NEXT GENERATION BROKER SYSTEM

Piyanath Mangkorntong[1], Willy Mong[1]

[1]FinanceIT Research Group,
School of Computer Science & Engineering,
University of New South Wales

ABSTRACT: The ever increasing popularity of the stock market as a revenue generating tool in recent years has seen it grow in size and complexity. Many investors seek the assistance of stock brokers to give them advice and to make trades on their behalf. However, the dynamic nature of the stock market makes it very difficult for brokers to do so without the aid of computerised systems. Existing systems are large and monolithic in nature, making them difficult to maintain and adapt to the changing environment of the stock market. This research proposes an architecture for a broker system that is based around a service oriented architecture (SOA) and Web service technology to modularise some of the tasks that need to be performed by the broker. This increases the adaptability of the system to a changing environment, allows existing stock market systems to be easily integrated, and minimises development time and costs. The focus of this paper is on the creation of a component of the broker system that can be used to investigate trading strategies. By combining basic web services for data extraction and trading management, it is possible to create a system that allows a broker to simulate and evaluate a strategy before applying it in a real market so that better trading decisions can be made.

INTRODUCTION

Stock brokers are one of the main participants in stock exchanges worldwide. They often act as an agent for their clients, making trades on their behalf. They also act as advisors, providing suggestions to their clients on what stocks to buy and sell.

However, the micro-structure of the stock market in which brokers work is highly dynamic and volatile. Many stocks are available to be bought and sold, each exhibiting its own patterns and characteristics that are highly unpredictable. With so many options and considerations that need to be taken into account, it is an extremely arduous task for a broker to investigate aspects of the stock market and consistently provide effective advice to their clients.

Thus, brokers perform their day to day tasks with the aid of a broker system. Such a system should provide tools for interacting with exchanges and performing analysis. A good broker system must be able to cope with an extremely complex and dynamic environment. As a consequence, these broker systems are quite large and complicated by themselves.

One of the problems with these systems is that they are created as a single, large software package. Thus, the time and costs of developing such a complex system is generally high, and so are the maintenance costs associated with them. It would be beneficial to develop a broker system with reusability in mind. By using a modular architecture and utilising web service technology, components are able to be created independently. Such modules would be easier to configure and replace, resulting in lower development time and costs.

This research aims at developing a novel architecture for a generic broker system using web services. The main advantage of such a design is the ease with which external services can be integrated. This helps increase the flexibility of the system, since the external components could be replaced with other modules that use the same standard protocol. Thus, this system has a high level of adaptability. The focus of this paper will then shift towards a detailed design of a specific part of this system that allows a broker to perform a very important task, namely, the analysis of trading strategies.

BACKGROUND AND LITERATURE REVIEW

The area of stock broker systems has not been explored to great extent in academic literature. Most systems for stock brokers are commercial by nature and thus details of such systems are difficult to gather. However, this is not to say that stock trading systems have not been an area of interest in information technology research. There has been some investigation into automated stock broker systems, many using intelligent agents to complete the tasks of human stock brokers. Sycara et al. [SD98] proposed that intelligent agents could be utilised in the area of portfolio management. The use of multiple agents was suggested, collaborating with each other in order to extract the desired information. The benefits of multiple agents were outlined, such as not having a single point of failure for the system (although point of failures still existed for individual agents that may hinder the software as a whole) and spreading the computation required across several entities, so as not to overwhelm a single agent. The main focus here was on monitoring a client's portfolio and gathering relevant information.

Luo et al. [LD02] utilised some of the work described in [SD98] and presented a multi-agent stock trading system using Distributed Problem Solving (DPS). They proposed the use of several different agents, each automating a specific task of the stock broker, and having these agents communicate and collaborate results with one another in order to provide an automated stock trading system. The breakdown of the system into different agents provides modularity in the architecture, making it easier to modify and maintain. This type of system however, automates the decision making process and hides it from the user. Thus, such a system would not be very suitable for a stock broker, as they would like to see the intermediate results of the analysis undertaken so as to make and validate recommendations to clients. In addition, the use of agents can often lead to a high degree of coupling as they require a multitude of information (from external sources and other agents).

