers related work.

## Agent Interactions with MAGNET

From the point of view of a customer agent (also known as the contractor agent), interactions fall into three phases: planning, bidding, and execution. In the planning phase, the customer selects one or more markets, each of which specializes in certain types of product or service categories. The customer uses an *ontology*, provided by the market, to develop a partial plan, and then estimates tentative values for plan components based on the value of its current goal and the "criticality" of each component.

Once a plan is developed, the customer initiates a *session* in the market to encapsulate the bidding and execution activities related to meeting its goal. The interactions involved in the basic bidding and execution cycle among the contractor, supplier, and market session are illustrated in Figure 1. A detailed description of this protocol can be found in (Jamison 1997). During the execution phase, the interactions can be much more complex than indicated here, since either party may decommit from a contract (and pay a penalty), and the contractor is continuously monitoring and repairing its plan by replanning and rebidding when events fail to proceed according to expectations. Finally, the customer and suppliers finalize financial settlements and the session is terminated.

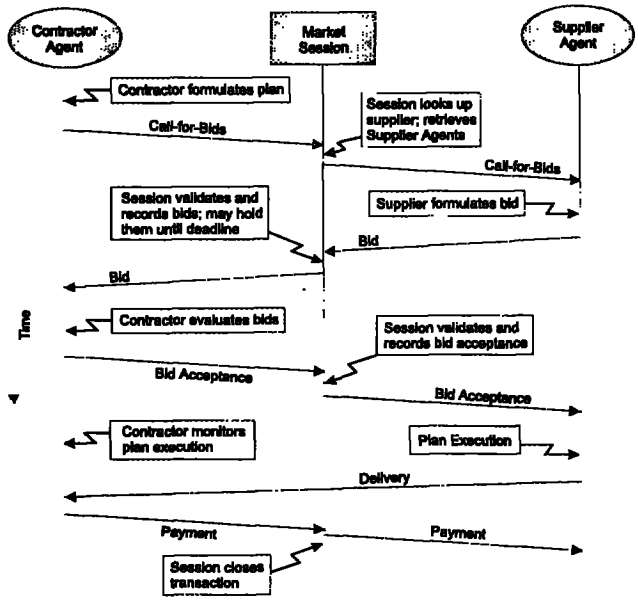


Figure 1: A Typical Session-Mediated Negotiation

## The MAGNET Architecture

The MAGNET architecture is a distributed set of objects that can support electronic commerce in a variety of domains, from the simple buying and selling of goods to situations that require complex multi-agent negotiation and contracting. The fundamental elements of this architecture are the *exchange*, the *market*, and the *market session*, as outlined below.

**The Exchange.** An *Exchange* is a collection of domain-specific markets in which goods and services are traded, along with some generic services required by all markets, such as verifying identities of participants in a transaction, or a Better Business Bureau that can provide information about the reliability of agents based on past performance. Architecturally, an exchange is a network-accessible resource that supports a set of markets and common services, as depicted in Figure 2.

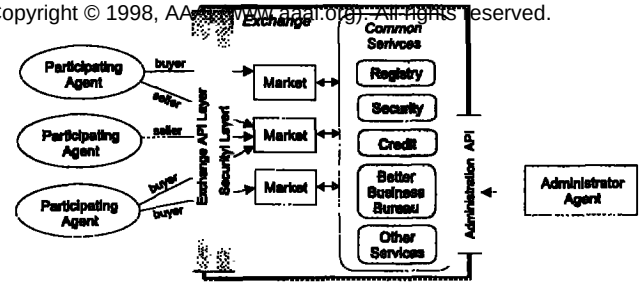


Figure 2: The Structure of an Exchange

**Markets.** Each *Market* within an exchange is a forum for commerce in a particular commodity or business area. There would be markets devoted to banking, publishing and printing, construction, transportation, industrial equipment, etc. Each market includes a set of domain-specific services and facilities, as shown in Figure 3, and each market draws upon the common services of the exchange.

