A Multi-Agent Decision Support System for Stock Trading

Yuan Luo, Kecheng Liu, Staffordshire University
Darryl N. Davis, University of Hull

Abstract

A distributed problem solving system can be characterized as a group of individual cooperating agents running to solve common problems. As dynamic application domains continue to grow in scale and complexity, it becomes more difficult to control the purposeful behavior of agents, especially when unexpected events may occur. This article presents an information and knowledge exchange framework to support distributed problem solving. From the application viewpoint the article concentrates on the stock trading domain; however, many presented solutions can be extended to other dynamic domains. It addresses two important issues: how individual agents should be interconnected so that their resources are efficiently used and their goals accomplished effectively; and how information and knowledge transfer should take place among the agents to allow them to respond successfully to user requests and unexpected external situations. The article introduces an architecture, the MASST system architecture, which supports dynamic information and knowledge exchange among the cooperating agents. The architecture uses a dynamic blackboard as an interagent communication paradigm to facilitate factual data, business rule, and command exchange between cooperating MASST agents. The critical components of the MASST architecture have been implemented and tested in the stock trading domain, and have proven to be a viable solution for distributed problem solving based on cooperating agents.

In a very general sense Distributed Problem Solving (DPS) assumes that complex and large-scale problems can be solved by organizing an assembly of cooperating experts (agents) possessing complementary problem solving skills. This approach has demonstrated results in medical diagnosis [1], industrial product design [2], knowledge acquisition of and fault root cause analysis in networks [3], pattern analysis and visual shape recognition [4], and natural language understanding [5].

As dynamic application domains such as stock trading continue to grow in scale and complexity, it becomes more difficult to control the behavior of agents in situations where unexpected events may occur. In recent years, there has been considerable growth of interest in the design of intelligent agent architectures for problem solving in dynamic and unpredictable domains. Most of today's intelligent agent architectures of distributed problem solving are limited to performing preprogrammed or human-assisted tasks in relatively static and predictable domains.

In multi-agent systems the agents should be able to interact with each other, and with the external environment in an adaptable manner by adjusting their behavior to the changes occurring in the environment. Each agent has a local view of the environment; generally it has been provided by specific operational goals, and it is known that the agent is unable to solve the system tasks alone, at least with the quality, efficiency, resources, and other constraints defined by the problem. The new global characteristics of such multi-agent system emerge from the behavior of its components (i.e., the cooperating agents). This cooperation, in turn, impinges on the interactions between the agents and subtly modifies the properties of the overall system [6]. In order to be more useful in complex real world domains, the agents need to be flexible in terms of their problem solving skills, communication capabilities, and utilization of internal knowledge and data.

It is highly possible that an agent with responsibility in a dynamic environment will face unexpected events. In order to be responsive, the agents should have enough knowledge to deal with these unexpected events. If an agent is not able to deal with a particular event on its own, it can take the following actions:

- Learn how to solve the problem by experimenting with alternative problem solving strategies
- Let some other knowledgeable agent solve the problem and then use the results
- Learn how to solve the particular problem by acquiring the necessary knowledge from other agents capable of solving the problem
- Ignore the unexpected event [7]

For real-time application domains such as stock trading, action 1 may not be suitable because it may take a long time to obtain the solution. Action 2 is a natural way for cooperating agents to solve the problem. However, it is not suitable for scenarios where there are large volumes of data to be processed. For action 3, there may be no large network transfer, but the agents themselves must be quite sophisticated. They need temporarily (maybe even permanently) to keep the knowledge acquired from other agents, and then learn how to use this knowledge to solve the particular problem.

0890-8044/01/$17.00 © 2002 IEEE

For building a distributed multi-agent system capable of solving complex and large-scale real-time stock trading tasks, we have developed the Multi-Agent System for Stock Trading (MASST) [8]. Although developed for stock trading, the MASST architecture extends beyond the stock trading domain. MASST is a closely collaborating agent system in which every agent has its own specialized capabilities and knowledge, and no one agent has the whole knowledge about the world. All the task-specific MASST agents are situated on the same machine. Hence, we do not need to worry about huge volume data transfers over a network. Based on the discussion above, the MASST agents will take actions 2 and 4 to deal with a particular event. The objective of this article is to investigate and recommend a framework to support distributed problem solving for action 2.

In this article we address two important issues:

- How individual agents should be interconnected so that their capacities are efficiently used, and their goals are accomplished effectively and efficiently
- How the information and knowledge transfer should take place among agents to allow them to respond successfully to users' requests and unexpected situations in the outside world

The breakdown of the rest of this article is as follows. We give some background about the applications of agents in the stock trading domain. We briefly describe the MASST framework. We discuss in detail the information and knowledge exchange framework used in the MASST agents. We describe MASST implementation and experiments. We then conclude this article.

Background

Intelligent agents are software programs that act on behalf of human users or other systems in order to carry out arduous information gathering and processing tasks, such as locating and accessing information from various information sources, resolving inconsistencies in the retrieved data, filtering away irrelevant or unwanted data, and integrating information from heterogeneous information sources. Agents can automate repetitive tasks, notify users on upcoming events or system changes, aggregate and summarize complex data, learn from past behavior, and even make recommendations to the user on alternative solutions.

The agent technology has proven to be suitable to address issues concerning portfolio management. Sycara *et al.* [9] pointed out that this is the task of providing an integrated financial picture for managing an investment portfolio over time, using the information resources already available over the Internet. The portfolio management domain has many interesting characteristics, including:

- The enormous amount of continuously changing and generally weakly organized data
- The variety of kinds of information that can and should be brought to bear on the task (market data, financial report data, technical models, analysts' reports, breaking news, etc.)
- The many sources of uncertainty and dynamic changes in the environment

The overall task of portfolio management is to provide the best possible rate of return for a specified level of risk, or conversely, to achieve a specified rate of return with the lowest possible risks. A multi-agent system approach is natural for portfolio monitoring, because the multiple control threads in such a computational model are a natural match for the distributed and ever-changing nature of the underlying sources of data and news that affect higher-level decision-making process. A multi-agent system can more easily manage the detection and response to important time-critical information that could appear suddenly at any of a large number of different information sources. A multi-agent system provides a natural mapping of the multiple types of expertise to be brought to bear during any portfolio management decision making.

