Develop Service Oriented Finance Business Processes: A Case Study in Capital Market

Aries Tao Tao\textsuperscript{1} Jian Yang\textsuperscript{2}

\textit{Department of Computing}
\textit{Macquarie University}
\textit{Sydney, NSW 2109}
\{tao\textsuperscript{1}, jian\textsuperscript{2}\}@ics.mq.edu.au

\textbf{Abstract}

Current business process development is a process that needs to apply software development principles and at the same time incorporate the special requirements of service oriented architecture (SOA). In this paper we will discuss the steps and principles involved in developing service oriented finance business process in the domain capital market.

\textbf{Keywords:} business process, service oriented architecture, capital market.

1. Introduction

Capital Markets (CM) is a virtual institution that supports money exchange cross multi-organizations such as investors, companies and banks. The CM trading cycle is characterized by a large number of heterogeneous systems including information distribution systems (e.g. real-time quotes, historic trade data), order processing systems (e.g. order routing, presentation, and execution), clearing and settlement systems, and research and analysis systems [1]. Due to the globalization of world economy in recent years, the needs of collaboration crossing heterogeneous system in CM grow tremendously. For example: investor wants to be able to invest money in multiple exchange services in other countries through one account. This involves activities such as setting up connections between the account and the heterogeneous systems used in those exchange services and related banks. In recent years, Service Oriented Architecture (SOA) and web services are becoming part of our everyday language and recognized as a suitable architectural style for developing modern enterprise applications. However there is a lack of methodological guidelines in SOA development. Furthermore there is a lack of reports in the application of SOA. We believe SOA can be applied in the domain of CM for two reasons:

1. SOC reorganizes a portfolio of previously developed software and a supporting infrastructure into an interconnected set of services, each of which is accessible through standard interfaces and messaging protocols [2].
2. The concept of service in SOC not only supports an autonomous computing entity as a basic service; by orchestrating a number of basic services crossing heterogeneous system, it also supports Business Process (BP) as composite service. In this way SOA supports adding new functionalities to existing systems without re-writing them. The relatively cast of development time and cost surely enables the CM systems to adapt to the dynamic change environment efficiently.

This paper is a report on applying SOC in building realistic Business Processes (BPs) in CM domain, a project sponsored by the Capital Markets Cooperative Research Centre (CMCRC). It contributes in evaluating the appropriateness of SOA by designing and implementing realistic BPs to extend functionalities of one existing system through a case study. Practical experience and issues on how to design and implement service oriented BPs in the CM domain will be presented and discussed.

The paper is organized as follows. Section 2 describes one existing system, the broker system, in CM and its problems. The rational and benefits of applying SOA will be discussed. Section 3 analyzes the BP for the case study as a necessary preparation for design and implementation phase. Section 4 demonstrates the design of the BP. Section 5 demonstrates the implementation of the BP. Section 6 discusses the practical issues and experiences in building service oriented BPs. The final section concludes the benefits and limitations of current SOA approach.

2. Problem Identification

As showed in Figure 1, Brokers constitute a necessary mid-layer in CM between Investor and Stock Trading. They 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.
Accordingly, the software application - Broker System has two main functionalities. The first one is Trading Service that enables clients to trade by placing orders in CM. The second one is Client Profit Management that formulates a trading strategy plan that advises customer on how or when to place an order by identifying some relevant market conditions to maximize profit. Currently trading strategy plan is implemented by a software package called Simulation Conductor as part of Broker System.

With the current trend of globalization of economics, the trading strategy plan in CM is affected by many factors from all over the world. For example, the raise of petrol price in Mid-East could affect the stock price of the car seller in Japan such as Toyota. As a result, the trading strategy plan becomes far more complex in order to response to these new factors, and more dynamic to adapt to the rapid changing environment. The original Simulation Conductor provided by the existing Broker System is no longer up to date. Rather spending considerable amount of money on both IT and financial expertise to redevelop Simulation Conductor inside Broker System, it will be much more economical and efficient to outsource Simulation Conductor to third party who is expert in this field. This case study aims to design and implement a realistic Broker System that enables trading strategy plan to be outsourced to third party (shown in Figure 2). This brings up the questions of how to design such a broker system and what existing applications can be re-used. We use SOA utilizing Web services (WS) approach for two reasons:

1. Short development time. By wrapping the related CM legacy systems into services, it provides a standard interface that enables rapid integration with an outsourced trading strategy service (TSS).

