Using ZEUS to Build a Multi-Agent Simulation System for Artificial Stock Market

Shi-Jen Lin
Department of Information Management, National Central University
No.300, Jung-da Rd., Jung-li City, Taoyuan, Taiwan 320, R.O.C
sjlin@mgt.ncu.edu.tw

Ying-Jia Lu
lisa9896@ms24.hinet.net

Abstract

Agent-based simulation is a new approach to the study of computer simulated financial markets. Santa Fe Artificial Stock Market (ASM) may be the most famous one among them. Many modifications of the ASM model have been made, but they rarely help with the actual details of designing a simulation framework or building a simulation system for it. Hence, we bring up an idea to utilize the ZEUS multi-agent toolkit to design and build a multi-agent simulation system, called MASS-Z. Besides, we also propose a general simulation model which is suitable for various financial market applications. And further we propose a mapping mode from the MASS-Z architecture to the general simulation model to provide an efficient way to implement the simulation applications. We believe that it will be a great help to the research workers who are not familiar with code writings to build various simulation systems readily.

Keywords : Agent-Based Simulation, Artificial Stock Market, Multi-Agent System, ZEUS.

1. Introduction

Agent technology is one of the most dynamic and exciting research fields in computer science today. The main areas of agent-based applications can be potentially divided into three broad categories: [13] (1) Assistant agents, (2) Multi-agent decision systems, and (3) Multi-agent simulation systems. For the multi-agent simulation systems applications, they are exactly in the research field of Agent-Based Social Simulation (ABSS) [4], which is characterized by the intersection of three scientific fields, namely Agent-based Computing, Social Science, and Computer Simulation. It can be said to investigate the use of agent technology for simulating social phenomena on a computer, or to provide models and tools for agent-based simulation of social phenomena.

The use of computer simulated markets with individual adaptive agents in finance is a new, but growing field. For a review in the last decade, the most famous one among those proposed computer simulated artificial financial markets is the Santa Fe Artificial Stock Market (ASM) [8][12]. So far, many modifications of the ASM model have been made, e.g. Swarm [3], but they rarely help with the actual details of designing a simulation framework or building a simulation system for it. Although there are some efforts supporting this kind of works recently, but generally, the common failings of them is the lack of interoperability, scalability, convenience and friendly user interface of the simulation system.

Hence, we bring up an idea to utilize the ZEUS multi-agent toolkit to design and build a multi-agent simulation system. We believe that applying multi-agent toolkits to construct this kind of system will greatly reduce the drawbacks that we mentioned above. And we believe that it will be a great help to the research workers who are not familiar with code writings to build various simulation systems readily.

2. Background

2.1 Agents Concepts

Here we discuss some definitions of what an agent is, and then introduce the essence of agency to avoid prescriptive arguments about how an agent should be. There's no general agreement as to what constitutes an agent, or as to how agents differ from programs. This situation is just like the very saying -- "The term ‘agent’ is so many that it is almost meaningless without reference to a particular concept of agent." [17] There are many different views of the “agent”, however, agents are characterized along certain dimensions, rather than defined precisely [6][19]. This leads to the concept of “Agency” or “Agentness” -- the essence of being an agent.

Russell and Norvig put it this way: "The notion of an agent is meant to be a tool for analyzing systems, not an absolute characterization that divides the world into agents and non-agents" [15]. Perhaps the most general way in which the term “agent” is used is to denote a hardware or software-based computer
system that enjoys the following properties [19]: (1) Autonomy, (2) Social ability, (3) Reactivity, and (4) Pro-activeness. And it is so called the weak notion of Agency. These characteristics are broadly accepted as representative of the key qualities that can be used to access “Agentness”.

Furthermore, for some researchers working in AI, they generally mean an agent to be a computer system that, in addition to having the properties identified above, is either conceptualized or implemented using concepts that are more usually applied to humans [19]. And it is so called the strong notion of agency.

2.2 Agent-Based System

An agent-based system may contain one or more agents. Although several agent-based systems now exist on the network, most are single-agent systems. However, the distributed, large-scale, dynamic nature of the Internet speaks to the need for multi-agent systems. Traditionally, research into systems composed of multiple agents was carried out under the banner of Distributed Artificial Intelligence (DAI), and has historically been divided into two main camps [7]: Distributed Problem Solving (DPS) [20] and Multi-Agent System (MAS). Recently, the term “multi-agent system” is now used to refer to all types of systems composed of multiple agents. MAS can be defined as a loosely coupled network of problem solvers that work together to solve problems that are beyond the individual capabilities or knowledge of each problem solver.