Rus and Subramanian [10] presented a customizable architecture for software agents that capture and access information in large heterogeneous distributed electronic repositories. The key idea is to exploit the underlying structure at various levels of granularity to build high-level indices with task-specific interpretations. Information agents construct such indices and are configured as a network of reusable modules called structure detectors and segmenters. The proposed architecture is illustrated by design and implementation of smart information filters in two contexts: retrieving stock market data from Internet newsgroups and retrieving technical reports from Internet FTP sites.

Delgado *et al.* [8] investigated a hybrid learning system that combines different fuzzy modeling techniques. In order to implement the different methods, they proposed the use of intelligent agents, which collaborate by means of a multi-agent architecture. This approach, involving agents that embody the different problem solving methods, is a potentially useful strategy for enhancing the power of fuzzy modeling systems. Working with stock markets requires constant monitoring of the continuously changing stock information, and the ability to make decisions instantaneously based on certain rules as the changes occur. Garcia *et al.* [12] recently reported a framework for implementing a deliberative multi-agent system for this domain. This system can be used as a proactive tool for expressing and implementing high-level stock trading strategies. In the framework, agents are able to monitor and extract the stock market information via the World Wide Web and, using the domain knowledge provided in the form of defeasible rules, can reason in order to achieve the established goals. The overall system is integrated using Java Inference Engine and Networked Interactor (JINNI), which provides a platform for building intelligent autonomous agents [13], and Defeasible Logic Programming (DeLP), which provides the agents with the capability of reasoning using defeasible rules in a dynamic domain.

Even though there are several agent-based approaches reported in literature that address the issues in the financial trading domain, most of the current agent-based approaches focus on how to get the information from a distributed source (the Internet). The use of intelligent agents to support decisions has not been thoroughly explored and merits serious consideration. In current practice, portfolio management is carried out by investment houses that employ teams of specialists for finding, filtering, and evaluating relevant information. Based on their evaluation and on predictions of the economic future, the specialists make suggestions about buying or selling various financial instruments. The current practice, as well as software engineering considerations, motivates our research in multiple-agent systems for stock management. A multiple-agent system approach is natural for portfolio management because of the multiplicity of information sources and the different expertise that must be brought to bear to produce a good recommendation for a stock buy or sell decision.

MASST Framework

MASST is a middle-layer agent system between the demand side of information (i.e., investors in the stock market) and the supply side of information (i.e., the Internet). Figure 1 shows the MASST scope and its context.

The major functions of MASST include:

- Stock information retrieval — a basic required data provi-

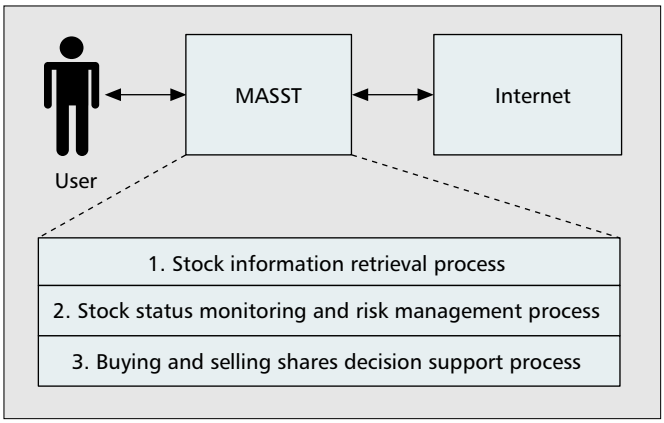


Figure 1. *MASST scoped organizational context.*

sioning function and a must for every commercial stock analysis software. MASST provides four categories of information retrieval:

–Trading history and current quotations, such as the information of opening price, highest price, lowest price, current price, abd trading volume for the selected stock on a given trading date

–Browsing stock technical analysis charts, such as a price movement chart (e.g., Candlestick Chart), trading volume chart, and various technical indicators chart

–Listed company's fundamental data and financial health information retrieval, such as total amount of stock trading volume, after-tax profits, earnings per share, annual earnings increases, number of common shares, new products or services, new management, and market news

–Market statistics information retrieval, such as the list of top 10 shares of maximum trading volume in the given trading day(s), the list of top 10 shares of maximum price upward (or downward), and the list of top 10 of the lowest price-earning ratios

- Stock status monitoring and risk management — MASST will automatically monitor the market status of the shares the user holds and is interested in. The share's market status includes the listed company's fundamental financial status and the status of the share's technical indicators. Based on the share's market status, MASST will automatically and promptly report any abnormal status to users. Indicators of abnormal status include:

  –Abnormal price fluctuation

  –Abnormal trading volume

  –Abnormal technical indicator's status

  –Abnormal price chart pattern

  –Some breaking news relating to the given shares

Furthermore, MASST will provide the profit and risk management, including calculation of profits/risk ratio based on shares' market status and user's investment, and reminders of stop-loss level for holding shares according to the user's profile.

- Buying and selling shares decision support — From the investors' perspective, the most important and concerning issues for investment in the stock market are buying share issue and selling share issue. Which share is the best one to buy? What time is the right time to buy the share? What time is the right time to sell your holding shares? It is very often difficult (maybe impossible) to find simple and accurate answers to these kinds of questions. Every investor has his/her own buying and selling share strategies and rules. MASST will provide buying and selling decision support based on business rules defined by the investors themselves. Through a combination of human and machine knowledge, using agent technologies, MASST aims to reduce investors' work overload in the process of stock analysis and investment decision making.

MASST provides a unified environment (Fig. 2) in which several agents are integrated. These intelligent agents interoperate to collect, filter, and fuse information from distributed network-based information sources, and to make buying and selling decision suggestions for investors in their daily stock trading.

One of the key components in this framework is the User Profile Database (UPDB), which is dynamic, changing, and shared among agents within the system. Each user has his/her own personalized interface agent, an individual user profile, and a trading strategy database (TSDB) that is the user's private trading strategies, while other agents in the system are shared by all users. The UPDB includes information such as username, password, the stock list the user possesses, the stock list in which the user is interested, monitoring instructions, planned tasks, preferences, and privacy settings. These agents are assigned to an individual user and must be able to learn a user's interests and behavior autonomously and adapt to the changing needs of the user over time. The profile is centrally available to all user agents.