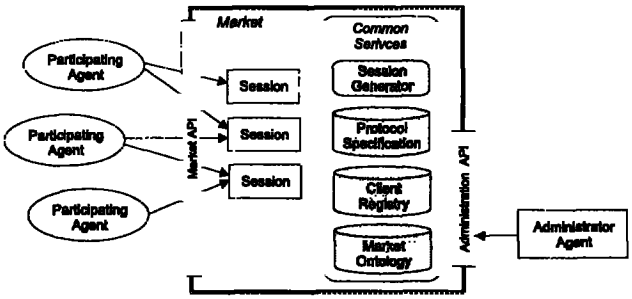


Figure 3: The Structure of a Market within the Exchange

An important component of each market is a set of current *Market Sessions* in which the actual agent interactions occur. Agents participating in a market may do so as either session initiators, or as clients, or both. As detailed in the next section, each session is initiated by a single agent for a particular purpose, and in general multiple agents may join an existing session as clients. Important elements of the market include:

- An *Ontology* that is specific to the domain of the market, specifying the terms of discourse within that domain. In a commodity-oriented domain, it would include terms for the products or services within the domain, as well as terminology for quality, quantity, features, terms and conditions of business, etc. In a planning-oriented domain, specifications of services would be in a form that supports planning.
- A *Protocol Specification* that formalizes the types of negotiation supported within the market. Within a planning-oriented market domain, these specifications would be limits on parameters of the negotiation protocol, such as the maximum decommitment

protocol, whether bids can be awarded before the bid deadline, etc.

- A *Registry* of market clients who have expressed interest in doing business in the market. Entries in this registry would include the identity of a client, a catalog (or a method for accessing a catalog) of that client's interests, products or capabilities, which can be used to locate clients to meet requests for new session participants, and a client agent that is empowered to negotiate contracts on behalf of the supplier. Client catalogs are required to express their interests and offerings in terms of the market's ontology.

**Market Sessions.** A *Market Session* (or simply a session) is the vehicle through which market services are delivered dynamically to participating agents. It serves as an encapsulation for a transaction in the market, as well as a persistent repository for the current state of the transaction.

We have chosen the term "session" to emphasize the temporally extended nature of many of these interactions. For example, in a construction-oriented market, if an agent wishes to build a new house, it initiates a session and issues a call-for-bids. The session extends from the initial call-for-bids through the negotiation, awards, construction work, paying of bills, and final closing. In other words, the session encloses the full life of a contract (or possibly a set of related contracts). The session mechanism ensures continuity of partially-completed transactions, protects against fraud by verifying the identity of agents, limits counterspeculation by enforcing negotiation rules, and relieves the participating agents from having to keep track of detailed negotiation status themselves.

Agents can play two different roles with respect to any session. The agent who initiates a session is known as the *session initiator*, while other participating agents are known as *session clients*. A session can be initiated either for the purpose of buying or selling, depending on the type of market. In the above example of building a house, the initiating agent was the buyer or customer, and the other participants would be sellers or suppliers, whether they were supplying materials, labor, advice, credit, or other services. A session could also be initiated to sell items or services at auction.

At any given time, a session can be *open* to new participants, or *closed*. A public auction would typically be open to new participants, while the house-building session described above would be closed once the contracts were let. The market maintains a list of open sessions which may be accessed by participating agents.

Figure 4 shows the structure of a session. Two APIs are exposed, one for the session initiator and one for session clients. Each session contains an Initiator Proxy that implements the Initiator API and persistently stores the current state of the session from the standpoint of the initiator.

A Client Proxy is provided for each client that similarly provides a Client API to the client agent, and

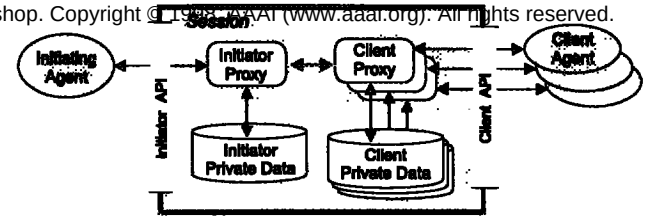


Figure 4: The Structure of a Market Session

persistently stores the current state of the session from the standpoint of the client. Proxies are market entities that act on behalf of the agents and enforce market rules.