2. Better service choices. Because WS has widely supported standard, very little IT support and cost is needed to switch between available TSSs to employ the best one at the time.

### 3. BUSINESS PROCESS ANALYSIS

In order to have maximum return-on-investment (ROI) in CM, BP analysis is necessary before design and implementation. There are two objectives: (1) to recognize the portfolio of existing systems by reuse their functionality as much as possible in new business process development; (2) to maximize the reusability of the new process for future use [3]. Thus, we adopt four steps in BP analysis: Service Identification, Component Identification, Process Scoping, and Process Realization Analysis.

In this case study, the aim is to implement a realistic process for Broker System – Broker Service (BS) based on SOA. The BS will use trading strategy plan as an external service to help maximize clients’ profit. Best trading strategy plan which generate maximum profit is calculated by re-running different trading strategy parameters over historical data. The design and implementation decision of BS will be discussed in the following sub-sections.

#### 3.1. Service Identification

This phase is to identify necessary services for the BS. Services are identified by recognizing essential functionalities of the BS.

In this case study, we identify three basic services needed for BS based on decomposing functionalities into three groups:

1. Trading Data Service (TDS) provides necessary historical data for simulation to calculate best trading strategy. The detail functionalities of TDS will be introduced in the next design section.

2. Trading Engine Service (TES) simulates the behavior of Exchange Services (E.g. ASX) that allows simulation over historical data. The detail functionalities of TES will be introduced in the next design section.

3. Simulation Conductor: It provides all the necessary Trading Strategy plans over historical data. Each plan will generate different new buy/sell orders that
will be sent to TES together with historical data. Comparison between the profits generated by each plan can be made based on the outputs of TES, so that the plan that generates maximum profit can be found. Unlike TES and TDS that can be developed as external service, Simulation Conductor must be an internal service that are developed and maintained by Trading Strategy Simulation (TSS) Provider because any new trading strategies is confidential and should not be accessible or known outside of trading strategy provider.

3.2. Component Identification

This phase is consisted of two tasks. The first one is to find those existing systems as valid components to realize services identified in the first phase. This is done by comparing the functionalities of the services identified with the existing systems. The more use of existing systems, the lower the cost to implement the underlying service functionality that supports the BS. In this case study, the basic services TES and TDS are discovered to be the typical services that can be built over existing systems. Those systems play an important role in a number of CM BP scenarios.

The second task is to find the functionality gap between services and components by comparing the functionalities identified in the new BS with the existing systems components. The gap needs to be fulfilled by developing new components connecting with third party services. In this case study, a component New Trading Strategies need to be developed to fulfill functionalities gap between what the component Old Trading Strategy provides and what the service Simulation Conductor requires.

3.3. Process Scoping

This phase is to decompose the main process into a number of sub-processes to accomplish the task. By doing this, we can prevent the main business process in BS from becoming overly large, complex and difficult to maintain. For each process, the message sequence and type from a service to another must be determined.

In this case study, we decide that Trading Strategy Simulation (TSS) is used to facilitate the main process in BS with client profit management. TSS takes trading strategy type and simulation related period of historical data as inputs, returns the best parameter which generates maximum profit as outputs. TSS is a composite service that composes TES, TDS as the external service and Simulation Conductor as the internal service. The sequence of how TSS orchestrates those services is shown in Figure 3. By outsourcing TSS as an independent process that can be developed and maintained by different Trading Strategy Providers, the BS which uses TSS could easily switch between Trading Strategy Providers that supports the TSS service interfaces. As a result, better product choice could easily be made with little IT support.

3.4. Process Realization Analysis

This phase is to choose the most appropriate BP realization approach based on evaluation of different approaches in turns of development time, costs and ROI. There are four BP realization options according to [4]:

1. Green-field approach. It is used when a service is already implemented, and subsequently the service interface needs to be derived from the new Web service implementation.
2. Top-down approach: It is used when firstly there already exits a service interface; secondly a new service is developed which conforms to
the service interface. Note that such service interface is usually an industry standard that can be developed by a number of service providers.