The history trading database (HTDB), real-time trading database (RTDB), and fundamental information database

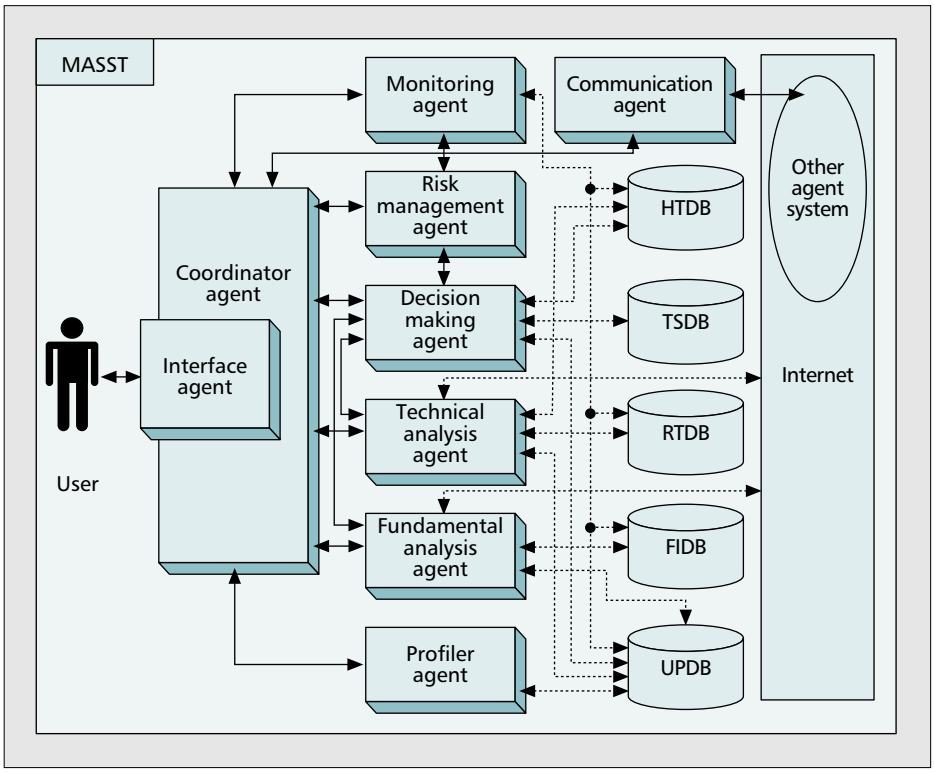


Figure 2. *MASST framework.*

(FIDB) are the internal data resources, continuously updated with relevant external information by the technical analysis agent (TAA) and fundamental analysis agent (FAA). The functions and relationships among the agents in MASST are as follows:

- The interface agent (IA) interacts with the user, receiving user tasks and specifications and delivering results. The IA passes the user's tasks to and gets returns from the coordinator agent.
- The coordinator agent (CA) is responsible for task decomposition and planning. The CA maintains a set of beliefs about the capabilities of all agents in MASST. It can decompose a given task into a number of subtasks and dispatch the subtasks to relevant agents to execution, in order to achieve its goals.
- The profiler agent provides the mechanism by which a user's profile and TSDB are generated and maintained. The profiler agent interacts with the CA to receive information from the user and the environment to determine the interests of the user.
- The monitoring agent monitors the status of the given stocks on behalf of users according to their profiles. This agent reports on the technical indicators status of the given stocks and notifies any abnormal changes in trading volume and price.
- The communication agent allows the framework to interact or communicate with other agents or programs developed by other developers. This is a reserved interface to other systems.
- The risk management agent (RMA), on the basis of the user profile, interacts with the monitoring agent and decision-making agent to analyze the risk levels of the user's share holdings, reports the profit status, and suggests a stop-loss level for the holding shares.
- The decision making agent (DMA) combines the outcomes of the TAA and FAA, according to the investment strategies selected through the user's TSDB. DMA will have two main functions:
  - –To give a list of stocks advised for the next trading day to buy
  - –To give suggestions for users holding shares to hold or sell
- The technical analysis agent (TAA) retrieves and processes the raw stock trading data from the Internet, stores the raw data to a relevant database (HTDB, RTDB), calculates various technical indicators, identifies various price and trading volume patterns, and gives the output to the DMA.
- The fundamental analysis agent (FAA) gathers the macroeconomics data, fundamental financial status of the listed companies, opinions of market commentators or experts, and some relative news, and puts this information into the FIDB. The fundamental analysis agent integrates the information and makes recommendations to the DMA.