2.3 Artificial Stock Market

In the last decade, many computer simulated artificial financial market have been proposed. Among the numerous agent-based simulations of financial markets [11][18], the Santa Fe Artificial Stock Market model is one of the pioneering models and thus probably the most well-known and best studied [10]. The ASM model was made famous in a very widely cited article by R.G. Palmer, W. Brian Arthur, John H. Holland, Blake LeBaron, and Paul Taylor (1994). The ASM was one of the first projects to demonstrate the potential of agent-based models for the exploration of theories of decentralized, individualistic, and rational bounded behavior. It has sparked a small cottage industry of research on artificial stock markets [9].

2.4 Agent Toolkits

There is no universal definition of agent toolkits. Some toolkits may offer only a platform for agent development, whereas others may provide features for visual programming [16]. By the works of Serenko and Dettor, four major categories of agent toolkits were identified: Mobile Agent toolkits, Multi-Agent toolkits, General Purpose toolkits, and Internet Agent toolkits.

A well-known multi-agent toolkit is ZEUS, which is a convenient tool for developers who lack Java programming skill. ZEUS is a product of British Telecommunications Lab. It is open source, freely available software. The aim of ZEUS project is to facilitate the rapid development of new multi-agent applications.

2.5 Complex Adaptive Systems

Systems that evolve over time as individuals within the system interact with each other and respond to their environment are called Complex Adaptive Systems (CAS) [5]. CAS comprise a heterogeneous population of agents, these systems can self-organize and show emergent structures [2]. As to the ASM, it is claimed frequently that it is a complex adaptive system. It consists of a computer model in which simulated traders update an array of trading rules or forecasts over time. The stock price and the market behaviors are entirely endogenously generated. Sometimes, they are also named as agent-based modeling of stock markets, or bottom-up approach.

3. Multi-Agent Simulation System

The goal of the multi-agent simulation system that we design is to provide an experimental environment to study the price formation resulted from the ASM model. In the first sub-section, we discuss the design issues of the ZEUS methodology that we utilize to develop the simulation system. In the second
sub-section, we discuss some issues of the simulation methodology that we considered. In the third sub-
section, we propose the MASS-Z simulation system.

3.1 ZEUS Methodology

The aim of utilizing ZEUS in this paper is to facilitate the rapid development of new multi-agent
applications, and to create a customizable multi-agent simulation system that could be used by social
science research workers with only basic competence in agent technology. By illustrating the ZEUS
methodology, we describe the three-stage process of the ZEUS methodology: Domain Analysis, Agent
Design and Agent Realization processes.

3.1.1 Domain Analysis Process

The purpose of this stage is to model and understand the application problem. The domain
analysis approach used here is the Role Modeling. The role models formalize the definition of an agent role
so that it can be modeled, designed, and implemented. Agent roles and role models provide a vocabulary
for describing agent systems, with each role describing a position and a set of responsibilities. Role models
are also patterns of interactions, providing a readily comprehensible means of analyzing the problem in
question. Typically an agent will play many roles, and so during design the previously identified roles will
be composed together into agent entities that can support all of the relevant roles. Finally the functionality
associated with each role can be implemented.

3.1.2 Agent Design Process

It involves the translation of role responsibilities into the agent-level problems they represent, and
deriving appropriate solutions. The activities and outputs of the agent design process are illustrated in
Figure 1.

1) Domain Study: To analyze the problem domain to identify potential agents and to create ontology
of the concepts in the domain.

2) Agent Definition: It involves identifying the tasks that the agent can perform; identifying the
number of tasks the agent can perform concurrently; identifying how far ahead in time the agent
normally plans its activities; and identifying the initial resources available to the agent.

3) Task Definition: Tasks are defined in terms of their preconditions, effects, cost, duration, and
constraints. The preconditions list the resources needed for execution of the task, while the effects
list the resources that will be produced upon execution of the task. The cost and duration are
arithmetic expressions that return in some notional units. The constraints of a task either relate some
preconditions or effects of the task to one another.

4) Agent Organization: To acquaint an agent A with another agent B, the resources that agent A
believes agent B can produce are specified, along with the primary organizational relationship.

5) Agent Coordination: It involves identifying the appropriate coordination protocols each agent is
likely to require for social interaction with other agents when performing its designated duties.