There are two reasons for the existence of the proxy components. The first is related to security: client proxy components cannot see the private data of the initiator or of other clients. The second is that in a distributed system environment, the processing and persistent data elements of the initiator and clients would presumably be at different locations in the network to maximize performance.

## Implementation

The MAGNET system is implemented in Java and consists of the MAGNET server and a client API, which enables developers to create client software suited to their needs. MAGNET currently includes several client packages implementing this API, including a GUI-based end user system and automated test clients for testing of the protocol.

The current MAGNET market implementation is built as a distributed Java server application, using Java RMI as the communication mechanism, and an object database to provide persistence. The core of the server is a singleton Exchange object and three Home interfaces, one for Markets, one for Sessions, and one for Participants. Each Home provides factory and finder capabilities for its respective collection. The Exchange is the only interface registered with the network naming service; everything else is accessed through the Exchange.

Figure 5 is a simplified UML diagram of the server components. The customer and supplier proxy components that appear as part of the Session in the abstract architecture have been separated out and consolidated as the Participant. Each registered Agent has a Participant object in the server's database. The Participant will be active whenever an Agent is actively logged in to the server, and whenever any Session that it participates in needs to send data to it. Using the Participant in this way allows Agents to disconnect from the server even when it has active Sessions, without risk of data loss.

The Session and the Participant, as well as the AgentInterface on the client side, are subclasses of EventListener. This class supports the receipt of proto-

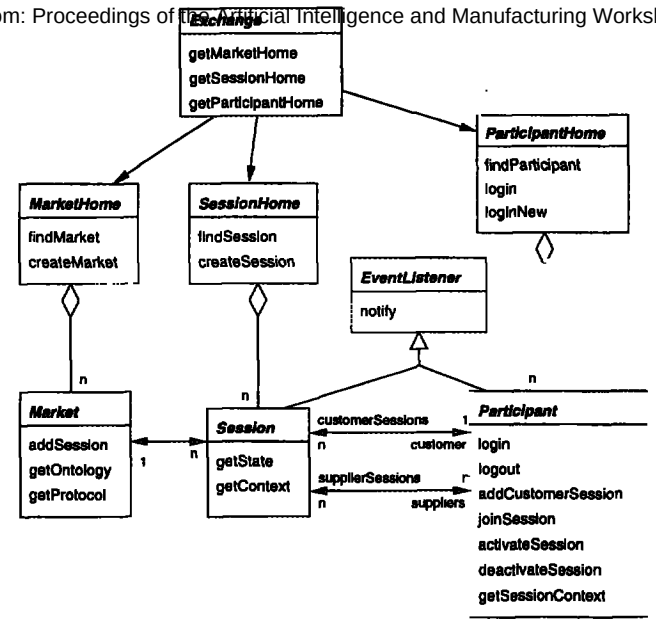


Figure 5: Exchange components

col events. Each subclass treats these events differently, as appropriate for its function.

The Session enforces the protocol rules, and maintains its internal state according to the protocol activity and the passage of time. For example, once the initial CallForBids has arrived, the Session will accept Bids, and once the BidDeadline passes, the Session will no longer accept Bids. The Participant receives events from the Agent, and checks security and optionally decrypts them before forwarding them to the Session. The Session interprets the messages and records them, possibly setting timeouts and updating its state, them forwards them to the appropriate Participants. The Participant then forwards them to its Agent just in case the Agent has activated that Session. This allows an Agent to start up multiple Sessions, and use its Participant to filter events to just the ones it is interested in at the moment. This is important for bandwidth management as well as to allow the Agent to control its allocation of computational resources.

The system is designed to be protocol-neutral. Only the Session and the Agent need to interpret messages, and the Agent specifies the protocol to be used when it initiates the Session. The initial prototype includes a version of the protocol outlined earlier implemented as a set of Java classes, and a KQML version is in the works.

## Example

We now describe the implementation using an example. A startup company, Acme Software and Screen Doors, Inc., is releasing version 1.0 of its new killer application. The product manager, who is also the president and chief programmer, has a goal: produce and ship 1000 shrink-wrap packages, including media, manuals, license forms, registration cards, and license key stickers. The old method would be to find a broker or publisher, and hand over the job.