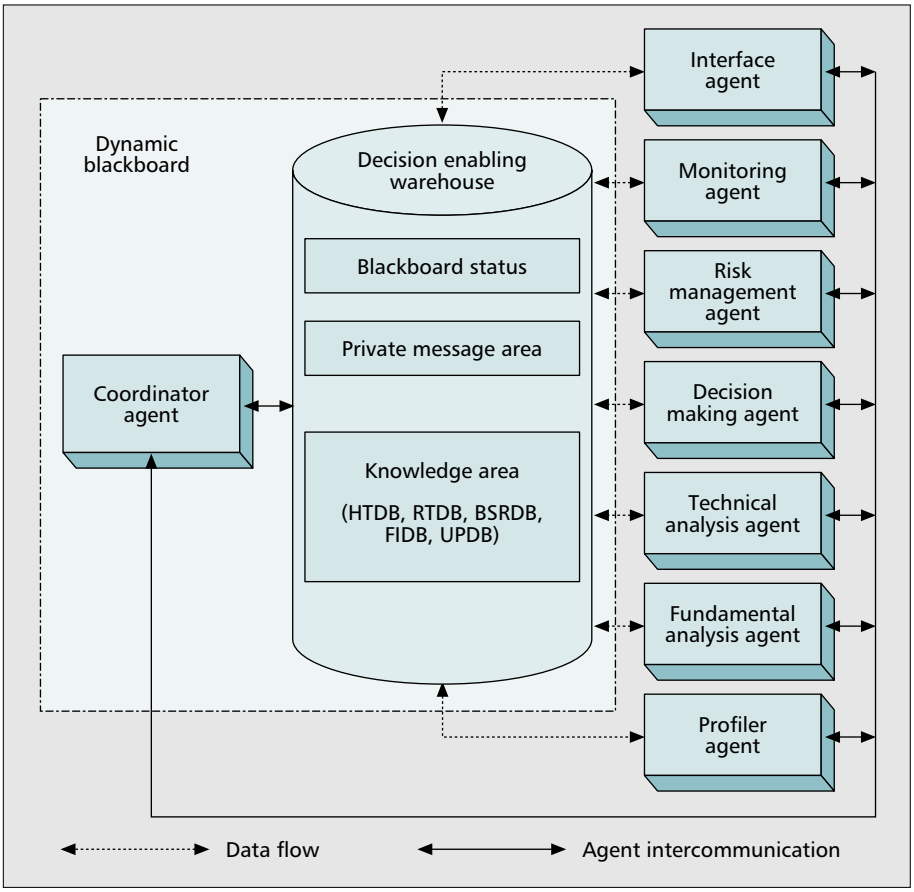


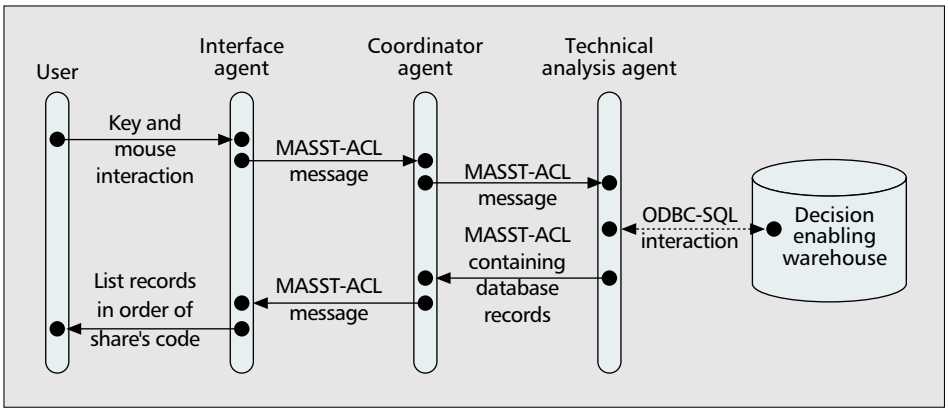
Figure 3. *Intercommunication architecture among MASST agents.*

Information and Knowledge Exchange in MASST

An Overview of the MASST Intercommunication Architecture

Agent-based approaches to decision-making applications are becoming more attractive in the business and commercial domain [14]. An intelligent agent operating in a dynamic environment will find its reasoning and other actions constrained by limitations of time, information, and other critical resources. These agents have to interact with dynamically changing and partially unknown environments, so they must be able to respond to unanticipated changes and events occurring in their operational environment. In such situations, agents must know what actions need to be taken. If the agents are not able to deal with the unexpected situation, they should seek help from other agents in the system to resolve the problem, as discussed in the first section.

Figure 3 shows the intercommunication architecture among MASST agents. As we stated earlier, the CA must have access to knowledge that models the functions and task capabilities of the other agents. Information relevant to the system's current task is held by the CA, which exchanges it with all the other agents. The CA is also responsible for ensuring that the data used within the framework is reliable and available as integrated information held in the decision enabling warehouse (DEW). From the perspectives offered by blackboard system design [15–17], the DEW together with the CA acts as a dynamic blackboard. The blackboard is a data structure that can be shared by all the intelligent agents simultaneously, and the coordinator agent is similar to the control in traditional blackboard systems. The blackboard is organized by the CA.



■ Figure 4. *Agent interaction through ODBC-SQL and MASST-ACL communication in response to the SA1 request.*

The DEW has three different sections:
- Blackboard status, which holds the processing status of the current task.
- Blackboard private message area, used by the CA to send task-related messages to other MASST agents, and used by other MASST agents to control their conversation session; when a task or a conversation is complete, the relevant messages in this area are deleted.
- Blackboard knowledge area, which holds rules, facts, and raw data to be used by the MASST agents. The knowledge area comprises the HTDB, RTDB, TSDB, FIDB, and UPDB.

There are four communication channels in the MASST framework:
- Internal agent–internal database
- Internal agent–internal Agent
- Internal agent–external data
- Internal agent–external agent

The communication protocol depends on the type of agents involved, and the information and knowledge being exchanged. The purpose of the internal agent–external agent category is to make the MASST an open system, whereby it can extend its functions. The technical issues involved here include ontology, agent communication language, security, and so on. The purpose of internal agent–external data is to collect and filter the necessary information from distributed data resources (e.g., the Internet). Much research has been done in this area [18–22] and is not repeated here. The focus of this work is on how to use the information for decision making rather than how to collect it. Thus, we only discuss the first two communication channels in this article.

Agent-to-database communication is based on ODBC using SQL requests and commands. This protocol is relatively straightforward. An agent connects to an active database via ODBC, and phrases information requests using SQL. In the MASST framework, the internal agents use this to obtain and place information on the DEW. Internal agents communicate with each other using the MASST Agent Communication Language (MASST-ACL), which is a combination of FIPA ACL [23] and XML [24]. The details about MASST-ACL are beyond the scope of this article. According to the description earlier, the major functions of MASST include:
- Stock information retrieval
- Stock status monitoring and risk management
- Buying and selling shares decision support

In the following subsections, the agent interactions will be discussed in detail for these processes.

Agent Interactions in the Stock Information Retrieval Process

MASST can provide seven functions of stock information retrieval based on the search actions (SAs) the user delegated. These include:
- SA1 — Query all share's quotation on the current day
- SA2 — Query a given share's quotation on the current day
- SA3 — Query a given share's real-time trading chart
- SA4 — Query a given share's history price chart over a period
- SA5 — Query a given share's price and technical indicator chart over a period
- SA6 — Query a given share's fundamental analysis data
- SA7 – Query the market statistic information over a period

Figure 4 shows how the agents interact and communicate in response to a request of SA1. A user delegates the task to the IA, which then passes the request to the CA using a MASST-ACL message by packaging the function ID (e.g., SA1) and the relevant parameters (e.g., Market-Code and Current-Date). The CA will distribute the task to the relevant task-specific agent based on the function ID. In this example it is the TAA. The TAA then interacts with the DEW through ODBC-SQL to obtain the database records. Then the TAA sends a MASST-ACL message containing database records to the CA. The CA then passes the message to the IA; in the meantime, the CA will update the blackboard status and delete the relative messages associated with this conversation session in the private message area of the blackboard, since this conversation (or task) is complete. Finally, the interface agent presents the results in a readily understandable format to the user.

