Multi-Agent Simulation of Financial Markets

Olga Streltchenko, Yelena Yesha, Timothy Finin
Department of Computer Science and Electrical Engineering,
University of Maryland Baltimore County,
1000 Hilltop Circle,
Baltimore, MD 21250, USA.
{streltch|yeyesha|finin}@cs.umbc.edu

1 Introduction

Simulation of financial markets is a new fast growing research area with two primary motivations. The first is the need to provide a development testbed for the ever increasing automation of financial markets. The second is the inability of traditional computational mathematics to predict market patterns that result from the choices made by interacting investors in a market.

Section 2 surveys the current state of financial market automation. Section 3 discusses the importance of simulation to help understand the patterns that arise from different investment strategies; it briefly surveys the literature and identifies some open problems, including the design of a general-purpose financial market simulator.

The design of a multi-agent simulator of a financial market is the subject of Section 4. It is a challenging task due to the operational complexity and computationally costly decision support. We discuss an approach in which the complexity of the financial market functionality is decomposed into relatively simple tasks and processes. In general, we separate the transactional and decision-support intelligence of the market agents. Futher, market entities are singled out and defined as software objects; the interaction protocols are specified; and the simulator architecture is presented.

We conclude with the discussion of possible applications of the simulator, stressing it extensibility to handle other marketplaces, such as emerging markets, energy and bandwidth markets, etc.

2 Automation of Modern Financial Markets

We define market automation as the execution of trades by software agents based on goals specified by human agents.

Modern financial market professionals recognize the benefits of finincial market globalization, worldwide trading through electronic interconnectivity, and around-the-clock market accessibility, as a means to increase liquidity and market efficiency. Automation is seen as a way to achieve these ends. Frank Zarb, chairman and CEO of NASD, formulated a vision of digital, global, continuously available security trading with real-time quotation and order execution systems accessible worlwide over the Internet via a number of computing devices in his speech to the National Press Club in 1999 [30]. It forecats rendering physical trading floors obsolete and completely replacing them by electronic transactions. Complete automation is necessary to support 7/24 availability.

This vision of financial market automation is not isolated. It is a part of a wider phenomenon of globalization and development of digital economy. The technology that can turn this vision into reality has arrived. The trend to automation is supported by developments in electronic commerce, agent technology, and achievements in mathematical and computational finance.

This section gives a brief introduction into the operations of financial markets and survey the current state of financial market automation. Electronic Communication Networks (ECNs) are of special interest here due to their pioneering role in automation of securities trading.

2.1 Securities Markets

Modern financial markets deal in standardiazed obligations in place of goods and commodities. In the current trading paradigm, the actual order execution, or securities exchange, is sepaprated from an investor by several levels of intermediaries (see figure 1). In general, a hierarchical structure is a characteristic feature of modern securities trading. Access to the actual financial market is open only to authorized brokerage houses. Institutional and retail customers contact a broker to place their orders. Once an order is initiated by an investor (placed with a broker) it goes through three distinct stages: order routing, execution, and clearing and settlement. Routing involves communicating, possibly through a number of intermediaries, the details of an order from a broker with whom the order has been placed to a market agent, human or software, responsible for order execution. Execution involves agreement to exchange securities, while clearing and settlement commits the transaction by checking availbility of the resources committted by both sides and exchanging the securities.

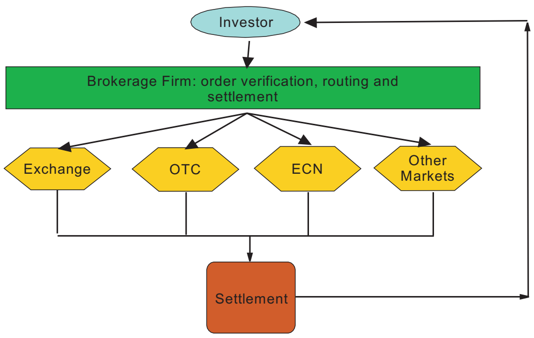

Figure 1. Current trading model; adopted from [4].

Not only the objects of exchange, but the exchange protocols have become standard. Exchange protocols are designed to achieve a price for a security that the trading community agrees on, i.e. is willing to trade; this procedure is called a *price discovery mechanism*. There are two major types of markets defined with respect to the price discovery mechanisms that they implement: the *dealer market* and the *auction market*, with a number of hybrid types between these two extremes. In an auction market a market specialist acts as a facilitator and does not own traded securities. Trading proceedes as a continous double auction with the market specialist acting as an auctioneerer assisting the sides in matching their offers and arriving at a mutually agreed on price for the trade. There is only one market specialist for a particular security. In a dealer market a market specialist owns the securities being traded. A dealer posts his/her ask and bid prices, and all the transactions occur between him/her and the investor (more precisely, an agent acting on behalf of the investor). Dealer markets allow several competitive dealers trade in a particular security.

Security trading occurs at exchanges, over-the-counter markets (OTCs), and electronic communication networks (ECNs). An exchange, such as New York Stock Exchange (NYSE) [19], is an example of a hybrid market with repect to its price discovery mechanism. It operates as an auction market most of the time, with the market specialist occationally acting as a dealer when the market conditions require so. This might happen when there is a considerable disparity between supply and demand that threatens to dramatically swing the prices, and a stabilising intervention of the market maker is desirable to the investor community to prevent trading from stalling. Over-the-counter markets, such as NASDAQ, are dealer markets. ECNs are neither auction nor dealer markets. In fact, they are not considered to be true markets since they lack a price discovery mechanism. They will be discussed in detail below.

A large degree of standardization exhibited by modern financial markets allowed to automate certain market procedures. We will now proceed to dicuss the current state of financial market automation.

2.2 Current State of Financial Market Automation

As was stated above, there are several levels of intermediaries to a financial transaction spaning order routing, execution and settlement. The efficiency of such a pyramid depends on the speed and quality of communications between the parties as well as streamlining of clearing and settlement for the trade.

Advances in clearing and settlement automation were made in in the 1970s [19]. The standardized obligations traded on financial markets readily yeilded to software representation due to their abstract nature; the security ownership information is now almost comletely relegated to digital information systems.

Throughout the century financial markets made heavy use of every advance in the communications technology. Currently, information exchange is almost fully automated. Electronic quotation systems came into existence in the early 1970's [19]. These display price quotes as well as post-trade information, such as transaction volumes, etc.

Order routing became automated with the development of order routing systems like SuperDOT (Super Designated Order Turnaround) at the NYSE, which significantly increased trading throughput. Currently, all exchanges and most OTCs receive most of their order volume through automated order routing systems.