Research has also been undertaken recently by Cheng et al. [CW04] into a multi-agent system for asset management. They propose a three-tier architecture comprising of a demand-tier, process-tier and supply-tier. The demand-tier is responsible for interaction with clients, while the process- tier performs analysis of data that is gathered by the supply-tier. Again, agent technology is utilised to perform each task. This paper focuses more on a complete asset management system of which stock analysis is only a small section. Hence, it provides only a very broad overview of the architecture. However, the three-tier architecture proposed does assist in dividing the system into maintainable components.

Instead of using an agent architecture, the proposed broker system will be based around an architecture where the different functions of a capital market system are modelled using Web services is described in [RB02,RD04]. Two important services are the Exchange Service (ES) [MY03] and the Trade Data Service (TDS) [CW03]. The ES in this architecture is responsible for the execution of trades, while the TDS performs tasks for accessing and performing queries on market data to obtain the desired information. By integrating the broker system with these services, the benefits of modularity can be utilised, as the broker system would simply need to communicate with these components to gain the necessary functionality rather than having it integrated into the system itself.

PROPOSED BROKER SYSTEM ARCHITECTURE

The proposed broker system is designed to facilitate a stock broker in performing their required tasks. These include risk profile definition, stock allocation and order submission. Risk profile definition refers to deriving how averse a client is to the possibility of loss. Stock allocation occurs when brokers determine which stocks to buy and sell for their clients. Order submission is the task that refers to sending transactions on behalf of clients to the stock exchange. This system design is focused on being easily configurable due to its modular nature and will provide a central interface to existing services (e.g. Exchange Service, Trade Data Service) for a stock broker.

Below is the diagram of the proposed broker system architecture, showing the main components of the system, with each line connecting components, representing communications between them.

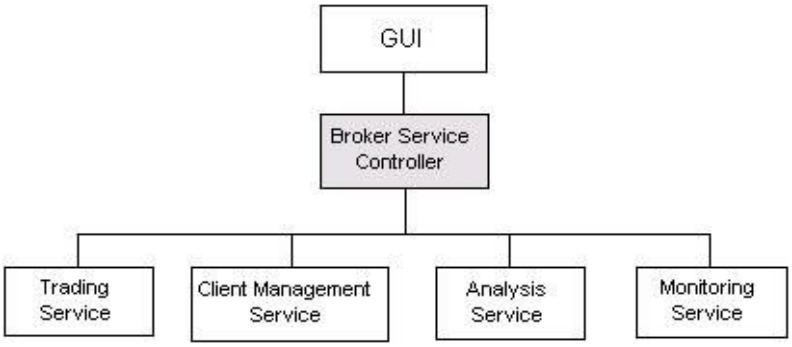

Figure 1. Top Level Diagram of System Architecture.

This architecture can be sub divided into three main sections – the graphical user interface (GUI), the Broker Service Controller and the components or services of the system. Such a design was based on the mediator design pattern, with the Broker Service Controller being the mediator. The main advantage of this architecture is to separate the business logic from the presentation in order to encourage reusability of components. The components that make up the bottom layer are not dependent on the type of visualisation that is utilised. Thus, these components can be freely modified and changed without adverse affects on others.

The four components in the bottom layer are web services. Each of them provides a service to the stock brokers to perform a group of related tasks. The core tasks supported by each component are described in the table below:

Table 1. Components and their tasks.

Component	Tasks
Trading Service	Order Submission. Submit financial transactions such as order placement, amendment and cancellation to the Stock Exchange.
Client Management Service	Manage the client profile, client portfolio and calculate the client risk levels.
Analysis Service	Analyse the stock market behaviour and the effect of trading strategy.
Monitoring Service	Monitor price movements for selected stocks. This can lead to the automated updating of client portfolios, making the broker's stock allocation task easier.

Each service is fairly complex and required further decomposition into smaller components that reflect the type of the market, the regulations in place, the information available etc. In the rest of this paper, we focus on the Analysis Service by providing a detailed design and describing a realistic case study that uses this service.