For the other stock information retrieval functions (i.e., SA2–SA7), similar agent interaction scenarios exist. Only the task-specific agents and the information formats presented to the user by the interface agent differ.

Agent Interactions in the Stock Status Monitoring and Risk Management Process

For the stock status monitoring and risk management process, the scenario of agent interactions is much more sophisticated than that shown in Fig. 4. MASST will automatically monitor the market status of the given shares. Based on the share's market status and monitoring actions (MAs) given by the users, MASST will also automatically report any abnormal status to the users. The MAs include:
- MA1 — Monitoring abnormal price fluctuation
- MA2 — Monitoring abnormal trading volume
- MA3 — Monitoring abnormal technical indicator's status
- MA4 — Monitoring abnormal price chart pattern
- MA5 — Monitoring breaking news relating to the given shares

Furthermore, MASST will provide profits and risks management, which includes calculation of profits/risk ratio based on shares' market status and user's investment, and reminders of the stop-loss level for a user's holding shares according to the user's profile.

As we discussed earlier, the DEW together with the CA acts as a dynamic blackboard in the system. Every MASST agent can watch the information change (or an event) on the blackboard, and an agent will react to or ignore the event according to the agent's responsibility. Figure 5 shows the

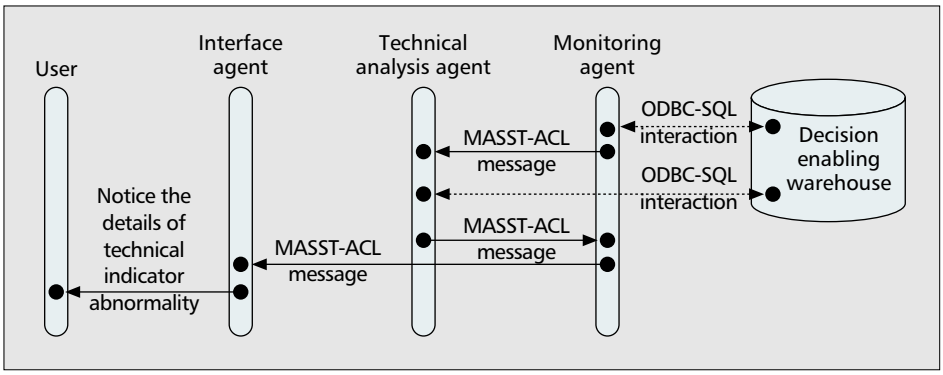


Figure 5. *Agent interactions for monitoring technical indicator abnormality.*

MASST agent interactions for monitoring the abnormal technical indicator. The MA requests the user profile database in the DEW to get the share codes needed for monitoring and the user's monitoring instruction (i.e., MA3). According to the monitoring instruction, the MA will ask the TAA for help in finding out the current value of the given shares' technical indicator by sending a MASST-ACL message with function ID (e.g., MA3) and parameters (e.g., share codes, technical indicators). Based on the function ID, the TAA interacts with the HTDB through ODBC-SQL and calculates the value of the technical indicators on behalf of a monitoring agent, then returns the results to the monitoring agent. The MA compares the value obtained from the TAA with the predefined appraisal thresholds of abnormal technical indicators that are obtained from the user profile database. If an abnormal event occurs, the ma will immediately send a MASST-ACL message to the IA to notify the user of abnormal indicators.

Agent Interactions in Buying and Selling Share Decision Support Process

For the buying and selling share decision support process, the scenario of agent interaction is variable in the MASST. It largely depends on the buying and selling rules that were defined by the users themselves. MASST provides the user with an interface to set up the TSDB, which holds the user's buying and selling rules (strategies). Users can change these at will by modifying the TSDB.

Figure 6 shows an example scenario of MASST agent interactions for this process. Suppose a user delegates a task to the MASST, such as "tell me the best share(s) to buy for the next trading day." The user delegates the task to the IA through key and mouse interaction. The IA passes the task to the CA with a function ID (e.g.,, Buy-Decision-1) together with parameters (e.g., Market-Code and Current-Date). According to the function ID, CA decides to assign the task to the DMA. The DMA needs to request the TSDB to get the buying rules through ODBC-SQL. The records in the TSDB are not only the buying and selling strategies but also the commands to be executed by the MASST agents. Therefore, the DEW together with the CA, acting as a dynamic blackboard, plus direct communication between the MASST agents make the dynamic exchange of facts, knowledge, and commands flexible and transparent in our framework.

Suppose the buying rules in the TSDB lead the DMA to walk along the following steps:

- Step 1: The DMA asks the RMA to carry out risk evaluation for all shares. The RMA evaluates the trade-off between risk and return based on the portfolio theory [25] or solely on the SAR algorithm [26]. The DMA throws out the relatively high-risk shares in order to keep the profit/risk ratio under control.
- Step 2: The DMA asks the TAA to assess the correlation between the individual share and the market trend. The TAA takes into account the relative stock trend compared

