

Novel Order Book Rules to Promote Trading System Price Stability

Lingcen Huang

University College London

MSc Financial Computing Project 2010/2011

Supervisors: Antoaneta Serguieva, Chris Clack

Submission date: 14 September 2011

This report is submitted as part requirement for the MSc Degree in Financial Computing at University College London. It is the result of my own work except where explicitly indicated in the text. The report may be freely copied and distributed provided the source is explicitly acknowledged but the real data used is confidential.

Abstract

This project focuses on designing an agent-based order book simulation model that applies novel rules to improve market price stability. The model is developed by Java language. To implement this model, Mason software and some java libraries are also applied. In order to compare the difference of price fluctuation when apply and not apply those order book rules, we develop a simplest model firstly. This model has an order book without any rules and all traders have same behaviors. And then, we improve this model by change simplest order book to double auction order book. The traders can be patient players or impatient players. The patient players place limit orders, which include the price and quantity they want to sell or buy. The impatient players place market orders, which execute immediately at current best price. The output graphics including price plot, volatility plot and order book for these two scenarios are compared and analyzed.

The report starts at introducing the motivations of this topic and using multi-agent simulation. And then, it defines the relevant concepts of order book. The trading rules and trading strategies are also descried. The report records the design and implement simulation model by each step. The simulation results are analyzed carefully. Finally, the report ends with description of finding and conclusion.

Acknowledgements

To Yao for the love and support.

To Toni and Chris for the guidance through-out this project.

To my parents for spiritual and financial support.

Content

Chapter 1 Introduction	3
1.1 Motivation.....	3
1.1.1 Order Book Motivation.....	3
1.1.2 Agent-based Modeling Motivation.....	3
1.2 Aim and Objectives.....	5
1.3 Project Report Overview.....	6
Chapter 2 Background and Research Information	7
2.1 Order Book.....	7
2.1.1 Order Book Definition	7
2.1.2 Limit Order Book and Model	8
2.2 Order Book Trading Rules and Strategies	10
2.2.1 Order Book Trading Rules.....	10
2.2.2 Limit Order Book Strategies	12
2.3 Agent-based Modeling	14
Chapter 3 Order Book System Design	16
3.1 Guidance from Related Research.....	16
3.2 Use Cases and User Requirements Analysis.....	17
3.2.1 Use Cases Analysis	17
3.2.2 User Requirements Analysis.....	19
3.3 System Requirements	20
3.4 The Order Book Model Design.....	21
Chapter 4 Implementation and Testing.....	26
4.1 Software Application Used.....	26
4.2 System Implementation	27
4.2.1 Description of Java Class.....	28
4.2.2 Description of Simulation Model.....	31
4.3 Testing.....	32
4.3.1 Testing Process	32
4.3.2 Scope, Approach and Results	33
Chapter 5 Results and Analysis	34
5.1 Simulation Plan	34
5.2 Scenario 1 – Cont Book Model	35
5.3 Scenario 2 – Double Auction Book Model.....	38
5.4 Analysis of Results	42
Chapter 6 Conclusions and Future Work	43
6.1 Project Summary	43
6.2 Conclusion	43
6.3 Further Work.....	44
Appendix A: System Manual	45
Appendix B: User Manual	46
Appendix C: Project Plan	53
Appendix D: Source Code List	55
Appendix E: Bibliography	56

List of Figure:

Figure 1 An agent	4
Figure 2 A Simple Order Book	8
Figure 3 Algorithm for Maslov Limit order generator	14
Figure 4 Action and Interaction between agents	15
Figure 5 Use Cases Diagram	17
Figure 6 Order Book Model	21
Figure 7 Basic System Design of Limit Order Book Model	22
Figure 8 The Sequence Diagram for Limit Order Book Simulation	23
Figure 9 The Workflow of the order book system	24
Figure 10 The Workflow of Insert a New Entry in Order Book to Buy Side	24
Figure 11 Class Diagram for Limit Order Book Model	25
Figure 12 System Package Diagram	27
Figure 13 Java Class Structure	30
Figure 14 Testing Process Diagram	32
Figure 15 Price for Cont Book Model at $D = 0.3$	35
Figure 16 Average Volatility for Cont Book Model at $D = 0.3$	35
Figure 17 Price for Cont Book Model at $D = 0.03$	36
Figure 18 Average Volatility for Cont Book Model at $D = 0.03$	36
Figure 19 Price for Cont Book Model at $D = 0.003$	37
Figure 20 Average Volatility for Cont Book Model at $D = 0.003$	37
Figure 21 Price for Double Auction Book Model at $D=0.3$	38
Figure 22 Average Volatility for Double Auction Book Model at $D=0.3$	38
Figure 23 Order Book for Double Auction Book Model at $D=0.3$	39
Figure 24 Price for Double Auction Book Model at $D=0.03$	39
Figure 25 Average Volatility for Double Auction Book Model at $D=0.03$	40
Figure 26 Order Book for Double Auction Book Model at $D=0.03$	40
Figure 27 Price for Double Auction Book Model at $D=0.003$	41
Figure 28 Average Volatility for Double Auction Book Model at $D=0.003$	41
Figure 29 Order Book for Double Auction Book Model at $D=0.003$	42

List of Table:

Table 1 The simple order book (indicate the source of the table)	9
Table 2 Order book after submit 400 shares buy limit orders at price 48.2	9
Table 3 Order book after submit 300 shares buy market orders	10
Table 4 Order Book Trading Rules of London Stock Exchange (source: London Stock Exchange, LSE, (2011). Rules of the London Stock Exchange. Effective1, 35-43.)	11
Table 5 Functional System Requirements	20
Table 6 Non-Functional System Requirements	21
Table 7 Explanations of Parameters	31
Table 8 Scenario 1 Parameters Setting	34
Table 9 Scenario 2 Parameters Setting	34

Chapter 1 Introduction

This chapter gives a brief introduction to the project. It explains the motivation for the project and its objectives. The motivation relates to financial concepts and multi-agent models.

1.1 Motivation

1.1.1 Order Book Motivation

In the modern time, many of the important stock exchanges like London Stock Exchange, New York Stock Exchange and Tokyo Stock Exchange, depend partially on limit order for liquidity provision. So it is necessary to understand the limit placement, the contribution to liquidity and price formation [Jiang, 2009].

We also are very interested in the dynamics of the order book and order flow: the order flow not only reacts to the information from the market and the order book state, but also influences the trading activity in the marketplace, affects the evolution of the order book afterwards, and updates the book state.

As known, an order book has extremely rich data and complex behavior. Therefore, it is important to model and study its behaviors and develop algorithms to limit its behaviors. It will be helpful in improving price stability and reducing trading costs.

1.1.2 Agent-based Modeling Motivation

Agent-based modeling is a novel way to build a system model that is comprised of interacting autonomous agents [Macal, 2006]. ABM could have a great positive effect on the way that scientists do research with electronic labs or businessmen use computers to make decisions.

We can take all types of independent components as agents. The agent behaviors in this project can range from simple ordering algorithm to complex limit order book matching rules. From this project model standpoint, the following agent characteristics will be relevant [Macal, 2006]:

1. Identifiable – the rules can be set to limit its behavior
2. Situated – all agents live in an environment and interact with each other
3. Goal-directed – each agent has goal to achieve
4. Autonomous and self-directed – each agent has independent function
- 5 Flexible – an agent can learn and adapt its behavior according to experience

The details show in Fig. 1.

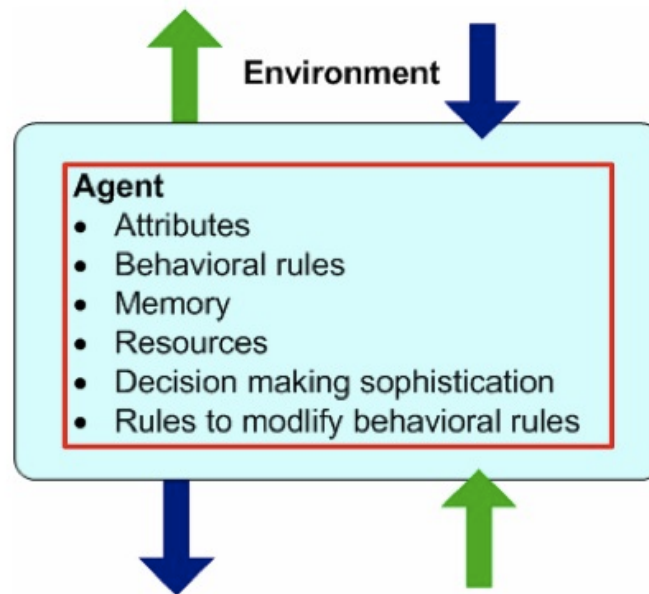


Figure 1 An agent

We consider applying agent-based modeling because the issue we study in this project is complex. The system that we try to analyze and build is much complex due to the agents' inter-dependence. A traditional modeling approach will not adequately model this system. Furthermore, this project will involve a large amount of data; agent-based modeling could provide much better approach to organize those data.

As a conclusion, applying agent-based modeling in this project is beneficial due to the reasons listed below [Macal, 2006]:

- The project has decisions and behaviors that can be defined discretely;
- The project needs agents who adapt and change their behaviors;
- The project needs agents who learn and engage in dynamic strategic behaviors;
- The project may require agents having a dynamic relationship with other agents, and agent relationships that form and dissolve;
- The project needs agents having a spatial component to their behaviors and interactions;
- The past of the market is no predictor of the future;
- Scaling-up to arbitrary levels is important;
- Process structural change needs to be a result of the model, rather than a model input;

1.2 Aim and Objectives

The main aim of the project is to find out the novel limit order book rules to improve the trading price stability. The objectives in order to achieve this aim are:

- To do overview of the existing limit order book models, And to analyze the issues related to building an effective limit order book.
- To build a simple order book model without any trading rules.
- To analyze the price movements for the simple order book model.
- To develop new algorithms to limit the behaviors of the order book, and to introduce them into the simple model to extend model capabilities.
- To analyze the price movements, having applied the order book algorithms.
- To compare the results output from the simple model and the extended model and discuss the improvements in price stability.

The objectives that must be achieved in the design order book model are:

- Three agents must be built to present the order book, a buy trader and a sell trader.
- Order book and traders must communicate with each other.
- The order book model must apply effective algorithms
- The system behavior should be visualized, so that the results could be shown directly.

1.3 Project Report Overview

Chapter 1 gives a brief introduction to the project. It explains the project's motivation and objectives. The motivation relates to financial concepts and multi-agent models.

Chapter 2 introduces the detailed background about order book and agent-based model. The first section is to give the order book definition, limit order book rules and trading strategies, by reviewing past published papers. The second section is to discuss more information about agent-based models.

Chapter 3 presents the design of the limit order book system process in details. These include gathering requirements, use case analysis, and limit order book design. The class diagram, activity diagram, sequence diagram are applied to show design process.

Chapter 4 explains the implementation and testing process. First section gives the descriptions about how to apply functionalities including initialization, input and output. The second section develops a way to test the system.

Chapter 5 analyzes the experimental results. This chapter displays all resulting output from the system. These results are compared and analyzed. The findings are also discussed in this chapter.

Chapter 6 summarizes the findings from chapter 5 and gives final conclusion. The future work is discussed in the second section of the chapter.

Chapter 2 Background and Research Information

This chapter introduces the detailed background about order book and agent-based models. The first section gives the order book definition, limit order book rules and trading strategies by reviewing past published paper. The second section discusses more information about agent-based models.

2.1 Order Book

2.1.1 Order Book Definition

An order book is a list of outstanding orders, manual or electronic, at an exchange venue, i.e. the London Stock Exchange (LSE). The book records the interest of sellers and buyers in certain financial instruments. It is used to decide which order could be fulfilled by the trade engine. The relevant definitions of terms in an order book are listed as follows [Marco, 2007]:

- Price levels – If several orders have the same price, they could be called as a price level. That means all orders on that price level have opportunities to fulfill that ask when an ask on that same price level comes.
- Cross the book – The orders could be matched according to the interest of sellers and buyers, as an order book is a part of the trade matching engine. When there is an order whose ask price is lower than the highest bid, the order should be fulfilled immediately. If it is not fulfilled and still a part of the order book due to errors, this situation is referred to as crossing the book.
- Top of book – Top of book is the lowest ask and highest bid. The difference between them is called the spread. They are the signal that orders need to be fulfilled in the market.
- Book depth – It means at particular time in the book how many price levels are available. Some books could have as many levels as wished, while other books have a fixed depth, i.e. the orders over the depth number will be rejected or ignored.

2.1.2 Limit Order Book and Model

An order book that contains limit orders is a limit order book [Black, 2010]. The Fig. 2 shows the concept of limit order book in details. The limit order book lists a current set of sell and buy offers for a particular instrument according to price levels. The 1st entry in the limit order SELL book is the lowest price to sell the instruments and the other orders are listed in increasing price in the SELL book. On the other side, the 1st entry in the limit order BUY book is the highest price to buy instruments, with other orders decreasing [Chen, 2007].

In a limit order book, the concept of last trade price is applied. If a market order could match the top of the limit order book, a trade will be done. We can see the book in Fig. 2.



Figure 2 A Simple Order Book

Trader A can sell a share of stock at price \$50 (best buy offer at the moment), while trader B can buy a share of stock at price \$53 (best sell offer at the moment). The limit order entry will be deleted from BUY/SELL book when any trade happens. For instance, if trader A sells a share with limit order book, they will get \$50, and the top of BUY book (\$50) will be deleted from the list. The last traded price will be set as \$50. Trader C must place a \$50 buy order to get a share stock, and give \$50 to trader A. The current model is simplified as trading in one stock share. But in the real world, a limit order book should hold a large number of different shares for every book entry.

In the modern market, most places apply an electronic double auction limit order book. A trader could choose to submit a market order or a limit order[Chen,2007]. A market order makes sure that the execution is done immediately, but cannot control its price. A limit order can control the execution price but cannot guarantee when the execution could be done. In the limit order book, depending on market rules the limit orders can be executed in two types – price priority and time priority. Ask orders that has lowest price will be fulfilled firstly, a sell market order gets executed at best bid price and vice versa. During a whole trading day, new limit orders keep adding into the order book, current orders in the book are continuously cancelled or executed. Therefore, the limit order book becomes highly dynamic and keeps changing throughout the trading day. Table 1 shows a simple order book. The order information is transparent to the traders. Table 2 shows the order book after submitting a buy limit order at price 48.2 for 400 shares. Table 3 shows a trader submitting a buy market order for 300 shares. (source: Chen, S., (2007). Computationally intelligent agents in economics and finance, Information Sciences, 177, 5, 1153–1168.)

ASK		BID	
Shares	Price	Price	Shares
400	49.3	49.8	600
100	48.2	50.1	450
50	47.7	50.6	700
150	46.5	50.9	500
200	45.4	51.2	100

Table 1 The simple order book

ASK		BID	
Shares	Price	Price	Shares
400	49.3	49.8	600
500	48.2	50.1	450
50	47.7	50.6	700
150	46.5	50.9	500
200	45.4	51.2	100

Table 2 Order book after submit 400 shares buy limit orders at price 48.2

ASK		BID	
Shares	Price	Price	Share
400	49.3	49.8	300
100	48.2	50.1	450
50	47.7	50.6	700
150	46.5	50.9	500
200	45.4	51.2	100

Table 3 Order book after submit 300 shares buy market orders

2.2 Order Book Trading Rules and Strategies

2.2.1 Order Book Trading Rules

The order book trading rules are applied to align with system and operation rules. They refer to the trading system and the parameters that are applicable. They are set according to change of segment and sector level system configuration in trading system. The rules include how a member firm enters the trading system and its responsibility for the order management. They could be member authorized connection or order routing. The rules also cover the reversal of erroneous trades and contra requests. When a trading system model is designed, it is necessary to take into consideration the order book trading rules. Table 4 shows these rules in details [LSE, 2011].