THE ANALYSIS SERVICE

One of the main ways in which a broker selects stocks is through the use of trading strategies. A trading strategy can be defined as a set of systematic rules that dictate how buy and sell decisions are made. A broker can often utilise various parameters in order to determine the time, amount and type of stock to buy and sell for their clients and this is an area where computerised systems provide major benefits.

The Analysis Service provides the facilities for a stock broker to explore the effect and performance of trading strategies on the market. This is done through the simulation of the historical data. Since the market behaviour is undeterministic, the broker may run the simulation several times using the same strategy, with different parameter values in order to obtain the optimal solution. The main design focus of this part of the research is to come up with a generic architecture where new trading strategies could be incorporated into the service with minimal development time and costs. The business processes involved in developing a trading strategy are described in Figure 2.

In Figure 2, six main steps in creating and evaluating a trading strategy are shown, along with the external modules (in grey boxes) that will aid in the completion of these activities. The scope of the Analysis Service will cover steps 1-4. These are the processes undertaken to simulate a market, while steps 5 and 6 utilise the strategy in a real market. Thus, the Trade Strategy Execution process will be similar to the Trade Strategy Simulation process, except it will communicate with a real market. Step 4: Iteration refers to modifying the parameters of a strategy after evaluation until the desired results are produced.

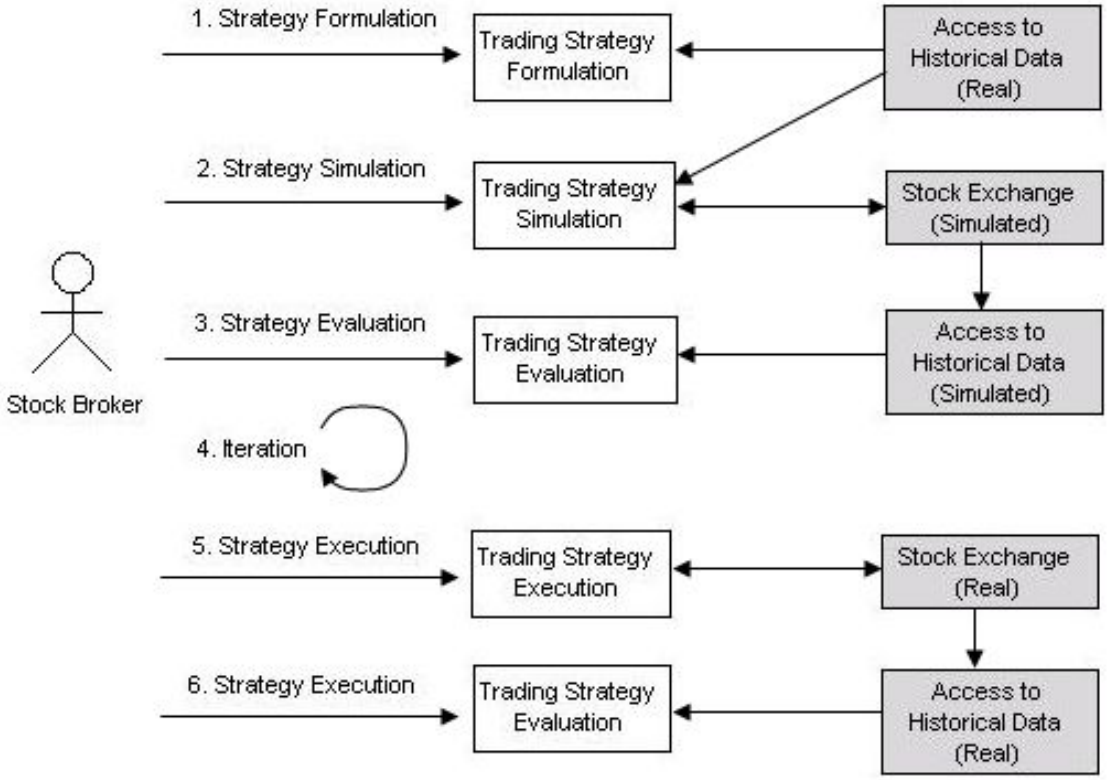

Figure 2. The business processes involved with specifying and executing trading strategies.