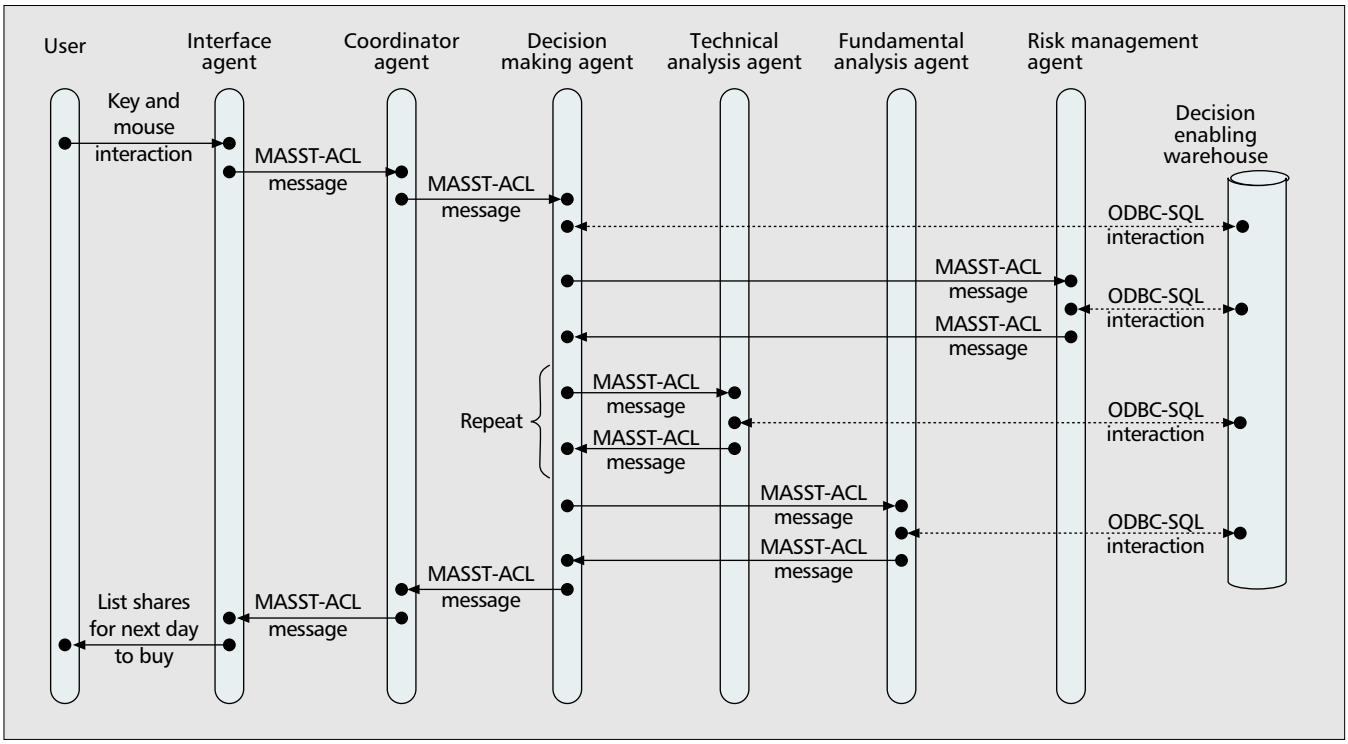


Figure 6. *An example scenario of agent interactions for decision support of buying shares.*

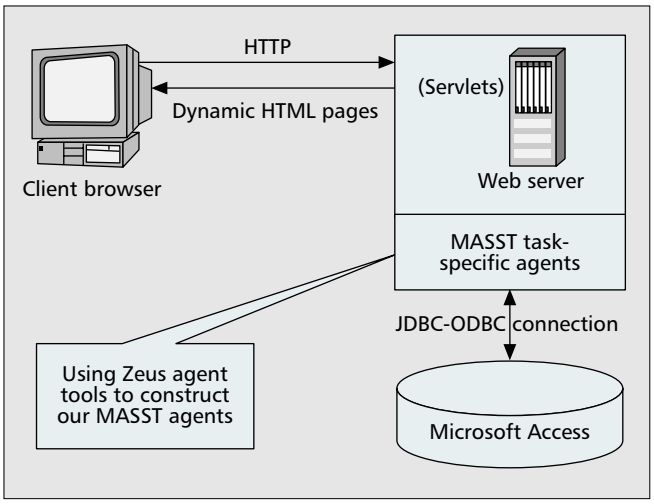


Figure 7. *One experimental platform of the MASST.*

to the weighted stock price index. The DMA rejects those shares whose price moves against the weighted stock price index in a given trading period as shares of this kind have probably been dominated by institutional investors.

- Step 3: The DMA asks the TAA to assess the share price trend and assumes the trend will continue. The TAA evaluates the share price trend through the slope of a moving average line and the relationship between short-term moving average and long-term moving average. The DMA further throws out those shares whose price trend is downward.
- Step 4: The DMA asks the TAA to assess the share's trading volume trend. The TAA assesses the share's trading volume trend by measuring the OBV indicator [1]. The DMA further throws out those shares whose trading volume trend is downward while the price trend is upward.
- Step 5: The DMA asks the FAA to evaluate the value of the share. The FAA evaluates the value of the share by taking into consideration the following factors:
  - –Net profit margin, which indicates how much profit the company is able to make for each dollar of sales
  - –Price/earning ratio (P/E ratio); the lower the P/E ratio, the better value the share
  - –Book value per share; the share's current price far below its book value is possibly an indication that the share is underpriced

The FAA can obtain fundamental analysis data from the company's financial reports or statements that have already been collected in the DEW. After evaluating the condition of the company, the FAA can determine if the company's share is overvalued, undervalued, or correctly valued. According to those results, the DMA further rejects the overvalued shares to get the final list of shares for the next trading to buy.

The DMA sends the final list of shares to the CA, which passes the message to the IA through a MASST-ACL message; finally, the result is delivered to the user.

Implementation and Experimentation

This research work is still ongoing. At the time of writing, the framework has been partially implemented using the following experiment platforms, as shown in Fig. 7.

Users delegate their tasks through the Web browser. The servlets running on the Web server, accept users' requests and pass them to the MASST. According to the tasks, MASST task-specific agents will complete the tasks on behalf of the users. Our agent resources (e.g., the raw trading data) are put into a Microsoft Access database. MASST agents interact with the database through a JDBC-ODBC connection. Servlets produce HTML pages in response to user requests. These experimental MASST agents are constructed using the Zeus Agent Builder Toolkit [28].

In our implementation, the blackboard status is a database table called TaskList with fields TaskID, Parameters, Status, and DMDate. TaskID and Parameters are based on our ontology definition, Status is the representation of current status of the task, and DMDate is the decision making date. The CA dispatches the task to the task-specific agents based on the TaskID; the task-specific agents can then change the value of the Status field in table TaskList to waiting, processing, dispatched, monitoring, finish, or delivered. Every agent can watch the status changes in the blackboard status area by starting a Java thread to monitor the status. The blackboard private message area is a Java Vector object in our implementation; each agent has a Vector object that holds the interaction message for a given task. When the task is completed, the messages will be removed from the Vector. The blackboard knowledge area holds the raw data, user profile settings, and trading strategies, which all are stored as database tables. MASST agents can access these tables through the JDBC-ODBC connection.