Trades On Exchange trades			
	2000		automatically effected on an Exchange order book
Order entry Access to the trading system and the responsibility of member firms			
	2100		Each order submitted shall be firm and subject only to the terms relating to benefit entitlements prevailing at the time of execution.
G	2101		The member firm shall, at all times, have sufficient order management systems, procedures and controls designed to prevent the entry of erroneous orders
GT	2102		A member firm should use the correct dealing

			capacity indicator
D G	2105		The Exchange reserves the right to refuse a member firm's request.
Contra request			
G GT	2110		If a member firm submits an order incorrectly which is subsequently executed, it may submit a request to contra the resultant trade(s).
Exchange enforced cancellation of erroneous trades			
D G	2120		The Exchange views all trades undertaken under its rules as firm. However, the Exchange may, in exceptional circumstances, undertake an Exchange enforced cancellation of an automated trade executed on the trading system , either at the request of a member firm or of its own volition. In considering a member firm's request for an Exchange enforced cancellation , the Exchange will have regard to a number of factors that are set out in the guidance below, and whether:
		2120.1	both parties to the trade(s) are unable to agree to use the contra facility;
		2120.2	the request for an Exchange enforced cancellation is submitted to the Market Supervision department within a time period specified by the Exchange in the guidance to this rule;
		2120.3	the member firm requesting the Exchange enforced cancellation provides appropriate information to the Market Supervision department as set out in the guidance below; and
		2120.4	A member firm has incurred an amount of loss through an automated trade conducted on the trading system as specified in the guidance to this rule.

Table 4 Order Book Trading Rules of London Stock Exchange (source: London Stock Exchange, LSE, (2011). Rules of the London Stock Exchange. Effective1, 35-43.)

2.2.2 Limit Order Book Strategies

Limit order book strategies is a set of rules to determine an amount of trade components aimed to improve price stability and reduce transaction cost. The components include how many orders should be submitted, size for each order, type of each order and what time should be submitted to market.

The trading volume of an order could be presented by a percentage of the average daily volume of the stock. If the volume of an order is less than 5% of average daily volume, it can be traded without any complex strategies [Kissell, 2006]. If the volume is more than 15%, it will lead to a high market impact. In order to reduce the impact, the order should be executed in several trading days. The normal size of an order is 5% to 15% of average daily volume. An appropriate trading strategy should be applied to execute these normal size orders.

The reason for applying a trading strategy when an order volume is more than 5% of average daily volume is that it has the potential to change the price of that asset. When price changes, there will be a huge market impact cost. If a large order could be divided, the cost will be reduced.

On the other side, suffering opportunity cost could be increased when large orders are divided. Therefore, a good trading strategy should try to balance these costs and reduce the total cost as much as possible.

In the next paragraphs, several strategies will be discussed that have been outlined in published research papers.

The simplest one is a pure market strategy [Cui, 2010]. An order that is executed through a trading day is divided into N smaller child orders. The child orders could have different sizes. Each child order is submitted as a market order, at regular intervals during the trading day. The time intervals are mostly set as 15 minutes or 30 minutes.

The limit order strategy is a child order submitted as a limit order through a trading day [Cui, 2010]. The trader must determine the limit price, lifetime and amendment frequency for each child order. A child order is set to have a T minutes' lifetime, Δt minutes' amendment frequency and price the same as the best available price. The limit order is placed at best price (ask/bid) at submission time. It will be amended to best price if it is not fulfilled during Δt minutes after submission. The process will keep going in Δt interval till T minutes. At T minutes, if it still has uncompleted orders left, they will be executed as market orders by crossing the bid-ask spread.

As described above, it is modest level to set the limit price as best available price. Except modest level, the trader also could submit the orders in aggressive level and passive level. An aggressive limit order is to sell/buy at the price that is one tick size below/above the best ask/bid price. A passive limit order is to sell/buy at the price that is one tick size above/below the best ask/bid price.

For a more complex strategy, the limit price can be set according to market condition dynamically [Daniel, 2006]. For instance, if a trader wants to make profit and minimize the cost, it will be a good choice to place passive orders when the price is decreasing and to submit aggressive orders when the price is rising up.

In Peter A. Whigham's paper, a limit order book generator model is developed, and it is tested whether a grammar-based evolution (GE) can evolve a strategy to make profit or not [Whigham, 2010]. Grammatical evolution is an evolutionary automatic programming technique to help seed the evolutionary process with the trading strategy.

The model used to generate a limit order book is called the Maslov model [Maslov, 2000]. For this model, the trader is chosen randomly at each time step and he may do a limit order or a market order. The number of stock sold or bought is not limited.

Fig. 3 shows the algorithm for Maslov generator to generate a limit order book L in a single time step [Maslov, 2000]. $L.Price$ is last trade price. $L.Sell$ and $L.Buy$ are limit

order sell/buy books. *L.Sell.Price* and *L.Buy.Price* are current sell/buy top entries in books.

```

1 Input : Limit-Order Book L, Tick-Size t, LimitOrderProbability plo
2 Output: Updated Limit-Order Book L
3
4 /* buy or sell? */
5 if Rnd(0,1) ≥ 0.5
6     /* Buy behaviour */
7     if Empty(L.Sell) or Rnd(0,1) ≤ plo
8         /* insert limit order to buy */
9         if Empty(L.Sell)
10            price = L.Price - RndI(1, t);
11            break
12        else
13            price = L.Sell.Price - RndI(1, t);
14            break
15        InsertLOBuy(L.Buy, price);
16    break
17 else
18     /* buy at lowest sell price */
19     LO = RemoveFirst(L.Sell);
20     L.Price = LO.Price;
21     break
22 break
23 else
24     /* Sell behaviour */
25     if Empty(L.Buy) or Rnd(0,1) ≤ plo
26         /* insert limit order to sell */
27         if Empty(L.Buy)
28            price = L.Price + RndI(1, t);
29            break
30        else
31            price = L.Buy.Price + RndI(1, t);
32            break
33        InsertLOSell(L.Sell, price);
34    break
35 else
36     /* sell at highest buy price */
37     LO = RemoveFirst(L.Buy);
38     L.Price = LO.Price;

```

Figure 3 Algorithm for Maslov Limit order generator

2.3 Agent-based Modeling

An agent-based model is effective model simulating the actions and interaction of individual agents with a view to assessing their effects on the system as a whole. Before we build an agent-based model, we should identify agents and their behavior, identify the relationships between agents, and get the necessary agent-related data. At a general level, building an agent-based model should follow the five steps described below [Macal, 2006]:

Step 1 – Agents: Identify the types of agents and their attributes.

Step 2 – Environments: Identify the environments that the agents will stay in and interact with.

Step 3 – Agent Methods: Set up the methods by which agents' attributes are renewed to make response to the agent interactions with environment or interactions between agents.

Step 4 – Agent Interactions: Use the methods to make interactions under control. The related issues can include: which agents interact, when they interact, and how they interact during the simulation.

Step 5 – Implementation: Implement the agent-based model into the computational software.

As we want to simulate an order book system and study its behaviors, an agent-based model should be built. The order book, BUY trader and SELL trader are three individual agents. Fig.4 shows their actions and the interaction between them.

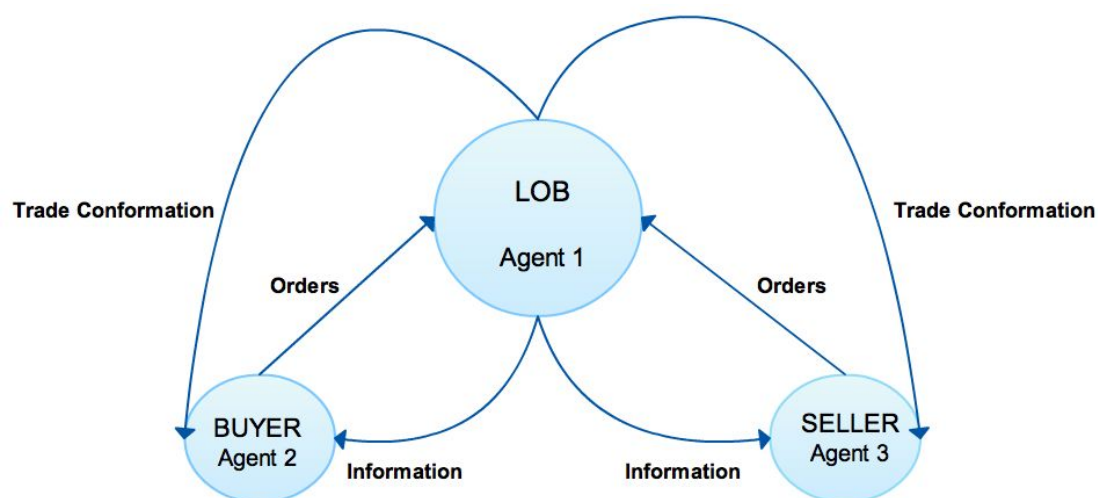


Figure 4 Action and Interaction between agents

Chapter 3 Order Book System Design

This chapter shows designs of limit order book system process in details. They include gathering requirements, use case analysis, and limit order book design. The class diagram, activity diagram, sequence diagram are applied to show the design process.

3.1 Guidance from Related Research

At the moment, a large amounts of trading platforms simulations are developed using agent-based modeling, most utilizing libraries. Researchers and research students design many of them. One of the recent most helpful simulators is designed by Preis et.al [Preis, 2006] as a multi-agent-based order book model of financial markets.

They introduce a simple model for simulating financial markets, based on an order book, in which several agents trade a kind of stock at a stock exchange continuously. For a fixed market structure of model, the used price time priority matching algorithm produces a diffusive behavior of price and order flow rates of the different kind of order types. They also show a market trend, for example, an asymmetric order flow of any type can result in a non-trivial Hurst exponent for the price development, but not “fat-tailed” return distributions. When an extra couples the order entry depth to the prevailing trend, our Order Book Model can also reproduce the stylized empirical fact of “fat tails”.

Another worthy research is carried out by Biails et al. [Biails et al., 1995], who do an empirical analysis of the limit order book and the order flow in the Paris Bourse. They analyze the demand and supply of liquidity. For instance, thick books result in trades while thin books elicit orders. In order to get time and price priority, investors quickly place orders within the quotes when the spread is large or depth at the quotes.

3.2 Use Cases and User Requirements Analysis

3.2.1 Use Cases Analysis

Before analyzing the user requirements and user cases, who will use this system must be defined. We design this system for traders, bank dealers, financial agents and financial market researchers. The parameters they need to specify could include: the best ask price, the best bid price, the ask depth, the bid depth and the spread. Hence, the system has 2 kinds of actors. One is system user, another is system developer. The developer is in charge of designing, developing, calibrating and testing the system. And the user is in charge to use the system functions to analyze their real order book problems. They also need a summary of this system's functionality, load the system and run the model. A use case diagram and task description are introduced as follows:

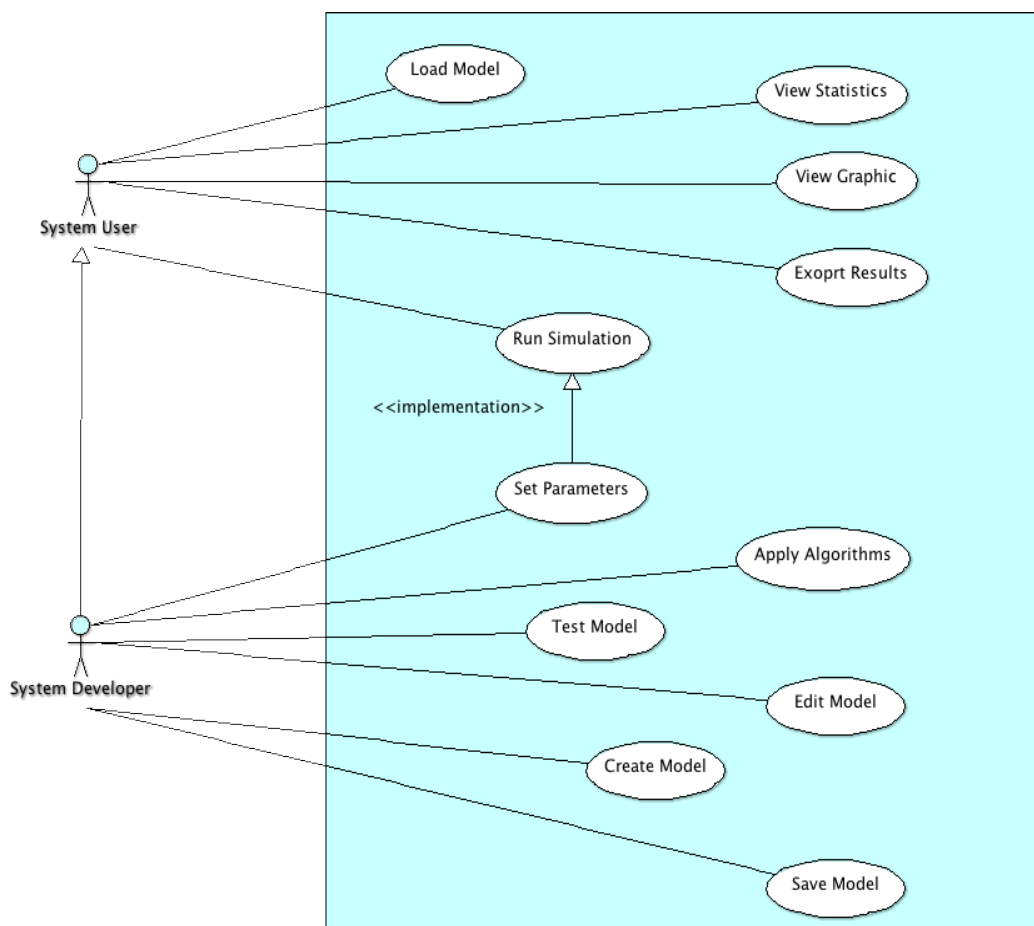


Figure 5 Use Cases Diagram

Load Model: The user chooses a developed model from the computer. The computer provides a list of models with different locations. The user finds the required model and selects it. The system loads the selected model and activates it. The details will be shown in the user manual.

View Statistics: The user chooses to see the output in a statistical way. The model should display all statistics in the interface windows.

View Graphics: The user chooses to see the output in a graphic way. The model should show the result visually by graphics.

Export Results: The user exports and saves the statistical results in a text file. The user exports and saves the graphic results in a jpg or gif file.

Run Simulation: The user changes the simulation parameters optionally and clicks the start button to run the simulation. The system begins the simulation and provides visual feedback while it is running. The user chooses to pause or stop the simulation at any time. The system stops running the simulation and saving the model's current state, allowing the user to run simulation continuously from where it was stopped.

Set Parameters: The developer sets the initialization value of parameters in the system. The users choose to change the value of parameters in the text file. The system will save new values of changed parameters.

Apply Algorithms: The developer applies the matching algorithm for the order book and trading algorithms for traders in developed simulation model.

Test Model: The developer calibrates/validates the simulation model in the computer system.

Edit Model: The developer edits and modifies the existing simulation model.

Create Model: The developer designs the simulation model and implements it into the computer system.

Save Model: The developer saves the existing model with a special name into hardware for users loading to use. The user selects the model or type the model name to accept.

3.2.2 User Requirements Analysis

Requirements gathering and analysis as the important step in the project are significant tasks for further understanding of the system's behavior and the project scope. It is critical to the success of the development process of the simulation model. They are high-level functions that user wish the system can provide. They should not include the technical details, but they should provide enough information to make developer decide whether to apply those functions into the model.

UR-1. The system must provide model creation utilities.

UR-2. The system must provide simple user interface.

UR-3. The system must allow user change the parameter setting before the simulation run or during the simulation run period.

UR-4. The system must provide a visualization window for the current state of the simulation, and animate this during simulation runs.

UR-5. The system must allow user to review all ask price and bid price together or separately.

UR-6. The system must allow user to review all ask depth and bid depth together or separately.

UR-7. The system must allow user to export and save the statistical results into text files.

3.3 System Requirements

There are two different types of system requirement – one is functional requirement, another is non-functional requirement. They are distinguished so that each refers to the implementation of one user requirement and the technical design of the system. The functional requirements define a function of a system or its components, while the non-functional requirements describe criteria used to access an operation of system, rather than specific behaviors. They could often be the limits on the system that are qualitative measurements such as cost, quality and performance.

The functional/non-functional system requirements are listed in tables 5 and 6 below. The priority of functions can be defined into three levels: **(M)** - Must Have (crucial to the system), **(S)** - Should have, and **(C)**- Could have (optional); is also given for each requirement.