6) Code Generation and Task Implementation: All the information necessary to automatically
generate source code implementations for each agent should be available. The Agent Building
Software contains a Code Generator Tool that can automatically generate individual agent programs
given their specification.

3.1.3 Agent Realization Process

The objective of this process is to realize working agent implementations from the conceptual
designs created during the previous stage by ZEUS tools. The agent realization process is illustrated as
follows:

1) Ontology Creation: The developer must firstly define the significant concepts, attributes and
values within the application domain. The tool used here is the ZEUS Ontology Editor.

2) Agent Creation: This process involves the ZEUS Agent Editor to complete four sub-stages: 1.
Agent Definition: specify its tasks, initial resources and planning abilities; 2. Task Description:
specify the applicability and attributes of agent activities; 3. Agent Organization: specify the social
context of each agent; 4. Agent Coordination: equip each agent with social abilities for interaction.
3) **Utility Agent Configuration**: This stage defines the attributes of the utility agents who provide the support infrastructure for the agent society. The tool used here is the Code Generation Editor.

4) **Task Agent Configuration**: This stage enables the runtime parameters of the task agents to be specified. The tool used here is also the Code Generation Editor.

5) **Agent Implementation**: The Code Generator can be invoked and agent source code automatically generated. This leaves the developer with the job of providing the application-specific implementations of the tasks, external resources, programs and interaction strategies. After this stage, the application is ready to be run.

![Diagram of agent design approach](image1)

![Diagram of MASS-Z simulation system](image2)

**Figure 1. The stages of the ZEUS agent design approach.**  
**Figure 2. The MASS-Z Simulation System.**

### 3.2 Simulation Methodology

It has been shown that the simulation techniques can be properly used in economics and financial markets. If the studied systems cannot be described completely by mathematics or they are analytically intractable, we can probably use simulation techniques to get better results. Market dynamics and macroscopic properties can be studied by modeling the microscopic details of participants. In such simulations, we specify elements of the systems and their interaction rules, and then we can reveal the macroscopic properties of such systems by observing these simulations and studying the simulation results with statistical methods. We discuss some important issues of microscopic simulations in the following sub-sections.

#### 3.2.1. Emergent Phenomena

Emergence is a coherent pattern that arises out of interactions among agents [14], or a by-product of individuals. Emergent results can be good as well as bad, and therefore must be considered when developing agent-based systems. Agents can work as non-interactive individuals or as a collective. When agents work as individuals with little or no interaction, what we get is just one thing that agents simply doing what they are asked to do. As a collective, however, something new and different can result what is more than the sum of the individual participants. Take the stock market for example, agents act independently to buy and sell shares of particular stocks and bonds. Yet from this independent behavior, an organism-like product called the stock market emerges. In other words, the rise and fall of the market is not controlled by a central process, but results from agents’ interactions. The stock market, its crashes, bubbles, and all are more than the sum of the individuals; it is an entity in its own right. Such entities are called **emergent structures** [14].

#### 3.2.2 Bottom-Up Approaches

The bottom-up approach starts with the parts of a system and then tries to figure out how the system
properties emerge from the interactions between these parts. Bottom-up models are computer simulations that describe the workings of CAS. These models are generally concerned with the system level properties that emerge from the actions and interactions of individual agents [5]. These agents are conceptually the bottom of the model and system level properties are built up from these individual agents as they respond to each other and their environment. The basic development of a bottom-up model is to define a population of heterogeneous, autonomous interacting agents. Parameters are specified at the agent level. The model is allowed to run to some endpoint and observations are made on system-wide entities that emerge from the actions and interaction of these agents over time.

3.3 MASS-Z

Firstly, we design the MASS-Z simulation system which is composed of Scenario Generator, Filter, Visualizer and the Simulation Framework, as illustrated in Figure 2. The goal of the MASS-Z is to provide an experimental environment to study the price formation mechanism resulting from the ASM model. Price formation in financial markets is a result of the interaction between the market mechanism and the trading decisions of multiple parties in the market [1]. In MASS-Z, it is illustrated in the MASS-F simulation framework.