Order execution requires the most human involvement. It has been automated for small-sized orders by some exchanges and OTCs. Such automated order execution systems handle both market and limit orders. As a rule, there is an upper bound on the transaction volume for such a sytem. This number varies for a particular security depending on how frequently the security is traded, i.e. depending on its liquidity. An example of an automated order execution system is SOES of NASDAQ which became ready for use in 1984 and allowed transaction volume up to 1000 shares.

Extending market activities into after-hours and competition from ECNs (see below) has forced further sophistication of automated execution systems. For example, in the 1990s SOES was enhanced by SelectNet which has certain negotiation capabilities. That allowed the sytem to implement basic price discovery, handle disproportinate orders, and, therefore, to be able to execute transactions of higher volume.

2.3 Electronic Communication Networks

ECNs were designed to compete with existing market makers (specialist) on the financial markets, particularly NASDAQ. The first ECN to become operational was Instinet, which opened in 1969. Currently the number of competitive ECNs is 9.

ECNs are comletely automatic: no human market maker is involved in trading. They emulate the operations of an auction-type exchange by electronicly matching buyers' and sellers' orders. However, the original matching procedure was very basic: orders to sell and orders to buy were gathered from the members and searched for matches in volume and price. If a match was found, a transaction went through. Otherwise, nothing happended. Such a procedure does not guarantee order execution; it remains a passive match-maker and has to avoid disproportionate client portfolios. Since there are no price negotiation capabilities, it does not lead to price discovery, and, therefore, cannot be considered an electronic market. Traditional exchanges became sources of price information for ECN members.

Despite the above mentioned drawbacks, ECN market share has been steadily growing. This is due to their after-hour trading capacities and low transaction costs - the benefits of complete automation.

The early example of automated order execution by the ECNs stimulated the development of automated order execution systems within the traditional markets. ECNs responded to the competition by extending their functionality to include order negotiation (price discovery). Once full market functinality is achieved, an ECN can apply for a change of status and become an exchange. Archipelago, Island, and NextTrade have done so.

2.4 Further Examples of Financial Market Automation

The major factor currently driving financial market automation is the Internet. Its influence on securities trading has been dramatic; investing is now as easy and accessible as playing an internet game. Its effect on the investor community is now under scrutiny by the academic community; for an early attempt to dicuss the issue, and, more generally, the effects of financial market automation, see [27].

The emergence of online trading was marked by the appearance of purely electronic brokerages, along with traditional brokerages opening online trading sites. The Inernet has gone beyond just being another channel for order placing; opportunities for faster and more comprehensive research into an individual company or security performance, visualization and analysis tools for processing historic data offered on online brokarage sites, 7/24 accessibility are just a few services offered by online brokerages. Note that 7/24 availablity of an online broker does not directly translate into 7/24 availability of financial markets. Orders can be placed with a broker continuously, but they will be executed according to the trading schedule of a chosen financial market. A number of issues concerning online brokarages were studied in a Securities and Exchange commision special study report [1].

Advances in automation have not so far reduced the number of intermediaries to a transaction. For the problems that arise as a result of the hierarchical structure, such as order internalization, and their effect on overall market efficiency, see [4]. The paper assumes an optimistic attitude and offers a direct trading model (figure 2), - disintermediation facilitated by completely automated order routing.

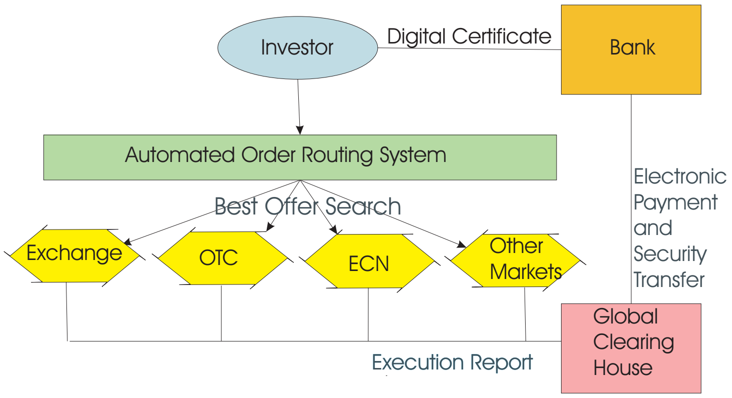

Figure 2. Direct trading model; adopted from [4].

Another matuaring aspect of financial market automation is automated decision support on the part of the investors. A large fraction of the investor community, e.g. all institutional investors, relies on mathematical/computational modeling of the market to support decisions concerning their portfolios. In most cases, the computational results are verified by a human agent before corresponding orders are made. A notable exception to this procedure is program trading. Currently program trading employs fairly straightforward algorithms to support decisions about portfolio rebalancing. They implement what is called technical trading, - searching for particular trends in the price movements of a chosen securuty(ies) and responding in a predefined manner once the trends have been observed. In general, the performance of technical trading is hotly debated by the financial community. In one particular instance, unsupervised technical trading - program trading - had a devastating effect on the stock market; during the 1987 stock market crash program traders picked up, followed and amplified the trend to sell [27]. The lesson of 1987 suggests the importance of the emperical study of large systems populated by automated decision-makers of various degrees of intelligence.

However, the importance of automated decision support for investors is almost self-evident. A financial market is a highly dynamic environment where the ability to act upon the latest price quotes is crucial: if a price moves away from the quote used to make a decision significantly while the decision is being made, the investor cannot be sure that his/her assesment of his/her position will hold in a significantly changed environment. The same argument applies to a market maker who needs to update price quotes of the securities that he/she is responsible for to maintain a dynamic balance between supply and demand. The speed of response to market signals is not the only incentive: automation is a precondition to global continuous, 7/24 trading.

3 Simulation of Finanacial Markets

The complexity of financial markets defies traditional mathematical and computational analysis [21]. Since many of the questions we would like to ask about financial markets are not amenable to theoretical analysis, experimental analysis suggests itself. The idea of experimental study originates in physical sciences, where a controlled experiment involves repeatability and parameter isolation. A financial market permits neither of these. (For example, we can not hold inflation and interest rates steady, as we experiment with a variety of pricing techniques.)

The impossibility of conducting controlled experiments has been identified as one of the major hindrances for transition of emperical finance into an axiomatic theory [5]. Multi-agent simulation of financial markets seeks to address this problem by providing the conditions for a controlled experiment, and thus allowing us to isolate cause and effect relationships in the market. The use of multi-agent simulation, therefore, may help greatly advance the theoretical developments of finance theory.

For instance, multi-agent simulation can aid in the understanding of derivative securities pricing for which intuition often takes the place of exact science among the practitioners [18]. This is especially true for energy and bandwidth contracts, since storability of the underlying - a fundamental assumption of traditional derivatives pricing techniques - is not a characteristic of these markets.