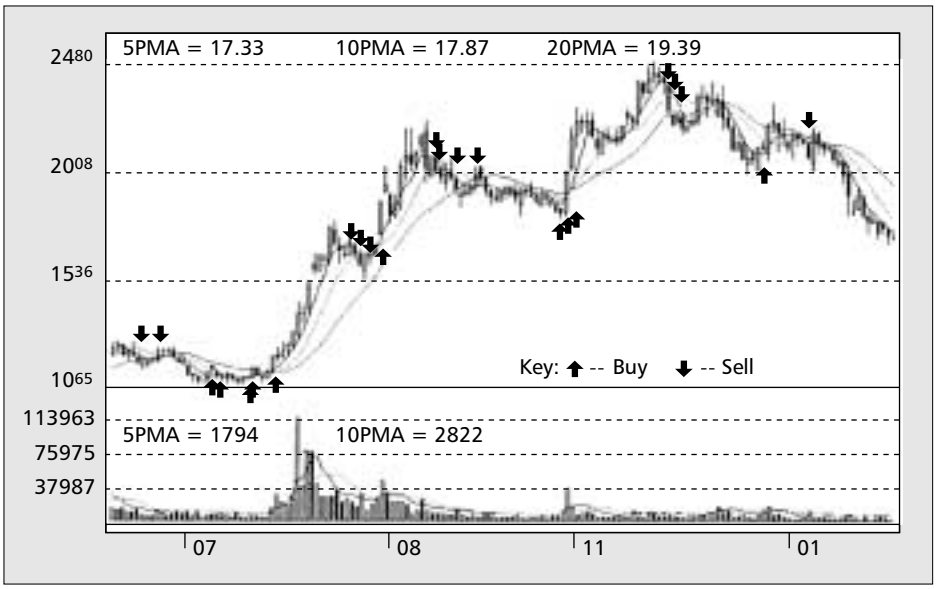
For the purpose of validating our framework prototype, we use real trading data collected from the China Stock Market (Shengzhen Stock Exchange) to test the functions of our system. The data used in our prototype is from June 14, 2000 to February 14, 2001. A number of experiments have been made; however, because of limited space, we only show an experiment in decision support for buying/selling shares.

For the task of decision support for buying/selling a given share, the user must provide two parameters: stock code and the date of decision making. Before submitting the task to the MASST agents, the user should go to the page User Profile Settings to define the strategies for buying/selling a given share. Then the MASST agents can automatically give the user a suggestion to buy or sell the given share on the given date. In this experiment, the share's code is 0020. Let the date of decision making go through from 14/06/2000 to 14/02/2001, recording each day's recommendation (i.e., buy or sell signals) given by MASST. The results are shown in Table 1.

In order to make the effectiveness of this experiment clearer, we put those Buy and Sell signals on the price chart of the share 0020, which is shown in Fig. 8. From Fig. 8 we can

Date (dd/mm/yyyy)	Buy/sell signals
19/06/2000, 26/06/2000	Sell
11/07/2000, 13/07/2000, 20/07/2000, 21/07/2000, 27/07/2000	Buy
21/08/2000, 22/08/2000, 24/08/2000	Sell
28/08/2000	Buy
13/09/2000, 14/09/2000, 19/09/2000, 26/09/2000	Sell
27/10/2000, 30/10/2000, 31/10/2000	Buy
24/11/2000, 27/11/2000, 28/11/2000, 05/12/2000	Sell
22/12/2000	Buy
04/01/2001	Sell

Table 1. *The results of decision support for buying/selling share 0020.*



■ Figure 8. *An experiment in decision support for buying/selling shares.*

clearly see that the MASST can give the Buy signals around the lowest point or relatively lower points and the Sell signals around the highest point or relatively high points.

It is necessary to point out that the MASST framework cannot guarantee the users will make easy money from the stock market; it aims to help investors make the buying or selling decisions in their daily investment activities and reduce the workload of investors to analyze investment-related information. Whether or not they profit from the stock market depends largely on the users' trading strategies.

Discussion and Conclusion

In this article we have proposed a communication architecture for the dynamic exchange of information and knowledge. The coordinator agent plays a vital role in maintaining the appropriate communication protocol. It decomposes system-level tasks to subtasks and distributes the subtasks to related task-specific agents, which are relatively simple. We recognize that a limitation to our MASST framework is the availability of the coordinator agent. The coordinator agent is the control locus of this framework. If it fails on its task, the whole system cannot work properly. Some generalized control heuristics will allow the system to recover, but remain unable to perform the precipitating task. An alternative is to change the design to that of distributed control. This can be done by making the task-specific agents hold onto beliefs about the address and abilities of other agents, and by giving them the ability to decompose the system-level tasks to subtasks. This approach can improve the reliability of a system but at the expense of increased complexity of design for each task-specific agent. Future computational experiments will determine whether this is necessary.

The focus of this article is dynamic knowledge exchange among MASST agents. We have introduced a framework in which MASST agents can exchange knowledge in a dynamic environment. The coordinator agent together with the decision enabling warehouse acting as a dynamic blackboard; plus direct intercommunication among the agents enables the transfer of facts, commands, and rules among MASST agents. Knowledge can be exchanged among the agents by using a combination of facts, rules, and command transfers. We believe that dynamic knowledge exchange is an important feature for any application in which unanticipated conditions or events occur. Using the proposed dynamic knowledge exchange capability, cooperative problem solving sessions can be initiated where each agent can share its problem-relevant knowledge with other agents to solve the problem. An obvious advantage of this capability is elimination of redundant knowledge and hence improved utilization of the system memory capacity.

We have partially implemented our proposed framework. It has proven that our multi-agent framework (MASST), as a generic solution, is a viable implementation for multi-agent approaches to decision support in stock trading.

References

[1] S. W. Tu *et al.*, "Ontology-Based Configuration of Problem-Solving Methods and Generation of Knowledge Acquisition Tools: Application of PROTÉGÉ-II to Protocol-Based Decision Support," *AI in Med.*, vol. 7, no. 3, 1995, pp. 257–89.

[2] C. Iffenecker and J. Ferber, "Using Multi-Agent Architecture for Designing Electromechanical Product," *Proc. Avignon '92 Conf. Exp. Sys. and Their App.*, Avignon, France, EC2.