ID	Functional Requirement	Priority
1	The model shall implement multiple agents – at least two agents (order book and trader).	M
2	The model shall implement an order book for the market and limit orders are placed and matched.	M
3	The model shall implement order book rules to limit the orders.	M
4	The model shall list bid/ask price from high to low/from low to high in separate columns.	M
5	The model shall use the visualization built-into MASON to show the model during a run and after.	M
6	The model shall represent the ask price in red line and the bid price in green line.	M
7	The model shall output results of the limit order price in graphs	S
8	The model shall provide an interface to modify the parameters such as time scale, etc.	S
9	The model shall implement the trading strategies for trader	C
10	The model shall allow simulation results history saved automatically	C

Table 5 Functional System Requirements

ID	Non-Functional Requirement	Priority
1	The model shall allow users to import native java libraries or external libraries to extend its functionality.	S
2	The model shall be extensible, allowing users to change functionality without changing many codes.	C

Table 6 Non-Functional System Requirements

3.4 The Order Book Model Design

In modern electronic exchanges, the order book saves demands and offers of different traders and supports a continuous trading. The minimum price change is called tick size. The price can be calculated by multiple of the tick size. The lowest offer price is called best ask while the highest demand is best bid. The spread is non-zero gap between best ask and best bid, as shown in Fig. 6. Limit orders at any price level will be added chronologically in the queue of that price tick according to the price of the limit orders. Then, a price time priority-matching algorithm could be achieved. Two orders at the price p_0 , on the offer and demand side, could be matched against each other. Therefore, a trade at price p_0 is done. The spread rises up to 3 ticks after the trade. [Withanawasam, 2010]

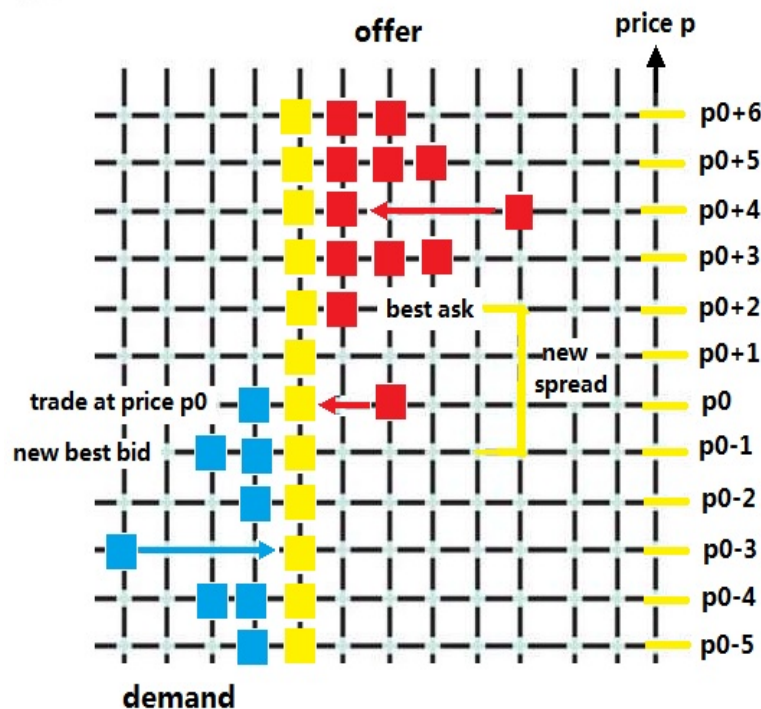


Figure 6 Order Book Model

The traders apply their own trading strategies (i.e. limit order or market order) to make the orders. The limit order book will decide to accept orders, reject the orders or place order in different ways by the limit order book rules. The orders accepted by the rules will store in the limit order book according to price and time priority algorithm. The trade will be executed when a counter trader (Buyer/Seller) offer a price that could be matched with that stored in order book. This trade confirmation message will both send to traders on two sides. On another hand, the limit order book also has to display all information about price and depth of the orders to all relevant financial agents. This information will have them to make decisions to send an order.

The diagram in Fig. 7 shows the basic system architecture.

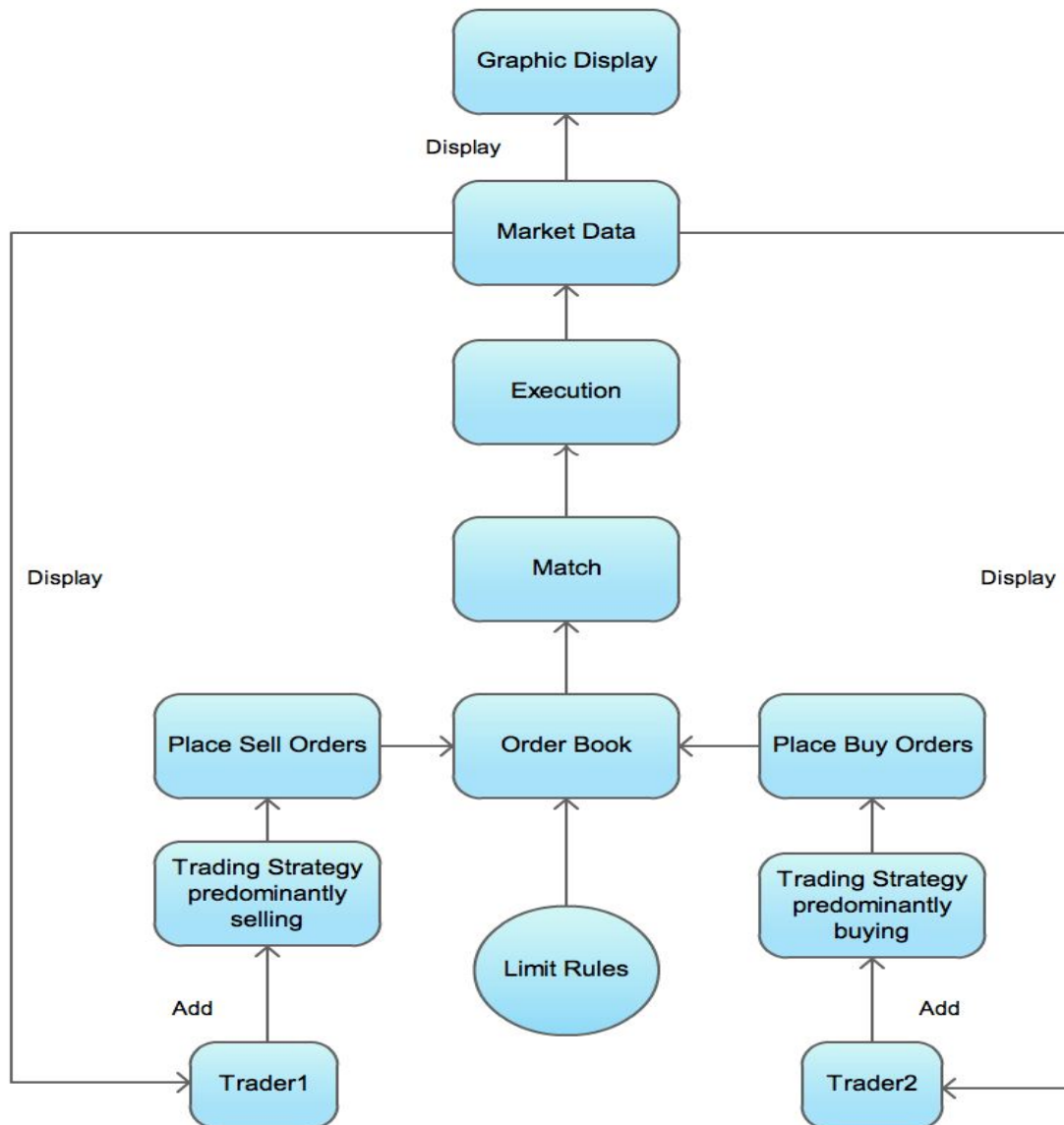


Figure 7 Basic System Design of Limit Order Book Model

The sequence diagram is a type of interaction diagram that shows the process operates with others and orders of operation. We use it to analyze the logic of a complex procedure, function or operation.

The diagram in Fig.8 displays the sequence of processes that operate during the model running period. It describes a single time-step. For normal use, this sequence would operate in a loop before simulation stops.

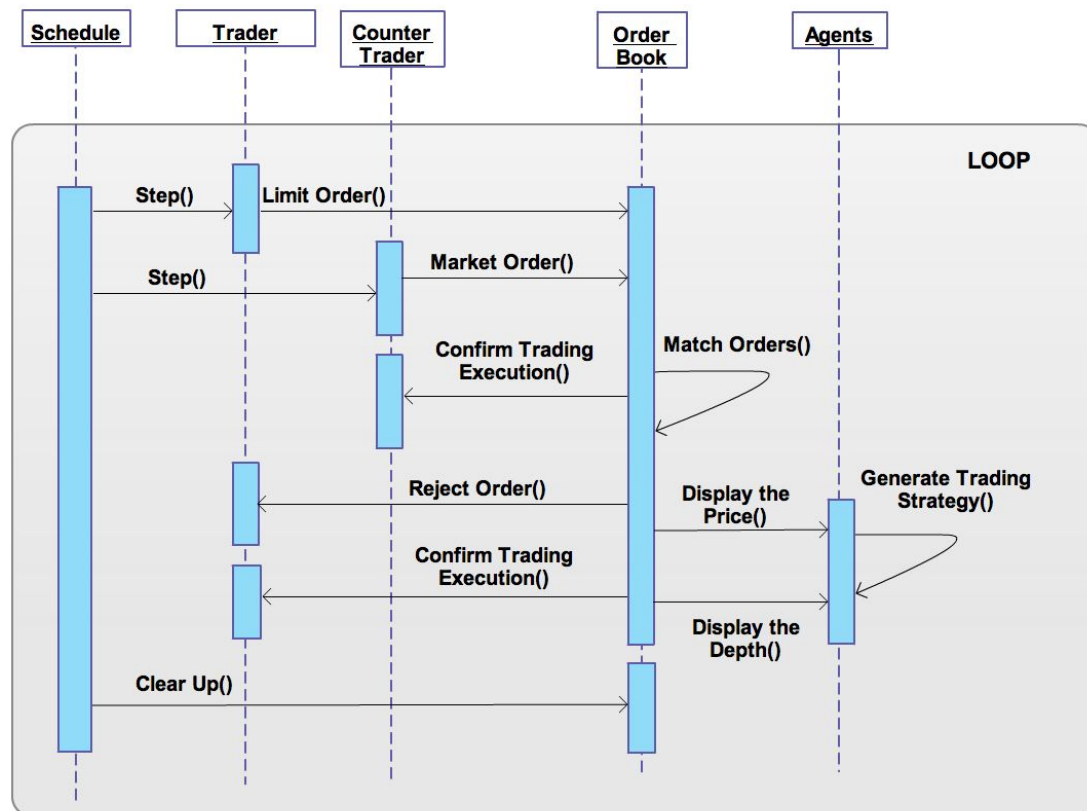


Figure 8 The Sequence Diagram for Limit Order Book Simulation

In order to analyze the model in more details, we use the activity diagram with conjunction with sequence diagram. Fig. 9 provides the visual presentations of workflow of how to order book system execution. Fig. 10 shows how to insert a new entry in order book on buy side. The sell side workflow is similar.

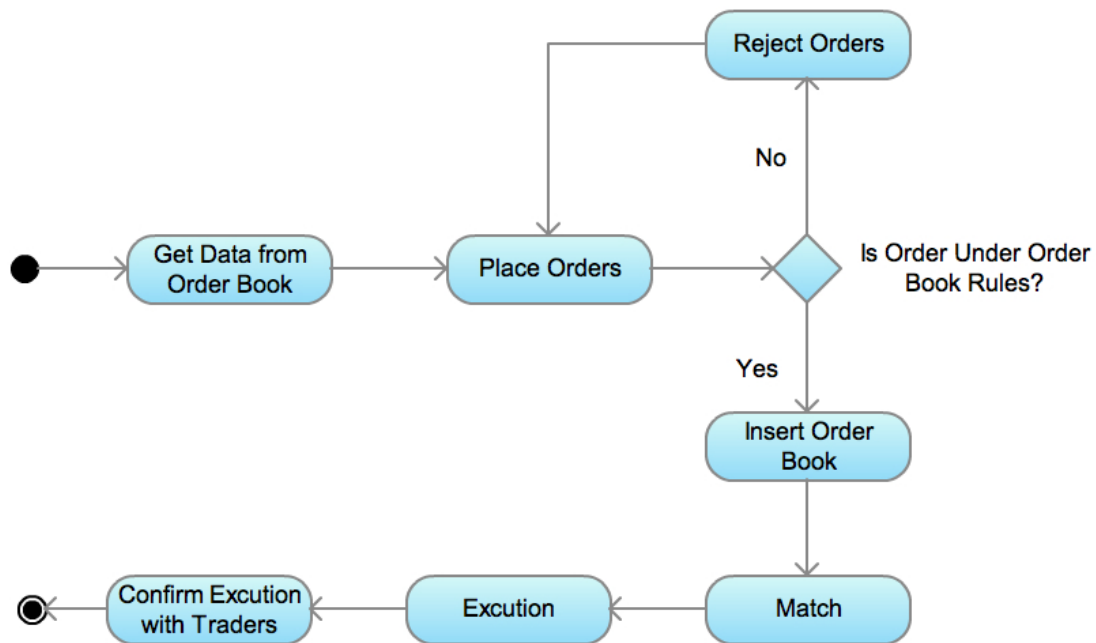


Figure 9 The Workflow of the order book system

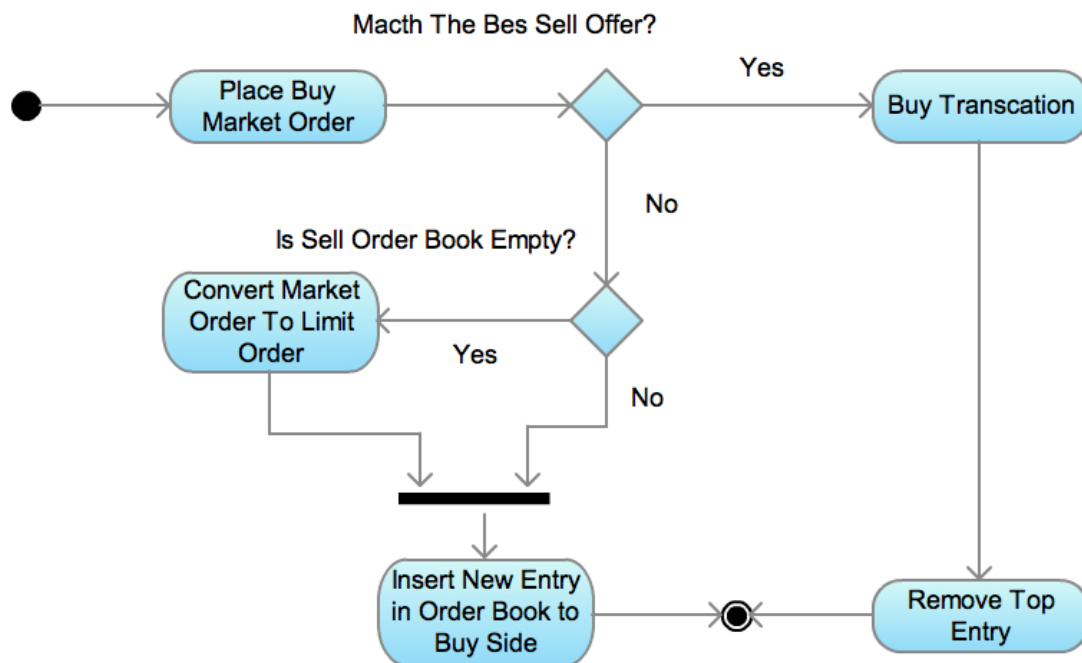


Figure 10 The Workflow of Insert a New Entry in Order Book to Buy Side

After analyzing the system architecture, operation processes and workflow, the next step for design system is to identify classes. We will combine responsibility-design approach and data-driven design approach to build a full class diagram – Fig.10. This diagram will describe the structure of the order book system by showing the all classes, their attributes, methods and their relationship among the classes.