3. Bottom-up approach: It is used to wrap up existing applications into services. Existing application can be developed from different programming languages or back end legacy systems. Such approach that is extremely suitable for an environment includes existing system implemented with several heterogeneous technologies.

4. Meet-in-the-middle approach: It is used when some Web services already exists and need to be mapped onto part of a new BP.

As showed in Figure 3, we choose combine top-down and bottom-up approach for this case study. Firstly, we use bottom up approach between the basic service layer and TSS because the basic services TES and TDS are already determined in the “Service Identification” phase. Bottom-up approach makes the best use of TES and TDS as existing services which needs to be deployed as the basic services for the BP development. Secondly, we use top-down approach between the BS and the TSS. By defining the TSS’ service interface by the BS according to trading strategy plan standards prior than its implementation, the service interface could be used as an open standards easily supported by a number of providers. This ensures better product choices for BS.

4. BUSINESS PROCESS DESIGN

As discussed in last section, we choose to build the BPs by combining bottom-up and top-down approach. Firstly we will introduce TES and TDS as basic services. Then we will develop the TSS by composing TES and TDS. Finally we will develop BS which makes use of TSS to provide client profit management service.

We use the Business Process Modeling Notation (BPMN) version 1.0 standard to draw Business Process Diagram (BPD) which shows the functionalities involved and the relationships between them. Due to the space limits, not all the functionalities or activities are sketched in the BPD.

4.1. Trading Engine Service (TES)

As showed in the bottom of Figure 4, TES simulates functionalities provided by a stock exchange. A TES takes Broker Systems’ buy and sell orders as input, and performs matching according to the financial market model it supports. After submitting an order, the Broker System receives a unique confirmed order identification that can be subsequently used in cancellation or amendment requests. Broker Systems can check the position of their orders by inspecting the orderbook which display all buy and sell orders. Once the price of a buy order and a sell order is matched, TES will send confirmation to both buyer and seller as output, related information such as orderbook will also be updated. The main functionalities of Trading Engine Service are summarized as follows:

1. Process Order: Orders (buy or sell) are processed when they are submitted to the TES either within the specific date or reloaded from the database at the start of trading (e.g. in the case of orders that do not expire until a specific future day comes). The orders also include cancelled and amended orders, which are occasionally submitted by traders. All the valid orders are collected in a data structure called the orderbook.

2. Generate Trades: Trades are generated by examining the orderbook and determining successful matches according to the algorithms implementing the rules of the exchange (they could be different according to the time of trade).

3. Information Dissemination: information related to all activities of the TES (including orders and trades) is stored and disseminated to the market participants through a suitable communication infrastructure (usually based on a publish/subscribe model).

4.2. Trading Data Service (TDS)

As showed on the top of Figure 4, A Trading Data Service (TDS) is dedicated to the provision of market data such as buy or sell orders and the resulting trades in response to queries. In general, there are two categories of data queries stored in the database: Intraday (data related to a specific time interval during a trading day); and Interday (daily, weekly or monthly summaries of trading activities over a period of time which is more than one day). The main functionalities of TDS are summarized as following:

1. Get Metadata: determines what markets and types of data queries that are supported by a particular service implementation.

2. ProcessQuery: executes a query formulated according to the metadata and containing the appropriate parameters.

3. Monitor Query Progress: checks a query’s progress in case a query involves a large amount of data and a long time to process.

4. Process Results: relates to the way query results are accessed (e.g. downloading the data from a network server).
4.3. Trading Strategy Simulation (TSS)

Recall that TSS takes required trading strategy type and related historical data period as inputs, return best trading strategy parameter maximize profit as output. TSS consists of re-running the market events of a specific previous trading period using a certain trading strategy then evaluate its effectiveness. The essential aspect of any strategy is to formulate how or when to place an order by identifying some relevant market conditions called trading signals. Having the ability to simulate a strategy gives a brief indication on how it may perform in the real market. Once a TSS instance is required, we can see there are three main functionalities involved:

1. Data Preparation: It involves in preparing the trade dataset that will be used during the simulation. This dataset is used to recreate the trading conditions that the strategy is to be used against. We can see that this functionality requires access to the basic service TDS.