In traditional mathematical finace, all market participants are modeled the same way, with each having equal powers, and being subject to the same constraints. This is an idealized setting. For example, airline companies and oil refineries both come to the market to trade in fuel, but they do so with fundamentally different perspectives, one of them being required to sell, and the other required to buy. Speculators have a perspective different from either. One can identify an arbitrary number of individual investor profiles in a single market. A theoretical study of such a model is virtually impossible. The strength of multi-agent approach is the ability to (experimentally) study large heterogeneous populations.

In this section we survey the work of three research communities whose work, in our opinion, comprises the background for further advances in the field. We distinguish two main branches in this area of research. One concernes itself with setting the environment, e.g. simulator and agent archtecture, functionality and implementation. The other one concentrates on specific application domains, special attention being given to agents' intelligence (decision-support algorithms) and environment modeling. Most research groups have not drawn this distinction, and the projects we survey below have made contributions to both branches.

3.1 Trading Agent Platforms

Motivated by the idea of facilitating electronic purchases over the Internet and reducing human involvement in corresponding search and, in some cases, negotiation, several research groups developed trading agent pletforms. Several such systems are surveyed in [16]. The features that received special consideration in such systems are

- search for the most suitable product or a set of alternative products;
- search for the best merchant or a set of merchants for further negotiation; and
- negotiation itself with a purchasing decision at the end of the process.

Not all of these features were necessarily implemented; most of the agent systems offered Web-search capabilities while leaving negotion and/or decision-making to the user.

Several agent platforms, such as Kasbah, e-Mediator, and AuctionBot, implmented an auction-type interactions or one-on-one price negotiation and offered a number of bidding strategies to be chosen by the user prior to negotiation. Such systems are of interest to us since they actually emulate a certain marketplace as well as well as supply basic regulatory and administrative infrastructuse.

eMediator, an electronic commerce server from the Multi-Agent Sytem Research Group at Washington University described by Sandholm in [22], is arguably the most advanced of this family. It features eAuctionHouse that implements a choice of auction types, strategic (price-quantity graph) bidding and combinatorial bidding (bidding on a set of products), and creation of personalized Java agents that perform trading on the server; eCommitter, a decision support engine for leveled commitment contract optimizer; and eExchangeHouse, an exchange planner. A number of other features are under development. The research that developed around the server mostly concerns semi-binding commitment (level commitment), agent strategic behavior in an auction-type environment, namely coalition formation, and combinatorial bidding winner determination. We recommend that the interested reader peruse a collection of publications by the Multi-Agent System Research group at `http://www.cs.wustl.edu/~mas/`.

The systems discussed above facilitate product search on the Internet while providing the user with a varying degree of negotiation and/or bidding support. Mostly they rely on fairly simple interactions and decision rules. Those building a financial or economic system simulator can benefit from their advances in agent platform development, agent architecture, distribution, and other system related issues as well as the decision support provided for bidding and negotiations. However, such systems do not provide the infrastructure and decision support specific to the financial market domain.

3.2 Simulation of Simple Economies

Auction as a price discovery mechanism has recieved significant attention in the experimental research of multi-agent interactions. Part of the reason for this is that individual agents rely on fairly simple decision rules in such an environment. However, the apparent simplicity of such systems may be deceptive.

Evidence of the emergent complexity in systems employing simple pricing techniques is presented in a series of publications on multi-agent simulation of simple economies by the Information Economies research group at IBM's T. J. Watson Research Center (see, for example, [10] and [11]). It is important to note that oversimplification of decision support in a multi-agent system can lead to disastrous consequences for the economy, such as price wars and stagnation of trading, examples of which are give in Tesauro et al [25] and Brooks et al [3]. This phenomenon is not confined to experimental systems - recall the example of the stock market crash of 1987 given above. Due to such possible outcomes, Kephart, Hanson and Greenwald in [9] stress the importance of multi-agent simulation as a testbed for novel decision support algorithms before implementing them in a real-world system.

The agent decision support algorithms chosen by this reasearch community are variations on Q-learning, an instance of a wider class of reinforcement learning algorithms. Kephart et al [9] overview a number of experiments with an information economy populated by heterogenous agents, some of which emply Q-learning. The model of the information economy presented in [9] assumes a dynamic posted pricing paragigm, that is a model in which buyers do not negotiate posted prices while sellers update their postings at will. In fact, the economy is modeled as a commodities market populated by rational self-interested agents optimizing their utility function via one of a number of algorithms ranging from a simple incremental price increase(decrease) algorithm to foresight-based Q-learning.

While financial markets are not explicitely considered here, the multi-agent systems under consideration possess the price discivery mechanisms present in financial markets. Insights into behavior of large systems presented by this community may help to better set up the expeimental environment for financial market simulation.

3.3 Financial Market Simulation

While the above examples use the price discovery mechanisms similar to those present in financial markets, they do not explore the financial domain explicitely. We now proceed to discuss simulations specific to financial markets.

Several early efforts in the area of simulated financial markets are surveyed by LeBaron in [12]. The paper recognizes the strength of an agent-based approach in understanding the dynamics of interaction among heterogeneous agents and agents learning from the environment. The survey concentrates on the concrete microeconomic (market) models and learning techniques, predominantly genetic algorithms, used in the surveyed work.

One of the most prolific research efforts in the area, also surveyed in [12], is the Santa Fe Artificial Stock Market. Its original design and experiments are presented in [17], [2] and [15]. For a brief digest of the model also see [26]. This line of research, along with further experiments on the same platform (see, for example [8]), studies success, in a game-theoretic sense, of technical traders relying on a variety of learning techniques borrowed from machine learning to improve their forecasting ability and gain a competitive advantage over other market participants.

The experience of the Santa Fe Artificial Stock Market and a number of several other simulation effors are summerized by LeBaron in [13]. The paper offers a classification of design issues that a builder of a financial market simulator may face. The first category, agents, discusses the types of market participant agents with respect to their intelligence. The stratification of agents on the bases of their intelligence parallels the classical AI classification offered by Russell and Norvig [20]. These agents are assumed to be price-takers, or regular investors, since the price setting issues are treated separately, in the next category, called trading. Trading covers both the trading protocol and the determination of asset prices. It differentiates between simulating the price movements and replicating market infrastructure with an appropriate price discovery mechanism. The former approach does not call for a market specialist agents while the latter does. The next design issue concerns the securities themselves. The author notes the tendency of simplification in modeling the traded assets and discusses the difficulties associated with incorporating the fundamentals into the simulated securities. The number of securiries in a simulation, and the related issue of diversification, are mentioned. Next, the issue of market evolution, or market agent learning, is tackled. Learning based on genetic algorithms is considered as a means of emulating dynamics and growing the expertise of the market participants. The necessity of model validation and possible approaches to it are discussed under the category of benchmarks/calibration. The category of time groups together several issues: agent memory for learning algorithms, lag in response to new information arriving to the market, and synchroneity. It fails to discuss simulation time horizon. The author also devotes a separate paper to the in-depth treatment of the issue of agent memory for learning algorithms in a financial market simulation and the influence of this parameter on the overall dynamics of the sytem [14].