THE SIMULATION PROCEDURE

As mentioned in the literature review, there are two services from [RB02,RD04] that are utilised to aid in the implementation of the Analysis Service. Such services are outlined below:

1. *Trade Data Service (TDS):* The TDS provides access to historical market data. This includes price, time and volume of orders and trades that have previously occurred.

2. *Exchange Service (ES):* The ES provides the common interface to the Stock Exchange, where the actual execution of financial transactions such as placing, amending and cancelling orders happen.

Figure 3 below illustrates the procedures and the components involved in the simulation.

The simulation is carried out by feeding historical data retrieved from a data source to a simulation controller. This data consists of the transactions that have taken place on a particular day, for example, all the buy/sell orders. The historical data is retrieved via the TDS, which extracts the relevant data from the database and makes it accessible through the internet. The data retrieved will depend on the simulation parameters. These parameters include the date range, security and market in which to run the simulation.

The simulation controller then passes these transactions to the ES which in turn submits the orders to the trading engine. Additional order(s) will then be made in accordance with the strategy, depending on the trading strategy parameters. Such parameters consist of, for example, the maximum volume to buy, and the time in which to begin execution of the strategy.

The feedback of the executed transactions from the trading engine is passed back to the components through the ES. This information is utilised by the strategy when determining whether to place additional orders. It is also used to keep track of results that are eventually displayed by the GUI.

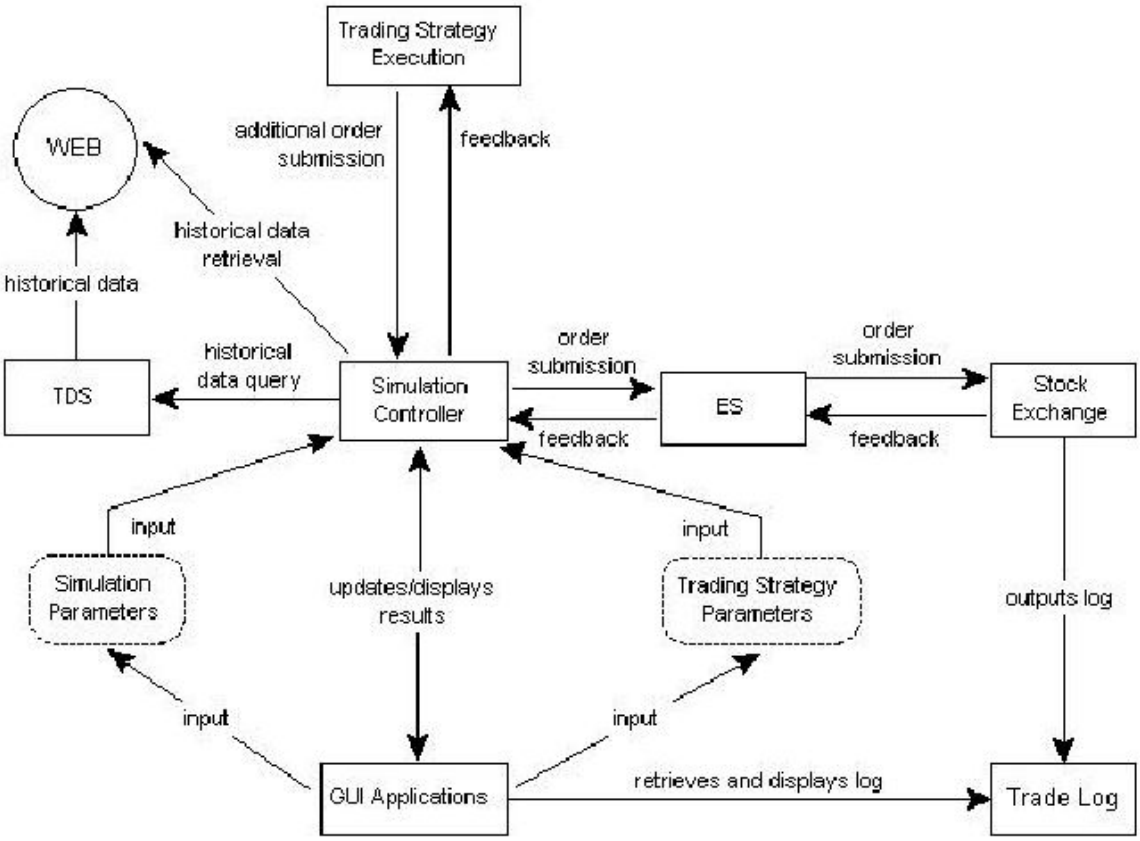

Figure 3. High Level Component Diagram for Analysis Component.

