

A Distributed Online Financial Market: Design, Implementation, and Experiments

by

Jonathan Paul Loflin

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the
Degree of Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 25, 1999

June 1999

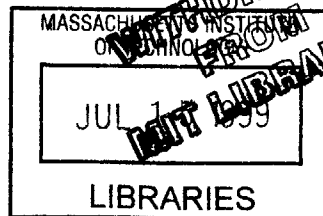
© Copyright 1999 Jonathan Paul Loflin. All rights reserved.

The author hereby grants to M.I.T. permission to reproduce and
distribute publicly paper and electronic copies of this thesis
and to grant others the right to do so.

Author.....
Department of Electrical Engineering and Computer Science
May 25, 1999

Certified by.....
Andrew W. Lo
Harris & Harris Group Professor
Director, Laboratory for Financial Engineering
Thesis Supervisor

Accepted by.....
Arthur C. Smith
Chairman, Department Committee on Graduate Students



ENG

A Distributed Online Financial Market: Design, Implementation, and Experiments

by

Jonathan Paul Loflin

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the
Degree of Master of Engineering in Electrical Engineering and Computer Science

May 25, 1999

Abstract

The MIT Artificial Financial Markets Exchange is a robust Java-based simulator for running flexible and configurable trading sessions with human and “robot” traders via the World Wide Web. The system is designed as a platform both for experimental research on financial markets and for training students of finance in a classroom setting. The AFM exchange is fully automated, featuring an algorithmic market maker to facilitate trading.

This thesis first presents a financial description of the market structure and the strategies used by the market maker. Second, it presents an overview of the design and implementation of the technical infrastructure. Third, it describes two experiments – an intense 20-minute trading session and an extended 5-day asset management simulation – designed to verify the system and the market maker. Finally, the results of these experiments are presented.

Thesis Supervisor: Andrew W. Lo

Title: Harris & Harris Group Professor, Massachusetts Institute of Technology
Director, Laboratory for Financial Engineering

Acknowledgements

I would first like to thank Nicholas Chan for guiding me through my graduate research at the Institute and for his help with the preparation of this document.

I am greatly indebted to the contributions of the Artificial Financial Markets Project team. Dan Hermann's technical expertise was invaluable to the development of the market. John Wang selflessly put his own Master's thesis on hold to help with the experiments. Robert S. Thau and Benny Suen contributed extensively before I joined the project.

I sincerely appreciate the guidance of my advisors, Professor Andrew Lo and Professor Tomaso Poggio. I also appreciate the patience of the students of Professor Lo's financial engineering course who eagerly participated in our trading experiments.

Finally, I thank Mom and Dad, who sent me to M.I.T. They have always supported me and offered me every opportunity to find out what I want to do in life.

Table of Contents

Chapter 1: Introduction	13
1.1 Motivation	14
1.2 Thesis Overview	15
1.3 Summary of Results	16
1.4 Related Work	17
Chapter 2: Economic Structure of the Market	23
2.1 Financial Background on Efficiency and Structure of Real Markets	23
2.1.1 Definition of Market Efficiency	24
2.1.2 Trading Mechanics	26
2.1.2.1 Call Auctions	26
2.1.2.2 Continuous Auctions	27
2.1.2.3 Continuous Agency Auctions	27
2.1.2.4 Multiple Dealer Markets	28
2.1.3 Components of Financial Markets	29
2.1.3.1 Market Makers	29
2.1.3.2 Exchange	30
2.1.3.3 Brokers	30
2.1.3.4 Clearinghouse	31
2.2 Financial Structure of the MIT Artificial Financial Markets Exchange	31
2.2.1 Types of Securities	31
2.2.2 Types of Orders	32
2.2.3 Order Routing, Order Execution, and Clearinghouse Mechanics	32
2.2.4 Surveillance and Regulation	33
2.2.5 Dissemination of Information	33
2.2.6 Trading Hours	34
2.2.7 Automated Trading Agents for Dynamic Hedging	34
2.3 Price Discovery Heuristics of the Electronic Market Maker	34
2.3.1 Goals of the Electronic Market Maker	35
2.3.2 Cost Functions and Parameters of the EMM	36
2.4 Conclusion	39
Chapter 3: Technical Architecture of the Exchange	41
3.1 Overview	41
3.2 Technical Design Goals	43
3.2.1 Dynamic Marketplace	43
3.2.2 Novice Users	43
3.2.3 Platform Independence	44
3.2.4 Simulated Markets	45
3.2.5 Simplicity and Compactness	45
3.2.6 Fault Recovery	45

3.3 Description of the Primary System Components	46
3.3.1 Trading Client	46
3.3.2 Market Server	50
3.3.3 Automated “Robot” Traders	53
3.3.3.1 Random Price Process Traders	54
3.4 Structure of the Database	57
3.5 Configuring the System for a Simulation	59
3.5.1 Database Version versus In-Memory Version	59
3.5.2 Setting up the Traded Securities	60
3.5.3 Setting up the “News Flashes”	61
3.5.4 Setting up the Users and Portfolios	62
3.5.5 The Salelog and Orders Tables	63
3.6 Application Programming Interfaces and Internal Mechanics	65
3.6.1 Trading Client API	66
3.6.2 Marketmaker API	67
3.6.3 Market Server API	68
3.7 Performance, Robustness, Scalability	69
3.8 Future Directions of the Technical Architecture	70
3.9 Conclusion	73
Chapter 4: Design and Results of the Simulated Trading Sessions	75
4.1 Charles River Logging (CRL) Simulation	76
4.1.1 Specification of the CRL Case	77
4.1.2 Recommended Strategies for the CRL Case	79
4.1.3 Experimental Results of the CRL Case	82
4.1.3.1 Convergence to the Efficient Price	82
4.1.3.2 Large Market Orders	85
4.1.3.3 Trading Laboratory Psychology and Panic Trading	86
4.1.3.4 Call Auction as an Alternative to Continuous Agency Auction	87
4.1.4 Conclusion to the CRL Case	87
4.2 Five Day Asset Management Simulation	88
4.2.1 Specifications of the Asset Management Case	89
4.2.2 Recommended Strategies for the Asset Management Case	91
4.2.3 Results of the Asset Management Simulation	94
4.2.3.1 Overall Price Parity between the AFM and NY Prices	94
4.2.3.2 Low Trading Volume	100
4.2.3.3 Effects of the Market Maker	100
4.2.4 Conclusion to the Asset Management Case	101
4.3 Conclusion	102
Chapter 5: Conclusions and Directions for Further Research	105
References	109

List of Figures

Figure 1: Electronic Market Maker Graphical Interface	38
Figure 2: Component Architecture of the AFM Exchange	42
Figure 3: Trader Applet Interface: Browser Window	47
Figure 4: Trader Applet Interface: Portfolio and Orders Window	48
Figure 5: Trader Applet Interface: Dynamic Graph Window	49
Figure 6: Trader Applet Interface: Historical Graph Window	49
Figure 7: Market Server Output	51
Figure 8: Market Server Graphical Interface	52
Figure 9: Electronic Market Maker Graphical Interface (Annotated)	53
Figure 10: Randomized "Robot" Trader Configuration File	56
Figure 11: "Robot" Trader Output	57
Figure 12: Five Database Tables	58
Figure 13: Database Table: Secinfo	60
Figure 14: Database Table: Man_Msg (News Flashes)	61
Figure 15: Database Table: Users	63
Figure 16: Database Table: Salelog	64
Figure 17: Database Table: Orders	65
Figure 18: Price and Volume: Charles River Logging Simulation	84
Figure 19: Price and Volume: Intel, Asset Management Simulation	97
Figure 20: Price and Volume: Merck, Asset Management Simulation	98
Figure 21: Price and Volume: McDonald's, Asset Management Simulation	99

List of Tables

Table 1: Trader Applet API (ProtocolClient.java)	67
Table 2: Primary Market Maker Functions API (MarketMaker.java)	68
Table 3: Primary Market Server Functions API (BackOffice.java)	68
Table 4: Charles River Logging Payoffs for Period One	77
Table 5: Charles River Logging Payoffs for Period One and Two	78
Table 6: Conditional Expected Values of CRL Stock	80
Table 7: Defeating Sharpe's Method for Risk-Adjusting Returns	92
Table 8: Price Differential between AFM Market and NY Markets	95

Chapter 1

Introduction

The total amount of financial assets managed online is projected to grow to \$474 billion by the year 2000 from \$11 billion in 1996, according to Forrester Research. The growth of online financial markets presents new opportunities to improve the efficiency of traditional financial institutions and to solve difficult resource allocation and information aggregation problems through novel applications of market structures.

The Artificial Financial Markets Project, a joint initiative of the MIT Laboratory for Financial Engineering (LFE) and the MIT Center for Biological and Computational Learning (CBCL), aims to study various market structures, examine the feasibility of an fully automated exchange, provide a platform for experimental markets, and study the interactions between human and automated agents.

The primary project effort of this thesis has been partially developing the market itself, developing the market experiments, and overseeing the execution of the experiments. These simulations verified the technical architecture and implementation and also provided insight into information aggregation in asset markets and the price parity between identical securities in disjoint markets.

1.1 Motivation

This thesis concentrates on efforts to implement a fully automated online securities exchange to serve as a platform for obtaining experimental results. Specifically, the purposes for implementing this exchange are to:

- Build a flexible online exchange that can be configured to a wide variety of market structures and security classes.
- Study market quality under various trading structures and market conditions.
- Provide an environment to assess the feasibility of automated strategies for market making.
- Test an electronic market maker that has been developed simultaneously as part of the MIT AFM project.
- Experiment with market psychology and assessing its effects on market quality.
- Assess the merit of various technical architectures for online exchanges.
- Provide a “playpen” for developing autonomous trading agents to apply learning algorithms in trading strategies.
- Assess methods of solving “hard” resource allocation problems by using markets in novel contexts.
- Educate MIT Sloan business school students using a case-based method with a real-time, simulated trading environment.

1.2 Thesis Overview

This thesis has three major sections. The first section offers background on the economic structure of securities markets, the details of the market implemented by the AFM team, and the methodology used by the automated market maker used in our experiments.

The second section presents the technical details of the exchange architecture. The primary goals of the technical design are presented. The major components of the technical architecture – the applet for human traders, the central server, the electronic market maker (EMM), and the automated “robot” traders – are all discussed. The user interfaces are presented to provide a sense of the “look and feel” of the system, and some implementation decisions are also discuss.

The third section discusses two trading experiments that used the exchange in a classroom environment. These experiments are designed to verify the robustness of the exchange, collect initial data on the performance of the electronic market maker (EMM), and provide simulated trading sessions for students of finance. The details of the cases, as presented to the students, are presented. Potential strategies for the cases are also discussed, and the resulting data from the simulations is presented. The final section offers insights interpreted from the data and from discussions with the students.

1.3 Summary of Results

The essential results are the completion of a robust Java-based system capable of running flexible and configurable trading simulations conveniently via the World Wide Web, the deployment and testing of an electronic market maker, the ability to test interactions between human traders and automated “agent” traders, and finally the development of two trading “cases” and the collection of data resulting from the experiments using these cases. The results from the technical development of the market simulator are discussed in Chapter 4.

The subject of the first trading simulation, which involved about 20 minutes of intense trading by 11 teams, is the assimilation of diverse imperfect information into the price of a stock, or the *price discovery*. A fictional company with a 2-year business model will announce earnings at the end of each year. During the year, the company releases hints about the future earnings announcement. In a given year, there are 3 possible earnings figures, X, Y, and Z. If the earning would be Y, half the traders will learn “Not X” and the other half will learn “Not Z.” By analyzing the price patterns and leveraging private knowledge, both groups should ultimately agree on an efficient single price. We conducted this experiment in the MIT Sloan Trading Laboratory, finding that the price convergence was obscured by excessive speculation on price swings, and interestingly that the presence of a market maker may have encouraged these price swings and therefore adversely affected market quality according to one hypothesis.

In the second simulation, which involved about 100 hours of trading over 5 days, the students were asked to trade real stocks in a disconnected market. The prices of the stocks were initially calibrated to the real-market prices on Monday, and the prices were also marked to the real-market prices at the Friday close. During the 5 days in between, the prices were determined by supply and demand in the simulated market. Traders were assigned non-identical objective functions to simulate hedge funds, mutual funds, and index funds. The heterogeneous performance metrics caused the risk-return preferences of traders in the simulated market to deviate from those risk-return preferences of real-market investors. This caused a temporary deviation in prices that gradually converged to zero as the simulation. In this simulation, characterized by low trading volumes, the market maker increased market quality. This simulation also demonstrates a genre of experiments that can be conducted in the future regarding the effects of risk-return preferences on market prices.

1.4 Related Work

This section aims to give a sampling of the types of exchanges that are on the Internet and also an overview of the electronic trading systems deployed by the NYSE, NASDAQ, CME, and the regional exchanges.

There are many markets of various types on the Internet. The AFM exchange is similar to all of these Internet markets in some ways. The AFM project, however, emphasizes researching automated market making strategies

and market structure flexibility as primary goals. This emphasis differentiates the AFM project from each of these exchanges.

1. *Iowa Electronic Markets* – A pioneer of online markets, the IEM trades real-money futures based on economic and political events. For example, a futures contract may pay a certain sum if the Republicans win the presidential election and a different sum if the Democrats win. This mechanism for predicting elections has been quite accurate in the past. Another advantage is that a continuous assessment of the presidential race can be obtained at relatively low cost. Polls can be taken only at discrete intervals (such as once a week) and are expensive to execute. Forsythe, Nelson, Neumann, and Wright (1991) further explain the presidential election security market.
2. *Hollywood Stock Exchange* – One of the most interesting markets on the Internet is the Hollywood Stock Exchange. The exchange trades with fictional cash and equities. The equities pay a dividend that is equal to the 4-week gross ticket sales of a particular movie. Therefore, traders are predicting the box-office draw of movies that are in theaters or will be released in the next few years. The exchange also trades “Star Bonds” that are tied to the box-office draw of actors and actresses. The HSX uses a market mechanism creatively to aggregate public opinion. This approach to market research may hold promise for further applications. The HSX can be accessed at <http://www.hsx.com>
3. *World Sports Exchange* – The world sports exchange is an application of futures markets to offshore gambling. A sample contract might pay \$100 if the Atlanta Braves win the world series, \$50 if they lose the world series, \$25

if they lose the National League Championship Series, and \$0 in all other cases. A contract such as this one would trade for the full duration of a baseball season. Other contracts trade actively during gametime. The contracts on this exchange generally carry prohibitively large spreads. The exchange can be accessed at <http://www.wsex.com>.

4. *CNNfn* – The CNN Financial Network has developed an online trading simulation that allows traders to test their trading strategies with fictional cash. The prices in this market are tied to the prices in the real U.S. securities markets.
5. *Amazon.com and eBay* – These two well-known eCommerce sites have both developed auctions on the web, primarily for collectibles and tangible goods such as sports tickets. The relatively simple auction system could perhaps be enhanced using ideas borrowed from securities markets. For example, it is not uncommon to see two identical products with different best bid prices.
6. *Priceline.com* – This eCommerce site allows consumers to submit “blind” bids for goods such as airline tickets and hotel rooms. For example, a traveler could submit a binding bid of \$120 for a plane ticket from the Boston area to the Chicago area. Priceline will send an email to the traveler within minutes if any airline is willing to sell at that price. The bids are “blind” because bidders have no information on past transaction prices (other than rates that a travel agent would quote). In this system, bidders are not allowed to bid twice for a single product or service.

There are also a number of automated electronic trading systems in the U.S. securities markets:

1. *NYSE* – SuperDOT, consisting of the Designated Order Turnaround (DOT) system and other peripheral support systems, transmits orders to market makers' books directly from remote traders. The ITS system links market makers in NY to market makers in the regional and OTC exchanges, transmitting quote information and transaction messages.
2. *NASDAQ* – SOES (Small Order Execution System), SelectNet, and ACES (Advanced Computerized Execution Systems) route small orders directly to dealers.
3. *Arizona Stock Exchange* – The AZX provides an electronic call auction for listed and OTC stocks. The call auction executes every day at 5:00. The exchange also holds three absolute price auctions during the day. The exchange traded 487,000 shares per day in 1994. Software for executing trades is available online. More information on AZX is available at <http://www.azx.com>.
4. *Chicago Mercantile Exchange* – The CME uses the Globex system for after-hours electronic trading of futures and options. The system accepts only limit orders, and executes trades according to a continuous auction model. The software displays the limit order book for all users, and trades are executed according to a set of priority rules.
5. *Optimark* – Optimark uses a complex algorithm to match buyers and sellers. Traders enter sophisticated profiles of their preferences. For example, a trader may be willing to sell 100,000 shares of IBM at \$101 and 200,000 shares of IBM at \$100. The Optimark system will aggregate the orders, and execute at the price that maximized transacted volume. The Pacific Stock Exchange began

using the Optimark system in Fall, 1998. Plans have been made to use the system on the CBOE (Chicago Board Options Exchange) and the NASDAQ.

In the area of academic trading simulators, Carnegie Mellon University has developed the Financial Trading System (FTS). This system enables trading laboratory simulations similar to the ones allowed by the AFM system. However, the FTS system does not run in a browser, and is primarily useful for simulations lasting a few hours or less.¹

Other trading simulators exist in the financial industry for training purposes. For example, Chase Manhattan Bank uses a trading simulator as an integral part of the interview process for Sales and Trading candidates. This simple simulator assesses the performance of the candidates making a market in a fictional energy commodity.

¹ The first simulation, Charles River Logging, is adapted partially from the REI case developed at Carnegie Mellon as part of FTS. The FTS website is at <http://www.ftsweb.com>.

Chapter 2

Economic Structure of the Market

The design of a financial market should aim to maximize *market efficiency*.

This chapter begins by defining market efficiency and various types of trading structures such as call auctions, continuous auctions, agency auctions, and multiple dealers markets. The role of the market participants is also examined. The financial structure of our online exchange, and its configurations, is then discussed. Finally, the design of the electronic market maker (EMM) and the price discovery heuristics used by the electronic market maker are examined.

The goal of this chapter is to provide background on the financial structure of securities markets, on the structure of the automated online marketplace that is the subject of this thesis, and on the electronic market maker that plays a crucial role in that marketplace.