The Santa Fe Artificial Stock Market group's work is the most advanced so far in the field. However, it is heavely biased towards machine learning techniques to provide agents' decision support, primarily genetic algorithms. The study is thus limited to technical trading, - the area that traditionaly relies on the use of genetic algorithms. The vast majority of the investment community does not rely on technical trading; optimization in conjunction with risk management is the technique of choice for large (institutional) investors. Hedging and risk management are not directly addressed in technical trading. The genetic algorithm approach, or the survival of the fittest, tends to favor a winner with the most money. Thus, risk-averse investors do not win in such a society, and get "weeded out" by the algorithm. Their success as hedges is not recognized by the procedure. Therefore, an important and large fraction of the investor community is not properly modeled by the genetic algorithm setting.

The issue of simulation horizon is important when modeling a community of investors with heterogeneous goals, like speculators, who invest for immediate reward, and long-term investors, who invest for future income. A financial market is populated by both types, and their interactions are important for maintaining dynamic balance. A simulation with a short horizon will concentrate on observing behavior of more active traders while a longer horizon will give insights into long-term investors' behavior.

3.4 Concluding Remarks

Existing research concentrates on a series of individual simulation problems. There has not yet been a unified study of the common properties of finacial market simulations. The focus of attention has been agents intelligence. AI techniques clearly dominate the landscape, while traditional matematical models of market agent decision support, such as (stochastic) optimization, (stochastic) differential equations, etc., have not been implemented for such systems. As was pointed out above, AI research operates under a heavy game-theoretic perception of success: finding a winner with the most wealth. This approach ignores the main concern of mainstream investors: risk management and hedging. In our opinion, multi-agent simulation of financial markets should

- elaborate its criteria of agent success and put more stress on risk management;
- broaden the choice of decision support techniques to include traditional portfolio management and security pricing techniques developed by computational finance community;
- combine the achivements of AI and computational finance to achieve better market performance;

Financial markets are highly regulated and standardized in their operation. While the motivation behind individual investors' actions can be extremely diverse, all market participants are subject to a fixed set of protocols that regulate securities exchange. These procedures are specific to the domain, i.e. a securities exchange operates differently than an auction. These considerations suggest the necessity of a unified approach to formal modeling of financial market infrastructure - the issue that we proceed to discuss below.

4 A Multi-Agent Environment for Financial Market Simulation

This section will adress one of the open problem identified above and offer a methodology for the analysis and design of a general-purpose financial market simulator. The exposition extends the work presented in [23] and [24].

The methology to be presented separates the transactional and decision-making components of the simulated environment. This is essential to achieve a flexible and extendable multi-agent environment which frees a researcher from the neccesity of implementing the underlying market infrastructure while allows him or her to choose among the available options for agent decision support or supply new algorthms to the agents. This also provides for independent developments in the areas of market functionality and decision support.

The multi-agent environment is modeled after a generic dealer market. While a number of features we judged to be essential for a valid financial market simulation a recreated within the environment, certain simplifications were made to reduce its compuational complexity. The simulator offers a universal quotation sytem, a unique (for each instance of the simulator) money-market, a security registration facility, order posting and trade clearing. The underlying infrastructure supports efficient and correct message flow among the trading agents.

Financial market participants are highly heterogeneous with respect to their ability to quote prices, engage in short-sales and borrow money, and many other respects. The simulator currently distigues one type of a priviliged market participant, the *broker*, or *market specialist*, who has the unique power to quote a price, and who provides liquidity to the market. Other market participants are modeled after regular investors and are assumed to have no such power and to be price-takers. This basic division can be enhanced to distinguish several layers of market specialists and intermediaries with a variety of market powers. However, such fine stratification is not necesary for most basic applications, and, therefore, was not implemened in the current version. Further we will refer to market agents as investors or brokers depending on there priviliges with respect to quoting prices.

The brokers communicate their prices to the investor community via the quotation system. Investors convey their orders to chosen brokers, and they trade. However simple this scenario may sound, it comprises a lot of intricacies that have to be unrevelled on the implementation level. Integrity of the exchanged information must be guaranteed, as well as timely delivery of the messages; trades must be appropriately cleared and monitored; investors' portfolios must be maintained, etc. In order to correctly implement the market operations the overal complexity of interactions and functionality is decomposed into relatively simple tasks and processes. We further present a number of steps that achieve this goal. The decomposition is done according to *Gaia methodology* [28], [29]. To facilitate our discussion we proceed with a brief review of Gaia methodology.

4.1 Gaia Methodology

Gaia methodology [28] is a body of high-level software engineering techniques particularly suitable for hierachical systems of heterogeneous agents which make use of significant computational resources. While offering a means of detailed analysis of the target application, it does not presuppose any particular implementation platform. It does not impose any checks on the overall system complxity or agent intelligence, but the organization structure of the system and the agent properties are assumed to be fixed during run-time.

Gaia recognizes *abstract* and *concrete* entities. Abstract entities are used during the analysis stage to conceptualize the system, while concrete entities are used within the design process and typically have direct counterparts in the run-time system.

The topmost abstract entity in the Gaia concept hierarchy is the system. Gaia views the system (organization) as a collection of roles that interact with each other. Each roles is defined by its *responsibilities*, *permissions*, *activities*, and *protocols*. Responsibilities determine role functionality. Roles correspond to atomic or elementary responsibilities. In order to realize its responsibilities, a role possesses a set of permissions or rights associated with the role. Access to information sources, like information generation, modification or reading, is under the purview of permissions. Computations associated with a role are called activities. Roles can interact according to a set of protocols. The dependencies and various relations among the roles are captured by the *interaction model*. The interaction model is essentially a directed graph with roles as nodes.

Once the analysis produces fully elaborated roles and interactions models, the goal of the design stage is to transform the analysis models (abstract entities) into concrete entities having implementation counterparts. The *agent* model defines the agent types present in the system. The *service* model defines the actions performed by agents. The *acquaintance* model represents the communication channels among the agents.

The agent model can combine several closely related roles as a single agent type for efficiency. It is convenient to think of an agent type model as a tree with the roles being leaf nodes.

The service model expresses agent functionality. The services are derived from the protocols, activities and responsibilities of the individual roles combined into an agent type.