Once a trading strategy has been implemented, it is important to assess how well such a strategy has performed. The trading engine produces a log of the simulation. This log is used to graphically display the events that have taken place. This will aid in the analysis of the effects that the strategy has had on the market. The amount of profit made will be one of the main factors in assessing the effectiveness of the strategy. After the evaluation has taken place, the variable factors contained within the strategy may be adjusted and results compared. This will be carried out for the continuous improvement of the strategy.

DETAILED DESIGN

The class-collaboration diagram shown in Figure 4 illustrates the interaction of classes within the Analysis Component. External entities such as TDS and ES are shown in grey boxes. The attributes and operations of the classes have been omitted for clarity.

The Model-View-Controller (MVC) paradigm is used as a basis for the design of the class diagram. In the diagram, the Simulation Controller class acts as a controller, establishing communication between the GUI and the rest of the modules in this component.

The Trading Strategy class represents the model as it is the responsibility of this class to manage the results and maintain other data that is used in the trading strategy. This class is also responsible for notifying the controller when trades take place, which in turn, will update the display on the GUI, which is the view in this paradigm. The GUI class places the results in a table and displays it on the screen.

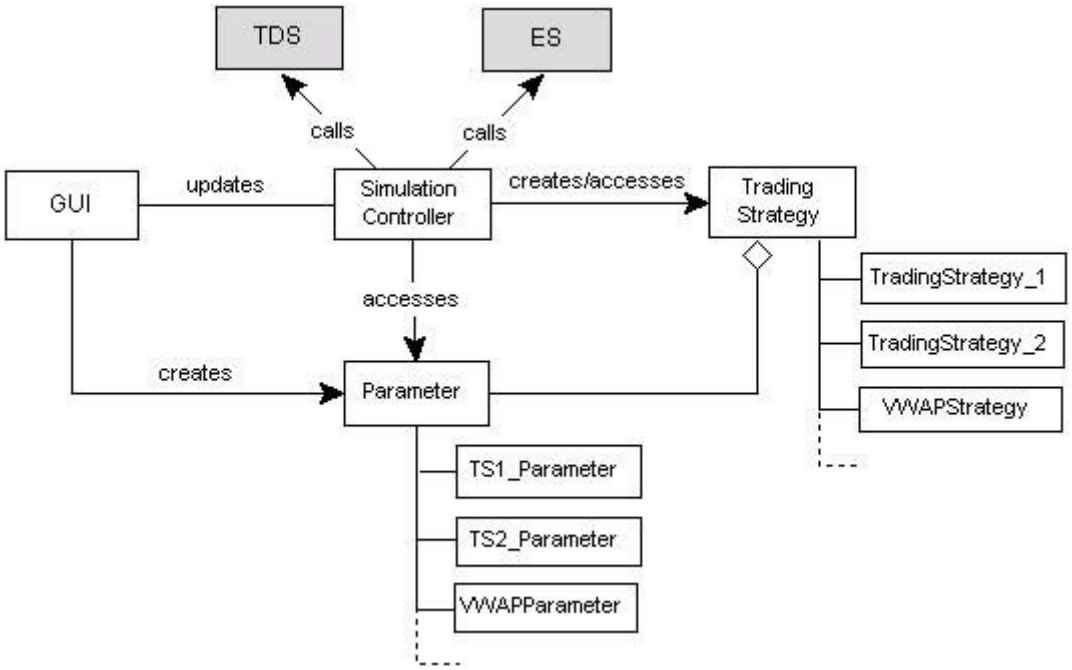

Figure 4. Class Diagram for Analysis Component.