2.1 Financial Background on Efficiency and Structure of Real Markets

This section examines market efficiency, common trading mechanisms used by markets (such as call auctions, continuous auctions, agency auctions, and multiple dealer markets), and the roles of market components (such as brokers, market makers, the clearinghouse, and the exchange itself). Schwartz

(1991) and Amihud, Ho, and Schwartz (1985) discuss market structure and the roles of market participants.

2.1.1 Definition of Market Efficiency

The hypothesis that markets are efficient (in its most common, or *semi-strong*, form) states that prices reflect all public information in an accurate and unbiased fashion. However, the efficiency of financial markets also implies characteristics beyond a close adherence to the “true price.” Efficient markets are characterized in terms of market depth, price continuity, immediacy, fairness, and low cost.

Market depth – A deep market is one in which sufficient shares are available for purchase at the current market bid and ask. For example, if there were 100 shares available in one market at the current bid and ask, and 400 shares available in another market at the same bid and ask, the second market would be said to have greater depth. Greater market depth implies that a particular transaction will have less market impact. Market impact is the price movement against a trade. For example, submitting a large buy order for 50,000 shares of a stock may cause the price of the stock to rise during execution.

Price continuity – Price continuity (or price stabilization) implies that prices tend not to jump discontinuously during trading. During time when prices would otherwise jump, the market maker is often obligated to provide a continuous

market by “buying on the way down” or “selling on the way up,” sometimes suffering trading losses by doing so.

Immediacy – In an efficient market, traders should be able to execute their trades as immediately as possible. Ideally, a seller of Warner-Lambert stock should not need to wait until an investor interested in buying Warner-Lambert appears.

True prices – Of course, prices in an efficient market should reflect the intrinsic present value of associated cash flows. Stock “bubbles,” panic selling, and other market pathologies are often cited as causing deviations from the “true” price of a security. A particular market structure may encourage or discourage these deviations.

Fairness – In a fair and egalitarian market, no investors should be intrinsically favored. Electronic exchanges may ultimately provide more equal access to financial markets to parties that have traditionally been at a disadvantage.

Low Cost – An efficient market should incur low costs. The total cost of a trade includes the commission charged (if any), the cost due to the bid-ask spread, and the cost of market impact. For an average investor, the cost of the commission dramatically outweighs any cost associated with market impact. For an institutional investor trading large blocks, the cost of the market impact can be quite significant.

2.1.2 Trading Mechanics

The mechanics of trading can have significant impact on overall market efficiency. One of the fundamental goals of the Artificial Markets Project is to research various market structures and they relate to efficiency. This section will present background on auction markets, agency auction markets, and multiple dealer markets.

2.1.2.1 Call Auctions

A call auction is an elegant mechanism for aggregating information by delaying execution of anonymous orders for a specified amount of time, then clearing all orders simultaneously at a single price. Traders submit binding orders, a bid price or an offer price paired with a quantity. The auctioneer simply accumulates these orders until the time of execution, and then simultaneously executes every order at a single price – the price that equates the shares bought and the shares sold. Traders often may have their orders cleared at a more favorable price than they submitted. Usually, a single price that clears all orders does not exist. In this case, the auctioneer may reveal information about an excess supply or demand and allow traders to adjust their orders, or the auctioneer may execute as many orders as possible at a price that maximizes transacted volume. In a call auction, there is an obvious sacrifice of immediate execution, but they may yield markets that are highly efficient by other measures. The opening prices on the New York Stock Exchange (NYSE) are determined by a call auction.

2.1.2.2 Continuous Auctions

In a continuous auction, a book of unexecuted limit buy orders (a price at which to buy and a quantity) and limit offer orders (a price at which to sell and a quantity) are maintained in the (passive) auctioneer's limit book. Execution occurs when two orders cross. For example, an offer to sell 100 shares at \$20 may exist in the limit order book. If another trader were to submit an order to buy 50 shares at \$21 (or lower), a transfer of 50 shares will occur at a price of \$20 (because the execution price is determined by the standing order). In this case, an order to buy 50 additional shares at \$21 will remain in the book.

Continuous auctions also accept market orders. A market order is executed immediately against limit orders until all shares of the market order have been filled, or until there are no more opposing limit orders on the book (in which case the order for the residual shares is cancelled).

In a continuous action market, traders have access to information regarding the best bid and best offer in the limit book.

2.1.2.3 Continuous Agency Auctions

In this market structure, the auctioneer who maintains the limit order book is not passive. That is, the auctioneer (or "specialist" or "market maker") actively trades. Furthermore, the specialist has an obligation to make markets

more efficient by supplying immediacy – the specialist is willing to buy and sell at any time by maintaining a bid and offer price. The market maker provides *price discovery* by monitoring the order flow, leveraging that information to set the bid and ask and therefore to determine the price at which the security will trade. The market maker also adds price continuity to the market by actively trading to dampen large price swings.

Of course, these market makers are sometimes forced to take unwanted inventory. Market makers generally make a small profit on each transaction on average due to the difference between the offer price and the bid price. A good market maker will generally earn a profit over time while simultaneously increasing the quality of the market.

The New York Stock Exchange (NYSE) and American Stock Exchange (AMEX) use the continuous agency auction model. Hasbrouck, Sofianos, and Sosebee (1993) provide a reference for trading procedures and systems on the NYSE. The market makers are *monopolistic* in that there is a single auctioneer in charge of any particular stock. The exchanges impose restrictions and requirements on the specialists to which other traders need not adhere.

2.1.2.4 Multiple Dealer Markets

Multiple dealer markets such as the NASDAQ allow multiple market makers to trade a particular stock. There is no central limit order book. Rather, each dealer maintains a separate book. The NASDAQ system provides a

consolidated quotation mechanism that disseminates the best bid and best offer of all the many dealers.

In multiple dealer markets, the market makers are not as tightly monitored as in monopolistic dealer markets. Another difference is the existence of a physical trading floor. In the NYSE, traders communicate face-to-face. This arguably is a richer medium for traders to interact, and may contribute positively (or negatively) to market efficiency. The NASDAQ does not feature a physical trading floor. Huang and Stoll (1996) and Neal (1992) compare transaction costs between auction markets and competitive dealer markets.

2.1.3 Components of Financial Markets

This section outlines the responsibilities of market makers, brokers, the exchange, and the clearinghouse.

2.1.3.1 Market Makers

The role of a market maker is dependent on the mechanics of the exchange. There are clearly differences between the role of a specialist on the NYSE and the role of a dealer on the NASDAQ. Regardless, the incentives of all market makers are structured such that the market makers promote market efficiency, price discovery, and information dissemination. Market efficiency implies price continuity, market depth, fairness, true prices, immediacy, and low cost. Price discovery is the process through which the marketmaker determines

the optimal trading price based on the observed order flow. The market maker's role in information dissemination is to make available a tradable quote at all times.

Market Makers are supposed to make money. While a human market maker may lose a lot of money on some days, one goal is to earn a living over the long term.

2.1.3.2 Exchange

The exchange itself performs the function of order routing, order execution, relaying quotes and other information, and enforcement of some regulations.

2.1.3.3 Brokers

Brokers provide market access to investors that are not on the exchange itself. Brokers essentially trade on behalf of the "end consumer," and maintain accounts and portfolios for their customers. Beyond these core functions, brokers may also sell research and advice to their customers. Brokers also enforce margin requirements on their customers.

2.1.3.4 Clearinghouse

The clearinghouse mechanism is responsible for transferring cash and securities between traders after the trade has taken place on the exchange.

2.2 Financial Structure of the MIT Artificial Financial Markets Exchange

This section provides financial background specific to the MIT Artificial Financial Markets Exchange. The MIT AFM Exchange combines the traditional role of the exchange, broker, clearinghouse, and market maker into a single entity. Traders log into the exchange directly using a password stored by the exchange. They may then submit market and limit orders, which may be executed by the electronic market maker, and then processed by the clearinghouse mechanism. The exchange provides some research (historical price and volume graphs, as well as “news flashes”) to the traders as a broker might, and the exchange also maintains the portfolio of each trader.

A graphical description (and technical discussion) of the market modules is presented in the chapter on the Technical Architecture of the Market (Chapter 3).

2.2.1 Types of Securities

The market has primarily been tested using standard equities (without dividends). However, it is feasible to incorporate securities that pay predefined

cash flows (such as bonds) or contingent cash flows (such as futures, options, or more complex derivative securities). Security prices should be greater than or equal to zero, and securities trade in integral share quantities. Tick sizes, the smallest price increment in which a stock is traded, must be defined in advance of a trading session.

2.2.2 Types of Orders

The market currently accepts market and limit orders only. Unfilled or partially filled limit orders can be cancelled. The market server architecture would allow for more complex orders to be submitted. For example, bundling orders for simultaneous execution may be useful in some circumstances. The limit and market order can be leveraged by automated trading agents to bundle orders or perform more complex strategies.

2.2.3 Order Routing, Order Execution, and Clearinghouse Mechanics

Order routing is a major logistical issue for large exchanges. In the AFM exchange, orders are routed to an electronic agent that is responsible for the particular security. This agent may be programmed to perform virtually any trading mechanics desired (call auction, continuous auction, multiple dealer market). Currently, our market is set up to perform either a continuous auction market or a continuous agency auction market.

The order routing agent could easily be programmed to pass along the orders to multiple dealers to create a NASDAQ simulation. Furthermore, novel market structures can be invented and simulated using human and “robot” traders.

Order execution is performed using the same interface as is used for order routing. The flexibility of the agent-based architecture provides a clean and efficient framework for experimenting with market structures.

2.2.4 Surveillance and Regulation

The system allows for enforcement of margin requirements. Other trading restrictions and surveillance can also be programmed into the system fairly painlessly as needed.

2.2.5 Dissemination of Information

The trader interface allows traders to request real-time quote and sale updates. The trader can have this information *pushed* from the server periodically (for example, every two seconds) or as needed (instantaneously whenever a trade occurs or a quote changes). The traders can also request price and volume graphs with data points representing 5-minute intervals or 24-hour intervals.

The server also contains a mechanism for pushing news flashes to the traders.

2.2.6 Trading Hours

The market is capable of running up to 24 hours a day for extended periods of time. During our simulations, we specified that the market would be closed for 1 hour each day in order to provide an opportunity for unforeseen maintenance. No maintenance was required during our test simulations.

The market can be set up to automatically open and close at specified times.

2.2.7 Automated Trading Agents for Dynamic Hedging

A technical audience with a modest knowledge of Java could build an automated trading agent that could submit trades implementing a dynamic hedging strategy. A simple strategy of this type might aim to replicate cash flows for a call option using a delta-hedging approach outlined by the Black-Scholes framework. An interesting case simulation might ask students to replicate a more complex derivative as a team project for a term.

2.3 Price Discovery Heuristics of the Electronic Market Maker

The Electronic Market Maker (EMM) is a module incorporated into the MIT AFM Exchange. The EMM aims to provide a fair, orderly, and efficient market, and also to make a reasonable risk-adjusted profit. As previously

discussed in this chapter, the incentives of market makers are structured such that the market makers promote market efficiency, price discovery, and information dissemination. The EMM is modeled after the specialists on the NYSE. The *NYSE Official Floor Manual* cites “the maintenance of a fair and orderly market in the stocks in which he is registered, which implies the maintenance of price continuity with reasonable depth and minimizing of temporary disparities between demand and supply...” as an affirmative responsibility of a market maker.

2.3.1 Goals of the Electronic Market Maker

The EMM releases a quote – a bid price, an ask price, and quantities for both at which the market maker is willing to transact – every time an order is submitted. The quote stands until another order comes into the market. The market maker has no access to information other than the order flow. That is, the market maker does not interpret news events or earnings announcements, or perform fundamental or technical securities research. The market maker simply views the order flow and releases a quote.

The EMM simultaneously pursues three goals:

- *Maximizing Market Quality* – Promoting market efficiency, price discovery, and information dissemination
- *Minimizing Risk Exposure* – Carrying inventory exposes the market maker to risk, particularly due to the nature that traders will push inventory onto the market maker at the worst time (due to adverse selection).

- *Speculation* – The market maker can also make money by carrying inventory (long or short). Because the market maker has access to the order flow, a market maker is in a unique position to infer possible future price movements. Hence, the market maker may intentionally carry modest inventory at times. (Commonly, exchanges regulate the speculation of maker makers).

2.3.2 Cost Functions and Parameters of the EMM

The preliminary version of the EMM uses five cost functions and eight parameters to determine a quote.

The cost functions are functions that produce a “score” for a potential quote. The market maker *minimizes* cost. The cost functions are not generally linear. The cost functions are the following:

- *Market Depth* – The EMM cost function is lower when the current best bid and best ask are backed by more shares.
- *Price Continuity* – The EMM attempts to keep discontinuous price jumps to a minimum.
- *Spread* – The EMM is programmed to minimize the spread. This function assigns a score to a potential spread.
- *Inventory* – Hitting the inventory target is important to the EMM. The EMM will bias the quote in order to make it likely that traders will add or subtract from the EMM’s inventory.

- *Imbalance* – The EMM will adjust the quote in order to account for imbalances in the limit order book.

The EMM used in our simulations also uses eight parameters:

- *Inventory Target* – Target inventories will depart from zero when a market maker desires to carry inventory (long or short). Currently, the EMM target inventory must be set manually until a speculation engine is developed.
- *Minimum and Maximum Spreads* – The EMM will consider spreads within this range.
- *Cost Function Weights* – The final “cost” of a potential quote is the sum of the five cost functions weighted (multiplied) by the cost function weights.

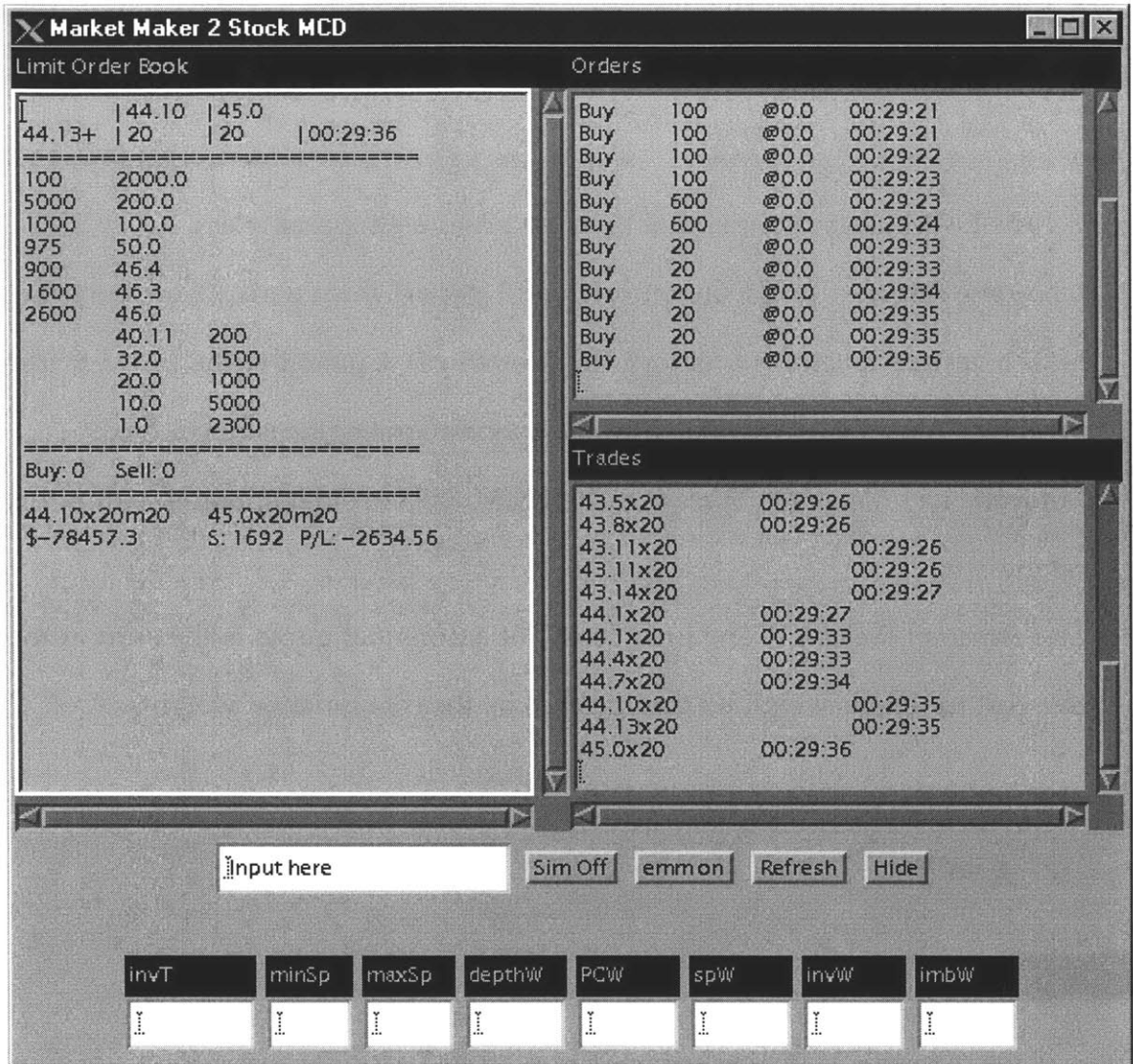


Figure 1: The Electronic Market Maker Interface

The EMM will issue the quote with the lowest total weighted cost within the upper and lower bounds imposed upon the spread. The EMMs can be monitored in real-time during trading, and the parameters can also be set during trading. The interface shows the limit order book, recent orders, recent transactions, and other relevant information. (See figure 1).

2.4 Conclusion

The flexibility of the financial structure of the AFM system is a primary advantage. The ability to build plug-and-play modules that implement various market structures enables future research to study various trading mechanics in an environment with human traders. Market modules could be switched in real-time during trading if desired.

The core functionality of the EMM needs to be further developed. The speculation module (that would dynamically adjust the inventory target) does not exist at this time. Setting inventory targets is an important means for market makers to perform well without losing money. Amihud and Mendelson (1980) and O'Hara and Oldfield (1986) discuss inventory issues in market making.

The EMM should be better able to defend against traders attempting to "game" the market maker. Perhaps a randomized strategy or a search-based algorithm (used to program adversarial intelligence for games such as chess) could improve performance in such circumstances.

Chapter 3

Technical Architecture of the Exchange

In order to develop and study trading strategies and theories of market efficiency, and in order to gain insight into the design of online exchanges, we implemented a fully automated electronic marketplace.