The trading decisions depend on the available information which received from the Scenario Generator in MASS-Z. The resulting trade actions determine the price developments which then output to the Visualizer for simulation analysis and observed by users. Besides, only part of all the information can be available to a group of trading agents that select it in different ways. Hence we apply the Filter as an information filtering gateway. Depending on their individual goals, preferences and their interpretation of the available information, the agents take different trading actions. For the MASS-F, all the details of simulation model design were illustrated in next section; as to the Scenario Generator, it is composed of Domain Ontology, Task Agent Module and Utility Agent Module. For the understanding of these three components, let us make clear the MASS-Z Architecture first.

Secondly, we propose the MASS-Z Architecture as illustrated in Figure 3. The 4-layer simulation architecture represents the abstract conception of a process while using ZEUS to build a simulation system.

They are illustrated as follows:

1) **Domain Ontology Layer**: The developer analyses the problem domain to identify potential agents and to create ontology of the concepts in the domain. The domain ontology is namely the declarative knowledge that represents the significant concepts, attributes and values within the application domain.

2) **Agent Definition Layer**: The developer identifies the significant attributes of each agent. Basically the layer can be divided into two modules, namely Task Agent Module and Utility Agent Module. The Task Agent Module includes identifying the tasks that the agent can perform; identifying the number of tasks the agent can perform concurrently; and identifying the initial resources available to the agent. And Task Definition is part of the module; it defines tasks in terms of their preconditions, effects, cost, duration, and constraints. The Utility Agent Module defines the attributes of the utility agents who provide the support infrastructure for the agent society. There are
two other parts in this layer, the Agent Organization that defines the acquaintances of each agent and the Agent Coordination that defines the appropriate coordination protocols each agent is likely to require for social interaction with other agents when performing its designated duties.

3) Simulation Framework Layer: The developer designs the market mechanism and its internal structure. The details of this layer were illustrated in next chapter.

4) Simulation Analysis Layer: The participants in the financial markets have different attitudes to risk, different time horizons and motivations. Intelligent agents can be used to model these differences, as well as the learning and the adaptation behavior of the traders. It is possible to study explicitly the influence of dynamics at a microscopic level, and to study how these dynamics eventually lead to the macroscopic behavior observed at the markets.

Thirdly, we propose a general simulation model which is suitable for various financial market applications, as illustrated in Figure 4. We found that most kinds of financial market models could be interpreted as the combination of four parts: namely Simulation Framework, Market Module, Participant Module and Price Discovery Mechanism. The Simulation Framework defines the market mechanism and its internal structure. The Market Module defines the market structure of the application-specific domain. It includes the issues of members, time-grain, activities occur during one time step and assets to trade in the market. The Participant Module defines the agent structure of the market. It includes the definition of the tasks that agent performs, as well as the motivation, preference, and goals that agent possesses. The Price Discovery Mechanism defines the mechanism of the simulation system exploring financial contract price formation.

Finally, we propose a mapping mode as illustrated in Figure 4. While applying the general simulation model to analyze financial market applications, the four parts of the simulation model could be fulfilled through the mapping to the MASS-Z architecture. In other words, the Participant Module of the general simulation model could be implemented into the Task Agent Module of the MASS-Z architecture; and the Market Module of the general simulation model could be implemented into the Utility Agent Module of the MASS-Z architecture. We believe that this concept could provide an efficient way to implement the simulation applications.

4. Simulation Model Design

We address the design details of some important components in the general simulation model that we propose in the previous section.

4.1 Simulation Framework

The simulation framework for ASM in MASS-Z is named MASS-F, meaning the general framework for various available models of financial markets. The MASS-F consists of a central market ASM Agent, a Market Maker and some Trader Agents, as illustrated in Figure 5.

There are two assets in the market available for traders to invest in. One is the risk-free asset called money, and the other is the risky asset called stock. The trader agents choose between investing in a stock and leaving their money in the bank. Each Trader Agent expects the movement of the stock price, and then changes the position of the risk-free and risky assets so that the utility of his expected return may become the maximum. ASM Agent maintains the market time step and records the market price which is determined by the interactions of all the traders in each simulation cycle.

In MASS-F, the price development process consists of four steps: Predicting, Ordering, Pricing and Learning.

1) Predicting: Each Trader Agent expects the change value of the stock price of this round using the weighted sum of the change value of past stock price, and then confirms the desire demand for holding shares of the risky asset.

2) Ordering: If the utility function of expected return with risk avoidance of each Trader Agent is derived, the optimum quantity of the position of the stock asset is then determined.

3) Pricing: Market Maker accumulates all the orders from Trader Agents in the market, and the market price of this round is determined as the value where the demand meets the supply.