Each of the specific Trading Strategy classes (TradingStrategy_1, TradingStrategy_2 and VWAPStrategy in the diagram) is a sub class of the super class Trading Strategy. In this way, the template design pattern can be applied to the trading strategies, making it easier to add and replace them. The general processing of feedback and maintenance of data can be placed in the super class while the intricacies of the strategies themselves are the responsibility of the concrete sub classes.

Each trading strategy has its own set of parameters, hence aggregation is used to model the relationship between these two classes. The set of parameters for a particular trading strategy inherit from a super class that contains generic parameters (for example, start date and end date for a strategy). More specific parameters, and the operations on them, are then defined in the sub classes to customise for each trading strategy.

A CASE STUDY: VWAP AS A TRADING STRATEGY

In this research, the VWAP (Volume Weighted Average Price) was chosen as a case study of the Analysis Component architecture. The details of this strategy and its implementation are described below.

VWAP is calculated by adding the dollars traded for every transaction in that stock (Price x Number of Shares traded) and dividing the total shares traded for a period of time. A VWAP chart shows the average price for a stock for a time period in the day. It is usually used as a performance benchmark for investors. If a broker bought a stock today at a price lower than the current cumulative VWAP, then that broker bought the stock at a good price (he/she did better than the average buyer of the stock). On the other hand, if a stock was bought at a price higher than the VWAP, such a stock has been over paid for, relative to other buyers of the day.

It has been observed in the past that the VWAP at the end of the trading day is very close to the actual stock price at that time. This is due to the fact that brokers are often contracted to sell stocks at the VWAP. Thus, there is a potential to generate profit if, near the close of market, the buy price is below the VWAP. If stocks can be bought when this phenomenon occurs, then it would be likely that the price for this particular stock would rise before market close. Thus, these shares can be sold a few minutes before closing at market price in order to generate profit for the broker's clients. To illustrate this, the graph below shows the stock price and VWAP for NAB near the end of the normal trading period.

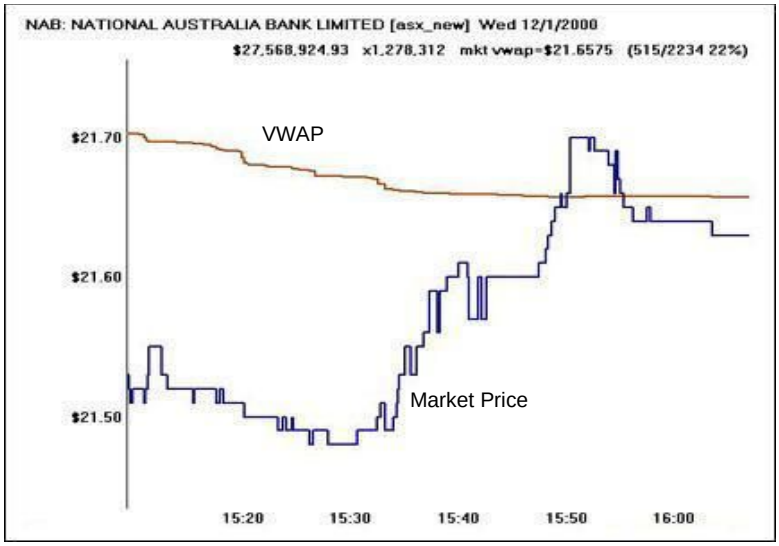

Figure 5. VWAP and price graph for NAB on 12/01/2000.

On this particular day, it can be seen that the price of the stock moves toward the VWAP near the end of the day. Thus, there is a potential to make profit if shares are purchased during 3:00pm to 3:40pm, and then sold at any time before the close of market (e.g. at 3:50pm). From this observation, a very simple algorithm and parameters required for implementing this strategy can be derived as show in Table 2 and Table 3 below:

Table 2. VWAP Algorithms for buy and sell.

Algorithm for Buy	Algorithm for Sell
From 3:00 – 3:40pm, For every sell order listed If sell price < VWAP by at least 2c then Buy all stock available	If stocks available to sell Then At 3:50pm Sell all stock available

Table 3. VWAP Strategy user input parameters.