This chapter begins with a high-level overview of the technical structure of our online exchange. Design tradeoffs are discussed and details of the major subsystems are exposed. The internal structure and the application programming interfaces (APIs) that define the subsystems' intercommunication are also described. This chapter concludes with a summary of possible future directions for the system.

3.1 Overview

The online exchange is a software system that enables human agents, automated "robot" agents, or both to trade securities with each other. The system consists of several systems: the central market server, the trading client, the administrator interface, the marketmakers, and a database.

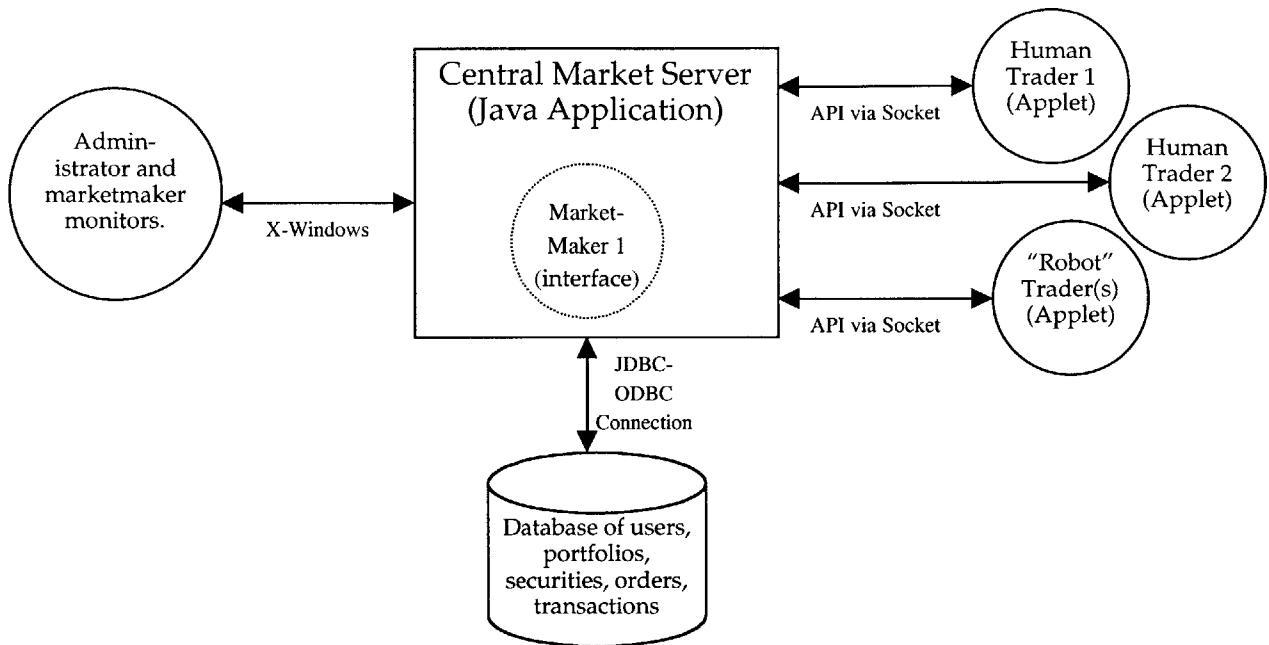


Figure 2: Component Architecture of the AFM Exchange. The subsystems of the electronic exchange include the Market Server (which encompasses marketmakers and the administrator functions), the client for human traders, the “robot” trader systems, and the database for storing the orders, sales, security specifications, and the user registry and portfolios.

Each of the components is written in Java, and therefore the system is portable. The central server is a Java application. The client applet (for human traders) is a Java applet and can therefore be conveniently run using almost any Netscape or Microsoft browser by accessing a URL. The client applet will then automatically load and begin the login process. The database can be any database that supports standard SQL. We used Microsoft’s SQL Server. The marketmaker resides within the server, and can be monitored and, if desired, manually controlled through the administrator graphical interface. The administrator interface also controls the opening and closing of the market, and also the sending of text messages (e.g. news headlines) to the clients. We have

also implemented a set of “robot” traders that generate trades based on random processes and/or artificial intelligence learning algorithms.

3.2 Technical Design Goals

The system is designed to meet particular criteria. The system is designed for multiple users to interact with each other in a dynamic marketplace. It is designed so that a novice trader can master the interface within minutes. It is designed to be platform independent (including the client, server, and database). The system is designed to be simple, compact, and easy to modify. It is designed to recover from technical problems without losing data. It is not intended to trade real money (with this prototype version).

3.2.1 Dynamic Marketplace

A trader’s actions may impact the prices that other traders see. That is, traders are not acting in an isolated environment. Rather, they are interacting with each other, either through price impact or by directly transacting. Prices are set by economic supply and demand among traders and price discovery may or may not be facilitated by a supplier of liquidity (such as a market maker).

3.2.2 Novice Users

The interface used by the trader is designed for non-expert users. Generally, a participant can learn how to log in, submit orders, and view market

data within 5 minutes. The traders use a graphical user interface, which is easier than a command-line driven interface for a beginner.

3.2.3 Platform Independence

The entire system is written in Java 1.02 to maximize portability.

Portability is most critical for the trader applet because it is crucial that participants can log in conveniently. The client has been tested extensively on PC platforms (using Internet Explorer 4.0, Netscape 4.5, and Sun's Appletviewer), on Unix platforms such as Linux and Solaris (using Sun's Appletviewer), and on Macintosh platforms (using Internet Explorer 4.0 with Apple's Macintosh Runtime for Java 2.1.1).

The server application has been tested on Linux and on Microsoft Windows NT Server 4.0. The server can be ported to additional platforms without rewriting code.

We have used multiple databases, including Microsoft Access, PostgreSQL (a free software SQL database), and, most extensively, Microsoft SQL Server.

3.2.4 Simulated Markets

The system is generally not designed to trade for real money in terms of privacy, authentication, and robustness. The currently implemented system is a prototype, and its intended use is for simulations in an academic environment.

3.2.5 Simplicity and Compactness

A primary goal of the system is simplicity and compactness. The code is generally concise, and future contributors can understand the overall system in a period measured in days, not weeks or months. By keeping the code simple, it is easier to make modifications to accommodate special needs in the future.

We used a single-server architecture. For a large, nationwide simulation it may be necessary (depending on trading volume) to modify the system to use multiple servers to divide the load. For simplicity, this first prototype was implemented assuming a single-server.

3.2.6 Fault Recovery

A system designed to run trading simulations that may last many months must be able to recover gracefully from system crashes. In our experience, the central server did not crash during a simulation. However, in the event of a crash, it would be possible to restart the server without loss of orders,

transaction, or user portfolios. The data is as persistent as the database that is used.

If a trader's Netscape or Internet Explorer browser were to crash, no data would be lost because no critical data is stored on the client-side.

3.3 Description of the Primary System Components

The primary system components include the trading client that human traders use directly, the central market server (which includes the EMMs), and the automated "robot" traders.

3.3.1 Trading Client

The trader interface is a Java applet. Traders access the applet with an http request. The applet automatically connects to a central server and asks the trader to log in. After logging in, the trader can subsequently submit orders to buy and sell stock, view graphs of historical prices, and view their portfolio and outstanding (unexecuted orders). A central server processes all stock orders and maintains records of all orders, sales, and trader positions.

The AFM Trader has 4 windows, the main window (inside the browser), the dynamic (self updating) chart, the historical (long term) graph, and the portfolio table.

The main window usually appears inside an Internet browser window. It is used to submit orders for trading, to view “news flashes,” and to summon the other client windows.

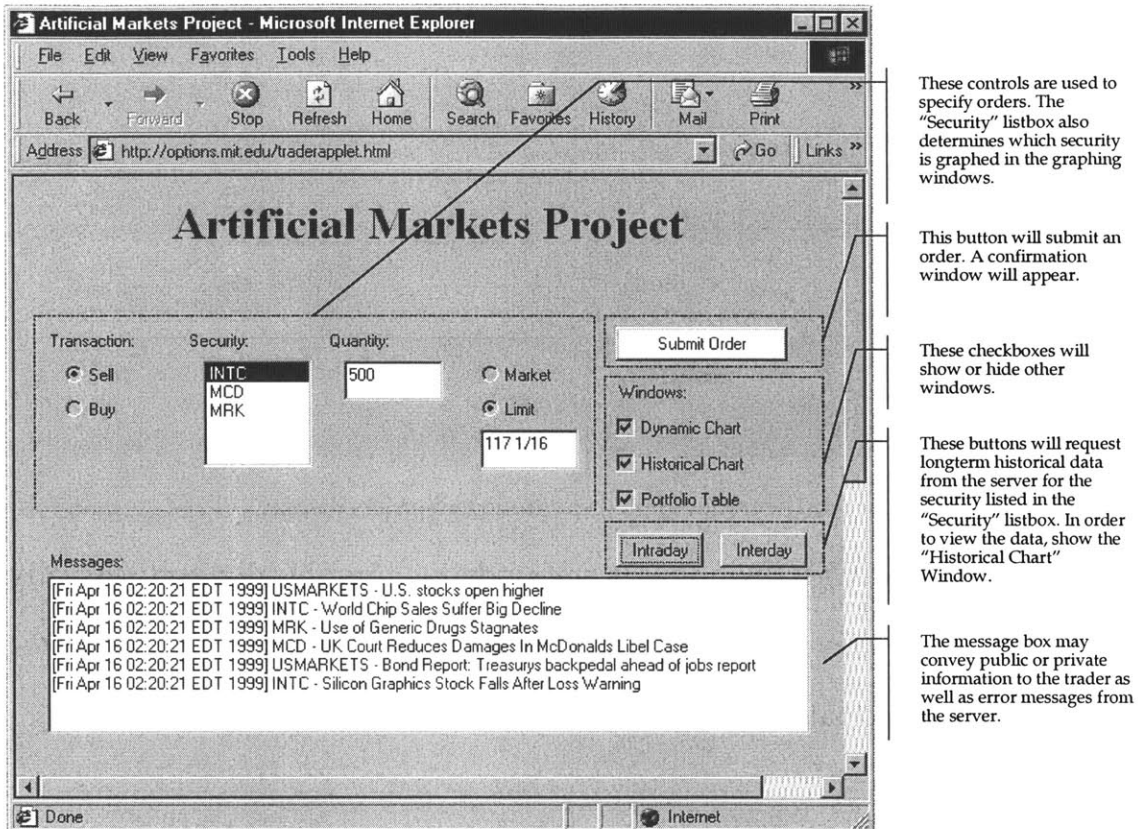


Figure 3: The trader interface loads automatically as a Java applet in the trader’s Internet browser

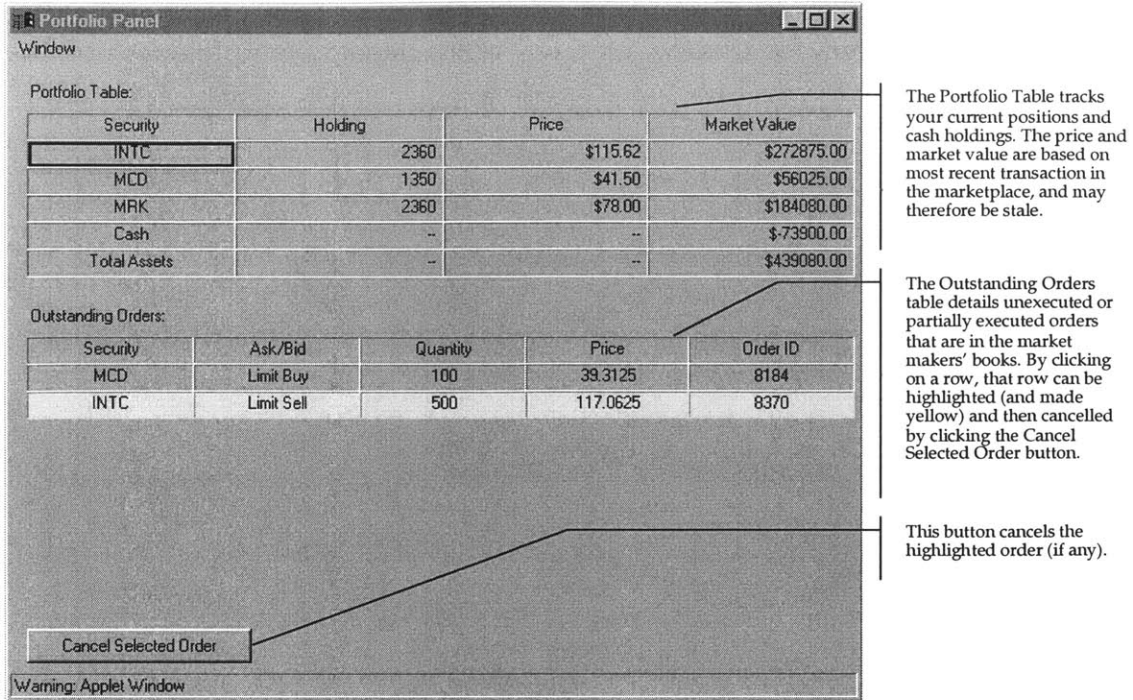


Figure 4: The trader applet's portfolio and orders display

The Dynamic Chart window is self-updating, graphically and textually displaying the most current quote every two seconds. This chart is useful for monitoring price fluctuations within the last few minutes.

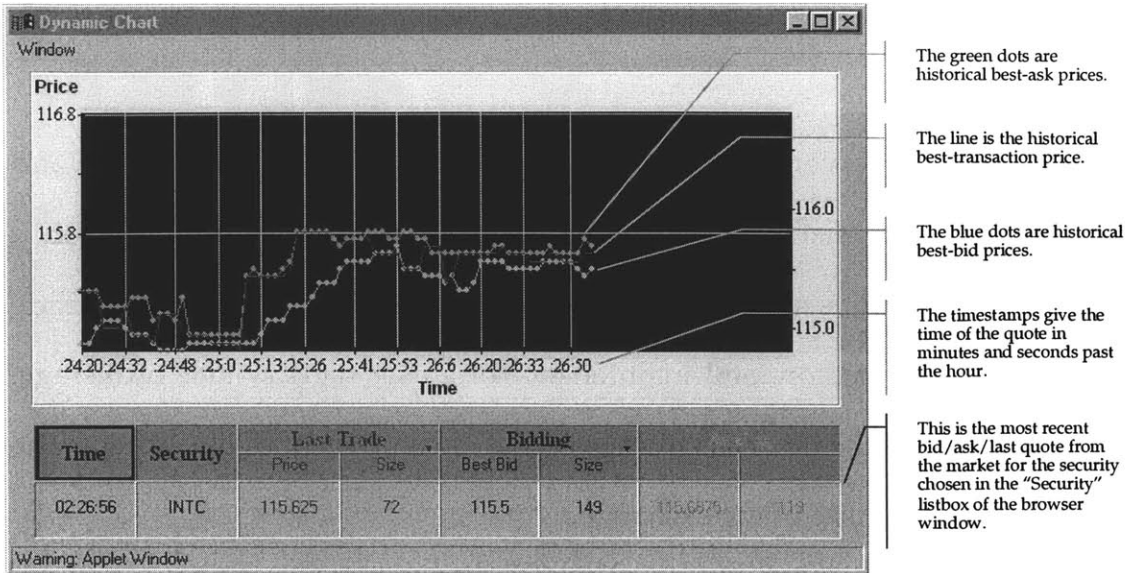


Figure 5: The trader applet's dynamic chart is self-updating, displaying recent bid, ask, and sale prices.

The Historical Chart window graphs longer-term price and volume series. It can show data in either 5-minute intervals or in 1-day intervals. This graph is most useful for simulations multiple weeks in duration.