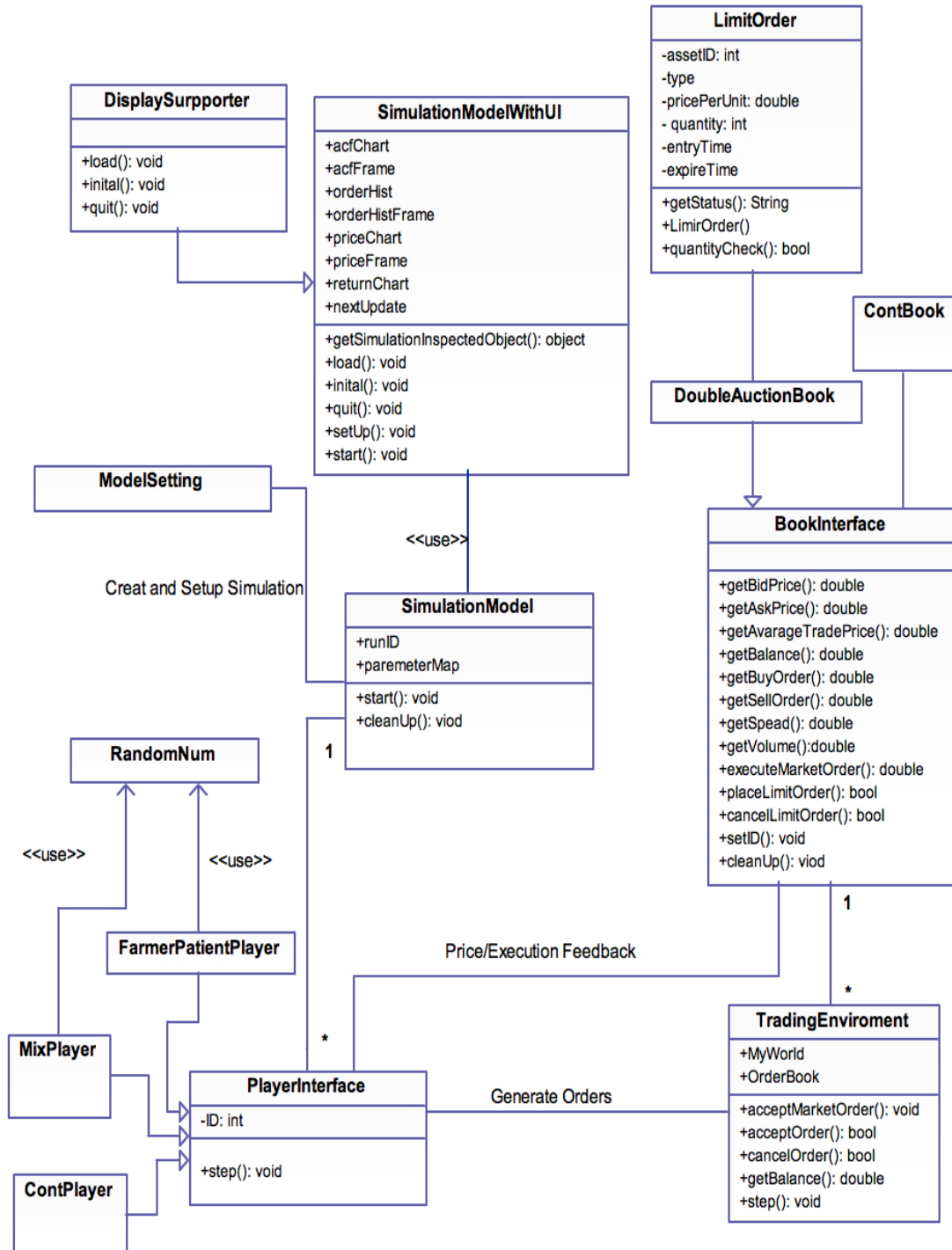


Figure 11 Class Diagram for Limit Order Book Model

Chapter 4 Implementation and Testing

This chapter explains the implementation and testing process. First section gives the descriptions about how to apply functionalities including initialization, input and output. The second section develops a way to test the system.

4.1 Software Application Used

Java SE JDK 6 Update 26

This is a java platform and can be run in any operation system. It is selected as MASON is applied in this project. Its agent modeling toolkits are developed by pure Java language.

NetBeans 7.0

This is a programming developer tool that both refers to an integrated development environment (IDE) for developing with Java and a platform framework for Java desktop application. It provides the environment to write codes, compile and debug them. It helps improve the efficiency of implementation work.

MASON

This is a kind of multi-agent environment developed in Java. It provides enough functionality for this project simulation. It is uploaded into Neatbeans to help me develop many agents and output visual results. With Mason, the implementation work improves a lot in efficiency.

Creately Desktop

This is a diagraming application from Cinergix Pty Ltd to help me create use case diagram, sequence diagram, activity diagram, class diagram, Gantt chart and anything visual needed for the project.

Java Libraries:

Java library is the compiled byte codes of source code developed by the JRE implementer to support application development in Java.

The libraries used for developing the system are as follows:

1. I-text – It is a library allowing the user to generate and manipulate PDF documents.
2. Jfreechart – It is a free java chart library that helps users to display professional quality charts in their simulations.
3. Jcommon – It is a java class library used by Jfreechart. It is used to support user interface classes for displaying information about applications and serialization utilities
4. Jmf – It is a java library called java media framework. The developer uses it to add audio, video and other time-based media to java applications and applets.
5. Vecmath – it provides a powerful way to handle and manipulate coordinates and vectors, being compatible with how the Source engine handles them.

4.2 System Implementation

The implementation of system was developed with Java, Java libraries and Mason software. It is a complex system that involves the object-orient design. The Mason software is applied to output visual results. Five packages are created according to system structure. The model package has two sub packages – agents and market. It use GUIdisplay package to generate graphics. The description of each package and package relationship diagram are show in following.

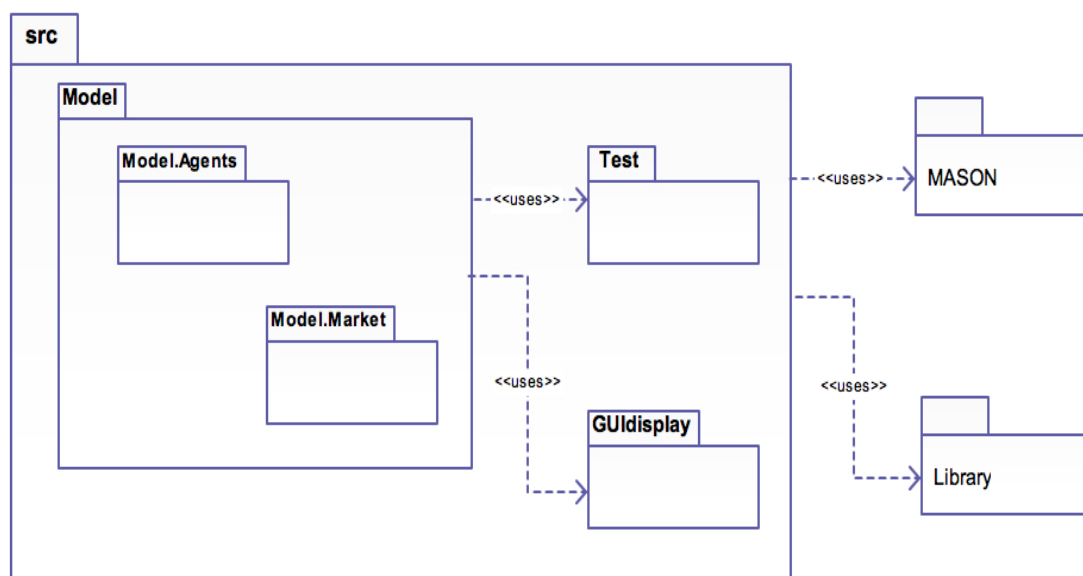


Figure 12 System Package Diagram

GUIdisplay – It includes a SimulationModelWithUI to plan chart style and generate the visual outputs. DisplaySupporter applied SimState simulation engine to load simulation scenarios and build GUI. An html file gives a description of simulation model for users.

Model – It includes two classes that builds a simulation model structure and controls setting. Two sub-systems (Agents and Market) are belonging to this package.

Model.Agents – It includes three different types of trades – ContPlayer, MixPlayer and FarmerPatientPlayer. A class defined as interface to connect with simulation model.

Model.Market – It includes a class that is used to simulate the trading environment. It can generate price for market. It also includes two order books. One is simplest order book; another is limit order book with some rules. An interface class is created in this package to do the communication between order book and market.

Test – It includes a test class to test the system whether can work with no any errors.

Mason – It includes all files from Mason.

Library - It includes all libraries that will support the system simulation.

4.2.1 Description of Java Class

In this section, the implementation approach and function of each java class will be described in details. There are fifteen java classes in this model in total.

- GUIdisplay package

Displaysupporter: This class implement Steppable simulation engine. It is used to communicate with other classes. ArrayList and Vector are applied to present XY series. Jfreechart library is called in this class to help draw the various charts.

SimulationModelWithUI: This class is in charge for building and managing the GUI. It builds the simulation interface for users and output different type graphics with using Jmf library. The time series chart generator is used to generate price plot. And histogram generator is used to generate order book chart. The controller is to control all related parameters about output charts.

Index: It is an html file that introduces the information about this simulation model. It will display in about page when user run the system.

- model package

ModelSetting: This class is responsible in connecting with step up files. User can select one types of the set up files that load into model. The class uses buffer reader to

get parameter values from one of property text files and get agents numbers from one of configuration files.

SimulationModel: It is main class of this model that initializes certain number of traders and runs traders for a maxT number of times. It extends SimState class used by Mason. It contains main method and use loop function to make simulation run continuously. The hash map is imported to this class to call the relevant maps that set up in ModelSetting class. The ArrayList is implemented for scheduling the traders.

- model.agents package

ContPlayer: This class simulates the trader agent's behavior that generates orders and updates thresholds.

FarmerPatientPlayer: This class simulates that the agent places limit orders according to the limit order books model.

MixImpatientPlayer: This class is a mix of cont and farmer impatient players. They place market order. It uses cont to determine when to trade and on which side. And it uses a double auction book to place orders.

PlayerInterface: It is an interface class for all agents. It is a steppable Mason object that helps agents connect with ModelSetting class. In this class trader's id is defined.

RandomNum: This class generates various random variables. The FarmerPatientPlayer and MixPlayer will call this class for random variables.

- model.market package

TradingEnvironment: This class simulates a trading environment like market. It generates the market price according to the orders placed by agents.

BookInterface: It is an interface class for all order books. This class provides the functions like placing an order for immediate execution, creating a limit order and cancelling a limit order.

ContBook: It is the simplest book that all traders follow the same behavioral rules.

DoubleAuctionBook: This class implements a complex order book. The market moves when market orders fulfill limit orders. The limit order doesn't ever execute when they are placed.

LimitOrder: This class contains the rules to improve the market price stability.

- test package

TestValidation: This class is developed to test system validation.

This figure shows the whole java class structure of system.

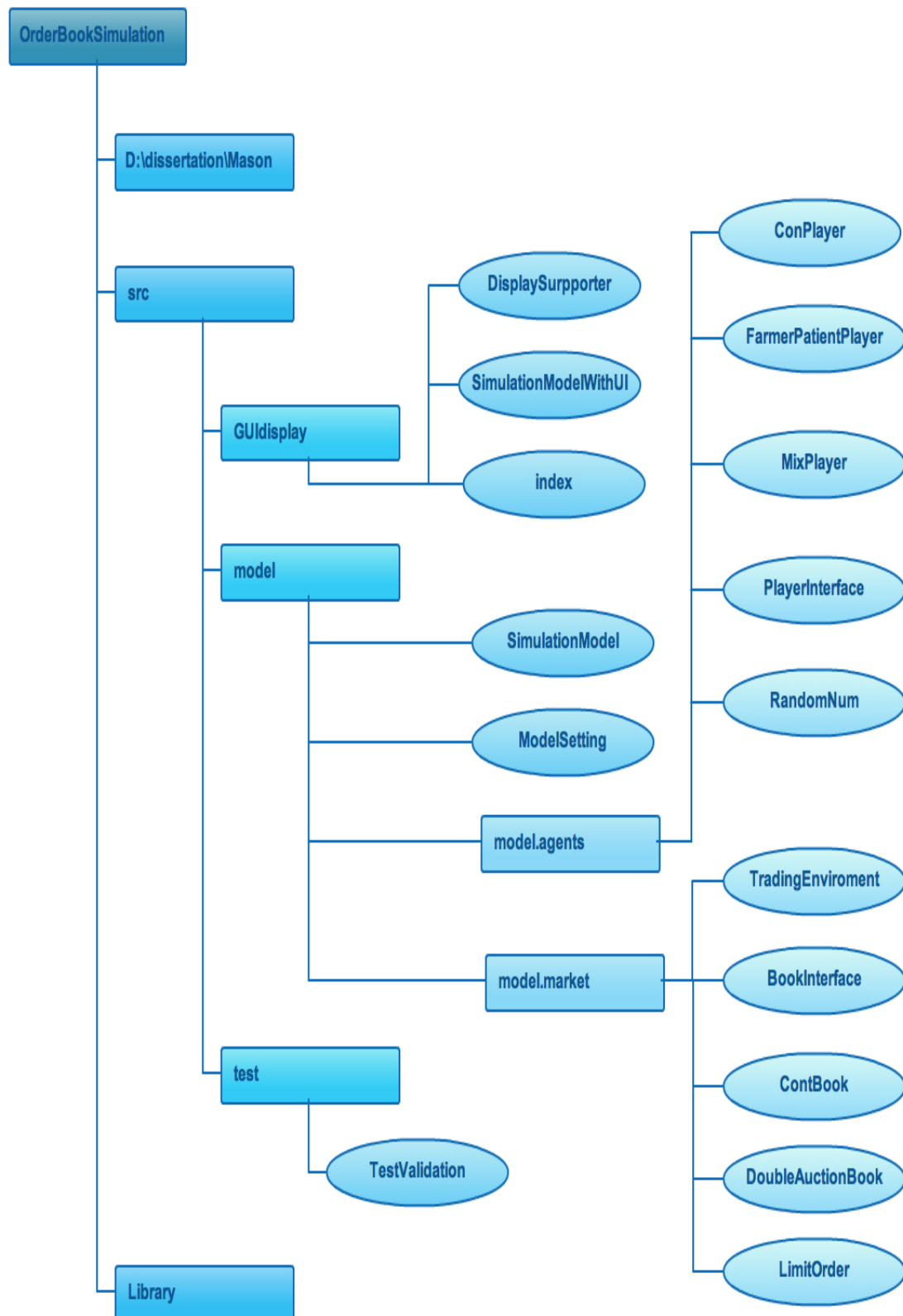


Figure 13 Java Class Structure

4.2.2 Description of Simulation Model

This model is designed by Java language to simulate an order book system. This system has different kinds of order books and traders. Each trader has his own trading strategy. A trading environment is simulated to place and execute the orders. The data of trading results like price is output by GUI. The GUI is achieved under help of Mason and java libraries. The whole order book simulation system contains two scenarios. One applies simplest order book model without any rules, another applies double auction book with limit rules.

The simplest order book model is called cont book. In this model, all trades will follow the same trading rules and have the same behavior. All traders are assigned volatility threshold independently. In the each period, the traders receive a common signal. The signal is seen as public information that simulated by random variables generator. To response the signal, each trader decide to sell, buy or sit out the period. After that, the market determines the excess demand and reaches the market-clearing price by means of a market impact function. At last, traders update the threshold so that can match the absolute value of the return rate for the current period.

The more complex order book model is called double auction book. This book organizes the limit orders and makes optimization between market order and limit order. To achieve this aim, the rules should be applied in order book. This model has two types of traders. One is to place market order, another is to place limit order. This model explains a large part of the price and spread diffusion of actual stocks given an order flow rate, which means that the double auction structure has a huge impact on the nature of market movements.

When users simulate different type of models, they also need change the setup file that connects to model in modelSetting class. The set up files specify the order book type applied and number of traders. DABook.properties file is for double auction book model. SBook.properties file is for cont book model. The number of trader can be changed in txt files. All parameters influenced simulation results in set up files are described in table. 7.

Parameter Symbol	Explanation
maxT	Max number of ticks
D	Standard deviation of the noise
numAssets	Number of assets present on market
initialPrice	Initial price
Cont_lambda	Cont market depth
Cont_s	Cont ratio of traders updating threshold
Farmer_alpha	Patient trader orders placed per step
Farmer_delta	Patient trader orders canceled per step
Farmer_mu	Impatient trader orders placed per step
Farmer_sigma	Shares per trade (order size)

Table 7 Explanations of Parameters

4.3 Testing

4.3.1 Testing Process

After finish implementation system, the next step must be done is testing. The proposal of testing is to make sure system operate normally, work as expected, and meet all functional/non-functional requirements that guide it design. In this project, testing is done exactly in the process that shown in Fig. 14.