The acquaintance model is a directional graph with its arcs corresponding to communication pathways.

Below we describe analysis and design of a multi-agent environment simulating a financial market based on the guidelines of Gaia methodology. As we proceed we will offer further explanation of the methodology when necessary.

4.2 Analysis

The analysis of the system aims to decompose system's functionality into a number of atomic functions or actions. These atomic actions are segregated and attributed to the entities that perform them. This leads to delineation of the roles and results in a roles model that describes the system.

Our intention is to separate transaction processing from decision making. Therefore, each role will be responsible for either or them but not both. This distiction ultimately propagates to the concrete implementation of a financial market simulator described below.

4.2.1 Roles Model

We introduce the following roles: Market, Investor, Broker, Investor Utility Function, and Broker Utility Function.

Orderly trading is impossible without a regulatory and administrative authority which provides the universal quotation system, security regulation, trade clearing, etc. The *Market* (M) role is responsible for providing market participants with a centralized adminstrative and regulatory environment. Its basic responsibilities include

- registering admissible securities upon the requests from the brokers and mainatining a list of registered securities (securities available for exchange);
- maintaining a centralized offer posting service and posting new offers for registered securities originating from brokers.

Additinally, if a particular simulated environment poses certain security concerns (as it may happen in a distributed simulation, for instance), the Market role may be responsible to act as a clearing house, an arbitrating authority, as well as perform some system functions as syncronization, quality-of-service assurance, etc.

As has been previously outlined, brokers establish prices and provide liquidity to the market. Investors are price-takers. All the trades have to be registered with the market (in order to verify the authenticity of the exchanged securities). Only brokers have write access to the quotation system, and, therefore, only they can be identified as potential counterparts to a trade. Thus, the investors trade solely with the brokers and are virtually transparent to each other.

The *Investor* role is responsible for

- mainataining a portfolio of currently held securities;
- obtaining information about desired changes in the current portfolio;
- obtaining appropriate price quotes along with the quoting broker information;
- choosing suitable Brokers based on their price quotes and serving them with corresponding market orders;
- maintaining a list of unacknowleged (open) orders.

Its responsibilities may also include serving limit orders to the brokers. The protocols that support these responsibilities are as follows. Once a need for a trade has been comunicated by the *Investor Utility Funcion role*, Investor's decision-making counterpart (described below), the Investor queries the Market, namely its quoting system, to obtain security offer information and choose the best offer depending on whether it needs to buy or sell, forwards this information back to the Investor Utility Function role, and recieves the transaction volume in reply. The Investor then issues orders for each individual security and initiates the transactions - forwards orders to appropriate brokers, and acknowledges the receipt of the ordered securities. Investor's read permissions open read access to the price quotation system while its read/write permissions allow portfolio and order manipulation.

The *Broker* role extends the Investor role with additional responsibilities which include

- registering securities that the Broker intends to offer for exchange with the Market;
- obtaining price quotes for those securities from the decision-making counterpart, *Broker Utility Function*;
- posting current security offers with the Market; and
- maintaining a list of unacknowledged (open) orders from the investors.

The protocols are augmented to facilitate the above activities. Brokers are granted write permission for the offer posting system.

A broker exercises direct control over security prices. The investors influence the security price evolution through their purchasing activity. The Investor and Broker roles are provided with their decision making counterpart, *Utility Function* (UF), whose responsibility is utility maximization or other optimal decision support.

The Broker UF role supplies the price quotes for the contracts as well as the desired portfolio holdings to the Broker. Its activity is the computation providing the above data, and the protocols it supports are: provide the security price and the desired volume. The Investor UF is analogous to the Broker UF, but the computation it performs supplies the volume quotes for the desired portfolio holdings only.

4.2.2 Interaction Model

The interaction model specifies message interchange in the system. The interatcions occur between the Broker and the Market roles, the Investor and the Market, the Brker and its Utility Function, the Investor and its Utility Function, and the Investor and the Broker. The nature of these interactions as well as some operations internal to the roles and triggered by external messages are described below.

As is mentioned above, the Market role maintains a pool of registered securities, - the securities available for trading. All the securities but the money market are offered by the broker community. The Market role is responsible for the money market; this security cannot be offered by a Broker. It performs the function of the *numeraire*, or the riskless security. Therefore, for every new security a Broker wants to offer it sends a message to the Market with the request to register the security. When the security is registered a designated area is created within the quotation system to accomodate further price quotes and the Broker is granted a write permission to the corresponding area of the offer posting service. A similar message exchage happens when a Broker takes upon itself to offer a registered security that it has not offered before; except that in this case there is no need to create a new quotation area and the Broker receives a write permission to the established area allocated to this security.

Price quotes are generated by the Broker Utility Functions and conveyed to their respective Brokers. Once the Broker obtains the price quote, it is conveyed to the offer posting service or quotation system within the Market. The Investors query the quotation system to obtain the price quotes. Those are forwarded by them to their respective Utility Functions for further processing. The Investor Utility Function role then generates the desired portfolio information and forwards it to its transactional counterpart, the Investor role. The Investor role initiates corresponding transactions to rebalance its existing portfolio and achieve the desired one by serving the chosen Brokers with their orders and making provisional chages to its portfolio. Until an order is fulfilled, the Investor maintains a list of open orders.

Upon the receipt of an order a Broker updates its portfolio accordingly and puts the order on the list of unacknowleged orders. The Broker then notifies its client, the Investor, about the fulfillment of the order. The Investor commits the portfolio update, acknowledges the receipt of the security to the Broker, and removes the order from its list of open orders. Upon the reciept of the acknowledgement from the Investor the Broker Removes the order from the list of unacknowledged orders.

The procedure described above makes several assumptions about the rights and responsibilities of the agents in the system and the communication infrastructure. The first assumption is that orders are binding; a Broker must fulfill an order that it recieves, and the Investor cannot withdraw an order once it has been placed. The second assumption is that no messages are lost or indefinately delayed by the system. This is a simple model serving as a basis for further elaboration. A Broker might not be able to fulfill all of the orders it receives due to certain portfolio constraints that could be imposed on it by the Market. If this happens, the Broker must remove or update its offer information with the Market, and notify the Investors about the change. The Investors roll back the provisional changes in their portfolios corresponding to returned orders. A system of penalties for the Brokers and restitutions to the Investors need to be implemented to prevent this from happening. These decisions are application-specific and should be made on a case-by-case basis.

A market order should generally remain binding for the Investor; a portfolio constraint check must be made while the provisional changes to the portfolio are carried through prior to the issue of the order. Note, that the changes made to the Investor's portfolio are determined solely by its corresponding Utility Function while in the Broker's case its Utilty Function cannot completely determine what changes to the Broker's portfolio will be requested by the investor community: it faces uncertainty about Broker's portfolio updates and can only make predictions/recommendations.