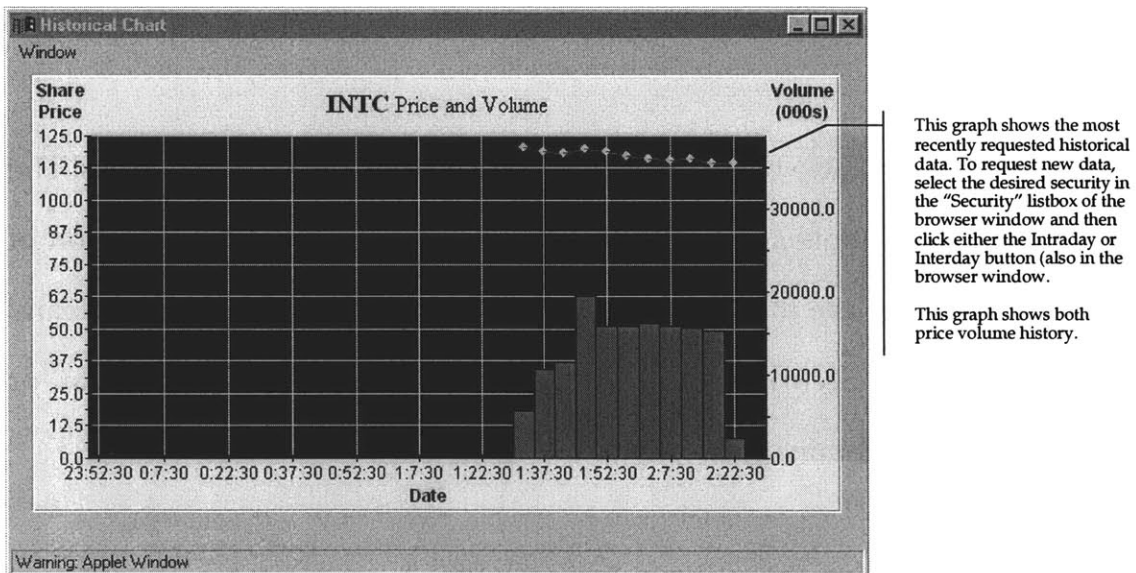


Figure 6: The historical chart displays longer term historical price and volume trends

3.3.2 Market Server

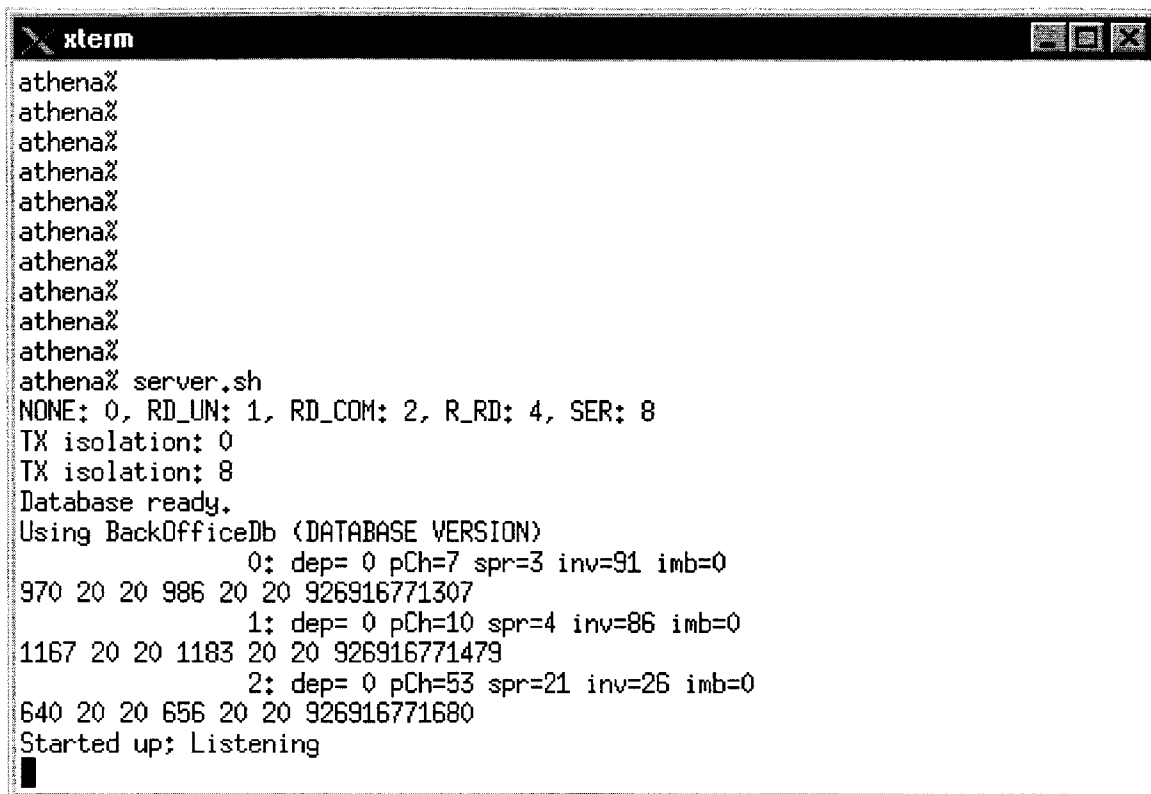
The market server provides an interface to every trader (human and automated), to every marketmaker, to the database, and to the administrator. It provides the functions of order routing, order execution, quote and trade information dissemination, and maintenance of a user registry and brokerage. The server abstracts the database from every other component of the system. The server also enforces rules of the exchange.

The server is highly configurable. The market structure is very flexible, allowing a variety of financial structures (continuous auctions, discrete call auctions, specialist systems), exchange regulations. This financial structure is abstracted into a market maker interface. The interface can be implemented by an automated market maker that performs any arbitrary type of marketplace in a plug-and-play manner.

The server can also be configured to trade various types of securities. The market can trade options, commodities, equities, futures, or a combination of these security types. The market can essentially be modified to trade almost any imaginable security. Securities can be traded in cents, sixteenths of a dollar, or almost any other denomination.

The server is a Java application that can run on any platform with a Java Virtual Machine that supports Java 1.02 (the most universally compatible version of Java). When the server is started, a connection to the database is made, the

marketmaking agents are started, the administrator graphical interface is started, and the server begins listening for traders who log in. When a trader logs in, a server-side thread is started to address requests from the trader.

The image shows a terminal window titled 'xterm'. The prompt 'athena%' is repeated ten times. Then, the command 'athena% server.sh' is entered. The output of the script is as follows:

```
athena%  
athena%  
athena%  
athena%  
athena%  
athena%  
athena%  
athena%  
athena%  
athena%  
athena% server.sh  
NONE: 0, RD_UN: 1, RD_COM: 2, R_RD: 4, SER: 8  
TX isolation: 0  
TX isolation: 8  
Database ready.  
Using BackOfficeDb (DATABASE VERSION)  
0: dep= 0 pCh=7 spr=3 inv=91 imb=0  
970 20 20 986 20 20 926916771307  
1: dep= 0 pCh=10 spr=4 inv=86 imb=0  
1167 20 20 1183 20 20 926916771479  
2: dep= 0 pCh=53 spr=21 inv=26 imb=0  
640 20 20 656 20 20 926916771680  
Started up; Listening
```

Figure 7: When the server is started, a database connection is established, the market makers are initialized, the server begins listening for human and “robot” traders, and the GUI is opened (see figure 8)

The main window of the administrator interface is used to open and close the market, to send “news flashes” to traders, and to open windows detailing the market makers’ books, positions, and actions. Note that closing the market does not terminate connections to the traders. Closing the market simply ceases the acceptance and execution of orders.

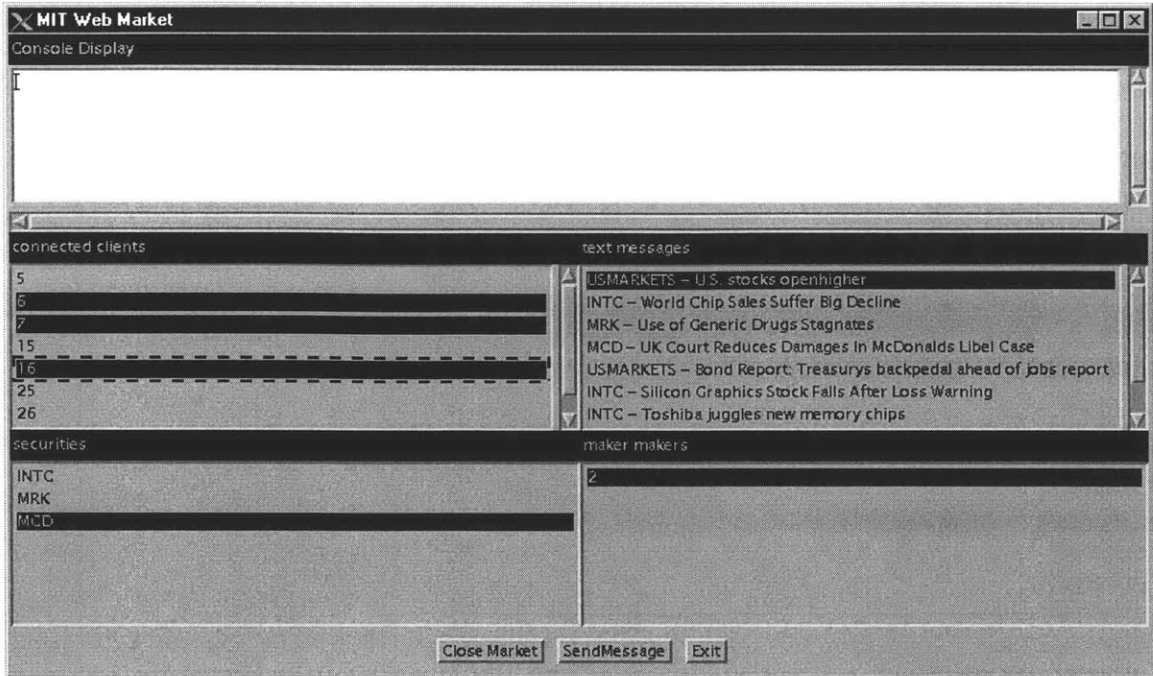


Figure 8: The Server GUI is used to open and close the market, open a window to monitor a specific market maker (see figure 9), and send messages to traders. Certain messages can be sent to selected traders by selecting the desired trader(s) in the “connected clients” box, selecting the desired message(s) in the “text messages” box, and clicking the “Send Message” button.

The market maker windows automatically refresh when an order is received, providing a dynamic, real-time view of the trades, order, and limit book. A market maker interface also shows the profit and loss, the current position, the current quote, and other information relevant to a particular marketmaker.

This window also allows an administrator to manually control the execution of orders, and to configure the relative importance of each of the financial objectives of the market makers (such as inventory target, minimum spread size, maximum spread size, market depth, price continuity, size of the spread, size of inventory, and order imbalance).

The market maker can also be inactivated entirely using this window.

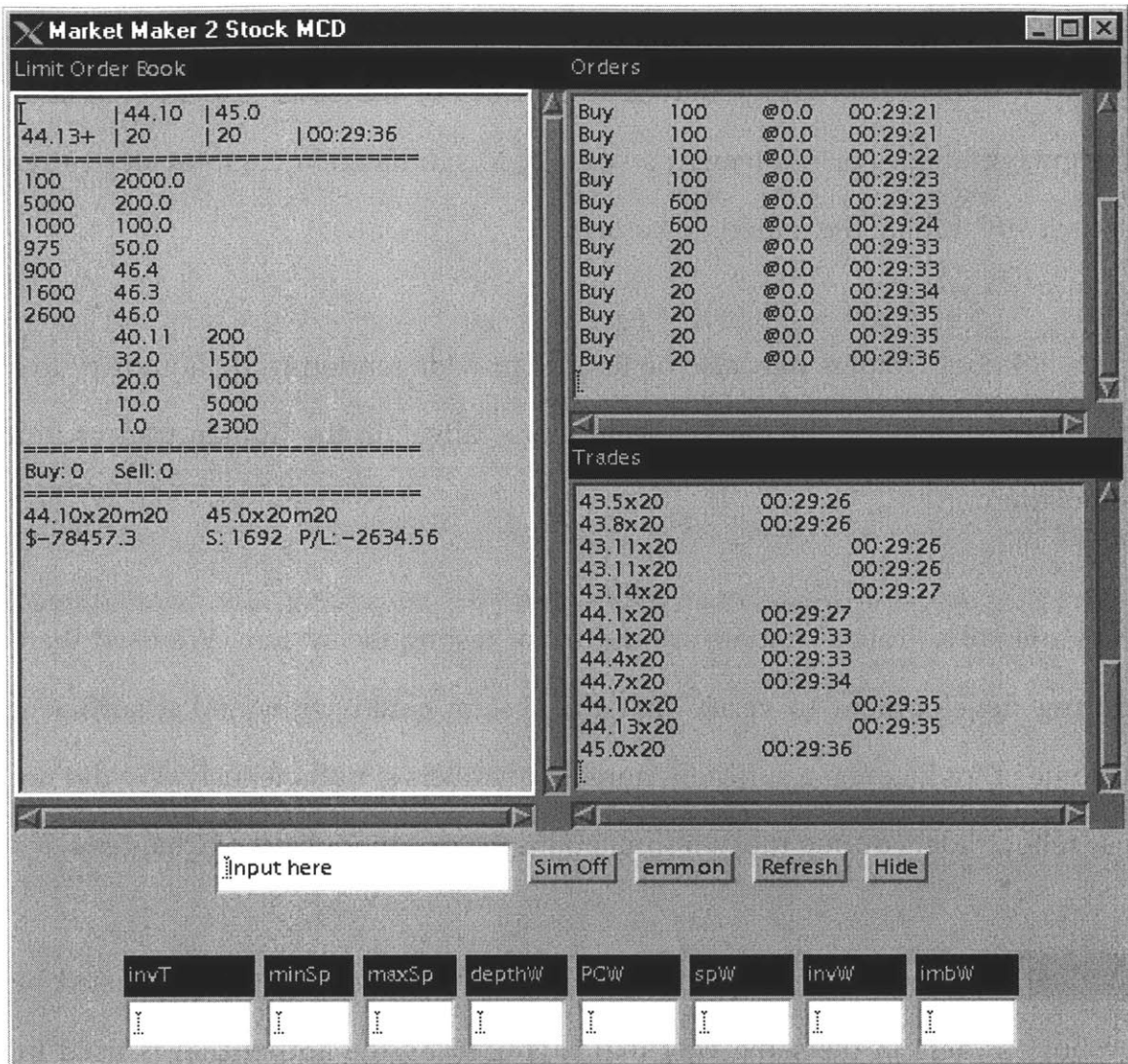


Figure 9: The Electronic Market Maker Interface

3.3.3 Automated "Robot" Traders

"Robot" traders are useful for a variety of tasks. As they are programmable, they may be used to test trading strategies against human traders or other "robot" traders. Simple "robot" traders may use a simple randomized price process to generate random orders. Somewhat more complicated traders

may execute a momentum trading strategy, buying when a stock is rising and selling when it is falling. More complex “robot” traders may use sophisticated machine learning techniques to simulate “intelligence.” “Robots” may also perform statistical arbitrage between securities or they may execute dynamic hedging strategies or implement a strategy to synthesize a complex derivative security by trading the underlying assets.

“Robot” traders may also be used to provide randomized “liquidity” to a thin market, making the market seem more “alive” to the human traders in a simulation.

Finally, “robot” traders are useful for testing the system. We used them during development to verify that the system could withstand a sufficient volume of trading and a sufficient number of traders simultaneously. We did not use robot traders during the two simulations that are the subject of Chapter 4.

The central server provides an interface to traders that may be used by “robot” traders in the same way that it is used by the applet that is used by human traders.

3.3.3.1 Random Price Process Traders

The simplest of the “robot” traders are configured using a text file (see figure 10). The file essentially allows the user to define a random-walk price process with a particular starting price, drift, standard deviation, and frequency

of change (as a Poisson inter-arrival time). The “robot” traders defined in the same file use these price processes to submit orders.

```

emacs@OPTIONS.MIT.EDU
Buffers Files Tools Edit Search Help
# this is a comment, empty lines are also ignored

# price processes
# InitPrice interArrivalTime mu sigma
1664 10000 0.0005 0.025
1136 50000 0.0010 0.005
648 10000 0.0050 0.050
1666 10000 0.0005 0.025

# special character "-" to indicate the end of price processes
-

# traders
# type: m=basic machine trader/uninformed
# i=informed
# arrivalTime = interarrival time of an order from the trader, in milliseconds
# priceProcess: -1 = no price process
# 0 = first price process
# username password secId type cancelOrder arrivalTime priceProcess
# sec0
r1 r1pw0 0 m 0 2500 -1
r2 r2pw0 0 i 0 1600 0
r3 r3pw0 0 i 0 1600 3
# sec1
r11 r11pw0 1 m 0 2500 -1
r12 r12pw0 1 i 0 800 1
# sec2
r21 r21pw0 2 m 0 2500 -1
r22 r22pw0 2 i 0 800 2
r23 r23pw0 2 i 0 800 2

```

--*--Emacs: sim1.dat (Fundamental)--L31--All-----

Figure 10: These simplest of “robot” traders can be configured to randomly submit orders according to parameterized processes


```
xterm
athena% genrobot.sh
price processes
1664 10000 0.0005 0.025
1136 50000 0.001 0.005
648 10000 0.005 0.05
1666 10000 0.0005 0.025
Traders
r1 r1pw0 0 m false 2500 -1
r2 r2pw0 0 i false 1600 0
r3 r3pw0 0 i false 1600 3
r11 r11pw0 1 m false 2500 -1
r12 r12pw0 1 i false 800 1
r21 r21pw0 2 m false 2500 -1
r22 r22pw0 2 i false 800 2
r23 r23pw0 2 i false 800 2
true price 40.9375
r23: Order #?:27 BIDs price=653 amt=200 of security 2 (0 served by 2)
true price 70.6875
r12: Order #?:16 BIDs price=1128 amt=800 of security 1 (0 served by 1)
true price 40.9375
r23: Order #?:27 ASKs price=655 amt=500 of security 2 (0 served by 2)
true price 40.9375
r23: Order #?:27 BIDs price=652 amt=1300 of security 2 (0 served by 2)
true price 40.9375
r22: Order #?:26 ASKs price=660 amt=100 of security 2 (0 served by 2)
true price 40.9375
r23: Order #?:27 BIDs price=652 amt=200 of security 2 (0 served by 2)
Trader 25: place no order
true price 105.562
```

Figure 11: The “robot” traders’ output as they submit orders to the market

3.4 Structure of the Database

The database can be any package that is compatible with standard SQL. We used primarily Microsoft SQL Server, although the server has run also with Microsoft Access and with PostgreSQL (a free-software alternative).

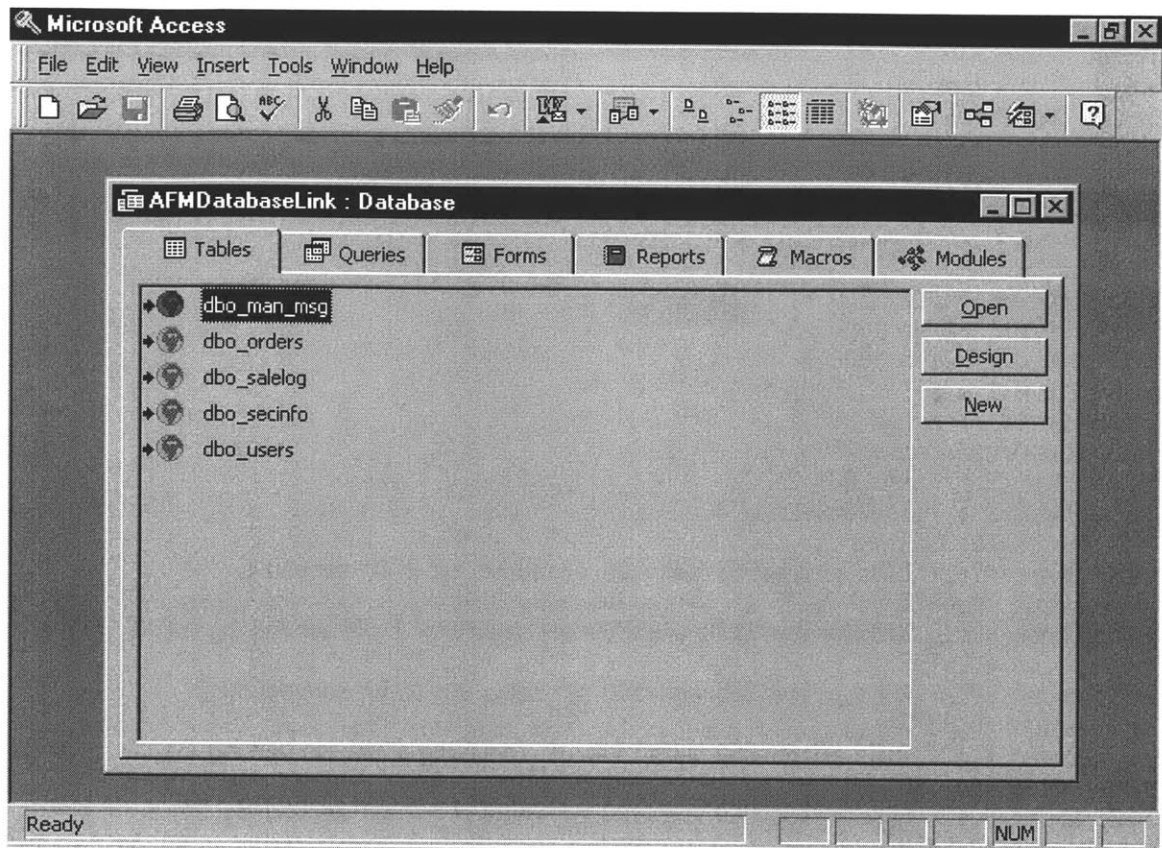


Figure 12: The five tables of the database (Man_Msg, Orders, Salelog, Secinfo, and Users) can be graphically edited with Microsoft Access

There are five tables in the database:

- *Users* saves information regarding user passwords and portfolios.
- *Salelog* saves a record of every transaction.
- *Orders* saves a record of every order.
- *Man_msg* is a registry of the news items that may be sent (manually) during a simulation.
- *Secinfo* contains a record for every security traded on the exchange.