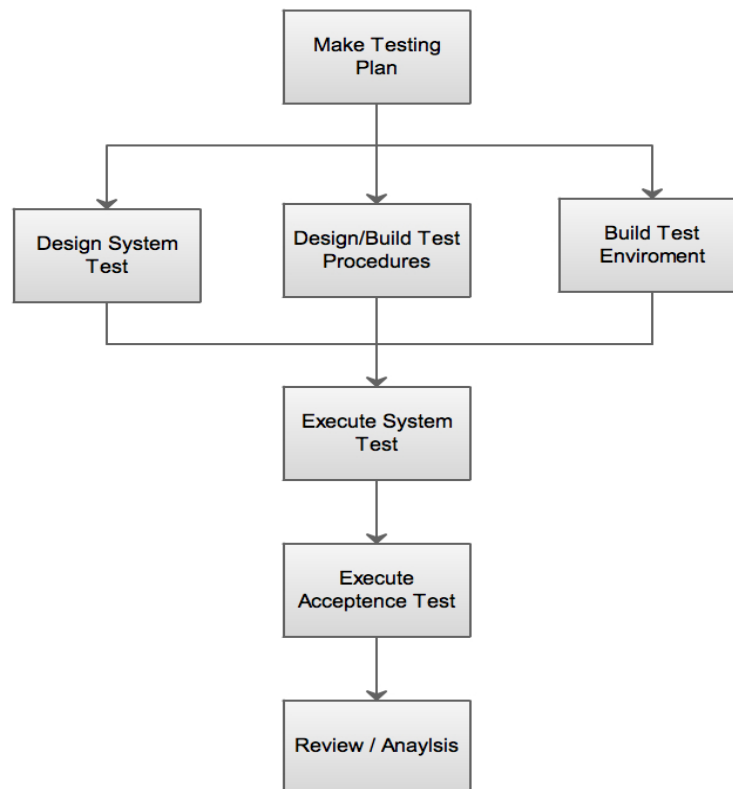


Figure 14 Testing Process Diagram

Making Testing Plan: Creates test schedule, plan and test approaches.

Design System Test: Identifies test cases, test cycles, exit & entry criteria and expected results.

Design/Build System Test Procedures: Sets up procedures like status reporting and error management.

Build Test Environment: Request and builds data set-up, software and hardware.

Execute System Test: Tests system performance and reports errors.

Execute Acceptance Test: Tests users' acceptance.

Review/Analysis of Results: Analyzes and summarizes tests results in report documents.

4.3.2 Scope, Approach and Results

Scope 1: Functional/Validation Testing

Objective: Make sure system meet functional requirements from design specification.

Approach: Run the system for 10 times with different settings and approaches. Compare the functions achieved actually by system with the requirements in design specifications. Do intensive testing of the new front-end fields and screens, windows GUI standards, screen & field look and appearance, and overall consistency with the rest of the application.

Results: The system can build order book and match orders correctly. The parameters can be set in set up files and be read in correctly. Simulation graphics is displayed as expected.

Scope 2: Technical Testing

Objectives: Make sure all unit java class run successfully and links between classes are correct.

Approach: A test package is developed in system. In the package, a specific class called TestValidation will test unit and links of classes. Click “Run” button and select “test project” tap to run test code. The code in details will be list in appendix.

Results: After run test class, netbeans reports 0 error found.

Scope 3: Performance Testing

Objective: Make sure the system provide acceptable response times

Approach: Run the system without GUI for 10 times and run the system with GUI for 10 times.

Results: Without GUI, the system response time is 3 seconds. With GUI, to view all charts 20k steps need to run. The response time is up to 3000 seconds.

Scope 4: User Acceptance Testing

Objective: Make sure the system operates in the manner excepted.

Approach: 25 test cases were performed for all supporting material like form and procedures are suitable and accurate for the purpose intended.

Results: There is no any error found during acceptance testing. 20 of 25 cases have been completed successfully.

Chapter 5 Results and Analysis

This chapter analyzes the experimental results. This chapter displays all resulting output from the system. These results are compared and analyzed. The findings are also discussed in this chapter.

5.1 Simulation Plan

The simulation will run with 3 times for two scenarios (cont book model and double auction book model). To alert scenarios, the code need be modified to import set up files.

For scenario 1, go to ModelSetting class to modify code in:

```
Properties properties = new Properties();
try {properties.load(new FileInputStream("setups//CBook.properties"));
}
```

For scenario 2, go to ModelSetting class to modify code in:

```
Properties properties = new Properties();
try {properties.load(new FileInputStream("setups//DABook.properties"));
}
```

For each time, the parameters setting are different. We use parameter D to control maximum and minimum threshold for trading. The maximum volatility is up to 0.3 and minimum volatility cannot be less than 0.003. The details of each parameter setting for each time and each scenario will be shown in table 8 and table 9.

Scenario 1 – Cont Book Model		
Run Times	Parameter D	Other Parameters
1st	0.3	maxT = 10000, $\alpha = 0.675$, $\delta = 0.29$, $\sigma = 1$, $\mu = 0.525$, initial price= 12.0, ContPlayer = 1500.
2nd	0.03	
3rd	0.003	

Table 8 Scenario 1 Parameters Setting

Scenario 2 – Double Auction Book Model		
Run Times	Parameter D	Other Parameters
1st	0.3	maxT = 10000, $\alpha = 0.675$, $\delta = 0.29$, $\sigma = 1$, $\mu = 0.525$, initial price= 12.0, PatientPlyer= 900, MixPlayer= 800.
2nd	0.03	
3rd	0.003	

Table 9 Scenario 2 Parameters Setting

5.2 Scenario 1 – Cont Book Model

This model is simple one without any order book rules. All traders follow the same trading strategy. This model will be run in three times and the parameters are set as details in table 7. The output charts include prices plots and average volatility plot. The simulation results are shown as following:

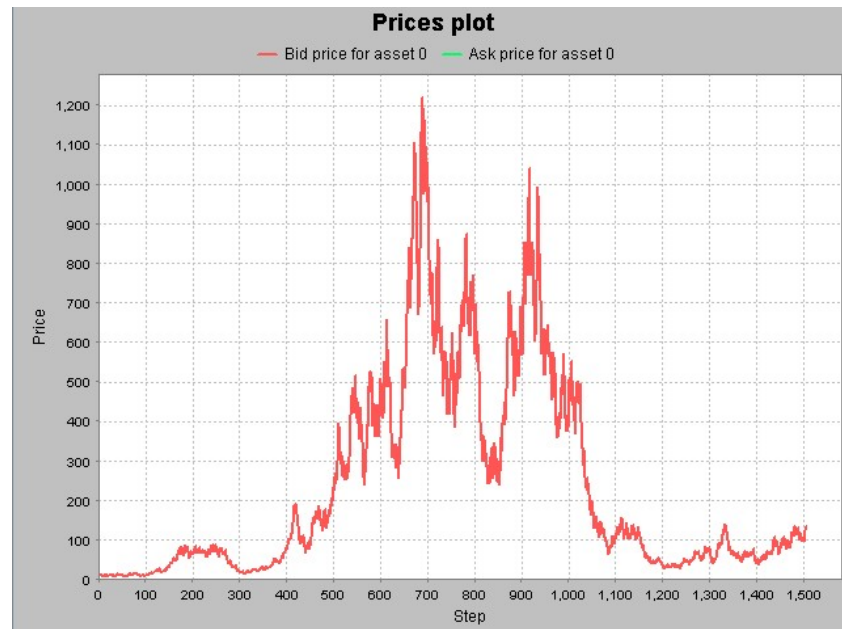


Figure 15 Price for Cont Book Model at $D = 0.3$

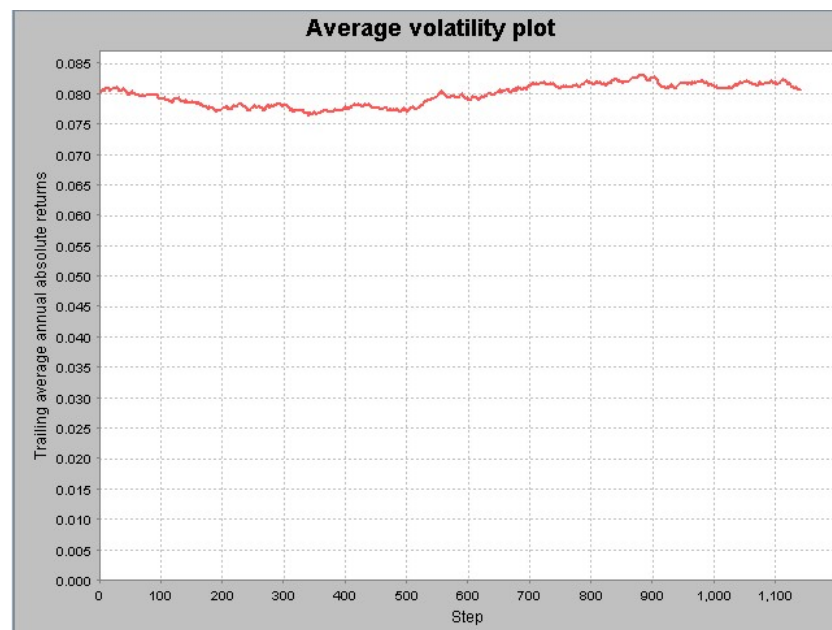


Figure 16 Average Volatility for Cont Book Model at $D = 0.3$



Figure 17 Price for Cont Book Model at $D = 0.03$

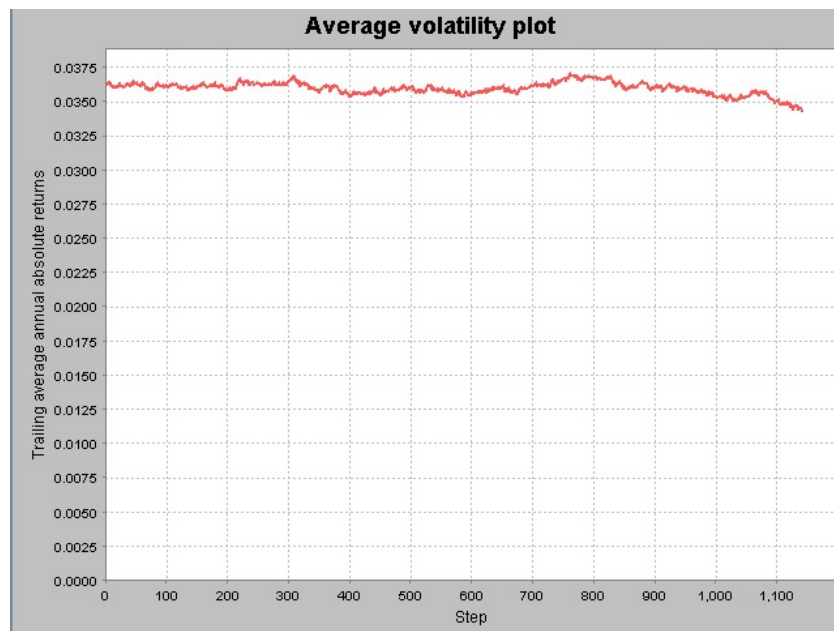


Figure 18 Average Volatility for Cont Book Model at $D = 0.03$



Figure 19 Price for Cont Book Model at $D = 0.003$

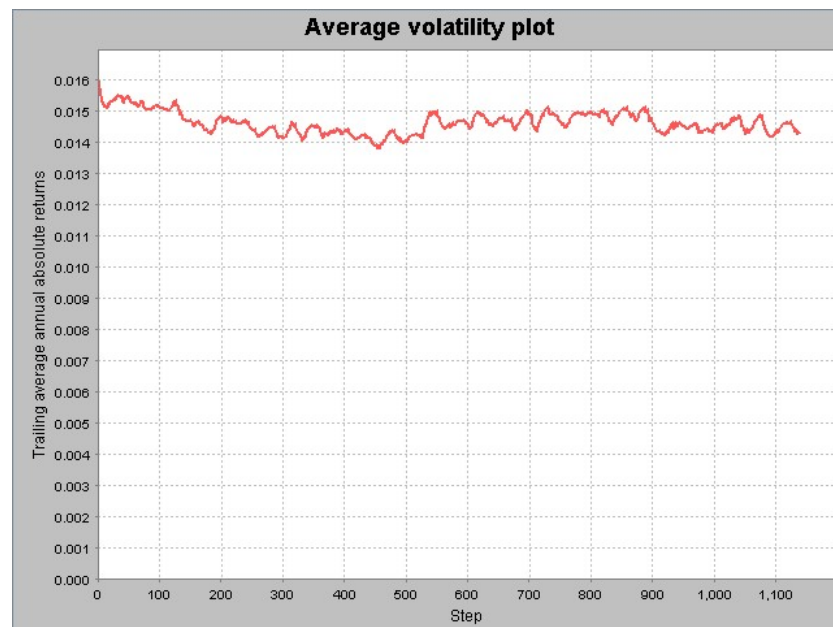


Figure 20 Average Volatility for Cont Book Model at $D = 0.003$

5.3 Scenario 2 – Double Auction Book Model

This model has two kinds of traders, which trade a certain asset by the approach of a continuous double auction. The traders are patient and impatient. This model will be run in three times and the parameters are set as details in table 8. The output charts include prices plots, average volatility plot and order book graphic. The simulation results are shown as following:

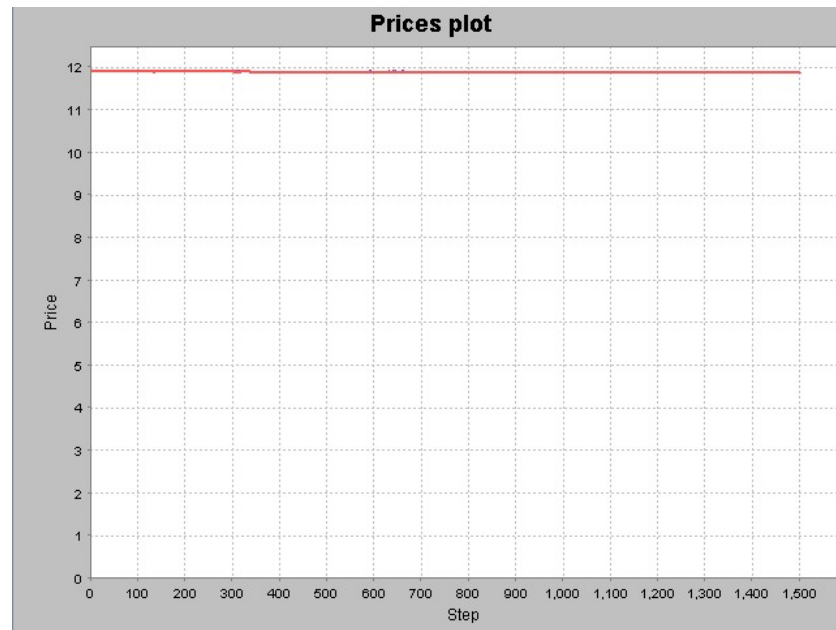


Figure 21 Price for Double Auction Book Model at $D=0.3$

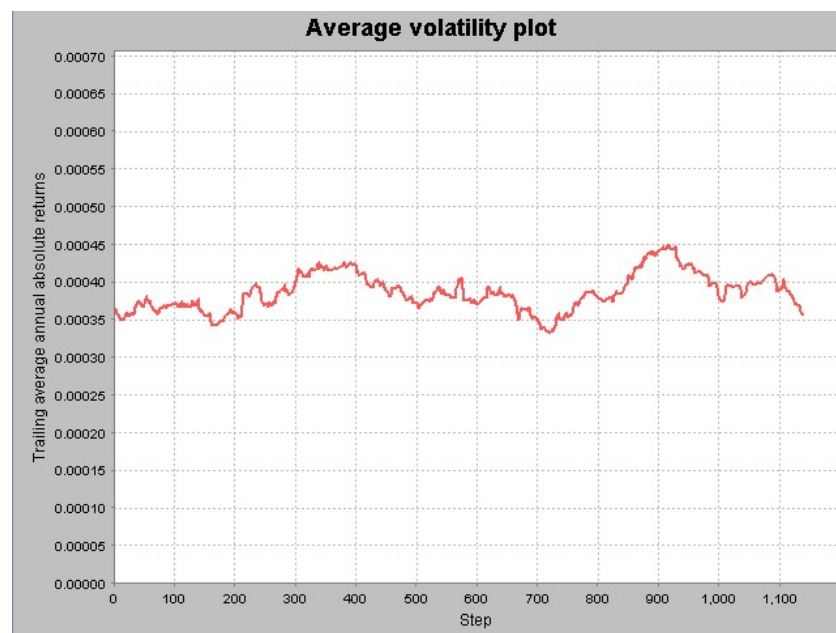


Figure 22 Average Volatility for Double Auction Book Model at $D=0.3$

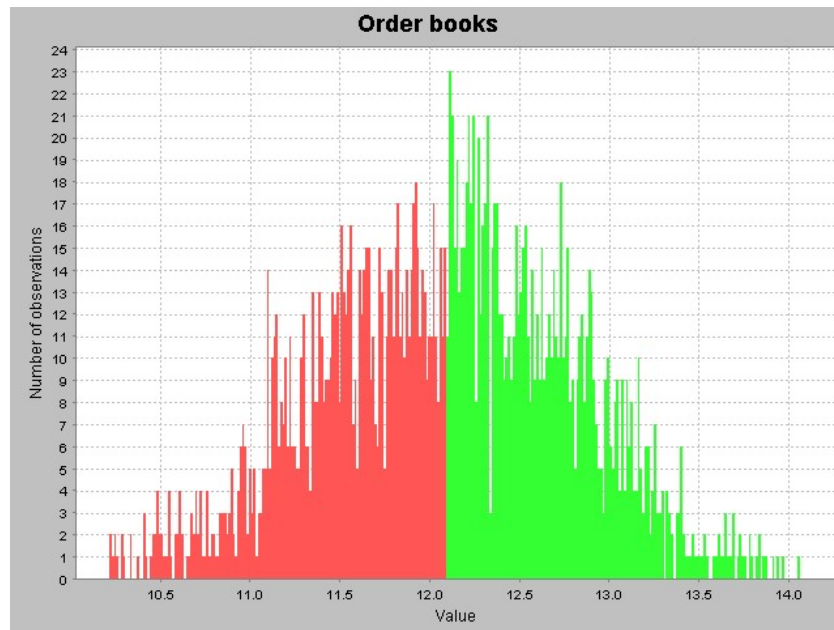


Figure 23 Order Book for Double Auction Book Model at $D=0.3$

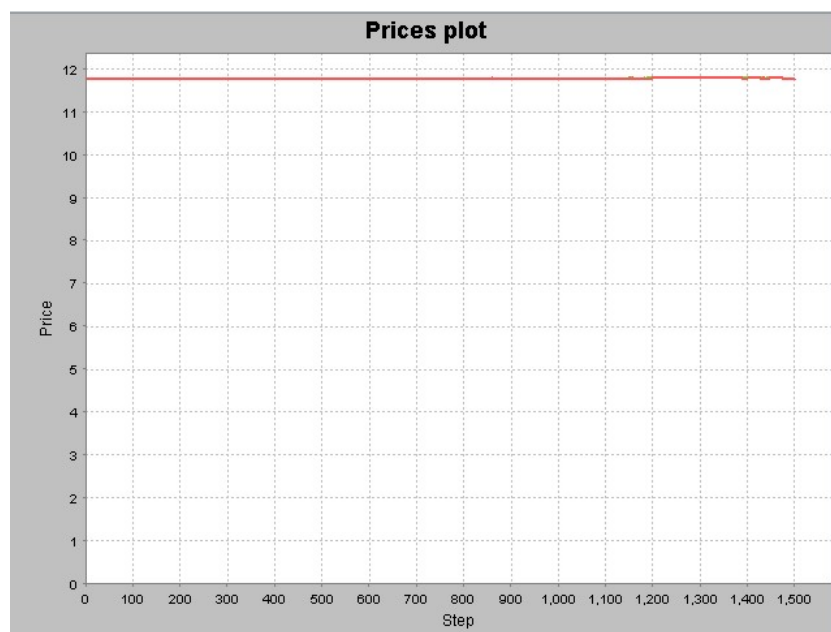


Figure 24 Price for Double Auction Book Model at $D=0.03$

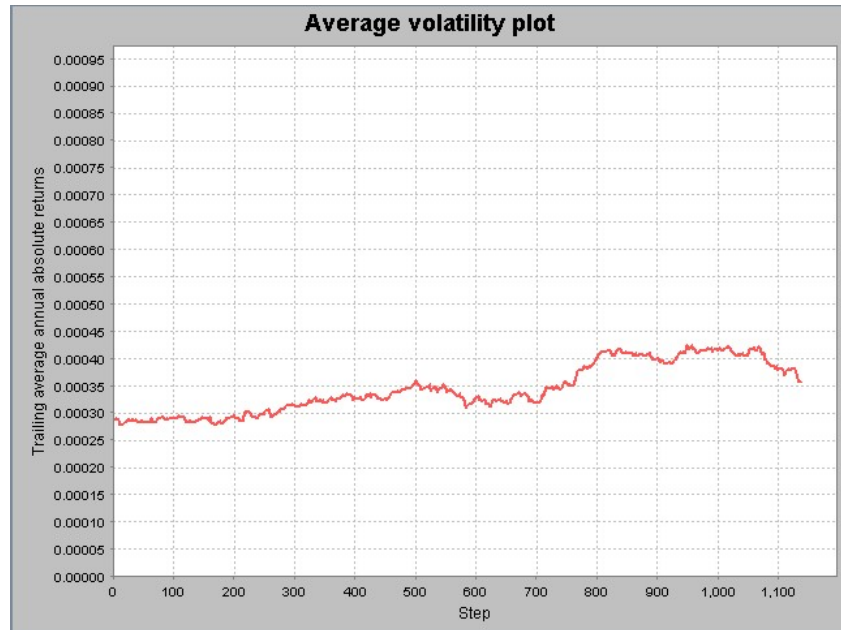


Figure 25 Average Volatility for Double Auction Book Model at $D=0.03$

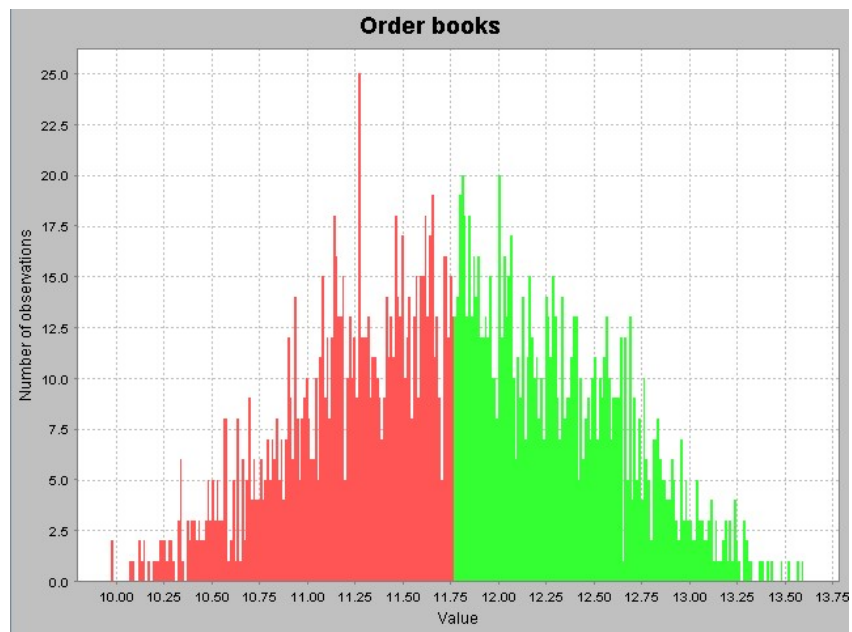


Figure 26 Order Book for Double Auction Book Model at $D=0.03$

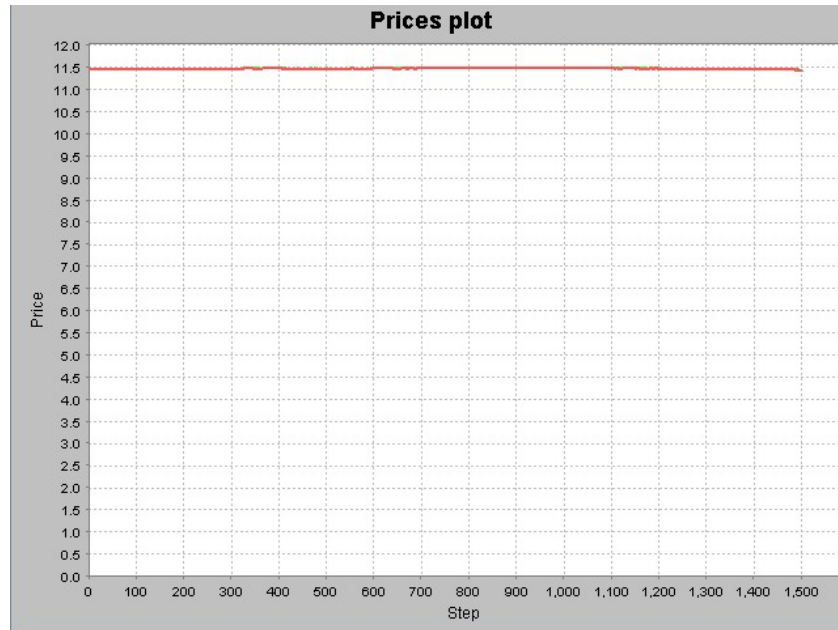


Figure 27 Price for Double Auction Book Model at $D=0.003$

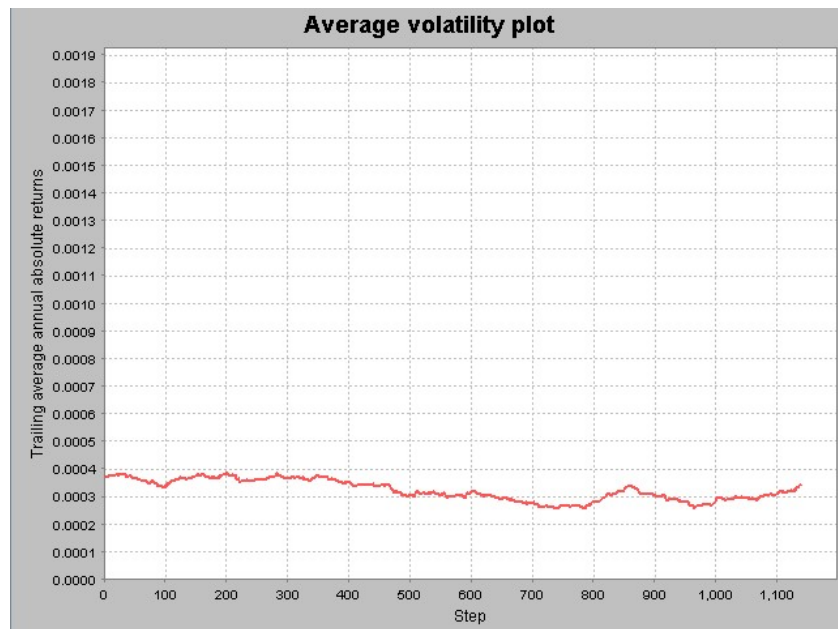


Figure 28 Average Volatility for Double Auction Book Model at $D=0.003$



Figure 29 Order Book for Double Auction Book Model at $D=0.003$

5.4 Analysis of Results

In the cont book model, the price movement is extremely huge. When $D = 0.3$, the maximum point can reach 1200. When D reduces to 0.03 and 0.003, the maximum price reduces to 25. But the price fluctuation is still significant, it is more than 100% of initial price. On another hand, whatever D equals to, the average volatility does not increase or decrease too much. It keeps about at 0.08, 0.036 and 0.015 when D equals to 0.3, 0.03 and 0.003. All of those chart results show the price is totally unstable in this model even the D is very small.

In the double auction book model, the simulation results are totally different. However the value of D is large or small, the price fluctuation is extremely small. The larger the value of D is, the closer to initial price the average price is. Whatever the value of D is, the average volatility is in a very small value. The order book charts are displayed as expected. They are very similar in three different values of D . They all look like sinusoidal distribution. Most buy and sell orders are placed in middle of prize axis. Only a few of sell orders is at a very high price, while also a few of buy orders is placed at a low price. This order distribution from charts also means the price of asset should be stable.

All in all, we can make conclusion from the charts from above. A limit order book with rules can help improve the price stability significantly.

Chapter 6 Conclusions and Future Work

This chapter summarizes the findings from chapter 5 and gives final conclusion. The future work is discussed in the second section of the chapter.

6.1 Project Summary

The main aim of project is to develop an order book system that can place market and limit order then match the orders. A set of rules must be implement into order book system to increase the market price stability.

Java language is used to develop codes due to it is object-oriented programming language, platform-independent. Each class is created for a special purpose. All of these classes can be used more times to become an object that can be then be executed. The Mason software and java library are applied to output simulation results in visual charts and graphics.

In order to complete the project, each step should be done successfully and in time. Firstly, we learned how to use Mason and combine it with Java. Then, we searched for related papers and did literature review. After the background and related definitions are clear, we analyzed the user requirements and functional requirements. The next step was to design system based on requirements. When design was completed, we developed java code to implement and test system. If the system could pass the testing, the final step of project is to generate simulation results and analyze them.

6.2 Conclusion

The main goal of this project was achieved successfully at end of project. To achieve this main goal, every objective in each step is done during the whole project period.

All user requirements from UR1 to UR7 that described in chapter 3 are achieved successfully in this system. All non-functional requirements are achieved in this system too. The functional requirements except FR10 are met in system implementation. The priority of FR10 is “could have” that means it is optional. This function can be done as a future work.

During the whole project process, a large mount of knowledge about limit order, order book and multi-agent modeling was obtained from literature review. And in the system simulation, a lot of conclusions could be made. If system applies the simplest order book without any rules, the price of asset will has a substantial fluctuation. If system applies double action book with rules, agents place market orders and limit orders, the stability of price will be improved significantly.

In conclusion, this project can be considered a success according to the reasons described above.

6.3 Further Work

In this project, we developed two kinds of models. One is simplest model with simplest trader behaviors. Another is complex one with order book rules. In the further work, the description in below could be direction for further development.

Combination Model

In this model, we can combine some aspects of the two models that have been developed successfully. The combination model will collect more positive characteristic of both models. It continually applies double auction mechanism. The two types of traders will interact via it. The patient trader's behavior remains same, but the impatient trader will be modeled after cont. They have heterogeneous thresholds, and place market orders according to the public information that is a normally distributed random variables compares with these thresholds. Some traders adjust the volatility thresholds in light of current volatility by setting it to the trailing period's return rate.

A Better User Interface

The user interface is still needed to do further development. When run the simulation model, an interface window for setting parameters should be displayed before simulation charts output. It will be much more convenient for changing set up files every times before run. An interface for select model also should be done instead of modifying codes every times.

Data Output and Storage

In this project, the model only can output results in charts style. The further work is to develop function that all simulation data can be store in a file as another results output. The data could be saved in txt file. But because of the huge mount of data, a more efficient approach for storing data is necessary. This approach should use less memory and is easy to manage data for users.

Appendix A: System Manual

Computer System Requirements:

Windows® XP / Vista / 7
512 MB RAM Minimum, 1GB RAM recommended
Hard Disk: 200 MB for NetBeans 7.0.1
DVD Rom Drive

Software Requirements:

JDK 6 Update 27 with NetBeans 7.0.1
Mason
Java 3D 1.5.1

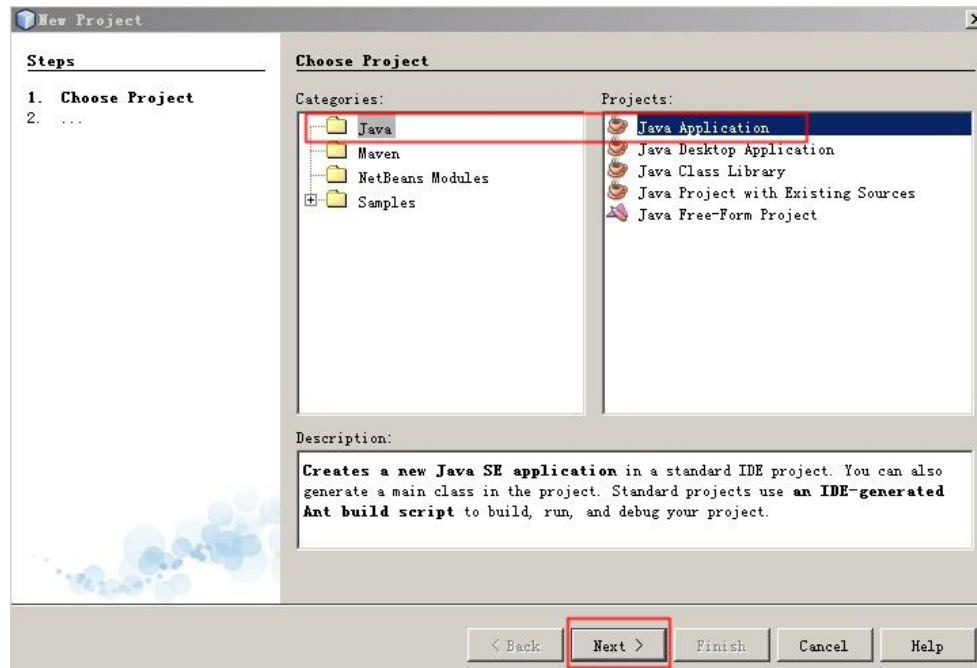
System Set Up Steps:

1. Download the file 'mason.zip' from <http://cs.gmu.edu/~eclab/projects/mason/>
2. Create a folder 'dissertation' in your D: drive.
3. Extract 'mason.zip' directly into 'dissertation'. It will create itself a sub-folder 'mason' while extracting.
4. Download the file libraries.zip from <http://cs.gmu.edu/~eclab/projects/mason/>, and extract into 'dissertation'. That will create the sub-folder 'libraries'.
5. Download the file jdk-6u26-nb-7_0-windows-ml.exe from <http://www.oracle.com/technetwork/java/javase/downloads/jdk-netbeans-jsp-142931.html>
6. Install JDK + NetBeans Bundle into computer.
7. Choose the 'Download Java 3D 1.5.1 Software' option from <http://www.oracle.com/technetwork/java/javase/tech/index-jsp-138252.html>, and download the file java3d-1_5_1-windows-i586.exe.
8. Install the Java3D framework into computer.
9. From your 'C:\Java' folder copy the sub-folder 'Java3D' into 'dissertation'.
10. Copy the JAR files 'itext-1.2', 'jcommon-1.0.0', 'jfreechart-1.0.1' and 'jmf' from 'D:\dissertation\libraries' to 'D:\dissertation\mason'. Copy the JAR files 'j3dcore', 'j3dutils' and 'vecmath' from 'D:\dissertation\Java3D\1.5.1\lib\ext' to 'D:\dissertation\mason'.