4) Learning: Each Trader Agent updates the expected return after the previous steps, and feedback to the first step for next round.
4.2 Market Module

The central market ASM Agent is responsible for the generating of market scenarios and price path. The basic components of the market are Trader Agents. They are heterogeneous in natural, and they don’t interact directly with each other, but only through the Market Maker, who acts as a broker and is responsible for the generating of market price and the liquidity providing.

In the market, time is discrete; and the time step is maintained by the ASM Agent. The money asset pays a constant interest rate \( r_f \), and the stock asset pays a dividend \( d_t \) per share at the start of time step \( t \) with price \( p_t \). In order to understand the market well, it will be helpful to make a description of the cycle during one time step in the market as illustrated in Figure 6.

At the start of any time step \( t \), the current dividend \( d_t \) is posted by ASM Agent, and observed by all Trader Agents. They use this information and, in addition, the general information on the state of the market to form their expectations of the next period’s price and dividend. They then calculate their desired holdings of the stock and pass their demand parameters to the Market Maker. The Market Maker then declares a price \( p_{t+1} \) according to the equilibrium decided by supply and demand mechanism of the market.

At the start of the next period the new dividend \( d_{t+1} \) is revealed, and the accuracies of the predictors’ active at time step \( t \) are updated. The sequence repeats till the end of the time horizon.

5. System Implementation

5.1 Specification

We implement MASS-Z by Java programming language in Windows 2000 Server operating system on platform with P III-850 CPU and 512 MB RAM. To keep the system small-scale and for simplicity, we assume there are 3 types of participants (Trader Agents) in the market, and each of them are capable of predicting, buying, selling and learning abilities. The trader agents are all initially endowed with 1 shares of risky stock and 1,000 units of money. They have to decide how much to invest in risky stock and how much to keep in money asset. In addition, we assume that the dividends of stock asset and the interests return are all paid in cash.

5.2 Application Analysis
As the trader agents of MASS-Z exhibit trading behaviors, this application is best situated within the Multi-Agent Trading Domain of the ZEUS Role Modeling Guide [21]. This domain contains two role models: Distributed Marketplace and Institutional Auction. From the comparison between them, we base our solution on the Distributed Marketplace role model. And we create some agents, as shown in Table 1, to fulfill the roles found within the role model.

<table>
<thead>
<tr>
<th>Agent Name</th>
<th>Roles Played</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trader Agent 1</td>
<td>Trader (Buyer, Seller, Inquirer, Registrant)</td>
</tr>
<tr>
<td>Trader Agent 2</td>
<td>Trader (Buyer, Seller, Inquirer, Registrant)</td>
</tr>
<tr>
<td>Trader Agent 3</td>
<td>Trader (Buyer, Seller, Inquirer, Registrant)</td>
</tr>
<tr>
<td>Market Maker</td>
<td>Broker (Facilitator)</td>
</tr>
<tr>
<td>ASM Agent</td>
<td>Agent Name Server</td>
</tr>
<tr>
<td>Visual</td>
<td>Visualiser</td>
</tr>
</tbody>
</table>

Each role played by an agent takes some responsibilities; we use the role descriptions to create a list of responsibilities for each agent. The responsibilities can be categorized as Social and Domain Responsibilities, the former involving interactions with other agents, and the latter involving some local application-specific activities. This results in Table 2 and Table 3.

5.3 Application Design

The application design process is a process of refinement, mapping each of the responsibilities identified in the previous stage to a generalized problem, and then choosing the most appropriate solution. Here, we take the Market Maker for example and summarize in Table 4.

<table>
<thead>
<tr>
<th>Market Maker – Social Responsibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Origin Role</td>
</tr>
<tr>
<td>Broker</td>
</tr>
<tr>
<td>Broker</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Market Maker – Domain Responsibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Origin Role</td>
</tr>
<tr>
<td>Broker</td>
</tr>
</tbody>
</table>

| Origin Role | Responsibility |
| Broker | Receive and Store |

| Problem: | Solution: |
| Receive and Store | Configure the ZEUS Facilitator Agent |
| Match Requirements and Send Message | Functionality provided by Facilitator Agent |

<table>
<thead>
<tr>
<th>Domain Responsibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Origin Role</td>
</tr>
<tr>
<td>Broker</td>
</tr>
</tbody>
</table>

| Problem: | Solution: |
| Store and Process | This functionality is not provided by ZEUS Facilitator Agent, it should be attached extra after the Code Generation stage. |
5.4 Application Realization

The purpose of these stages is to translate the design we have derived from the role models into agent descriptions that can be automatically created by the ZEUS Agent Generator tool. The realization process, as described in chapter 3, consists of some activities and is described in the following subsections respectively.