Sales and Orders should be cleared before each simulation. They are used as a way to record a history of a trading session. Man_msg and Secinfo are used to configure the server before a simulation. Users is used to set up usernames,

passwords, and initial portfolios before the simulation, and contains the final portfolio of each user after the simulation (although these final portfolios could be reconstructed by scanning the Salelog).

3.5 Configuring the System for a Simulation

Configuring the system for a simulation primarily involves running a SQL script to set the preferences in the database. For example, user accounts and passwords must be set up, the securities must be specified, and the order and sale logs should be initialized.

3.5.1 Database Version versus In-Memory Version

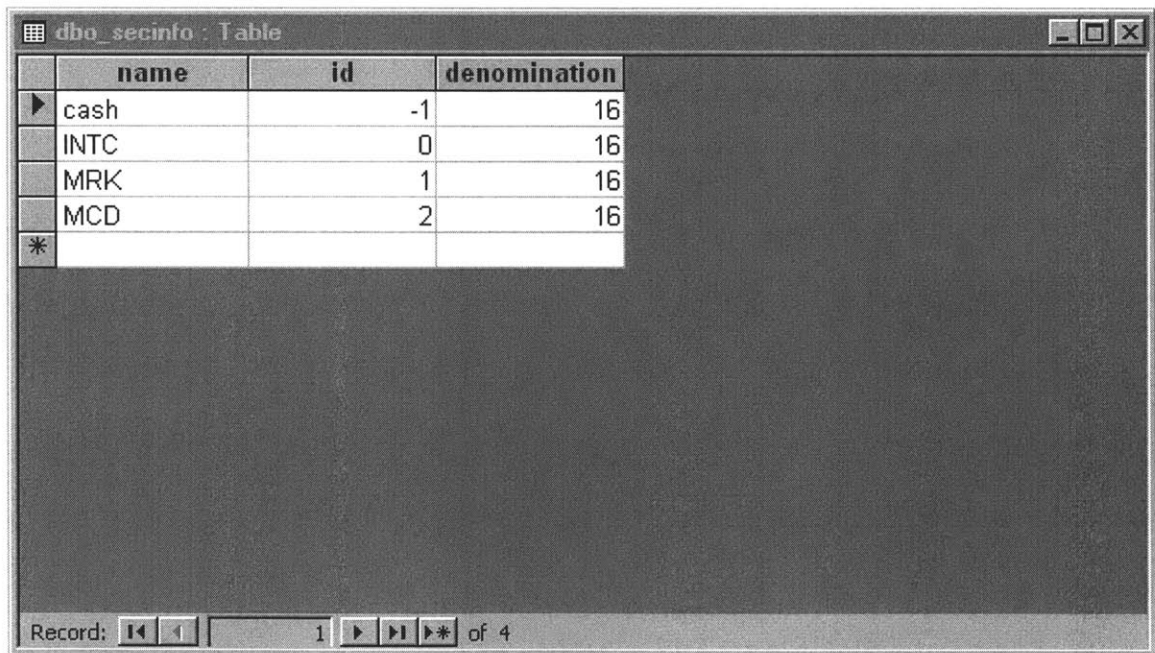
We created two versions of the market server. One version, referred to as the “BackOfficeDb” version, uses the database continuously (updating the database every time a sale or order occurs). This version is more resilient to system failures because there is little opportunity for data loss. The other version, referred to as the “BackOfficeMem” version, saves data to RAM and copying it to the database periodically. The advantage is that there is a slight performance gain by caching the data in Memory.

A primary reason for creating two versions of the server is to evaluate the advantages and disadvantages of each design. The non-caching BackOfficeDb version seems advantageous in almost every situation. While the BackOfficeMem version can handle roughly twice the volume of transactions and orders, the

BackOfficeDb version is fast enough and is more robust. Furthermore, a faster database and especially a better JDBC/ODBC driver would likely close the performance gap. Additional performance details are available in the section on performance.

3.5.2 Setting up the Traded Securities

The Secinfo table stores parameters for the traded securities. It contains three columns: name, id, and denomination.



The screenshot shows a window titled 'dbo_secinfo : Table'. The table has three columns: 'name', 'id', and 'denomination'. The data rows are as follows:

	name	id	denomination
▶	cash	-1	16
	INTC	0	16
	MRK	1	16
	MCD	2	16
*			

At the bottom of the window, there is a record navigation bar showing 'Record: 1 of 4'.

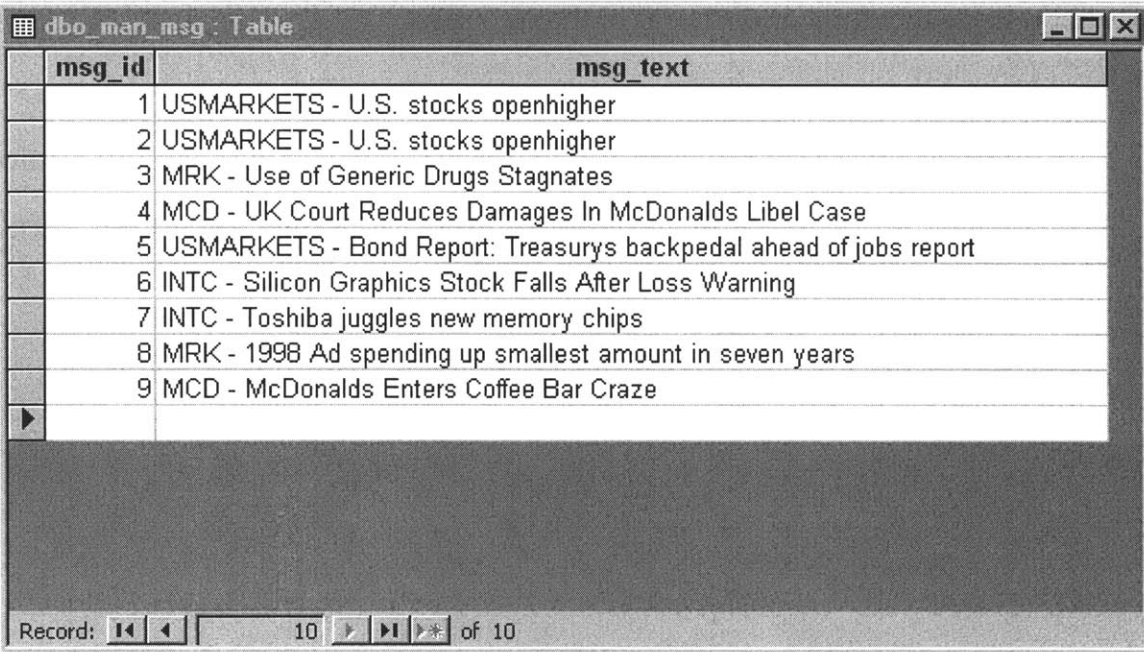
Figure 13: The Secinfo table configures the security names, the security IDs, and the denomination (or tick size) in which they trade

The 'cash' security should always have an ID of negative one. The other securities should have sequential IDs starting with zero. The securities may have virtually any symbol. Denomination declared the tick size for each security. If the

denominations were set to one hundred, the securities would trade in cents. Cash should trade in a denomination that is a greatest common multiple of the denominations of all the other securities. In most real markets, securities trade in denominations that are a power of two (1, 2, 4, 8, 16, 32, 64, et cetera).

3.5.3 Setting up the “News Flashes”

From the administrator window, the server can send “news flashes” to all or to selected traders. These messages are entered into the database beforehand in the Man_msg table.



The screenshot shows a window titled 'dbo_man_msg : Table' containing a table with two columns: 'msg_id' and 'msg_text'. The table contains 9 rows of data. At the bottom of the window, there is a record navigation bar showing 'Record: 10 of 10'.

msg_id	msg_text
1	USMARKETS - U.S. stocks openhigher
2	USMARKETS - U.S. stocks openhigher
3	MRK - Use of Generic Drugs Stagnates
4	MCD - UK Court Reduces Damages In McDonalds Libel Case
5	USMARKETS - Bond Report: Treasurys backpedal ahead of jobs report
6	INTC - Silicon Graphics Stock Falls After Loss Warning
7	INTC - Toshiba juggles new memory chips
8	MRK - 1998 Ad spending up smallest amount in seven years
9	MCD - McDonalds Enters Coffee Bar Craze

Figure 14: The “News Flashes” are configured in the Man_Msg table

3.5.4 Setting up the Users and Portfolios

The Users table in the database contains the username and password of each trader (including human traders, “robot” traders, and marketmakers). Each trader also has a unique ID.

The amount of cash and stock is also specified in this table. Initial portfolios for each trader are set before the simulation begins. At the end of the session, the table will have been modified to reflect the final portfolios of each user. The integers in the “cash” column reflect the amount of cash units (in the denomination specified for cash in the Secinfo table). The integer under the stock symbol reflects the number of shares of that security.

Note that this table contains a ‘CRL’ security that is not in the Secinfo table. Because ‘CRL’ is not in the Secinfo table, the server will ignore the ‘CRL’ column.

	name	password	id	cash	INTC	MCD	MRK	CRL
▶	group1	4320	401	240000	300	150	200	0
	group2	2016	402	1972245	-116	3	-3	0
	group3	4295	403	240000	300	150	200	0
	group4	3591	404	240000	300	150	200	0
	group5	6178	405	887837	0	0	0	0
	group6	1304	406	-39961	0	0	0	0
	group7	5241	407	1974178	-860	-139	0	0
	group8	8370	408	721804	0	0	140	0
	group9	3083	409	296411	568	0	0	0
	group10	5600	410	2276274	-799	-580	212	0
	group11	3397	411	240000	300	150	200	0
*								

Record: 208 of 218

Figure 15: The Users table configures the usernames, passwords, user IDs, and initial portfolios. After the market is closed, this table will be altered to reflect updated portfolios.

3.5.5 The Salelog and Orders Tables

The records in the Salelog and Orders tables should be deleted before each trading simulation. There are two reasons for this:

- Orders in the database that were not executed or only partially executed during the previous session will be loaded into the limit order book when the server is started.
- If orders and sales accumulate perpetually in the database, the database might suffer a decrease in speed or memory efficiency.

Each record in the Salelog saves the ID of the sale, the marketmaker that performed the transaction, the security ID, the buyer ID, the seller ID, the number of shares, the price (in ticks) and the time.

	id	marketMaker	security	buyer	seller	amount	price	time
▶	6	0	0	7	0	20	986	5/17/99 1:06:33 AM
	7	0	0	7	0	20	989	5/17/99 1:06:33 AM
	8	0	0	7	0	20	992	5/17/99 1:06:34 AM
	9	0	0	7	0	20	992	5/17/99 1:06:34 AM
	10	0	0	7	0	20	995	5/17/99 1:06:34 AM
	11	0	0	7	0	20	998	5/17/99 1:06:34 AM
	12	0	0	7	0	20	998	5/17/99 1:06:35 AM
	13	0	0	7	0	20	1001	5/17/99 1:06:35 AM
	14	0	0	7	0	20	1004	5/17/99 1:06:35 AM
	15	0	0	7	0	20	1004	5/17/99 1:06:35 AM
	16	0	0	7	0	20	1007	5/17/99 1:06:35 AM
	17	0	0	7	0	20	1010	5/17/99 1:06:36 AM
	18	0	0	7	0	20	1010	5/17/99 1:06:36 AM
	19	0	0	7	0	20	1013	5/17/99 1:06:36 AM
	20	0	0	7	0	20	1016	5/17/99 1:06:36 AM
	21	0	0	7	0	20	1016	5/17/99 1:06:37 AM
	22	0	0	7	0	20	1019	5/17/99 1:06:37 AM
	23	0	0	7	0	20	1022	5/17/99 1:06:37 AM
	24	0	0	7	0	20	1022	5/17/99 1:06:37 AM
	25	0	0	7	0	20	1025	5/17/99 1:06:37 AM
	26	0	0	7	0	20	1028	5/17/99 1:06:38 AM
	27	0	0	7	0	20	1028	5/17/99 1:06:38 AM

Record: 1 of 173

Figure 16: The Salelog table logs details of every sale during a simulation. This data can be exported for analysis.

The Orders table saves the ID of the order, the marketmaker that received the order, the type of the order (encoded here as an integer indicating a market buy, a market sell, a limit buy, or a limit sell), the trader's ID, the security ID, the price (in ticks), the number of shares, the amount that has been transacted (fulfilled) so far, and the time.

id	supersedes	marketMaker	otype	client	security	price	amount	fulfilled	time
2	-1	2	3	27	2	615	600	0	5/17/99 1:06:32 AM
3	-1	2	3	27	2	618	2200	40	5/17/99 1:06:33 AM
4	-1	2	3	27	2	617	1500	0	5/17/99 1:06:33 AM
5	-1	0	3	7	0	1623	1100	1100	5/17/99 1:06:33 AM
61	-1	1	4	15	1	1193	500	0	5/17/99 1:06:46 AM
62	-1	0	1	7	0	0	20	20	5/17/99 1:06:47 AM
64	-1	1	2	16	1	0	20	20	5/17/99 1:06:47 AM
66	-1	0	1	5	0	0	20	20	5/17/99 1:06:48 AM
68	-1	2	2	26	2	0	20	20	5/17/99 1:06:48 AM
70	-1	2	2	27	2	0	20	20	5/17/99 1:06:49 AM
72	-1	1	2	15	1	0	20	20	5/17/99 1:06:49 AM
74	-1	0	1	5	0	0	20	20	5/17/99 1:06:49 AM
76	-1	2	2	27	2	0	20	20	5/17/99 1:06:50 AM
78	-1	2	1	25	2	0	20	20	5/17/99 1:06:50 AM
80	-1	0	1	6	0	0	20	20	5/17/99 1:06:51 AM
82	-1	2	2	25	2	0	20	20	5/17/99 1:06:51 AM
84	-1	1	1	15	1	0	20	20	5/17/99 1:06:51 AM
86	-1	0	1	6	0	0	20	20	5/17/99 1:06:52 AM
88	-1	2	2	27	2	0	20	20	5/17/99 1:06:52 AM
90	-1	2	2	26	2	0	20	20	5/17/99 1:06:53 AM
92	-1	2	2	26	2	0	20	20	5/17/99 1:06:53 AM

Figure 17: The orders table logs every order during the simulation.

The IDs used by the Orders table and Salelog table are based on an incrementing counter shared between the two tables. That is, no order and sale will share the same ID. IDs are assigned in chronological order by the server.

3.6 Application Programming Interfaces and Internal Mechanics

This section details the key interfaces that define the modularity of the system. The ProtocolClient object is used by human and “robot” traders to log in, request information, and place trades.

3.6.1 Trading Client API

The ProtocolClient object (defined in ProtocolClient.java) provides a trader's access to the marketplace. The "robot" trader or web browser applet starts by logging the user into the system:

```
ProtocolClient market = new ProtocolClient ("options.mit.edu");
market.login ("user1", "pass1");
```

A challenge response scheme is used to avoid transmitting the password over the Internet. If the login is successful, the applet can make requests to the server regarding the client's portfolio, for example, as follows:

```
SecurityInfo IntelInfo = market.getSecInfo ("INTC");
int IntelID = IntelInfo.getId ();
int IntelHoldings = market.getHoldings (IntelID);
int CashHoldings = market.getCashHoldings ();
```

The trader might want to place a limit order to buy Intel:

```
market.placeorder ( new Order (Order.BID, marketmaker, myid,
                             IntelID, amount, price/IntelDenomination));
```

An interesting thing to do is to request periodic updates from the server. The trader can specify which types of transactions are of interest using the "requestUpdates()" method. The server will then notify the trader either

asynchronously (that is, as information becomes available) or at set intervals (every two seconds, for example).

The API for ProtocolClient is characterized in Table 1. This summarizes the functions available to trading clients such as “robot” trader programs.

Routine name	Description
<i>RequestUpdates(update template)</i>	Request periodic updates of security and portfolio data
<i>login(name, pw)</i>	Log into market server
<i>placeOrder(order)</i>	Place an order
<i>cancelOrder(order)</i>	Cancel an order
<i>showOrders()</i>	Show current orders (that are not fully executed)
<i>getSecInfo(secname or secid)</i>	Request information on a security
<i>getSecInfos()</i>	Request information on all securities
<i>getHoldings(secname or secid)</i>	Request information on current holdings
<i>getCashHoldings()</i>	Request information on current cash holdings
<i>describeSale(sale)</i>	Return a String describing a sale
<i>describeOrder(order)</i>	Return a String describing an order
<i>getIntraday(secid, starttime)</i>	Get price and volume history in 5-minute intervals
<i>getInterdat(secid, starttime)</i>	Get price and volume history in 24-hour intervals

Table 1: Trader Applet Interface (ProtocolClient.java)

3.6.2 Marketmaker API

The marketmaker use methods in BackOffice, such as noteSale, to execute a sale. The marketmakers also provide methods that the server calls to notify the marketmaker of new events. The server calls other functions to query the marketmaker for information. Table 2 presents these functions.