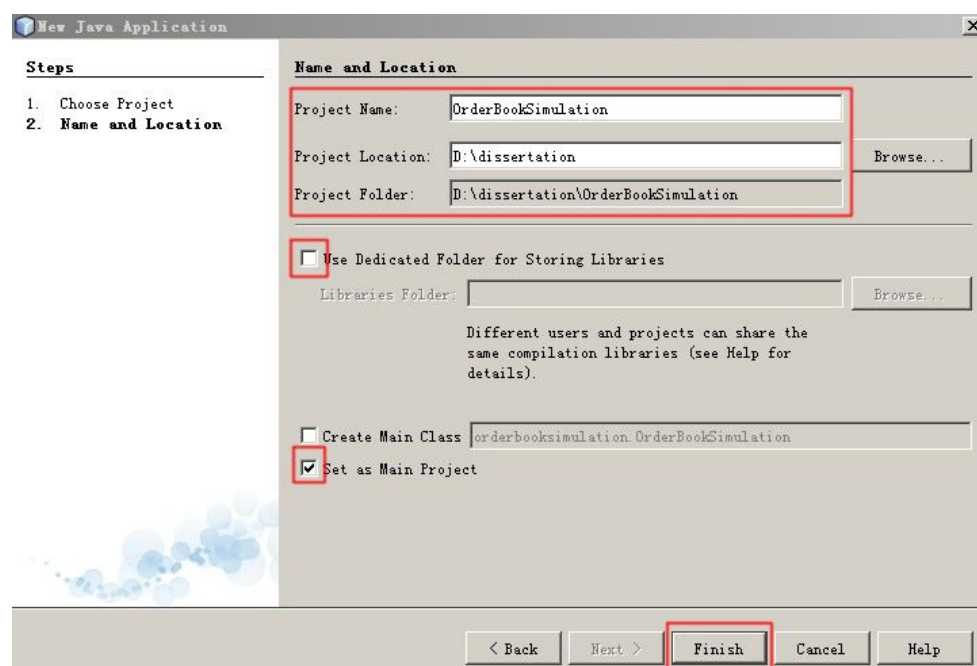
Appendix B: User Manual

Create Project and Import Codes:

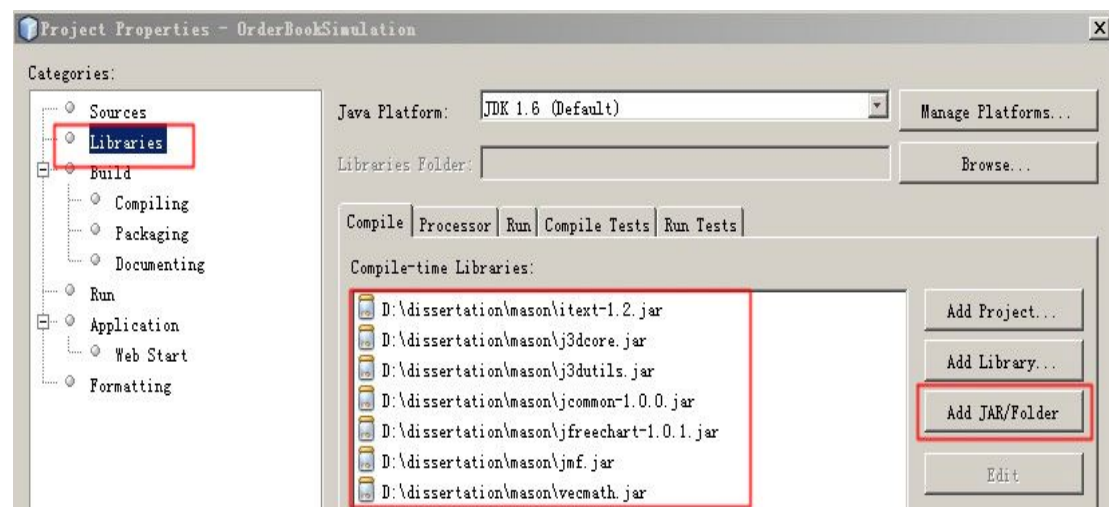
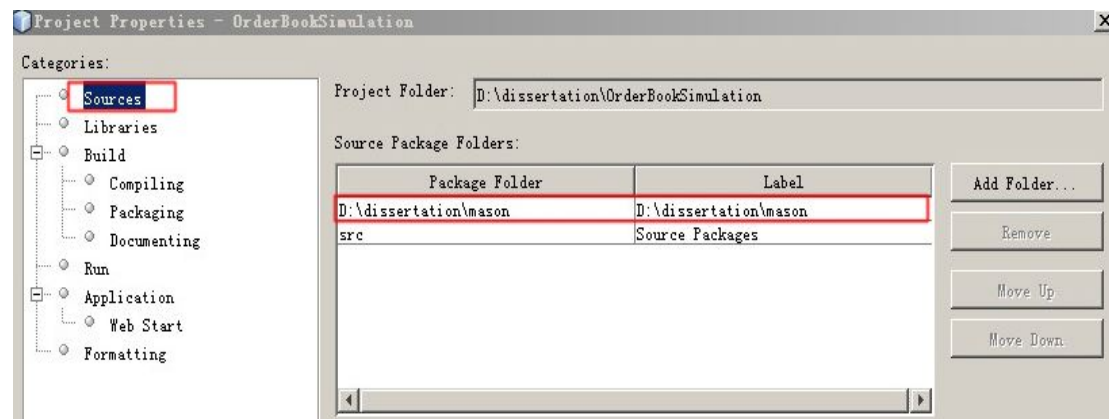
1. In the main window, select 'File>New Project'. Then choose the category 'Java' and the type of project 'Java Application'. Click 'Next'.



2. In the next window, give the project a name – 'OrderBookSimulation', which by default will also create a sub-folder 'OrderBookSimulation' in your 'D:\dissertation' folder. Select the option 'Set as Main Project', while not selecting the option 'Use Dedicated Folder for Storing Libraries'. Click 'Finish'.



3. In the main project window, right-click on project 'OrderBookSimulation' and select 'Properties'. In the next window, choose the category 'Sources' and add the folder 'D:\dissertation\mason' under 'Source Package Folders', while not adding any folders under 'Test Package Folders'. Then in the same window, choose the category 'Libraries' and under the tab 'Compile' click on the option 'Add JAR/Folder' and select all JAR files available in folder 'D:\dissertation\mason'. Click 'OK'.



4. Extract the Code.zip from the DVD to a destination folder.

5. Copy all sub-folders in 'Code' folder and paste them into 'OrderBookSimulation' folder.

Set Up Simulation Model:

1. Go to 'ModelSetting' java class and active the command
`"Properties properties = new Properties();
try {properties.load(new FileInputStream("setups//CBook.properties"));
}";`

```

26
27      /* Read in logging settings. */
28
29      if (target == null) {
30          returnSim = true;
31          target = new SimulationModel(System.currentTimeMillis());
32      } else {
33          this.target = target;
34      }
35
36      /* Read in simulation settings and set them to the target. */
37
38      Properties properties = new Properties();
39      try {
40          properties.load(new FileInputStream("setups//DABook.properties"));
41          // properties.load(new FileInputStream("setups//CBook.properties"));
42      } catch (IOException e) {
43

```

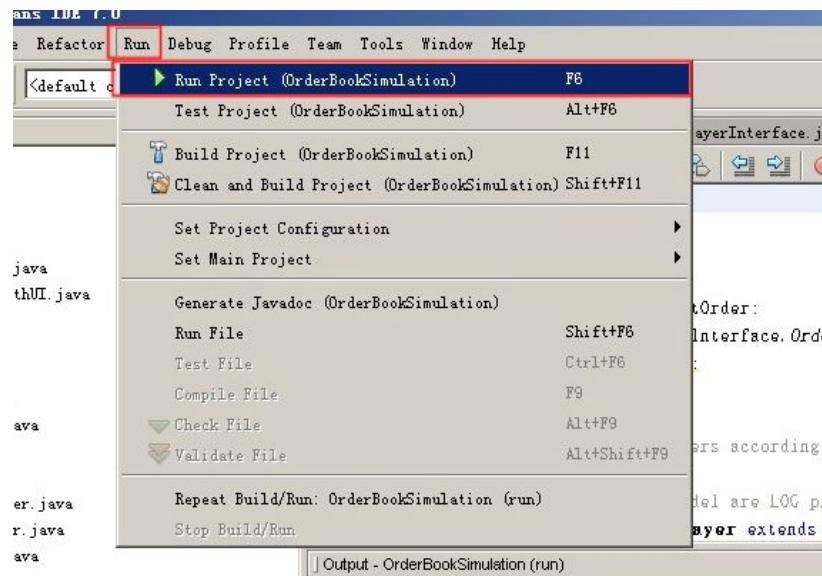
2. Find the 'setups' folder in 'OrderBookSimulation' folder. Open it and find a prosperities file called "CBook".
3. Open "CBook" file and set the value of α , δ , σ , μ according to table 8.
4. Save and close "CBook".
5. Open a txt file called "cont" in same folder.
6. Set ContPlayer number according to table 8, save and close the file.
7. Run simulations.

After run this model and collect the results, we change model setting.

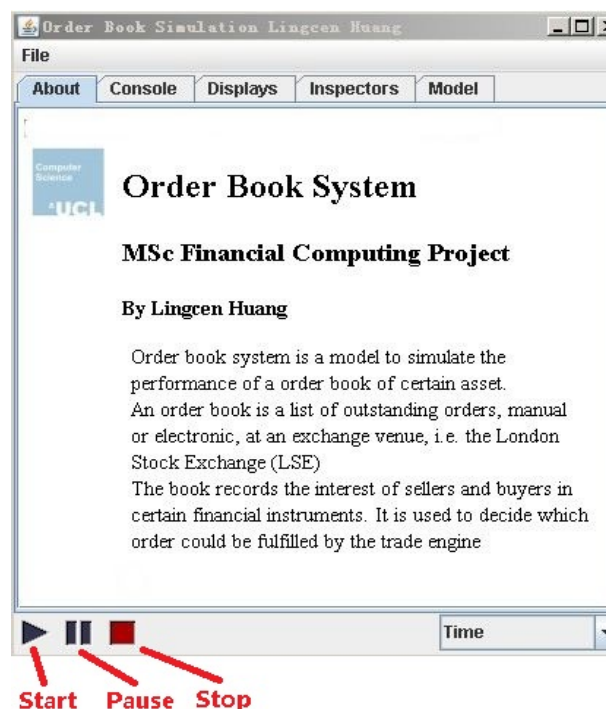
8. Go to 'ModelSetting' java class and active the command
`"Properties properties = new Properties();
try {properties.load(new FileInputStream("setups//DABook.properties"));
}";`
9. Open "DABook" file in 'OrderBookSimulation' folder and set the value of α , δ , σ , μ according to table 9.
10. Save and close "DABook".
11. Open a txt file called "farmercont" in same folder.
12. Set MixPlayer and FarmerPatientPlyer number according to table 9, save and close the file.

Run System:

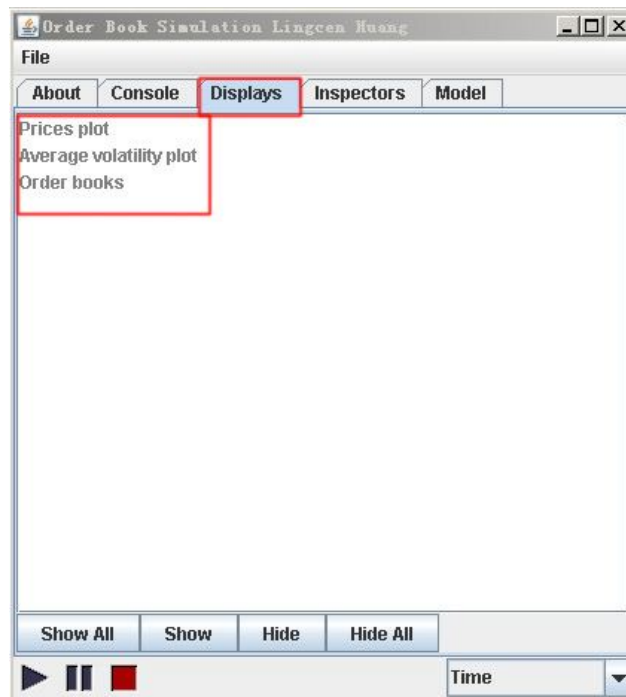
1. Go to NetBeans main menu and select 'Run' button. Single click first line 'Run Project (OrderBookSimulation) F6'.



2. The console window of the simulation pops up, which allows the user to start, pause and run the simulation.
3. Click on the run button to start the simulation.
4. If need to stop or pause the run, click on the on the stop button or the pause button respectively.

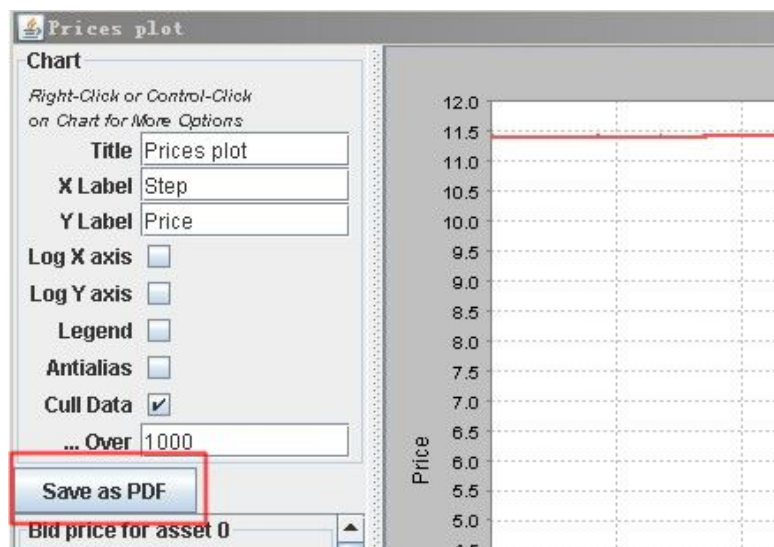


5. Click display tag to view the graphics.

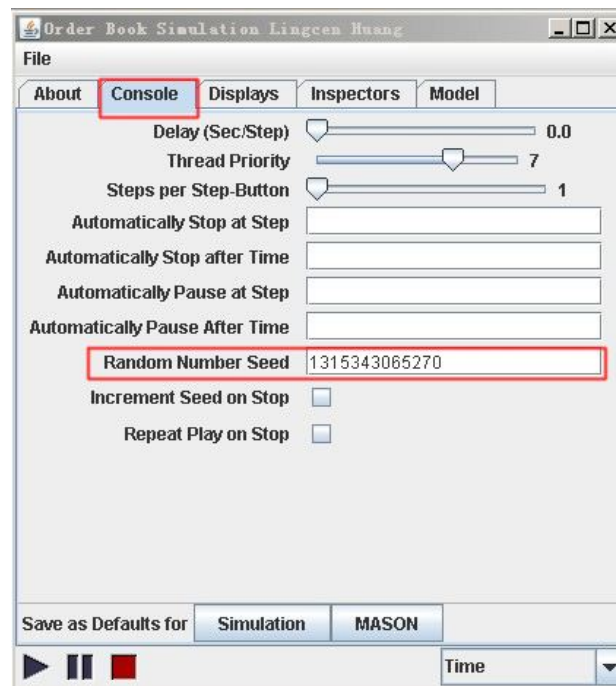


6. Double click the name of graphics to display them. The separate graphic for each one will pop up.

7. Click the button “Save as PDF” to save those graphics.



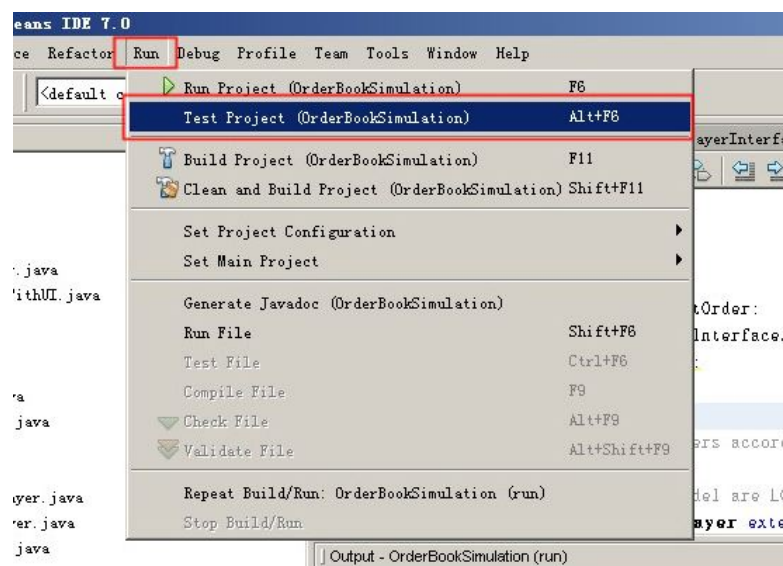
8. Click the ‘Console’ tab to change simulation speed and re-set random number.