5.4.1 Ontology Creation

This stage involves using the Ontology Editor tool to define the ASM Ontology, such as the significant concepts, attributes and values within the Artificial Stock Market. The ASM ontology and the content of the Ontology Editor are shown in Figure 8.

![Figure 7. The Problem Ontology of the ASM model](image7.png)  
![Figure 8. The Ontology of the Artificial Stock Market](image8.png)

Figure 7. The Problem Ontology of the ASM model  
Figure 8. The Ontology of the Artificial Stock Market

![Figure 9. The Agent Definition details of the Trader Agent.](image9.png)  
![Figure 10. The specification of the utility agents.](image10.png)

Figure 9. The Agent Definition details of the Trader Agent.  
Figure 10. The specification of the utility agents.
5.4.2 Agent Creation

At the stage, the generic ZEUS agent is configured to fulfill its application-specific responsibilities. After this stage, some design decisions should have been taken, such as: What is the time-grain for the application? What agents exist? What activities will each agent perform? How will each agent interact with other agents? This stage consists of several sub-processes that are repeated for each different task agent in the application. Here, we take the Agent Definition Process for example as illustrated in Figure 9.

5.4.3 Utility Agent Configuration

We decide what utility agents will be created and then configure them accordingly at this stage. Next, we configure the runtime parameters of these agents. This is achieved through the “Utility Agent” tab pane as illustrated in Figure 10.

5.4.4 Task Agent Configuration

The application design process states that some functions are not provided by ZEUS, they should be implemented through the external programs. Except that, the configuration of task agent is illustrated in Figure 11.

5.4.5 Code Generation and Implementation

At last, with all agents defined, we can generate their implementations from the “Generation Plan” tab pane as illustrated in Figure 12. This will create Java code for each agent and we can find the source code files in the directory where we specified. Then we should work with the new generated code to implement the application specific components that proposed by our design.

6. Conclusions

We bring up an idea which applying the multi-agent toolkit to design and build a simulation system that not used before. We use the ZEUS multi-agent toolkit to design and build a multi-agent simulation system, called MASS-Z. And we propose a MASS-Z architecture which extracted from ZEUS methodology. We also propose a general simulation model which is suitable for various financial market applications. And further we propose a mapping mode from the MASS-Z architecture to the general simulation model to provide an efficient way to implement the simulation applications. Besides, we found that ZEUS not only provides a well-defined methodology for the design of the multi-agent systems, but provides reusable components, convenience tools and friendly user interface for the implementation of the multi-agent systems. It could improve the efficiency of the development process. Hence we believe that
applying the multi-agent toolkit to construct the simulation system will be a great help to the research workers who are not familiar with code writings to build various simulation systems readily.

However, during the development process, the MASS-Z simulation system does face some troubles. First, MASS-Z is a simulation system based on ZEUS; hence it is restricted by ZEUS to some extent. ZEUS is not developed to solve the simulation applications by birth, so it is confronted with some difficulties. In fact, there are lots of improvements to do for the ZEUS toolkit to meet the very goal that we propose in this paper. Second, MASS-Z is the first developed simulation system that integrated with multi-agent toolkits; hence the architecture and framework we propose in MASS-Z might not be complete and perfect in consideration. We hope that in the near future we can make a breakthrough in these works.

Towards our system’s limitations and drawbacks, we propose the possible improvements to do in the future, where we would like to:

1) Support the ZEUS Simulation Role Model at the Application Analysis stage. The role models that pre-built in ZEUS Role Modeling Guide [21] are not really suitable for the simulation applications; hence they should be adjusted to meet the purpose that we set.

2) Improve the functionalities of the Task Agent. The configuration of the tasks also needs to enhance. For example, as we have no clear idea of how to design a task of a Trader Agent, we have to try out all the possible methods that ZEUS provides. In some cases, the methods we tried would not work out at all, and we have to implement it through external programs.

3) Support more available negotiation strategies. ZEUS only provides two negotiation strategies, and it’s not sufficient for many multi-agent systems. In this case, we wish we can develop the agent’s learning strategy which integrated with the genetic programming technique in the future.

Reference