Routine name	Description
<i>placeOrder(order)</i>	Notify the market maker of a new incoming order
<i>cancelOrder(order)</i>	Notify that an order has been cancelled
<i>bestBid(secid)</i>	Return the best bid
<i>bestAsk(secid)</i>	Return the best ask
<i>getLastSale()</i>	Return the last sale

Table 2: Functions the Marketmaker provides for the Market (MarketMaker.java)

3.6.3 Market Server API

The following table defines the essential functions of the central server. These are primarily the functions that access and modify the database. These functions are called on behalf of market making modules and traders.

Routine name	Description
<i>reportOrder(order)</i>	Report a new order
<i>noteSale(buyorder, sellorder)</i>	Report a new sale
<i>cancelOrder(order)</i>	Cancel an order
<i>ordersForClient(id)</i>	Get all orders for a specific client
<i>addMarketMaker(mm)</i>	Add a new market maker to the server
<i>getSecInfo(name or id)</i>	Retrieve information on a security
<i>getPassword(name)</i>	Get a user's password
<i>getUserId(name)</i>	Get a user's login ID
<i>getHoldings(client id, secinfo)</i>	Get a user's current holdings
<i>getCashHoldings(client id)</i>	Get a user's current cash holdings
<i>addClient(protocolserver)</i>	Add a client connection to the server
<i>dropClient(protocolserver)</i>	Remove a client connection from the server
<i>sendMessage(client id, string messags)</i>	Send a text message to a client
<i>saveToDb()</i>	Save all current information to the persistent store (meaningful only in the BackOfficeMem version that caches data in memory temporarily)

Table 3: Primary Market Server Functions (BackOffice.java)

The market server also provides an interface for the administrator to perform tasks such as opening and closing the market.

3.7 Performance, Robustness, Scalability

The system has been tested in two sessions using MBA and Undergraduate candidates from the MIT Sloan School of Management as traders. In the first session, eleven groups traded excessively for 20 minutes of total trading, executing over 3100 trades through an electronic marketmaker. For this simulation, we used the "BackOfficeMem" version, which caches the data in RAM before saving it to the database. The server handled this load averaging over 2 trades per second and peaking at over 8 trades per second without any sign of delays or technical problems.

The second test was a 5-day trading simulation with significantly lower volume. Traders logged in from browsers at home. The market was open for trading over 23 hours a day during the week (over 98 1/2 hours), using the BackOfficeDb version. We experienced no technical problems during this simulation.

There are two apparent bottlenecks in the system: Processing orders and sales, and sending periodic updates to the clients. When the system is overloaded by a large number of hyperactive "robot" traders, the system tends to fall behind on order processing rather than crash.

We have positively verified that the server can handle at least 50 clients simultaneously logged in simultaneously without showing any slowdown in performance. The primary bottleneck for simultaneous logins is sending updates

to clients when a trade occurs, and keeping the clients informed regarding their portfolio and open orders. The growth of this bottleneck is approximately $O(\#Securities \times \#ActiveClients)$. This bottleneck can be addressed by reducing the update frequency, changing the architecture so that clients must specifically request updated information, or further optimizing the architecture for updates.

The other bottleneck is the processing of orders and sales. Using the BackOfficeMem version, the system is capable of processing at least 10 trades per second for an extended time. The BackOfficeDb version is capable of processing at least 5 trades per second. The primary issue with the BackOfficeDb version is the speed of the database driver. Our JDBC/ODBC driver, while adequate for our needs, did offer room for improvement from a performance perspective.

These results are based on a server running Linux on an Intel Pentium III 450-Mhz computer with 384 megabytes of RAM and a 14-gigabyte disk drive. The server accessed via Ethernet a SQL Server database on a neighboring Intel Pentium Pro 200-Mhz computer with 192 megabytes of RAM and a 2.1-gigabyte disk drive.

3.8 Future Directions of the Technical Architecture

This marketplace represents a solid experimental platform for academic simulations of financial markets. It is capable of running trading simulations with on the order of 30-50 traders logged in simultaneously, on the order of 8 transactions per second, and on the order of 5 to 10 securities traded. In order to

scale these numbers by an a factor of 10, some parts of the system may need to be re-worked:

- The database performance was disappointing. This may be because the queries were not optimized. However, the queries were not complex in nature and the more likely explanation is that the JDBC driver is the bottleneck.
- Using Remote Method Invocation (RMI) may help reduce the overhead associated with accessing methods over the network. The server code includes essentially a home-implementation of RMI functionality written in Java. This was written because, at the time, Java's RMI did not exist.
- Using multiple servers would possibly improve performance. Architecturally, the first division of labor between servers may be to divide the task of information dissemination (such as sending periodic updates to every client and replying to requests for historical data from the database) from the task of executing orders.
- The trader applet could be optimized to request less information from the server.

From a functionality perspective, additions to the architecture would be based on the requirements of any additional market structures or security types. That is, if a more complex auction structure (like that used by Optimark) were implemented, additional functionality would be needed.

A graphical interface designed for experienced users would an improvement if the system were to be used as the basis for a business school

laboratory course (which is a possibility that is being considered). Typing in orders using a keyboard would be quicker than clicking a GUI, although the GUI is clearly superior for novice users.

Using a Java servlet architecture for the front end would be an improvement as well. A servlet would essentially move the Java associated with the trader applet into a back-end "agent." This agent would then serve html to the trader's browser as needed. This would be advantageous by reducing load time (which is a few minutes over dialup, but only a few seconds over Ethernet). Also, it would further improve compatibility and robustness on the client side. Serving html is generally a better solution than running Java on the client side.

Enforcing a constant time cadence would be necessary for more professional uses of the server. By constantly clicking on requests for historical data, a trader might be able to slow down the server enough that it would be noticeable to other traders. That is, if a trader were in a position to benefit from delays in other trader's executions, the trader could possibly load the server intentionally. Ideally, a mechanism to prevent this should exist.

Building tools for automated configuration and data analysis would be useful. We configured the system manually, sometimes by slightly modifying code and typing SQL commands to the database. A clean application to configure the system would be a nice addition. Also, data analysis was performed manually using Excel. Automating the data processing would also increase convenience.

3.9 Conclusion

The core design goals have been satisfied. The system is a simple, compact, robust, fault tolerant, platform independent, dynamic simulated marketplace.

From a technical standpoint, the system is quite successful. The system has withstood stress tests intense, high volume trading using human traders, and has withstood the test of a week of operation virtually 24-hours a day. Furthermore, the system has been tested using hyperactive “robot” traders submitting 100 orders per second without breaking (although execution turnaround was not instantaneous).

The insights gained from this prototype will be useful when building the next-generation, which is underway.

Chapter 4

Design and Results of the Simulated Trading Sessions

After designing and implementing our online exchange and market makers, and testing the system with simulated “robot” traders, we ran two simulated trading sessions in which human traders logged into the system and competed with each other.

The participants were volunteers from Professor Andrew Lo’s Financial Engineering course at the MIT Sloan School of Management. The students were primarily Wall Street bound second-year MBA candidates and senior undergraduates. The students were grouped into eleven teams, each with about six members. The winning teams would receive twenty bonus points on their final exam, with no points allocated to or detracted from a team not finishing first.

The first session involved about 25 minutes of trading of a single stock, with a short break between the two trading periods. The participants logged in from the Sloan Trading Laboratory, and were given earnings information during the simulation that would affect the price of the fictional stock, Charles River Logging.

The second session lasted about 98 1/2 hours, running 23 hours a day from Monday at 9:30 am to until Friday at 4:00 pm during a single week. In the second session, students traded Intel, Merck, and McDonald's equity in the fictional "AFM" exchange. The prices of the three equities were pinned on Monday morning to the week's NYSE/NASDAQ opening price, and they were pinned on Friday to the week's NYSE/NASDAQ closing price. During the week of trading between the first open and last close, the price of the stocks were allowed to deviate from those of the NY markets due to forces of local supply and demand. The traders were divided into three fund types – hedge funds, active mutual funds, and index funds – with distinct risk and return preferences.

From a technical point of view, the simulations served to verify our online exchange system. The simulations ran without interference from technical issues and trading load did not reach a level that would stress the server. For information regarding technical performance, please see the section on Performance, Robustness, Scalability (Section 3.7).

4.1 Charles River Logging (CRL) Simulation

The Charles River Logging case was based partially on a case previously developed at Carnegie Mellon as part of the FTS system [citation]. The CRL case was run one time in the Sloan Trading Laboratory. Students were given the case in advance in order to prepare strategies for the fast-paced simulation.

4.1.1 Specification of the CRL Case

A fictitious company, Charles River Logging Incorporated (Symbol: CRL), has recently embarked on a 2-year logging project and they have financed this project partly through equity traded on the AFM exchange. At the end of year two, the earnings from the 2-year project will be distributed as dividends to the shareholders and the company will be dissolved.

CRL trades in increments of 1/16 of a dollar.

Demand for logs and the costs of production are unknown. These factors will affect the earnings of CRL, and therefore they should have significant bearing on the trading price of the stock. In year 1, there are 3 possible outcomes, each leading to a different earnings announcement at the end of year 1:

Event in Year 1	Terminal Future Value of Year 1 Earnings
Very weak demand, some costs renegotiated (X1)	\$0
Weak demand, some costs renegotiated (Y1)	\$12
Weak demand, successful cost reduction (Z1)	\$24

Table 4: There are three possible outcomes of period one.

For example, if it turns out that CRL experiences weak demand and renegotiates some costs (i.e. event Y1) then the stock of CRL should theoretically trade at no less than \$12 at any time during year 2 because a risk-free cash flow with a future value of \$12 per share is guaranteed to the shareholder at the end of year two. There is no chance of CRL dissolving before the end of year 2.

The earnings in year 2 depend on the business environments both of year 1 and of year 2. The full earnings schedule for year 1 and year 2 is detailed in the following table:

Event in Year 1	Event in Year 2	Terminal Future Value of Year 1 Earnings (per share)	Terminal Future Value of Year 2 Earnings (per share)	Terminal Value of Stock at end of Year 2 (col. 3 plus col. 4)
Very weak demand, some costs renegotiated (X1)	Very weak demand, some costs renegotiated (X2)	\$0	\$0	\$0
"	Weak demand, some costs renegotiated (Y2)	\$0	\$0	\$0
"	Weak demand, successful cost reduction (Z2)	\$0	\$12	\$12
Weak demand, some costs renegotiated (Y1)	Very weak demand, some costs renegotiated (X2)	\$12	\$0	\$12
"	Weak demand, some costs renegotiated (Y2)	\$12	\$12	\$24
"	Weak demand, successful cost reduction (Z2)	\$12	\$24	\$36
Weak demand, successful cost reduction (Z1)	Very weak demand, some costs renegotiated (X2)	\$24	\$12	\$36
"	Weak demand, some costs renegotiated (Y2)	\$24	\$24	\$48
"	Weak demand, successful cost reduction (Z2)	\$24	\$24	\$48

Table 5: This chart presents all nine possible two-period scenarios, and the corresponding final values of the stock prices

The probability of each state is equal *a priori*. Therefore, the expected final value of CRL stock is the average final value over all states, which is \$24.

The objective of a trader (or "Market Taker") is to achieve the highest possible trading gains during the simulation. Because the objective is to maximize return, this simulation is a zero sum game, meaning that a good trade for one party is necessarily a bad trade for the other party.

Traders are also directly interacting with market, similar to the specialist system on the New York Stock Exchange. The specialist has knowledge of every trader's orders and sets the bid and ask price based on these orders.

During the course of the simulation, traders receive *private information* very soon after the trading period begins. For example, a trader may get a message during year 1 that reads "CRL has renegotiated some of its costs (not Z1)." If the actual earnings for a period will be "x", then half the traders will be told "not y" and the other half will be told "not z." If traders were to speak directly, they may be able to deduce with certainty the actual earnings. At the end of each period, the market will close before the earnings announcement is made. This announcement may read "CRL Reports earnings of \$24 per share in year 1 (Z1)." Everyone will receive identical earning announcements.

Every trader begins with an identical portfolio of \$10,000 and 400 shares of stock. For this case, there are no restrictions on borrowing or short selling. Traders are encouraged to take large positions but are warned that large market orders may significantly impact the price of CRL.

4.1.2 Recommended Strategies for the CRL Case

The first insight of this case is to find the conditional expected value of the stock given the known information. For example, knowing that the earnings in period one will not be zero should obviously affect the valuation of the stock. It is important to make explicit what the conditional values should be given

different information and states of nature. Table 6 shows all of possible conditional expected values.

Information	Value
None	\$24

Information	Value
Not X1	\$34
Not Y1	\$24
Not Z1	\$14

Information	Value
X1	\$4
Y1	\$24
Z1	\$44

Information	Value	Information	Value	Information	Value
X1, Not X2	\$6	Y1, Not X2	\$30	Z1, Not X2	\$48
X1, Not Y2	\$6	Y1, Not Y2	\$24	Z1, Not Y2	\$42
X1, Not Z2	\$0	Y1, Not Z2	\$18	Z1, Not Z2	\$42

Information	Value	Information	Value	Information	Value
X1, X2	\$0	Y1, X2	\$12	Z1, X2	\$36
X1, Y2	\$0	Y1, Y2	\$24	Z1, Y2	\$48
X1, Z2	\$12	Y1, Z2	\$36	Z1, Z2	\$48

Table 6: All possible conditional expected values of CRL. Given certain information, our expectations of CRL's value changes.

These conditional expected values provide guidelines for trading ranges. However, a clever trader should always be able to infer with certainty what the upcoming earnings announcement will be.

For example, assume that a trader receives the information "Not Y1." Then that trader will value the stock at \$24. However, the trader also has access to a graph of transaction prices. The trader may notice that other traders are willing to buy for as much as \$34. Observing a tendency for the price to go above

\$24, a trader can infer that some other traders must have been told "Not X1." The clever trader should then know that CRL will earn \$24 (or "Z1") in period one and submit large-quantity limit buys at $\$43 \frac{15}{16}$ and large-quantity limit sells at $\$44 \frac{1}{16}$, essentially making a tight market at \$44 until the first public earnings announcement.

Now assume the clever deductive trader was correct, and the announcement was "Z1." The trader might receive information revealing "Not Y2" early in period two. The trader may observe that the final price is either \$42 if X2, \$48 if Y2, or \$48 if Z2. So any other market participant who might learn "Not X2" clearly knows (with very little analysis) that the terminal price will be \$48. Hence, the clever trader may observe that nobody is making a tight market at \$48 and therefore nobody received "Not X2." Since nobody received "Not X2," then "X2" will be the upcoming earnings announcement and the clever trader should make a tight market at \$36 to maximize profits.

By leveraging private information and the information contained in recent transactions, it is almost always possible for a trader to deduce the upcoming earnings announcement. To see why there is little room for exception to this rule, consider an extremely devious and clever trader who swiftly makes a tight market at a "false" price in order to fool all other market participants. For example, assume that the devious trader learns "Not X1" at the beginning of period one. This trader might submit very large limit orders to sell at $\$4 \frac{1}{4}$, essentially taking a loss on these trades. The devious trader might hope other market participants might zero in on that \$4, assuming that it is the efficient

price, only to then start making a market at a higher price and hoping that the other teams dismiss his actions at those of a naïve and foolish trader. This devious strategy will not work, however, because he is not the only trader to have received not X1, and that group of traders will choose to profit enormously from the devious trader's plan.

In other words, the price may fail to converge to any value, but it will never converge to the wrong value unless traders are irrational or collusive.

4.1.3 Experimental Results of the CRL Case

This section examines the quality of the market and cites factors and conditions that might affect the market quality.

4.1.3.1 Convergence to the Efficient Price

The price path taken during the CRL simulation in the Sloan Trading Laboratory was indeed Z1 and X2. Price convergence in period one was marginal, and convergence in period two was even less evident.

In the price graph of CRL, we can see that that the price started at \$24, but almost immediately increased by \$10 (due to traders who were informed "Not X1"). This could have been a signal to the other half of the traders (who were informed "Not Y1") that the true earnings would be "Z1," and hence the trading price should be \$44. Note a very vague and noisy trend that the price of CRL is

rising during period one. Perhaps given more time, the price would have converged to \$44.

In period two, the price should have converged to \$36. It appears that the market failed in period two. At the end of the simulation, most of the teams were long CRL. Only one team was short CRL, and some teams had zeroed their holdings. It appears that those traders who were informed "Not Y2" noted the price floating above \$42 and took the position that the final price would be on the high side of their conditional expected value. The other half of the traders (who were told "Not Z2") were not in a strong position to infer the settlement price due to the ambiguity of the data. Hence, the session closed with the price of CRL far higher than its efficient price of \$36, and only one team held a short position to benefit from this.