Another consideration should be given to a case of imperfect communication infrastructure. For instance, this may be the case in a large distributed simulation. In such a simulator certain rules have to be implemented to resolve the issue of outdated requests and timeouts. This issue has been exhaustively covered by the mainstream database research[7]. Once a transaction has been initiated, agents may utilize protocols developed for distributed database transaction processing to achieve a mutually consistent state of their respective portfolios.

4.3 Design

Now we proceed through unification of some of abstract entities to achieve concrete ones and further translate them into their implementation counterparts. As will be seen below, we unify transactional and decision-support roles to obtain a reasoning and acting agent, a market participant. When translating this design into a conctrete implementation it is desirable to keep the agent archtecture highly modular so that one type of decision support can be easily exchanged for another one depending on the application and user preferences. Certain applications call for adaptible agents, i.e. the agent itself must modify its decision-making procedure to accommodate changes in the environment. These decisions are application-specific and should be treated on a case-by-case basis.

4.3.1 Agent Model

Following the idea outlined above, we unify the Investor role with its Utility Function role to obtain an Investor agent; and the Broker role together with its Utility Function role yeild a Broker agent. These are the market participants populating the simulated environment. There could be an arbitrary number of these agents per instanciation depending on the user choice and the available computational resources. We will further refer to them as Investors and Brokers.

The Market agent encompasses only one role, the Market role. This agent is unique for each instanciation of the simulator and performs the function of the centralized administrative and regulatoty authority. It regulates the marketplace and provides the means to the Investors and the Brokers to find each other.

4.3.2 Service Model

The service model describes the interactions on the agent level and is derived directly from the Interaction Model for the roles.

The Broker agents contact the Market agent to register new securities and to post quotes for tradable securities. The Investor agents direct their price queries to the Market agent to obtain current quotes for registered securities. Once an Investor decides to initiate a transaction it sends an order message to a set of chosen Brokers. (Investors choose the best offer for each security, so individual security orders may be directed to different Brokers). Depending on whether the market model hold the orders binding for the brokers, the Broker either fulfills the recieved order or notifies his client about his inability to do so. The number and semantics of the messages exchanged in order to complete or abort the transacton between the Investor and the Broker may differ depending on the protocol utilized to achieve a mutually consistent state of their respective portfolios.

4.3.3 Acquaintance Model

This model represents communication channels in the simulated market. Observe that, until an Investor is ready to initiate a transaction, there is no communication between the Investor and a Broker. Also, individual Investors do not communicate with each other. All the information available to an Investor is contained in the quotation system. Brokers have access to their respective trade histories. In most applications, Brokers will be assumed competitive and will not share their private trading histories.

Once an Investor wishes to make changes to its portfolio it obtains relevant offer information from the Market. All the message traffic concerning securities exchange happens between the Investor and a chosen Broker. Brokers communicate to the Market to post their offers or register new securities as well as to obtain information from the quotation system. Note that, since a Broker is also an Investor, it may initiate transactions with other Brokers. The graph corresponding to this model is presented in figure 3.

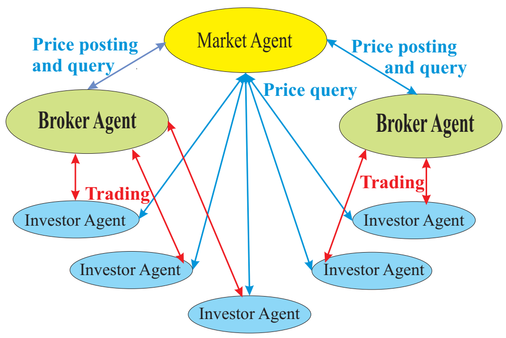

Figure 3. Acquaintance model graph.

4.3.4 User's Perspective

The operations of the tranasactional part of the simulator are modeled by the general trading rules of a dealer market, and therefore can be assumed to apply without major change through a number of different applications. The decision support used by the agents is highly application-specific and varied even within the same application. For example, a stock price evolution in a community of technical traders can be studied under a variety of diverce technical trading rules. Within a particular application domain a wide spectrum of decision support algorithms need to be supplied.

In the market model described above, where the market community is stratified into invetors and brokers, there are two types of decisions to be made. A security price quote needs to be generated (this action is available only to the Broker agents), and the protfolio rebalancing information needs to be generated (this is done by both Brokers and Investors). These decisions are made based on the following:

- the information, available from the Market, such as current securities prices and the money market interest rate, as well as, possibly, the price and interest rate history;
- agent's internal believes about the market properties.

The above considerartions suggest that it is easy to develop a standard interface between the transactional and the decision-support components of the agent architecture, since the inputs and the outputs are determined. Thus, the decision support becomes the only application-specific component of the simulator. Once the necessary decision support modules have been supplied the simulator is ready to perform experiments.

The basic overall architecture is depicted in figure 4. A user instantiates a model by specifying the agents using the *Agent Creation* module. At this step, as the Market agent is created the simulation horizon is set. Broker and Investor agents receive only their most basic properties, such as thier identification numbers. Further, the simulator requires a set of initial data which is supplied at the Data Generation step:

1. Using the *Utility Function* module, the user specifies the utility function and the portfolio constraints (e.g. short-selling and borrowing constraints) for each Broker or Investor agent. The set of utilities is application-specific and extendible. The ability to handle a particular utility depends on the decision support algorithm chosen for the simuation.
2. The *Decision support and learning* module is used to endow the agents with their respective reasoning capabilities. The algorithms should be chosen to support agents' utilities if optimization is used as a decision support technique.
3. An initial portfolio is assigned to each Broker or Investor agent using the *Portfolio Definition* module. The initial portfolio may reflect a position in the money market as well as other types of securities.

Notice, that every agent in the system is initialized individually. This implies that even within the same type of agent, like a Broker or an Investor agent, a great variety of personal features can be implemented, like distinct utility functions and market constraints, different pricing algorithms, and so on. The simulator aims to recreate a variety of investor types and any number of investors and brokers, as well as to accommodate an array of market models.