Now assume Acme has an intelligent agent that can act as a broker in the Exchange. The manager gives the goal to her automated agent, which enters the exchange to assemble a set of contracts to satisfy the goals. We will assume the agent is already registered with the exchange. The agent searches the exchange and finds two markets that provide services to satisfy its goals. One deals with services for software publishing - printing services, digital media, and one with packaging and shipping (Figure 6)

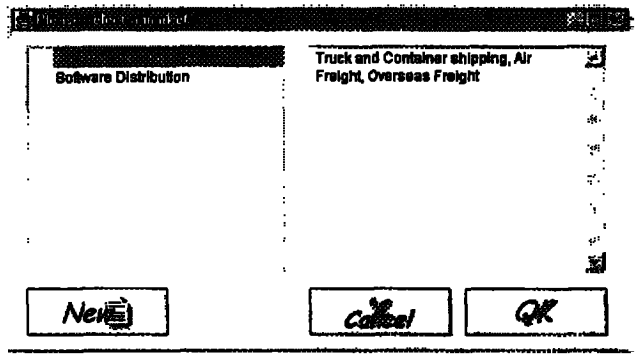


Figure 6: Markets for Software Publishers

Using operators obtained from the ontologies in the selected markets, the agent formulates a plan to satisfy its goals. Its plan looks like Figure 7, and is entered into the Magnet client using an interface that consists of a Gantt chart and a property sheet for entering and editing task parameters (Figure 8)

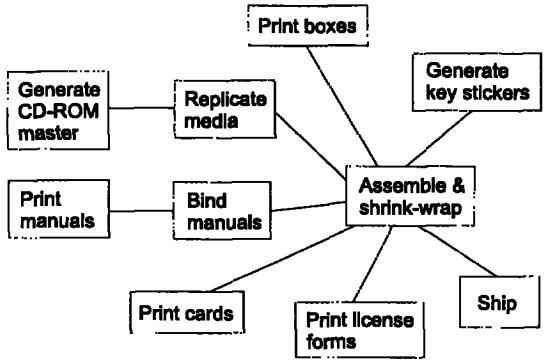


Figure 7: Plan for Release 1.0 Production

Next, the agent establishes the necessary credit with exchange, as specified by the chosen markets, and requests contract negotiation sessions in all 3 markets. The plan is now broken down by market, and the agent submits a call-for-bids to each session. It is acceptable if the submitted subplans overlap. This is because some

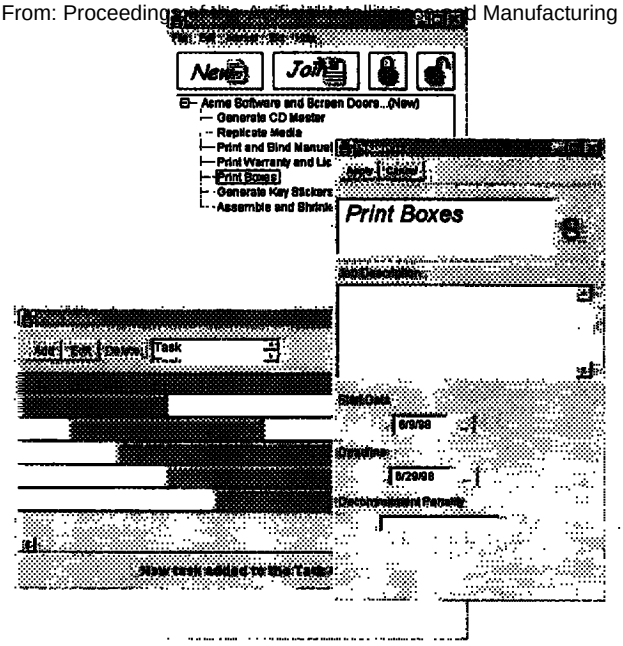


Figure 8: Customer Creates a Plan of Action

potential contractors may give lower bids on combinations of multiple tasks than they would on single tasks. Figure 9 shows the customer agent in the process of setting timing parameters for its call-for-bids.