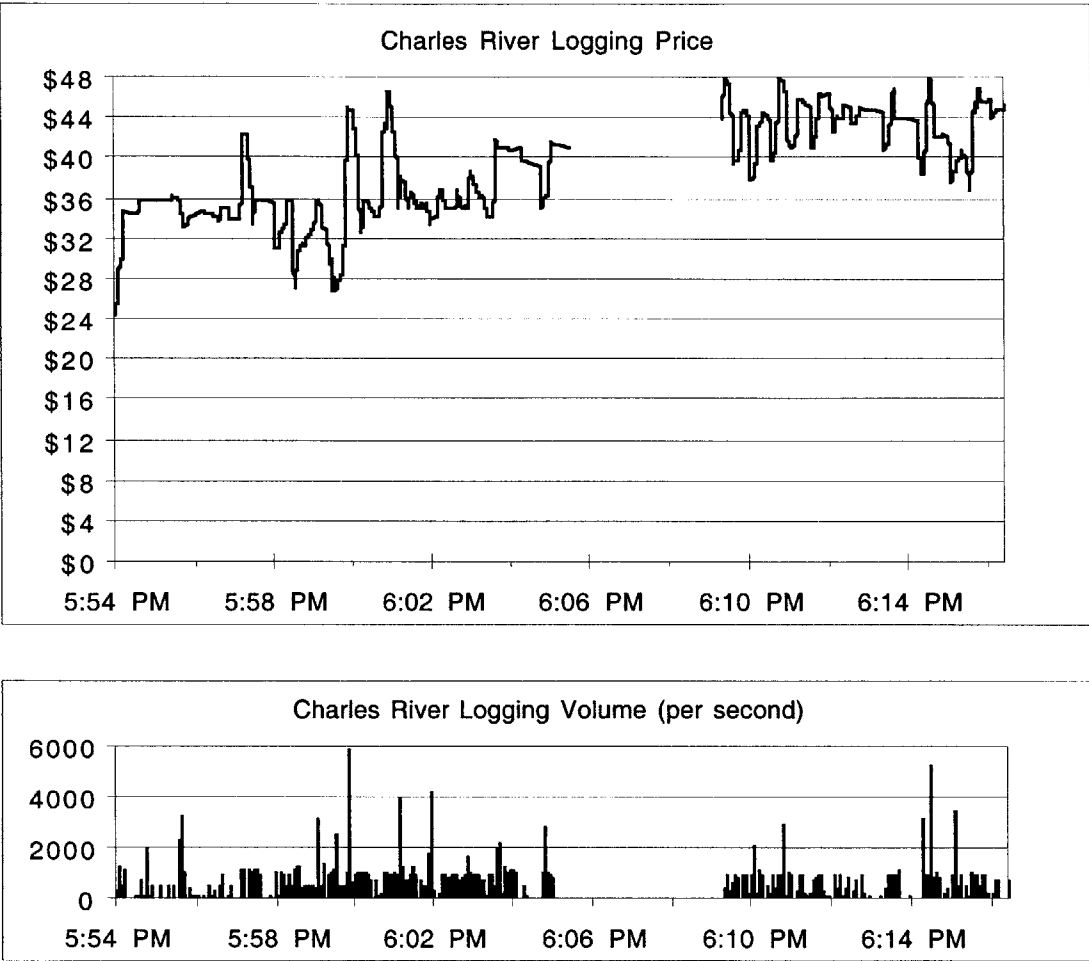


Figure 18: The price path and volume of CRL stock during simulation. The gap in the price graph represents the trading halt between periods one and two. Note the hyperactive volatility in the price of CRL. Also note that the volume often approached or exceeded 4,400 shares transacted in 1 second. There were a total of 4,400 shares outstanding of CRL between all traders collectively.

The stock should have converged to \$44 in period one, and we see a late trend in that direction. The stock should have converged to \$36 in period two, although the efficiency of the market has been completely obscured by large chaotic market orders.

4.1.3.2 Large Market Orders

During this simulation, note the number of large swings in price throughout the simulation. These swings are not due to new information (because there was no new information other than at the start of each period). These swings are due to large market orders submitted by traders for 5000 shares (the maximum allowed by the software for a single order). The electronic market maker (EMM) divided these orders into 50 smaller orders of 100 shares each, and was kind in executing each sub-order with price continuity.

These market orders caused price swings that obscured the information contained in the price history, information that is critical to price convergence for this simulation. Interviews with the traders confirm that the traders were confused.

The large market orders seem to reflect a deep faith in the benevolence of market makers. Perhaps the EMM should have executed the first 100 shares of a 5000 share market order at a reasonable price, and execute the remaining 4900 at a harsher price. Such a draconian practice may have hurt the trader in the short run, but it would potentially prevent traders from submitting these large market orders that, as we can see, significantly damage the quality of the market.

Students seem to have ignored the large costs associated with market impact. That there were no fixed transaction costs (commissions) also encouraged heavy trading.

4.1.3.3 Trading Laboratory Psychology and Panic Trading

There seems to be a psychology in the Sloan Trading Laboratory that trading should be a frantic activity that may have been accentuated by a number of factors:

- Students see other students trading in large volume, and assume the cost of sitting idle is higher than the cost of doing something.
- The continuous auction lends a feeling that time is running out, inducing a feeling of panic.
- The students shared a faith that the market maker would give a fair price.
- The reward structure is that the winner gets 20 points on the final, while every other trader neither gains nor loses points. Hence, traders have incentive to take as much risk as possible in order to maximize the probability of the largest gain (without particular regard to maximizing the expected value of the gain).
- Students are not trading with real money.

In real markets, a sense of confusion and time pressure might lead to panic selling. In this simulation, rapid trading (both buying and selling) occurred. In about 20 minutes, there were 3,124 transactions accounting for 366,350 shares changing hands. Only 4,400 shares of CRL existed in the market.

4.1.3.4 Call Auction as an Alternative to Continuous Agency Auction

In a call auction, traders tend to submit more limit orders than market orders. Submitting a limit order requires a trader to determine a price, which incurs more analysis than does madly clicking a “huge market buy!” and letting the market maker work out the details.

That is, it seems that the existence of a quote (in addition to the high volatility) encouraged the students to speculate on price the next price swing instead of on the fundamental value of the stock. Popular strategies included momentum trading (buying at market on upswings and selling at market on downswings), and also contrarian trading (buying at market on downswings and selling at market on upswings).

Perhaps when a market becomes overactive, a circuit breaker mechanism could switch the trading structure to a call auction executed at 15 minute intervals until the clearing price stabilizes. This would allow traders a longer time to consider the value of the security.

4.1.4 Conclusion to the CRL Case

After the first period, many groups claimed that they ignored the information regarding earnings. This is perhaps due to their observations during period one that the market was chaotic, and therefore a chaotic strategy would yield the best results. However, it is clear that during period one there was at

least a ten-minute window during which a team could have taken the game by deducing that \$44 is the efficient value. It is important to remember, however, that the students engaging in the simulation were using the trading system for the first time and were faced with understanding the software, managing group dynamics of relatively large teams (with six students in each trading group), and understanding the subtleties of the case in a distracting environment.

The simulation demonstrated that the online exchange works correctly, and that markets are sometimes less than efficient.

4.2 Five Day Asset Management Simulation

The second trading session lasted 5 days, nearly 24 hours a day, from Monday at 9:30 am until Friday at 4:00 pm. There were about 98 1/2 hours of trading. The same groups of students who participated in the CRL simulation participated in this simulation. However, different groups were given different performance metrics.

The students traded three stocks, and each of these stocks would be automatically liquidated from their portfolio on Friday at 4:00 using the stocks' NYSE/NASDAQ Friday closing price. Therefore, these stocks were tied to the NY markets at the end of the simulation but were free to deviate prior to that time.

The reasons a stock might deviate from its New York price are, of course, due to supply and demand in the local market. The equilibrium price could deviate because the risk preferences in the AFM exchange are not identical to those of the real market.

4.2.1 Specifications of the Asset Management Case

This case focused on the roles of asset managers in financial markets. For 5 trading days, teams took on the role of one of 3 types of asset management institutions: that of a Conventional Mutual fund, an Index fund, a Hedge fund, and a Pension fund. Each of these fund types is graded by a distinct performance metric and is subject to different restrictions on margin buying and short selling.

The exchange listed 3 stocks: Intel (INTC), McDonalds (MCD), and Merck (MRK). Hence, portfolios consisted only of these 3 stocks and cash. Every fund initially held an identical portfolio.

The AFM exchange will open the stock at the stock's latest price on the NYSE or NASDAQ. At the end of the final trading day, each portfolio's net value will be calculated using the final day's closing price on the NYSE or NASDAQ.

Arbitrage opportunities would suggest that the price of these securities would equal the "real world" price during the entire simulation if arbitrage were allowed. However, the AFM exchange is disconnected from the larger exchanges during the simulation.

The market was open almost continuously from Monday morning until Friday afternoon. The exact hours were the following:

- Monday 9:30am to Monday 5:00pm
- Monday 6:00pm to Tuesday 5:00pm
- Tuesday 6:00pm to Wednesday 5:00pm
- Wednesday 6:00pm to Thursday 5:00pm
- Thursday 6:00pm to Friday 4:00pm

The market was to be closed for about an hour each day in case there was a need for unforeseen maintenance. On some days, the market was reopened a few minutes before 5:00 pm.

Each fund type pursued a different objective and was subject to different constraints. The performance measures were the following:

- *HedgeFundPerformance* = $\max(0, \text{return})$ where the return used is the cumulative return from the Monday opening to the Friday close.

- *ActiveMutualFundPerformance* = $\frac{\overline{\text{return}}}{\sigma_{\text{return}}}$ where the five returns are the daily returns to each day's close on the AFM exchange from the most recent open on the AFM exchange. (This is the Sharpe ratio with a risk free rate of 0.00%).

- *IndexFundPerformance* = $-\sum_{\text{Each Day}} (\text{return} - S \& P500\text{return})^2$ where the returns are the five cumulative returns at each close, each measured from the Monday open.

The hedge fund manager is subject to these restrictions:

- The leverage must be less than or equal to 2:1. This means that if the portfolio is worth \$100,000, the manager can go long a maximum of \$200,000 worth of stock, or alternatively go long \$125,000 and short \$75,000... so long as the positions add up to less than 2 times market value.
- The hedge fund can short or buy long.

The active mutual fund manager and index fund manager manage long-only equity funds, and they cannot short stocks.

Every fund began with identical portfolios of \$15,000 plus 300 shares of Intel, 200 shares of Merck, and 150 shares of McDonald's. The winner in each of the 3 fund categories received 20 bonus points on their final exam for the Financial Engineering course.

4.2.2 Recommended Strategies for the Asset Management Case

Successful strategies for this case cannot be as explicitly defined as they could be for the CRL case. However, there are a few core points to consider when forming a strategy.

Hedge funds are maximizing returns. Performing leveraged statistical "arbitrage" is a good way to make money when prices on the AFM exchange fall out of line with those on the real exchange. In order to maximize risk (which is

clearly advantageous given the reward structure), holding cash is perhaps not optimal. Furthermore, holding some stocks long and some short will cause the systematic risk to partially cancel and therefore a risk maximizer would be either all short or all long. If I were participating, I would have strongly considered a strategy to hold zero cash and to short the most overpriced stock (by percentage of value) at all times.

Active mutual funds are measured by a peculiar metric – the Sharpe ratio with a risk free rate of zero. The Sharpe ratio is meant to measure risk-adjusted returns over a long period of time. A winning strategy would have been to make exactly \$1 every day. This would result in a high Sharpe ratio. Consider the following calculation where v is the initial value, and c is profit on the first day. Assume $v, c > 0$ and $v \gg c$:

	NAV	Daily Return
Monday Open	v	-
Monday Close	$v + c$	c / v
Tuesday Close	$v + 2c$	near c / v
Wednesday Close	$v + 3c$	near c / v
Thursday Close	$v + 4c$	near c / v
Friday Close	$v + 5c$	near c / v

Table 7: This table shows how the Sharpe measure could be exploited when measured over a short period of time by making a very small, constant profit each day. Although no fund managers recognized this opportunity, subsequent simulations should perhaps use an alternative method of risk-adjusting mutual fund portfolios. Perhaps using an appraisal ratio or another alternative would be more appropriate.

In this case, the average return is very close to $\frac{c}{5v}$ and the standard deviation of the returns is very close to zero. Hence, the Sharpe ratio would be very high. In our case, v was about \$54,000. If c were \$10, the ratio would be over 23,000,000,

and it approaches infinity as the daily profit c gets *smaller*. The winning group scored a ratio of about 1.39 because no active fund managers exploited this flaw, although the winners did practice a strategy that focused on a low standard deviation. They won their division easily with a 3% return for the week, while the second place group earned a 21% return for the week but by taking considerably more risk.

Matching the S&P proved to be the most difficult. The obvious strategy is to run regressions that would reveal the optimal mix of these three stocks that statistically match the S&P 500 index. Consider the following regression as an example:

$$return_{S\&P} = a \cdot return_{INTC} + b \cdot return_{MRK} + c \cdot return_{MCD} + \varepsilon$$

Using daily returns to find the a , b , and c that minimize the variance of the epsilon would translate into a static trading strategy for matching the S&P 500. However, it turns out that the optimal values of a , b , and c are *highly* dependent on the starting and ending dates of the stock data used such that the results from this method would not yield a meaningful strategy. Furthermore, this approach would assume that the AFM prices would follow price distributions similar to those that their real counterparts have in the past. It would seem that the most effective strategy for matching the S&P 500 for five days is to watch the S&P every day, attempting to make money, however possible, when the S&P rises and attempting to lose money when the S&P 500 falls. The most successful

indexing group followed this strategy and outperformed other groups who attempted to create portfolios based on beta-weighting or other regression methodologies.

4.2.3 Results of the Asset Management Simulation

The simulation yielded low trading volume. The prices tended to track those in the real markets, with the largest deviations occurring in the middle of the week, pricing the AFM stocks higher than the stocks in the NYSE.

4.2.3.1 Overall Price Parity between the AFM Prices and the NY Prices

One of the most interesting points in the asset management simulation is whether the prices in the AFM exchange would correspond to those in the real markets. A reasonable hypothesis circulated before the simulation is that the price would be very closely related on Monday and Friday but not necessarily related on Tuesday, Wednesday, or Thursday. This is based on a conjecture that, since the AFM prices would tend to converge near the end to the NY prices, the deviation would follow a “Brownian Bridge” process. In fact, the prices in the two markets were reasonably close, and did depart the most on Wednesday.

To compare, it is useful to construct an index that represents the total equity in the AFM exchange. Because all traders started with 300 shares of Intel, 200 shares of Merck, and 150 shares of McDonald’s, the index will use these proportions as a measure of the total market capitalization in the AFM exchange:

$$AFMIndex = 300 \cdot P_{INTC} + 200 \cdot P_{MRK} + 150 \cdot P_{MCD}.$$

From the two tables, showing this AFM index calculated using actual AFM prices, and also with prices from the NYSE/NASDAQ, it is clear that the cumulative returns are closest on Friday. Furthermore, the prices in the AFM exchange deviated the most on Wednesday. Not the difference in cumulative returns on Wednesday is 6.55% to 2.98% with the AFM stocks having higher prices.

Using AFM Prices	AFM Index	Cumm AFM Return
Monday Open	38703	-
Monday Close	39119	1.07%
Tuesday Close	40144	3.72%
Wednesday Close	41238	6.55%
Thursday Close	39125	1.09%
Friday Close	39063	0.93%

Using NY Prices	AFM Index	Cumm AFM Return
Monday Open	38703	-
Monday Close	39413	1.83%
Tuesday Close	38803	0.26%
Wednesday Close	39856	2.98%
Thursday Close	38903	0.52%
Friday Close	39181	1.24%

Table 8: This table shows the price premium for shares on the AFM exchange versus the same shares on the NYSE/NASDAQ. The prices are generally higher in the AFM market. This premium is most pronounced on Tuesday and Wednesday, and the prices nearly converge by the Friday close.

It is an open question whether it is chance that the AFM prices were generally higher than the NY prices. One explanation of the price premium could be that the investors were, on the whole, seeking to maximize risk. Clearly the hedge fund managers were in this position because they would gain no points by earning the second highest return. The index managers, had incentive to match

an index that is fully invested in equity, but the initial portfolio is only 72% invested in equities (with the remaining 28% in cash) A theory is that the indexers and hedge funds both have an incentive to increase their holdings of equities above the level of their initial portfolio.

Analysis of Betas also supports that indexers and hedge funds would want to increase holdings of equity (relative to the initial portfolio). The Beta of the initial portfolio, including the \$15,000 in cash, is 0.81.² The S&P 500 has a Beta of 1.0, and hence the indexers and hedge fund traders would need to buy more equity in order to reach a Beta of 1.0. This analysis supports the theory high demand for market risk by the index funds that would cause a bias toward high stock prices.

These two fund types may systematically create a bias for higher prices due to risk-return preferences that are different from those observed in the real markets.

² Using Betas for Intel, Merck, and McDonald's of 1.36, 0.89, and 0.94, respectively, as published by Yahoo! Finance.

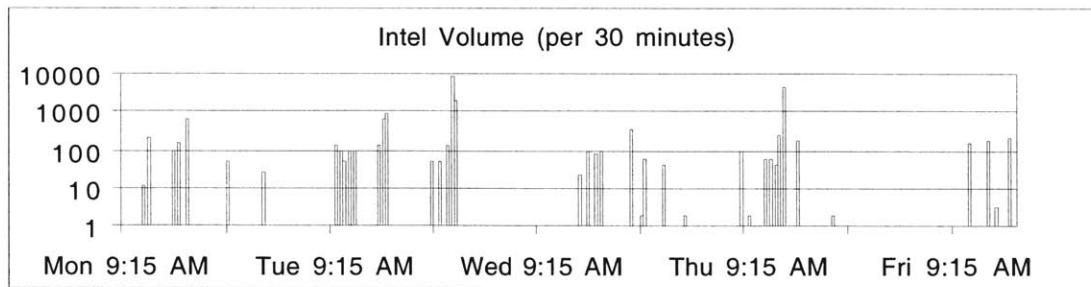
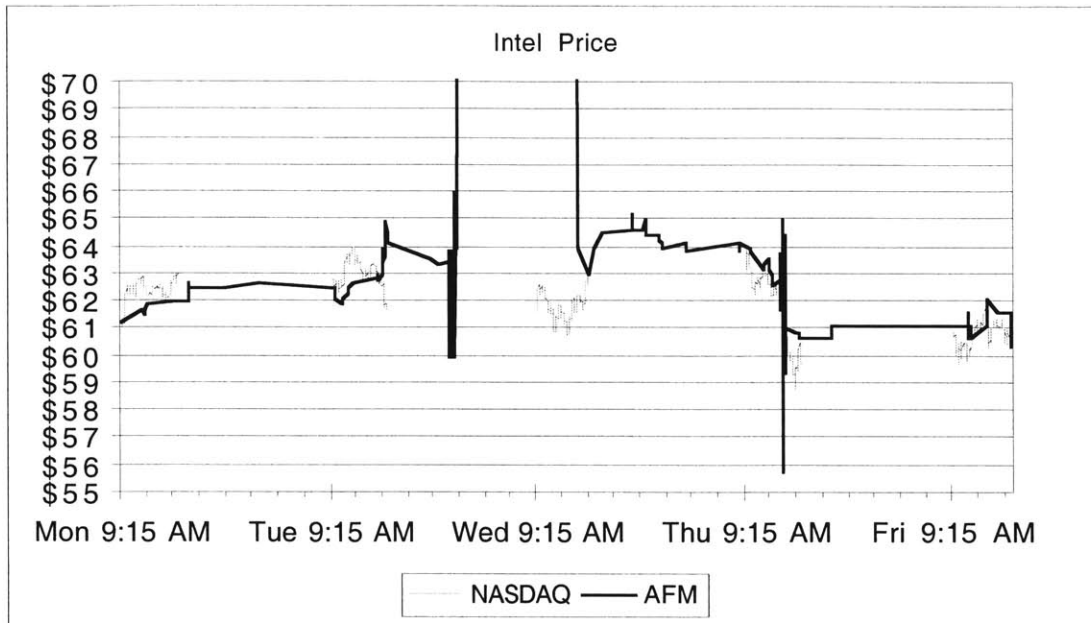


Figure 19: This chart presents the price of Intel stock in both the AFM exchange and the NASDAQ, and volume in the AFM exchange. The Price convergence is generally evident. Note the simultaneous \$4 drop on Thursday afternoon. From late Tuesday until Wednesday afternoon, the EMM is turned off. Market quality declines and a reasonable price quote is not available.