Once the setup is over the user starts the simulation. The *Recorder* component performs the necessary data capture and visualization. The price history is made available for the market participants. The transaction log is

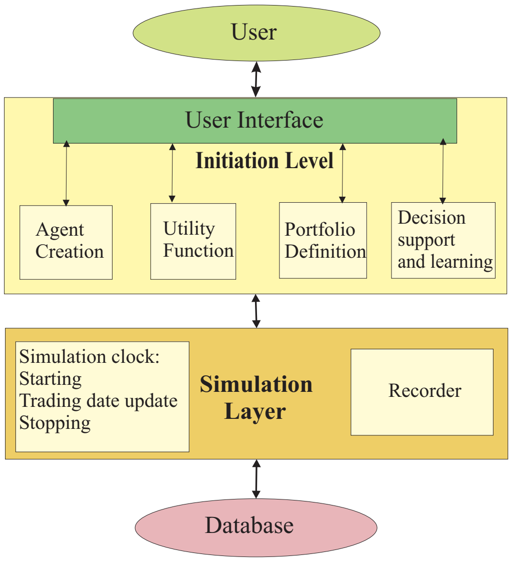

Figure 4. Reference architecture.

not seen by the agents. Broker agents may maintain their own transaction histories, which, in most applications, will not be shared among the Brokers (remember the competitiveness assumtion). However, certian studies may target broker coalition formation or other types of cooperative behavior. In this case, the user may modify the permissions on the inidividual transaction histories to accomodate such behavior.

The simulation ends as it reaches the horizon. Agent success is measured through the value of the terminal portfolio. Establishing an unambiguous procedure to value the resulting portfolio is a challenge to the decision support designer. If the terminal portfolio consists of securities for which the market community agrees on a price, i.e. highly liquid securities, the portfolio can be valued against the market. If there are thinly traded securities in the agent portfolios the valuation becomes subjective. The latter situation can be avoided by ensuring that such illiquid securities expire before the simulation ends.

The above describes an operation of a general-purpose financial market simulator. Only the most general assumptions are made and universal activities are described. We will proceed with a specialization of the simulator to a particular application domain, namely a derivatives market.

4.4 Simualting a Derivatives Market

Understanding the ties between derivative securities pricing and trading incentives for both parties of the trade is crucial to the development of a mature finanace theory. The key to it lies in modleing of the trading parties as heterogeneous entities that are motivated to trade in order to meet their previous obligations or hedge their future exposures. While it is prohibitatively hard to theoretically model heterogeneity of the investor community, an agent-based system offers ample opportunities for experimantal research. Below we describe an experimental environment that can be employed for such a study.

To concentrate only on the aspect of price formation for the derivative securities, the simulation does not reproduce the price formation mechanism for the underlying assets. Instead, following an established procedure of computational finance, it models the evolution of the prices of the underlying via a discrete stochastic process (a scenario tree).

An example of a scenario tree for a discrete stochastic process with a finite time horizon is given in figure 5. Each node represents a possible state of the price process for the underlying at a particular time step. Every state except the initial one has a unique parent, and every non-terminal state has a set of child states. The outgoing edges of a particular state in the tree connect it to all possible states of the price process evolution for the next time step, provided that the given state occurs. Each path from the root to a leaf node corresponds to a single scenario. The probability of each scenario is a product of the probabilities on the edges corresponding to it. For further details of the mathematical modeling of market processes we refer the reader to [6].

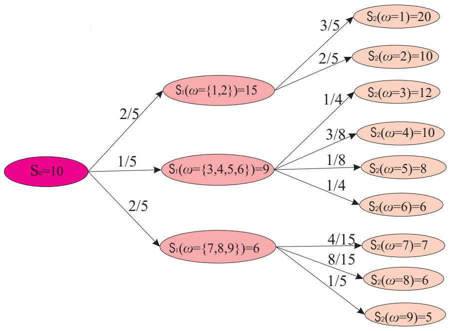

Figure 5. A sceanario tree.

To supply the underlying prices to the rest of the agent community we introduce a single Broker agent, called *Stock Broker*, whose Utility Function component performs a random walk through the scenario tree to establish the underlying price vector at each trading date. The scenario tree is given to the simulation during the data generation step. An additional module within the Initiation level, namely a *Scenario Definition* module, is responsible for this step (see figure 6 for extended architecture). The Scenario Defenition module presents a choice of a number of frequently encountered discrete stochestic processes and allows the user to generate a custom tree from user data. For instance, a binomial tree can be generated given the upward price move coefficient and the probability of such a move.

The interactions in a derivatives market simulator proceed as follows. The Stock Broker generates a vector of the basic securities prices and enters the quotes into the quotation system. A Broker trading in derivative securities observes the prices of the underlyings, and decides on the derivatives prices according to its private valuation algorithm. Investors obtain price quotes for both the underlying assets and the contracts written on them (derivatives) and make their portfolio rebalancing decisions. The brokers provide liquidity to the market, i.e. satisfy the investors' requests.

The goal of such experiments is to make a Broker strategic, i.e. to enable him to explicitely include market

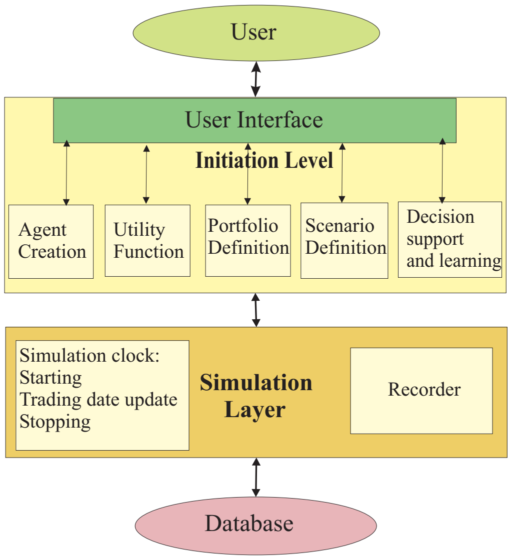

Figure 6. Reference architecture for a derivatives market simulation.

response in its pricing of derivative securities.

Further possible applications may include emergent market, electricity and bandwidth exchanges, etc. These types of markets are of special interest because of the lack of established decision support techniques and the need to experimentally validate the proposed ones.

5 Conclusions

Given a financial market, there are two main reasons to simulate a it. One is to provide a testbed for the components (such as price discovery) necessary to fully automate the market. The other is to provide an experimental setting to observe the consequences of a range of investor behaviors. The second case can be thought of as a special case of the first (since decision support is one of the components required for an automated market), but even if market automation were not a goal, an experimental environment would still be desirable.

Once an order is placed, there are three stages to a financial market transaction: order routing, execution, and clearing and settlement. The first and last of these have been fully automated, while the negotiation and/or decision necessary for execution is automated only in limited situations. In general, decision support, whether for an investor or a market maker, is still to be automated. Research here has primarily focused on the investor employing AI techniques such as genetic algorithms. These have been used to build agents that engage in technical trading with the objective of maximizing their profit from market transactions. Institutional investors, however, do not engage in technical trading, and, for the most part, are more interested in managing risk than in maximizing return on their market investments. Automation of the market maker's decision support has not yet received any attention.