[3] N. Jennings, J. M. Corera, and I. Laresgoiti, "Developing Industrial Multi-Agent Systems," *1st Int'l. Conf. Multi-Agent Sys.*, San Francisco, V. Lesser, Ed., MIT Press, 1995.

[4] Y. Demazeau, O. Boissier, and J. L. Koning, "Using Interaction Protocols to Control Vision Systems," *IEEE Int'l. Conf. Sys., Man and Cybernetics*, San Antonio, TX, 1994.

[5] G. Sabah, 1990 "CARAMEL: A Computational Model of Natural Language Understanding Using Parallel Implementation," *Proc. 9th Euro. Conf. AI*, Stockholm, Sweden.

[6] M. P. Gleizes, P. Glize, and A. Leger, "Abros: An Adaptive Brokerage Tool Based on Multi-Agent Framework," *IEEE Comp. Commun.*, Jan. 2000.

[7] Y. Cengeloglu, S. Khajenoori, and D. Linton, "A Framework for Dynamic Knowledge Exchange among Intelligent Agents," *Amer. Assn. AI Symp.; Control of the Physical World by Intelligent Agents*, New Orleans, LA, Nov. 1994.

[8] K. Liu, Y. Luo, and D. N. Davis, "A Multi-Agent System for Stock Trading," *Proc. Conf. Intelligent Information Processing, 16th World Comp. Cong. 2000*, Beijing, China, Aug. 21–5, 2000.

[9] K. Sycara, K. Decker, and D. Zeng, "Intelligent Agents in Portfolio Management," N. R. Jennings and M. Wooldridge, Eds., *Agent Technology: Foundations, Applications, and Markets*, Springer, 1998, pp.267–82.

[10] D. Rus and D. Subramanian, "Customizing Information Capture and Access," *ACM Trans. Info. Sys.*, vol.15, no.1, 1997, pp.67–101.

[11] M. Delgado *et al.*, "A Multi-Agent Architecture for Fuzzy Modelling," *Int'l. J. Intel. Sys.*, vol. 14, no. 3, 1999, pp. 305–29.

[12] A. Garcia *et al.*, "Deliberative Stock Market Agents using Jinni and Defeasible Logic Programming," *Proc. 14th Euro. Conf. AI Wksp., Eng. Soc. Agents' World*, Berlin, Germany, Aug. 2000.

[13] P. Tarau, "Inference and Computation Mobility with Jinni," K. R. Apt, V. M. Marek, and M. Truszczynski, Eds., *The Logic Programming Paradigm: A 25 Year Perspective*, Springer, 1999, pp. 33–48.

[14] T. Bui and J. Lee, "An Agent-based Framework for Building Decision Support Systems," *Dec. Supp. Sys.*, vol. 25, 1999, pp. 225–37.

[15] D. D. Corkill, "Blackboard Systems," *AI Expert*, vol. 6, no. 9, Sept. 1991, pp. 40–7.

[16] N. Carver and V. Lesser, "The Evolution of Blackboard Control Architectures," *Exp. Sys. with Apps.*, Special Issue on the Blackboard Paradigm and Its Applications, vol. 7, no. 1, 1994, pp. 1–30.

[17] N. M. Sadeh *et al.*, "A Blackboard Architecture for Integrating Process Planning and Production Scheduling," *Concurrent Eng.: Res. and Apps.*, vol. 6, no. 2, June 1998.

[18] P. Maes,"Agents That Reduce Work and Information Overload," *Commun. ACM*, vol. 37, no. 7, 1994.

[19] K. Sycara *et al.*, "Distributed Intelligent Agents," *IEEE Expert*, vol. 11, no. 6, 1996.

[20] O. Etzioni, "Moving up the Information Food Chain: Deploying Softbots on the World-wide Web," *Proc. 13th Nat'.. Conf. AAI-96*, Portland, OR, 1996.

[21] L. Chen and K. Sycara, "Webmate : A Personal Agent for Browsing and Searching," *Proc. 2nd Int'l. Conf. Auto. Agents*, Minn./St. Paul, MN, May 1998.

[22] A. Caglayan *et al.*, "Open Sesame — A Learning Agent Engine," *App. AI*, vol. 11, no. 5, 1997, pp. 393–412.

[23] FIPA 97 Spec., "Part 2, Agent Communication Language, Foundation for Intelligent Physical Agents Fipa 97 Specification," Geneva, Switzerland, Oct. 10, 1997.

[24] XML v. 1.0, 1998, "eXtensible Markup Language (XML) 1.0, XML v. 1.0 Model and Syntax," Feb. 10, 1998, REC-xml-19980210, http://www.w3.org/TR/REC-xml

[25] H. Markowitz, "Portfolio Selection," *J. Finance*, Mar.1952.
[26] J. W. Wilder, *New Concepts in Technical Trading Systems*, Greensboro, NC: Trend Research, 1978.
[27] S. B. Achelis, *Technical Analysis from A to Z*, McGraw-Hill, 1995.
[28] H. Nwana *et al.*, "ZEUS: A Toolkit for Building Distributed Multi-Agent Systems," *App. AI J.*, vol. 13, no. 1, 1999, pp.129–86.

Biographies

YUAN LUO (Y.Luo@staffs.ac.uk) received his B.Eng. in electronics engineering at Chongqing University, China, in 1983, and his M.Eng in communication and electronics systems at South China University of Technology in 1986. He is an associate professor in Foshan University, China, and currently a Ph.D. researcher at Straffordshire University, United Kingdom. His areas of interest include: software agents/multi-agent system, e-commerce, and software engineering.

KECHENG LIU (k.liu@staffs.ac.uk), a Fellow of the British Computer Society, holds a chair as professor of computing science at Staffordshire University. He has published over 100 papers in the field of computing. His research interests span from information systems methodology, requirements studies, systems analysis and design, software engineering, object methods for systems design, and normative modeling of software agents to HCI and CSCW. He has been awarded a number of research grants from national and international foundations, including two major ones in information systems reengineering and systems integration. His research monograph "Semiotics in Information Systems Engineering" (2000, Cambridge University Press), waS one of the first to treat the topic in a systematic manner, followed by a number of edited books on organizational semiotics. As a key figure in the international research community of organizational semiotics, he chaired a series of international workshops in this field and was Program Chair of the International Federation of Information Processing (IFIP) WG8.1 Working Conference (Montreal, Canada, 2001).