Volume was very low and inconsistent. Notice that typical volume is under 100 shares per 30 minutes, yet with spikes up to 10000 shares per 30 minutes.

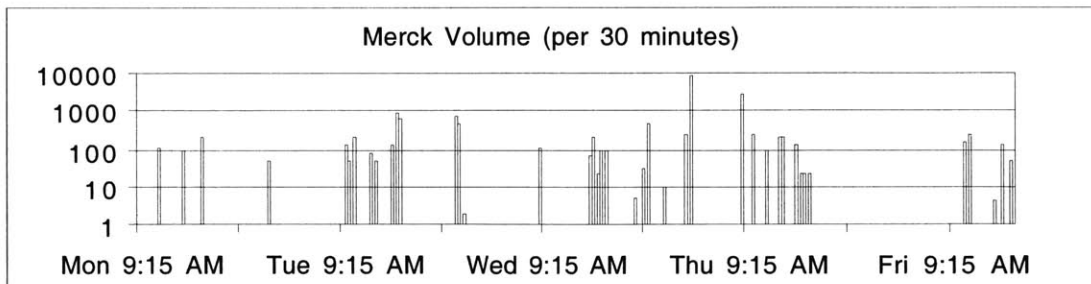
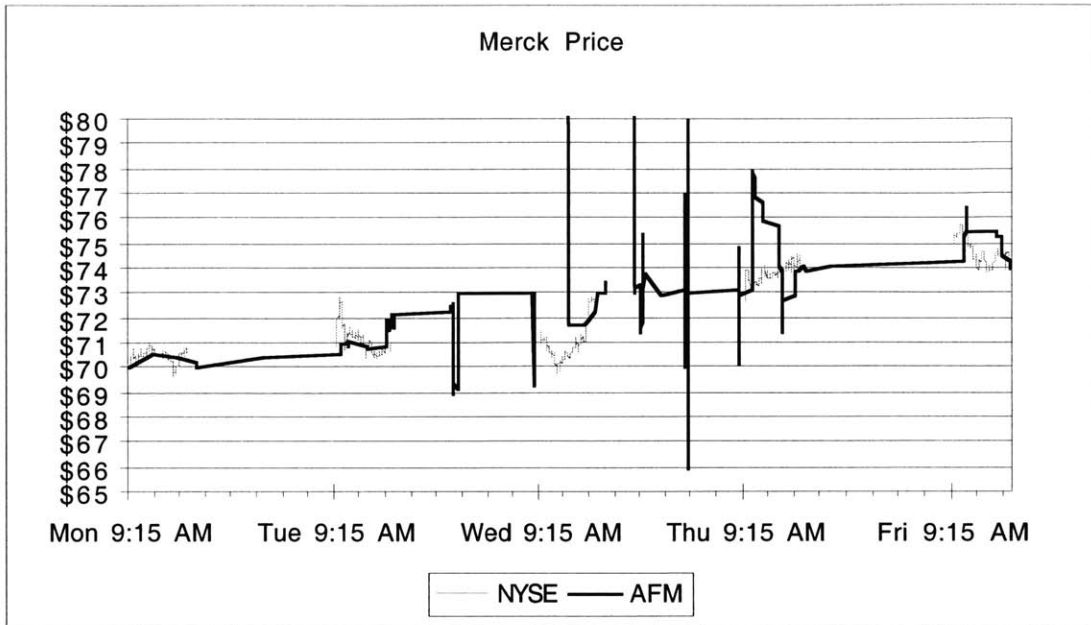


Figure 20: This chart presents the price of Merck stock in both the AFM exchange and the NYSE, and volume in the AFM exchange

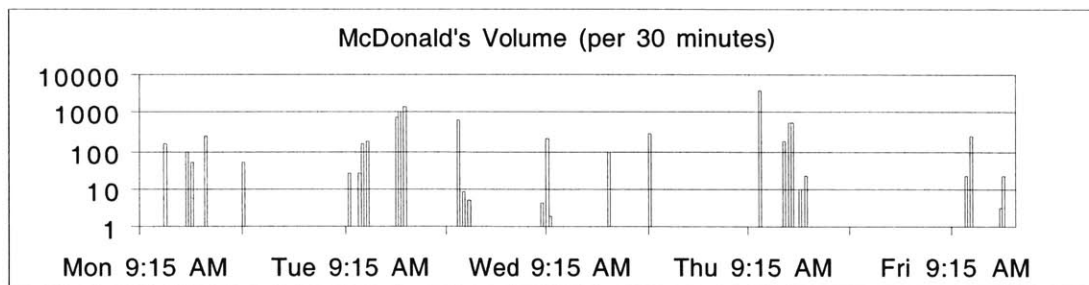
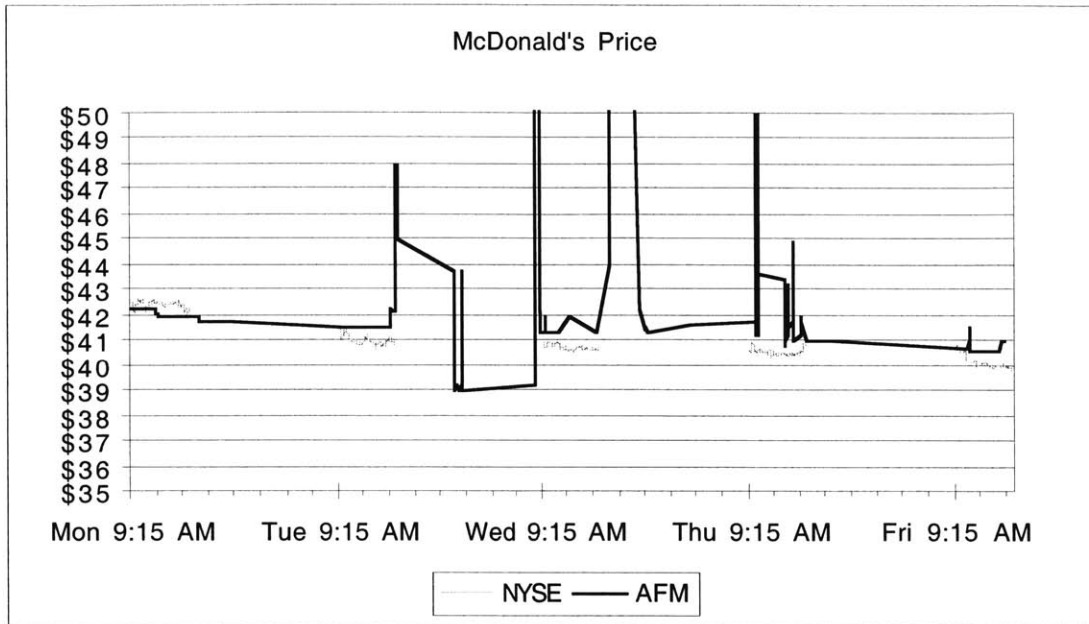


Figure 21: This chart presents the price of McDonald's stock in both the AFM exchange and the NYSE, and volume in the AFM exchange. McDonald's stock was consistently overvalued in the AFM market when compared to the NYSE.

4.2.3.2 Low Trading Volume

Trading volume was extremely low during the simulation. Often, we observed hours without any transactions. At times, traders would be logged in and watching the market without trading.

This is in sharp contrast to the CRL simulation (which was conducted using the same teams as participants). Perhaps we are seeing the negative of the “Trading Laboratory” effect. That is, in the Trading Laboratory, students seem eager to reverse from very bullish to very bearish (and then back to bullish) within a few seconds. During the asset management simulation, students logged in remotely (from home or from an on-campus laboratory). The simulation lacked the feeling of a trading “crowd” because the price graph was generally flat for hours at a time.

4.2.3.3 Effects of the Market Maker

Due to the general inactivity of the market, traders had an opportunity to test the skills of the electronic market maker (EMM). A few traders did learn to take money from the market maker and earned solid returns by this method.

On Tuesday, the EMM was removed from the market and a continuous auctioneer was installed. This took many traders by surprise. At least one group, a mutual fund, attempted to play the role of market maker by leaving standing limit bids and offers. This fund found itself in a short position – which is illegal

for this fund type – and ultimately paid excessively for stock to cover their short position. During the downtime on Tuesday, the EMM was tuned to prevent it from losing money to persistent traders. The EMM was put online again on Wednesday.

During the period when the EMM was not in the market, the prices varied dramatically and it was impossible to find a satisfying quote for traders wishing to learn the current price. Trading volume fell dramatically. It is reasonable to conclude that the EMM increased the general quality of the market.

4.2.4 Conclusion and Potential Amendments to the Asset Management Case

The Asset Management Case could be amended in a few ways. Increasing market activity would make the simulation more interesting.

Using volatile stocks may increase interest. For example, AOL, Microsoft, and Dell could be traded. Instead of matching the S&P 500, perhaps the Goldman Sachs Technology Index could be substituted. This would increase price movement in the market, encouraging additional trading. Furthermore, it may be more feasible to match the GSTI index with three stocks than to match the S&P 500 with three stocks.

Allowing teams to choose their fund type may also increase interest. Given that there are three fund types, a simple iterated market mechanism could be used to allocate teams to fund types. That is, let all teams choose their fund type.

Reveal to all teams how many are in each group, and allow teams to change to a different fund type. Repeat until no teams desire to change fund types (or until the repetition becomes tiresome). This may allow teams to be allocated to the fund type that they have the best strategy for. The market force equating demand for the three fund types is the 20 points that go to the winner of each type.

Also, restricting trading hours may bring more participants together simultaneously.

Running the case during the first week of the semester, not the last, would allow students more freedom to spend time on the simulation.

Finally, researching a more robust measure of risk adjusting the returns for the active mutual fund managers would add significant value to the simulation. Some candidates for this measure include the Treynor Ratio, Jensen's measure, and the appraisal ratio.

4.3 Conclusion

The data indicate that the electronic market makers can either improve or degrade market quality, depending on the psychology of the traders and the velocity of the market, and perhaps other factors.

In neither simulation did the market maker have access to any external information (news items or real-world stock prices) that would help determine

the price. Regardless, the market maker did make a significant profit in the 20-minute simulation – in fact, a significantly larger profit than even the best trading group. Informed traders should generally outperform an uninformed market maker. This is partially explained by traders ignoring information, especially in the second period. The large swings in price essentially represented “noise trading,” and market makers can capture the spread in noisy markets without suffering losses from traders who are acting on information.

In the 5-day simulations, the market makers lost money, as expected. This is for two reasons. First, the traders had excellent information regarding the fair price of the stock (whereas the market maker had none other than the order flow). A very common strategy for traders was to “arbitrage”³ between the markets when prices were out of line. This works well, because a stock price is a reasonable estimator of the expected value of the price in a few days. Due to the information advantage, the vast majority of the trades with the market maker were losing trades for the market maker. This accounts for much of the market maker’s loss. As previously mentioned, a speculative module of the market maker could reset inventory targets dynamically based on the information in the order flow. However, such a system would not be useful in a very thin market because the information content in the order flow is sparse. Making a profit as a market maker when the market is thin and the traders have overwhelming informational advantage (as they do in this 5-day simulation) is quite difficult.

³ Arbitrage is in quotations because this is not true arbitrage. There is significant risk to this strategy.

The second reason the market maker lost money in the 5-day session is that at least one group learned how to beat the EMM head to head by a closed cycle of buying and selling. While the profit gained from this exploitation was not the most significant source of loss, it is clearly a primary concern. Building an electronic market maker that cannot be “gamed” by traders is a significant challenge. Humans have developed computers that are good at chess and other games. A robust market maker may need to incorporate intelligence into defensive strategies. Of course, if the spread is wide enough, all such opportunities vanish. The goal is to minimize the opportunity for exploitation without sacrificing narrow spreads.

Finally, testing market structures and market makers is inherently challenging. Market structures and market making algorithms can only be verified by the test of time. These systems may work 90% or 99% of the time, but trusting a new algorithm or trading structure to stabilize the market in the most difficult circumstances (periods of high volatility) is at some level a leap of faith. Even if human market makers receive high compensation for arguably mechanical tasks, this may be preferable to trusting a computer algorithm. The costs and perceived costs of low-probability, high-loss trading session (a “blowout” like those that destroy careers and sometimes institutions) may restrain the acceptance of electronic market making until it is determined that human market makers are as likely to incur “blowouts” as are electronic market makers. The argument in favor of automated market makers is that they can run 24 hours a day and they can be replicated cheaply enabling the use of markets in novel ways.

Chapter 5

Conclusions and Directions for Further Research

The key results of this thesis are the configurable World Wide Web based exchange, the technical insights and recommendations gained from implementation of the exchange, the insights gained by testing the automated market maker against human traders, the two trading simulation cases, and the insights from the data from running the two simulations.

The WWW based market worked well in simulated testing and in practice. There are additional features and architectural changes that could improve the market, as discussed in the conclusion of Chapter 4. In its current state, the market is useful as a research platform for the EMM and for developing “robot” traders that may use learning algorithms borrowed from Artificial Intelligence.

A possibility is that this implementation or the next major version of this market will be a platform for a laboratory course in the MIT Sloan School. Such a course would likely attract considerable interest from students who are eager to learn more about careers on Wall Street or who are simply interested in learning more about the mechanics of trading.

One interesting note is that only one of the eleven teams was short the CRL stock at the end of the simulation. The other groups were all very long (if they were bullish) or neutral (if they were bearish). Of course, they should have all been bearish because the stock was \$9 1/4 (or 25%) overvalued. This may be an indication of a psychology among a young generation of day-traders that shorting overvalued stocks is often not a good idea.

The electronic market maker demonstrated a core ability to set reasonably good quotes, and to improve the market quality in some cases and perhaps degrade the market quality in others. Additional research is needed to dynamically adjust the inventory target and to make it hard to take profit from the market maker in one-on-one trading.

A common argument against moving the major markets to a distributed electronic platform with no trading "pits" is that information critical to the efficiency of markets is conveyed on the trading floor. In the CRL simulation, information was more or less ignored (which is an oddity in itself). The irony is that there was a reasonably straightforward strategic insight that would have won the game for any team that pursued it. Perhaps a chaotic strategy is extremely contagious when traders are gathered in a crowd. Distancing the traders from each other (in different rooms) may have an effect on the character of the market. Certainly something is lost when traders are not centrally gathered. Whether the information conveyed in a trading crowd is stabilizing, destabilizing, or irrelevant is not obvious. This raises a critical question. As we improve technology, it will become possible to trade securities exclusively

through computers and the primary question will be “Do we want electronic markets?”

A general direction of further research on electronic markets should aim to determine if electronic, distributed markets are superior to traditional, centralized markets by the measures of price efficiency, fairness, transaction cost, and overall market quality. Further, it is important to determine what types of auctions or trading structures are suitable for distributed markets, how to govern electronic exchanges in a way that promotes the desirable qualities of markets, how to build these exchanges, and how to best make the transition.

References

Amihud, Yakov, and Haum Mendelson. 1980. "Dealership market: Market making with inventory." *Journal of Financial Economics*. 8:31-53.

Amihud, Yakov, Thomas S.Y. Ho, and Robert A. Schwartz. 1985. *Market Making and the Changing Structure of the Securities Industries*. D.C. Heath and Company.

Bodie, Zvi, Alex Kane, and Alan J. Marcus. 1996. *Investments*. Irwin.

Brealey, Richard A. and Stewart C. Myers. 1996. *Principles of Corporate Finance*. McGraw Hill.

Forsythe, Robert, Forrest Nelson, George Neumann, and Jack Wright. 1991. "The Iowa Presidential Stock Market: A Field Experiment." *Research in Experimental Economics* 4:1-43

Hasbrouck, Joel, George Sofianos, and Deborah Sosebee. 1993. "New York Stock Exchange Systems and Trading Procedures." NYSE Working Paper.

Ho, Thomas, and Hans R. Stoll. 1983. "The dynamics of dealer markets under competition." *Journal of Finance*.

Huang, Roger D. and Jans R. Stoll. 1996. "Dealer Versus Auction Markets: A Paired Comparison of Execution Costs on NASDAQ and the NYSE." *Journal of Financial Economics* 41:313-357.

Jensen, Michael C. 1969. "Risk, the pricing of capital assets, and the evaluation of investment portfolios." *Journal of Business*.

Neal, Robert. 1992. "A Comparison of Transaction Costs between Competitive Market Maker and Specialist Market Structures." *Journal of Business* 65:317-334.

"NYSE Floor Official Manual." NYSE, June 1998.

O'Brien, John. 1991b. "Dynamic stock markets with multiple assets: An experimental analysis." *Journal of Finance* 46:1811-1838

O'Hara, M. and G. Oldfield. 1986. "The microeconomics of market making." *Journal of Financial and Quantitative Analysis*.

Plott, C. R. and Sunder, S. 1998. "Rational expectations and the aggregation of diverse information in laboratory settings." *Econometrica* 56:1085-1118

Schwartz, Robert A. 1991. *Reshaping the Equity Market: A Guide for the 1990s*. Harper Business.

Sharpe, William F. 1966. "Mutual fund performance." *Journal of Business*. 39.

Treynor, Jack L. 1966. "How to rate management investment funds." *Harvard Business Review*. 43.

Arizona Stock Exchange URL: <http://www.azx.com>

CNN Financial Network URL: <http://cnfn.com>

Estats URL: <http://etats.com>

Financial Trading System URL: <http://www.ftswb.com>

Hollywood Stock Exchange URL: <http://www.hsx.com>

World Sports Exchange URL: <http://www.wsex.com>

8071-01