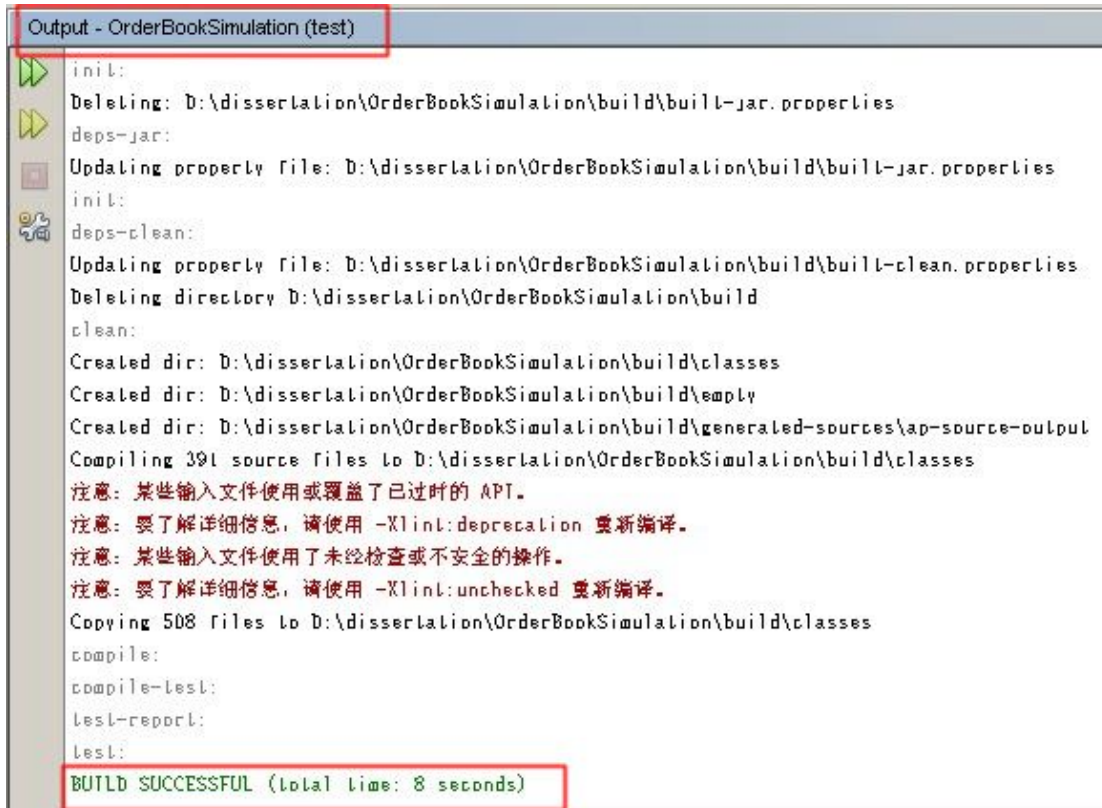
9. Close all pop up windows to finish simulation.

Test:

1. Go to NetBeans main menu and select ‘Run’ button. Single click second line ‘Test Project (OrderBookSimulation)(T)’.



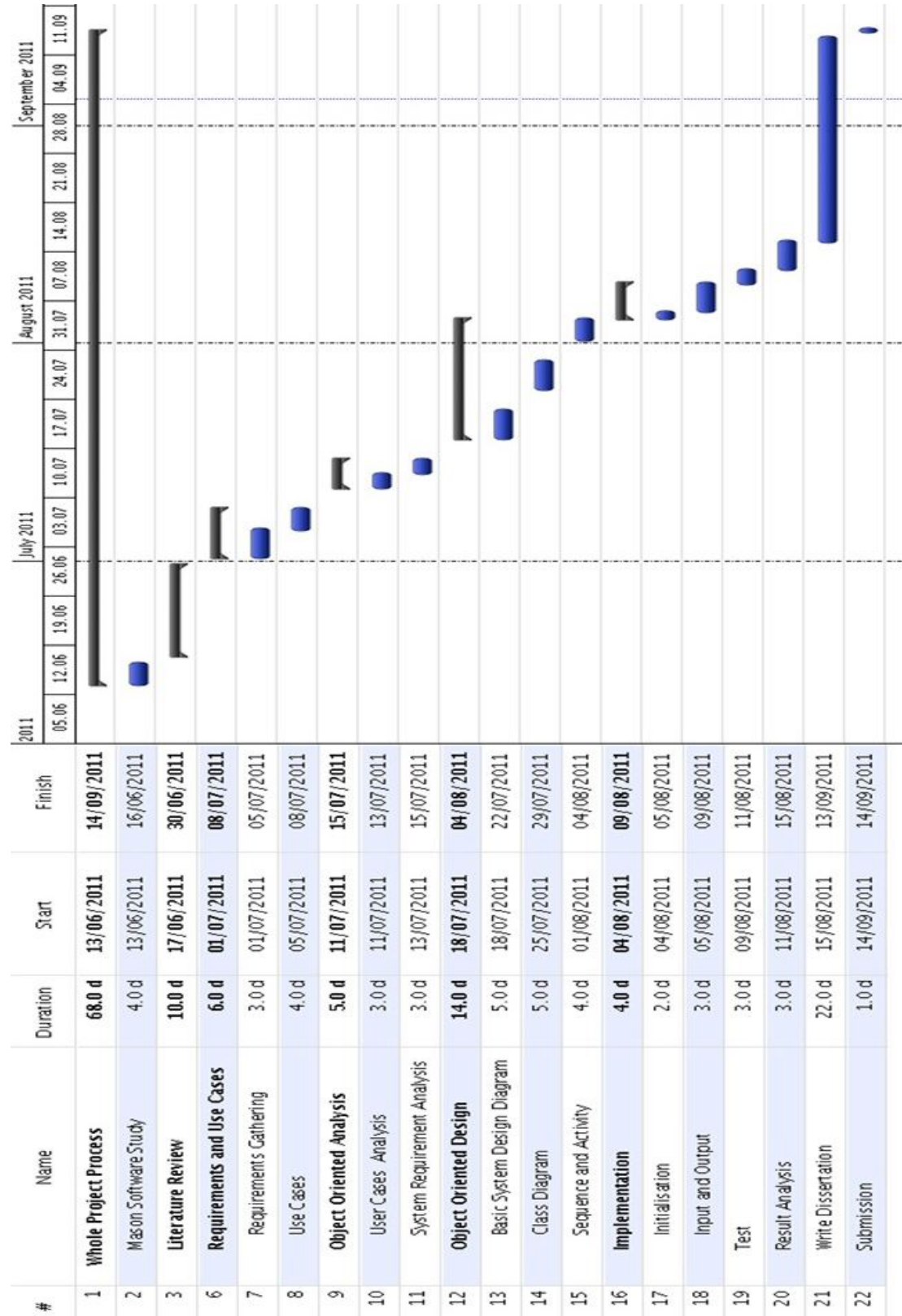
2. Wait for 8 seconds and test result will display in bottom window. The test result is displayed in below.



```
Output - OrderBookSimulation (test)
init:
Deleting: D:\dissertation\OrderBookSimulation\build\build-jar.properties
deps-jar:
Updating property file: D:\dissertation\OrderBookSimulation\build\build-jar.properties
init:
deps-clean:
Updating property file: D:\dissertation\OrderBookSimulation\build\build-clean.properties
Deleting directory D:\dissertation\OrderBookSimulation\build
clean:
Created dir: D:\dissertation\OrderBookSimulation\build\classes
Created dir: D:\dissertation\OrderBookSimulation\build\empty
Created dir: D:\dissertation\OrderBookSimulation\build\generated-sources\ap-source-output
Compiling 391 source files to D:\dissertation\OrderBookSimulation\build\classes
注意: 某些输入文件使用或覆盖了已过时的 API。
注意: 要了解详细信息, 请使用 -Xlint:deprecation 重新编译。
注意: 某些输入文件使用了未经检查或不安全的操作。
注意: 要了解详细信息, 请使用 -Xlint:unchecked 重新编译。
Copying 508 files to D:\dissertation\OrderBookSimulation\build\classes
compile:
compile-test:
test-report:
test:
BUILD SUCCESSFUL (total time: 8 seconds)
```

Appendix C: Project Plan

Gantt Chart



Project Schedule

Week	Description	Finished or Not
1	1 st Project Group Meeting Introduction to Mason Think about project topic	Finished
2	Review Mason tutorial Choose project topic Write objectives	Finished
3	Install the relevant software Set up	Finished
4	Run tutorial program under instruction	Finished
5	Literature review <ul style="list-style-type: none"> • Trading strategy • Multi-agent trading system model 	Finished
6	Project plan and Gantt chart 2 nd project group meeting 1 st individual meeting	Finished
7	Object oriented analysis 2 nd individual meeting	Finished
8	Object oriented design 3 rd individual meeting	Finished
9	Implementation 4 th individual meeting	Finished
10	Implementation Test 5 th individual meeting	Finished
11	Implementation Results analysis Write dissertation	Finished
12	Write dissertation	Finished
13	Write dissertation	Finished
14	Write dissertation	Finished

Appendix D: Source Code List

Because a lot of codes are developed in this project, the report will be too long if all of them are listed in appendix. The appendix D will only list all packages, class and other files names. The detail code of each class will be found in DVD enclosed.

Package:

GUIdisplay
model
model.agents
model.market
test

Java Class:

BookInterface
ContBook
ContPlayer
DisplaySupport
DoubleAuctionOrderBook
FarmerPatientPlayer
LimitOrder
MixPlayer
ModelSetting
PlayerInterface
RandomNum
SimulationModel
SimulationModelWithUI
TestValidation
TradingEnviroment

Other Files:

index.html
ucl_logo.jpg

Appendix E: Bibliography

- [1] Avellaneda, M, (2008). High-frequency trading in a limit order book. *Quantitative Finance*, 8, 217-224.
- [2] Bartolozzi, M, (2010). A multi agent model for the limit order book dynamics. *The European Physic Journal B*, 78, 265-273.
- [3] Biais, B., (1995). An Empirical Analysis of the Limit Order Book and the Order Flow in the Paris Bourse. *The Journal of Finance*, 5, 1665 -1779.
- [4] Black, C., (2010). Challenges for Equity Front Office Architecture. M.Sc. Dissertation. London: University College London.
- [5] Chen, S., (2007). Computationally intelligent agents in economics and finance, *Information Sciences*, 177, 5, 1153–1168.
- [6] Class Project Team, CSS 739 , 2009. Simulating Financial Markets using MASON Framework. Ph.D.. Center for Social Complexity: George Mason University, USA.
- [7] Cui, W., (2010). Evolving Efficient Limit Order Strategy using Grammatical Evolution. In 2010 IEEE World Congress on Computational Intelligence. Barcelona, U.S.A: IEEE Press. 2408--2413.
- [8] Cui. W., Brabazon A., O'Neill M., (2010). Evolving dynamic trade execution strategies using grammatical evolution, in *Proceedings of EvoFin 2010, Applications of Evolutionary Computation*, Lecture Notes in Computer Science 6025, Springer-Verlag, Berlin, 191–200.
- [9] Daniel G., (2006). Asynchronous simulations of a limit order book, Ph.D. Thesis, University of Manchester, U.K.
- [10] Ghoulmie, F, (2007). Effects of diversification among assets in an agent-based market model. In *Proceedings of the Complex Systems II Conference*. Australian National University, 4-7 December . ACT Australia: Canberra. 12.

- [11] Farmer, J., Patelli, P., and I. Zovko. I. (2005) The Predictive Power of Zero Intelligence in Financial Markets. ArXiv Condensed Matter, September 2003. Sean Luke, Claudio Cioffi-Revilla, Liviu Panait, Keith Sullivan, and Gabriel Balan. MASON: A Multiagent Simulation Environment. *Simulation*, 81:517–525.
- [12] Frank H. Westerhoff. Multiasset Market Dynamics. *Macroeconomic Dynamics*, 8(05):596–616, November 2004. URL http://ideas.repec.org/a/cup/macdyn/v8y2004i05p596-616_04.html.
- [13] Jiang, J., (2009). Revealing Intraday Market Efficiency -- Estimating Diurnal Price Densities in Limit Order Books. In ICIFE '09 Proceedings of the 2009 International Conference on Information and Financial Engineering. Washington DC, USA: IEEE Computer Society, 8-12.
- [14] Kissell R., Malanur R., (2006). Algorithmic decision-making framework, *Journal of Trading*, 1, 1, 12–21.
- [15] Lamba, H, (2008). Rational expectations, psychology and inductive learning via moving thresholds. *Physica A*, 387, 3904-3909.
- [16] London Stock Exchange, LSE, (2011). Rules of the London Stock Exchange. Effective1, 35-43.
- [17] Luke, S., (2011). Multi-agent Simulation and the MASON Library. PhD Thesis. USA: George Mason University.
- [18] Macal, Charles M., (2006). Tutorial On Agent-based Modeling and Simulation Part 2: How to Model with Agents. In *Proceedings of the 2006 Winter Simulation Conference*, U.S.A : IEEE Press, 73-91.
- [19] Marco, A, 2007. High-frequency trading in a limit order book. *Quantitative Finance*, Vol.8, 217-224.
- [20] Market model website - <http://svn2.assembla.com/svn/MarketModel/>.

[21] Maslov S., (2000). Simple model of a limit order-driven market, *Physica A*, vol. 278, pp. 571–578.

[22] Manyumwa, E, (2010). A Multi Agent Simulation of an Intraday Liquidity Market With the Use of Prediction Markets. M.Sc.. London: University College London.

[23] Preis, T., (2006). Multi-agent-based Order Book Model of financial markets. PhD Thesis. Germany: Johannes Gutenberg University of Mainz Staudinger Weg.

[24] Rama Cont., (2006) Long Memory in Economics, chapter Volatility Clustering in Financial Markets: Empirical Facts and Agent-Based Models. Springer Berlin Heidelberg.

[25] Rinaldo, A, (2004). Order aggressiveness in limit order book markets. *Journal of Financial Markets*, 7, 53-74.

[26] Rosu, I, (2009). A Dynamic Model of the Limit Order Book. *Review of Financial Studies* , 22, 4601-4641.

[27] Sanmay, D, (2008). The Effects of Market-Making on Price Dynamics. *Proceeding AAMAS '08 Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems* , 2, 24-32.

[28] Whigham, P.A., (2006). Evolving trading strategies for a limit-order book generator. In *Proceedings of the Congress on Evolutionary Computation (CEC)*. Barcelona , 18-23 July. New Zealand: IEEE Press. 1-8. (cannot be both in Barcelona and New Zealand. select the correct one!)

[29] Withanawasam R., Whigham P., Crack T., and Premachandra I., (2010). An empirical investigation of the Maslov limit-order market model, *Information Science Dept., Univ. of Otago, Tech. Rep. 2010/4*