Input Parameter	Description
Buy time	The time in which to begin checking the condition of the market for the potential to buy.
Sell time	The time in which to sell all the stocks bought
Buy Time Interval	The time interval in which to look for buy opportunities.
Max Volume	The maximum volume to buy
Number of cents below VWAP	The number of cents below VWAP that the price can at most be for a buy order to be submitted.

The parameters and algorithms defined above lead to the creation of the two classes for the VWAP trading strategy. These classes are the *VWAPStrategy* class that contains the logic of the strategy and the *VWAPParameter* class that contains the input parameters used by the strategy.

Apart from the user input parameters to the strategy, there are also internal attributes of the VWAP trading strategy which are maintained within the VWAPStrategy class:

- *Aggregate Volume:* The cumulative total of all the volume that has been traded so far.
- *Aggregate Value:* The cumulative value of all the trades that have occurred so far.
- *Best Sell Price:* The price at which a market buy order would execute at if entered at the current point in time.

CONCLUSION

This research work has introduced a new architecture for creating a generic broker system. The system design reuses other services proposed in the literature. This demonstrates the ease with which external services can be integrated. This is part of a research effort for widespread use of such

service-oriented architectures in this domain, moving away from the large single package systems that are currently available. This in turn would lead to other benefits such as providing a more easily manageable system, where components can be distributed and maintained separately. This is an important feature in the current globalised environment.

The paper has focused on the implementation of a market analysis service for brokers. More specifically, the service provides a means of executing and evaluating a trading strategy. The paper also described a realistic application based on the VWAP trading strategy. By following the same procedure, other trading strategies can be easily incorporated into the system. This is of great benefit to brokers as it gives them the ability to try their strategies and enhance them before using them in real situations.

ACKNOWLEDGEMENTS

This research work is funded and supported by the Capital Markets CRC Limited (www.cmcrc.com). It is part of the Interoperability Program at the CMCRC led by A/Prof. Fethi Rabhi, which focuses on the development and deployment of e-services in the area of financial market systems. We would like to acknowledge the support and contribution of the current and former members of this team, including Sunny Wu, Joshua Mok, Standley Yip, Johan Fischer, Anthony Cheung, Feras Dabous, Dr. Pradeep Ray and Dr. Hairong Yu.

We also would like thank Andreas Furche of the CMCRC and A/Prof Rabhi for their invaluable advice and assistance throughout the course of this research.

REFERENCES

[CW03] Tak-Hung Anthony Cheung and Sunny Yuk-Ho Wu. *Trade Data Service in Capital Markets*. School of Computer Science and Engineering, Honours Thesis, The University of New South Wales, 2003.

[CW04] Tung Wan Cheng, Wan Ling Wang, and Shung Kwan Chan. Three-tier multi-agent architecture for asset management consultant. In Soe-Tsyr Yuan and Jiming Liu, editors, *2004 IEEE International Conference on E-Technologies and E-Commerce E-Service*, pages 173 – 176. IEEE Computer Society, 2004.

[LD02] Yuan Luo, Kecheng Liu, and Darryl Davis. A multi-agent decision support system for stock trading. *IEEE Network*, 16(1):20 – 27, 2002.

[MY03] Joshua Mok and Stanley Yip. *Exchange Services for Financial Markets*. School of Computer Science and Engineering, Honours Thesis, The University of New South Wales, 2003.

[RB02] Fethi Rabhi and Boualem Benatallah, A Service-Based Architecture for Capital Markets Systems, Vol 16, no 1, *IEEE Network*, 2002, pp. 15-19.

[RD04] Fethi Rabhi, Feras Dabous, Hairong Yu, Boaulem Benatallah, and Yun Ki Lee. Case study in developing web services for capital markets. In S.T. Yuan and J. Liu, editors, *IEEE International Conference on e-Technology, e-Commerce and e-Service*. IEEE Press, March 2004.

[SD98] Katia Sycara, Keith Decker, and Dajun Zeng. *Agent Technology: Foundations, Applications and Markets*, chapter Intelligent Agents in Portfolio Management,

pages 267 – 282. Springer, 1998.