To facilitate experimental study of a wide range of financial market problems, we propose a unified approach to simulator construction which recognizes the objects common to all markets. Our approach decomposes the complexity of financial markets into relatively simple tasks and processes, and clearly separates the operational complexity of financial markets from the decision support that drives market agents. Our approach enables the design or simulation of a variety of markets, as well as the deployment of agents with a variety of decision support mechanisms. We ourseleves have simulated a derivative securities market using this approach, and are using it to study derivative price formation.

We see two main directions on the research agenda for the area of multi-agent simulation of financial markets. One direction elaborates on the system and agent architecture, functionality, and properties. The goal is twofold: to improve the modeling potential of such systems, and to enable the incorporation of software agents into real-world financial marketplaces (i.e. to automate marketplaces). The other direction concentrates on elucidating the patterns that result from the interaction of heterogeneous market agents. We feel that our proposed reference architecture will be useful in both pursuits.

References

[1] Securities and Exchange Commission special study: On-line brokerage: Keeping apace of cyberspace, November 1999. Available online at http://www.sec.gov/news/studies/cyberspace.htm.

[2] W. B. Arthur, J. H. Holland, B. LeBaron, R. G. Palmer, and P. Tayler. Asset pricing under endogenous expectations in an artificial stock market. In W. B. Arthur, D. Lane, and S. N. Durlauf, editors, *The Economy as an Evolving Complex System II*, Menlo Park, CA, 1997. Adisson-Wesley.

[3] C. H. Brooks, E. H. Durfee, and R. Das. Price wars and niche discovery in an information economy. In *Proceedings of ACM Conference on Electronic Commerce (EC-00)*, Minneapolis, MN, October 2000.

[4] M. Fan, J. Stallaert, and A. B. Whinston. The internet and the future of financial markets. *Communications of the ACM*, 43(11):82–88, November 2000.

[5] S. Focardi and C. Jonas. *Modeling the Market: New Theories and Techniques.* Frank J. Fabozzi Associates, 1997.

[6] J. M. Harrison and S. R. Pliska. Martingales and stochastic integrals in the theory of continuous time trading. *Stochastic Processes and their Applications*, 11:215–260, 1981.

[7] T. M. Őzsu and P. Valduriez. *Principles of Distributed Database Systems.* Prentice Hall, 1999.

[8] S. Joshi, J. Parker, and M. A. Bedau. Technical trading creates prisoner's dilemma: Results from an agent-based model. In Y. S. Abu-Mostafa, B. LeBaron, A. W. Lo, and A. S. Weigend, editors, *Computational Finance*, Cambidge, MA, 1999. The MIT Press.

[9] J. O. Kephart, J. E. Hansen, and A. R. Greenwald. Dynamic pricing by software agents. *Computer Networks*, 32(6):731–752, 2000.

[10] J. O. Kephart, J. E. Hansen, D. W. Levine, B. Grosof, J. Sairamesh, R. Segal, and S. R. White. Emergent behavior in information economies. In *Proceedings of the International Conference on Multi-Agent Systems*, 1998.

[11] J. O. Kephart, J. E. Hanson, D. W. Levine, B. N. Grosof, J. Sairamesh, R. B. Segaland, and S. R. White. Dynamics of an information-filtering economy. In *Proceedings of the Second International Workshop on Cooperative Information Agents*, July 1998.

[12] B. LeBaron. Agent-based computational finanace: Suggested readings and early research. *Journal of Economic Dynamics and Control*, 24:679–702, 2000.

[13] B. LeBaron. A builder's guide to agent-based financial markets. *Quantitative Finance*, 1:254–261, 2001.

[14] B. LeBaron. Evolution and time horizons in an agent based stock market. *Macroeconomic Dynamics*, 5:254–261, 2001.

[15] B. LeBaron, W. B. Arthur, and R. G. Palmer. The time series properties of an artificial stock market. *Journal of Economic Dynamics and Control*, 23:1487–1516, 1999.

[16] P. Maes, R. Guttman, and A. Moukas. Agents that buy and sell. *Communications of tha ACM*, 42(3):81–91, 1999.

[17] R. G. Palmer, W. B. Arthur, J. H. Holland, and B. LeBaron. Artificial economic life: a simple model of a stock market. *Physica D*, 75:264–274, 1994.

[18] F. Partnoy. *FIASCO The Inside Story of a Wall Street Trader*. Penguin Books, New York, 1997.

[19] A. Picot, C. Bortenlänger, and H. Röhrl. The automation of capital markets. *Journal of Computer-Mediated Commerce*, 1(3), December 1995. Available online at http://www.ascusc.org/jcmc/vol1/issue3/picot.html.

[20] S. Russell and P. Norvig. *Artificial Intelligence*. Prentice Hall, New Jersey, 1995.

[21] J. Rust. Dealing with the complexity of economic calculations, 1996. Workshop on Foundamental Limits to Knowledge in Economics.

[22] T. Sandholm. eMediator: A next generation electronic commerce server. In *AAAI Workshop Technical Report WS-99-01*, pages 46–55, Orlando, Fl, 1999. AAAI Workshop on AI in Electronic Commerce.

[23] O. Streltchenko, N. C. Narendra, and Y. Yesha. A reference architecture for multi-agent simulation of derivatives markets. In *Proceedings of the International ICSC Congress on Computational Intelligence: Methods and Applications*, Bangor, Wales, 2001.

[24] O. Streltchenko, N. C. Narendra, and Y. Yesha. A multi-agent environment for exploring price formation in financial markets, 2002. Submitted to the *International Journal of Intelligent Systems in Accounting, Finance and Management.*

[25] G. J. Tesauro and J. O. Kephar. Foresight-based pricing algorithms in an economy of software agents. In *Proceedings of International Conference on Information and Computation Economies*, October 1998.

[26] L. Tesfatsion. Notes on the Santa Fe Artificial Stock Market model. Available online at http://www.econ.iastate.edu/classes/econ308x/tesfatsion/sfistock.htm, 2002.

[27] H. Varian. Effect of the internet on financial markets, 1998. Available online at http://www.sims.berkeley.edu/∼hal/Papers/brookings-paper.html.

[28] M. Wooldridge, N. Jennings, and D. Kinny. The gaia methodology for agent-oriented analysis and design. *Journal of Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, 2000.

[29] F. Zambonelly, N. Jennings, and M. Wooldridge. Organizational abstractions for the analysis and design of multi-agent systems. In *Proceedings of the Firstst International Workshop on Agent-Oriented Software Engineering*, pages 127–141, Limerick, Ireland, 2000.

[30] F. G. Zarb. The coming global digital stock market, speech at the National Press Club, 1999. Available online at http://www.nasdaqnews.com/views/speech/digmarkets.htm.