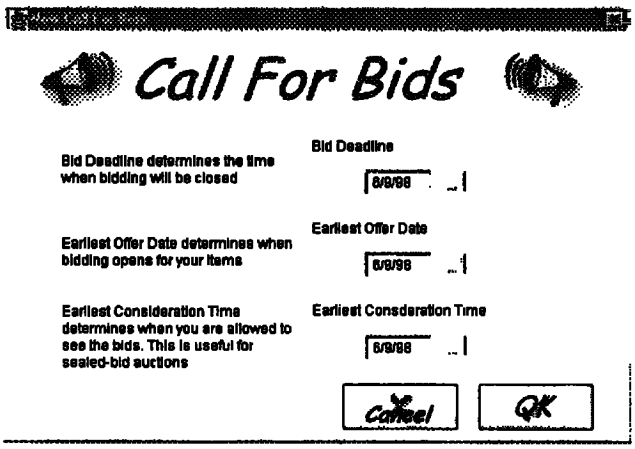


Figure 9: Customer Issues a Call-for-Bids

Each session now invites potential contractors who have expressed interest, through the market registry, in the subject matter of the bids, to join the sessions as clients. Some of those contractors submit bids.

The customer agent receives the bids, and attempts to assemble them into an optimal feasible plan. Unfortunately, no supplier has bid on printing the license key stickers. At this point, the customer agent must decide whether its own resources can satisfy this need, whether the plan can succeed without the stickers, or whether to risk starting plan execution on the assumption that a supplier for the stickers can be found before plan execution reaches the point where they are needed. It decides (or is told by our manager) that the stickers can be done in-house.

The customer agent now awards the selected bids, and work commences. Part way through the process, the box printer backs out, and pays the required decommitment penalty. Our agent re-opens bidding for box printing, receives and awards a bid. The customer agent monitors task progress, posting payments as agreed at task or task completion. Once all transactions are completed, the customer agent requests the respective markets to terminate the sessions.

## Related Work

To the extent that we require the existence of an external market mechanism as an intermediary, our proposed framework is similar to that of Wellman's market-oriented programming used in AuctionBot (Wurman, Wellman, & Walsh 1998). AuctionBot supports a variety of auction types each imposing a set of market rules on the agent interactions. Hence, the auctions, themselves, become the intermediaries. The entity that sets up the auction can specify certain parameters for the auctions. In contrast, our framework provides explicit market mechanisms which can not only specify and enforce auction parameters, but also support more complex interactions such as automated contracting in a variety of types of markets based on a market ontology. Furthermore, these market mechanisms also enforce general market rules and "social laws", such as government regulations, by which all participants must abide.

Rosenschein and Zlotkin (Rosenschein & Zlotkin 1994) analyze a variety of domains and propose a classification of problems into domains that are characterized by different types of negotiation among agents. They show that the behavior of multiple, interacting agents can be influenced by the set of rules (the protocol) that the system designers choose for the agents' environment. The purpose of these rules is to allow the agents to make constructive agreements. Their analysis assumes that the negotiating agents have similar capabilities. The protocol we present in this paper does not require that assumption.

Mechanisms to reduce counterspeculation, such as the Clarke tax mechanism (Ephrati & Rosenschein 1996) or the Vickrey auction (Vickrey 1961) have been proposed for automated negotiation of self-interested agents. The architecture we present can support the Vickrey auction, and eliminates one of its limitations by providing a structure that can act as a trusted auctioneer (Sandholm 1996).

A variety of architectures have been proposed for single and multiple agents in different domains (see, for instance, (Eriksson & Finne 1997; Kinny & Georgeff 1997; Lejter & Dean 1996; Rodriguez *et al.* 1997)).

MAGMA (Tsvetovatyy *et al.* 1997), an open architecture for agents interested in buying and selling, supports both manual and automated negotiation with a limited form of Vickrey auction. Even though MAGMA already includes many of the features of the architecture we present here, MAGMA is intended for a more limited domain. An earlier version of the system we describe here is outlined in (Collins *et al.* 1998).

Substantial work is underway in standardizing an open architecture for electronic commerce (Tennenbaum, Chowdhry, & Hughes 1997; McConnell *et al.* 1997). Our architecture improves on these proposals by adding support for more complex negotiation protocols. For example, our architecture could be implemented by extending the framework presented in (Tennenbaum, Chowdhry, & Hughes 1997).

## Conclusions and Future Work

In this paper we have brought together ideas from recent work in market architectures for electronic commerce, and work in multi-agent contracting protocols. We have presented a generalized market architecture that provides support for a variety of transaction types, from simple buying and selling to complex multi-agent contract negotiations. We have also presented a protocol that takes advantage of the services of the market. Our market architecture is organized around three basic components: the exchange, the market, and the session. We have shown how the existence of an appropriate market infrastructure can add value to a multi-agent contracting protocol by controlling fraud and discouraging counterspeculation.

We have begun implementing software-agents to interact within MAGNET, and intend to benchmark their performance against human participants as well as each other. We are exploring how modifying the parameters of the market and session protocols affects the performance of the system. Eventually, we would like to open our testbed to outside participation over the Internet.

This work raises several interesting questions for future research. A game-theoretic analysis of the protocol could be done to determine optimal strategies for its use by agents. In particular, how should the decommitment penalties be used, and how should proposed decommitment functions be evaluated when computing marginal costs of plan alternatives. The methods for composition of possibly overlapping bids into feasible or even optimal plans must be worked out.

## References


Collins, J.; Jamison, S.; Gini, M.; and Mobasher, B. 1997. Temporal strategies in a multi-agent contracting protocol. In *AAAI-97 Workshop on AI in Electronic Commerce.*

Collins, J.; Youngdahl, B.; Jamison, S.; Mobasher, B.; and Gini, M. 1998. A market architecture for multi-agent contracting. In *Second Int'l Conf on Autonomous Agents*, 285-292.

Ephrati, E., and Rosenschein, J. 1996. Deriving consensus in multiagent systems. *Artificial Intelligence* 87:21–74.

Eriksson, J., and Finne, N. 1997. Marketspace: an open agent-based market infrastructure. Master's thesis, Computing Science Dept., Uppsala University.

Jamison, S. 1997. A negotiation protocol and market agent model for complex transactions in electronic commerce. Technical Report TR97-037, University of Minnesota, Dept of Computer Science and Engineering.

Kinny, D., and Georgeff, M. 1997. Modeling and design of multi-agent systems. In J.P. Mller, M. W., and Jennings, N., eds., *Intelligent Agents III*, Lecture Notes in Artificial Intelligence. Berlin: Springer.

Lejter, M., and Dean, T. 1996. A framework for the development of multi-agent architectures. *IEEE Expert.*

McConnell, S.; Merz, M.; Maesano, L.; and Witthaut, M. 1997. An open architecture for electronic commerce. Technical report, Object Management Group, Cambridge, MA.

Rodriguez, J. A.; Noriega, F.; Sierra, C.; and Padget, J. 1997. FM96.5 - a Java-based electronic auction house. In *Second Int'l Conf on The Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'97).*

Rosenschein, J. S., and Zlotkin, G. 1994. *Rules of Encounter.* Cambridge, MA: MIT Press.

Sandholm, T., and Lesser, V. 1995. Issues in automated negotiation and electronic commerce: Extending the contract net framework. In *1st International Conf. on Multiagent Systems*, 328-335.

Sandholm, T. W. 1996. Limitations of the Vickrey auction in computational multiagent systems. In *Second Int'l Conf on Multi-Agent Systems.*

Smith, R. G. 1980. The contract net protocol: High level communication and control in a distributed problem solver. *IEEE Trans. on Computers* 29(12):1104 - 1113.

Tennenbaum, J. M.; Chowdhry, T. S.; and Hughes, K. 1997. eCo System: CommerceNet's architectural framework for internet commerce. Technical report, Object Management Group, Cambridge, MA.

Tsvetovatyy, M.; Gini, M.; Mobasher, B.; and Wieckowski, Z. 1997. MAGMA: An agent-based virtual market for electronic commerce. *Journal of Applied Artificial Intelligence* (6).

Vickrey, W. 1961. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance* 16:8–37.

Wurman, P.; Wellman, M.; and Walsh, W. 1998. The Michigan Internet Auctionbot: A configurable auction server for human and software agents. In *Second Int'l Conf. on Autonomous Agents*, 301–308.