2. Best Parameter: It is responsible for the overall control of the simulation in order to find the best parameter for specific trading strategy. Best Parameter runs a number of different set of
parameters over the same historical data in the simulation, and the set of parameters generates maximum profit will be returned to the external BPs (e.g. BS). We can see that Simulation Conductor is used as an internal service.

3. Trading Strategy Execution: implements the strategy being evaluated by monitoring the market real time data and generating orders according to the strategy being evaluated (i.e. new orders generated from trading signals). Orders are submitted to the basic service TES for processing.

4.4. Broker Service (BS)

As showed on the top of Figure 5, the main purpose of the BS is to make use of TSS to formulate trading strategy plan to clients in order to maximize their profits. The BS takes each stock as input, it runs TSS for a number of different set of trading strategies over recent historical data overnight, so the best trading strategy plan that generates most profit will be updated in the database daily. Whenever clients want to submit an order to buy or sell a specific stock, the updated trading strategy plan will be return as output. The main functionalities of the BS are follows:

1. Order Execution: Once an order is received from a client by BS, a trading strategy and its related parameters are recommended according to the stock ID (e.g. BHP) for clients’ consideration. The order will be submit to the stock exchange. Settlement such as transferring money between bank accounts will be executed once a trade is executed.

2. Client Profit Management: There a set of different trading strategies for each stock (e.g. BHP), TSS will be run to find the best parameter for each trading strategy. The best trading strategy generates most profit will be stored in database for Order Execution to use later.

5. BUSINESS PROCESS IMPLEMENTATION

We use the paradigm of two-level programming for WS [5]: Programming in the small for implementing the basic services and additional functionalities used by a BP, and programming in the large for orchestrate the basic services into the BP.

5.1. Programming in the small

This step develops basic services to be accessed by BP, it involves three steps:

1. Modifying the existing systems (E.g. Old Trading Strategies, TES and TDS) which determined by Service Identification phase in BP analysis section, make sure after they are deployed to basic services, their functionalities are compatible with SOC message and interface standard.

2. Developing the necessary additional applications identified by Business Gap Analysis phase in the BP analysis section. In our case study, the component New Trading Strategies (Showed in Figure 3) need to be developed.

3. Deploying the applications produced in step 1 and 2 into WS as basic services.

All services are implemented as autonomous entities using Enterprise Java Beans (EJB) components, and deployed on the Tomcat J2EE-based application server. The Web Service [6] is supported by the Apache Axis SOAP engine and a MySQL relational database is used for storing administrative information such as security permissions and submitted query details. The reason of
choosing J2EE and Apache Axis is because they are open sourced software and widely supported.

5.1. Programming in the large

The second step is to orchestrate those basic services deployed in last step into BPs. There are a number of standards that support BP orchestration which assembling a set of discrete services into an end-to-end process flow [7]. Widely supported by large organizations such as Oracle, IBM, Microsoft and BEA, Business Process Execution Language (BPEL) is totally compatible with BPMN that provides a smooth and quick transformation between design and implementation phase. In a number of free software which support BPEL (e.g. ActiveWebflow, ActiveBPEL, BPEL4J and OracleBPEL), we select the free software OracleBPEL Process Manager (10.0.2), because with a graphical user interface, it offers a comprehensive and easy-to-use infrastructure for creating, deploying and managing BPEL business processes comparing to the other software.

BPEL is used to model message flows among services for BPs. Taking the implementation of the TSS as an example, Figure 6 shows part of BPEL code for TSS to highlight the message flow between services. We can see that TSS interacts with “trading-engine-service”, "trading-data-service" as external services, it also make use of internal service "simulation-conductor" in order to optimize the Trading Strategy parameters.

6. DISCUSSION

There are several major concerns during the design and implementation of this case study:

1. **Service granularity** refers to the scope of functionality exposed by a service. Different level of granularity will enforce the trade off between performance and flexibility. Take TES in Figure 1 as an example, we could have the finer-grained interfaces such as “validate broker”, “Validate order” and “Execute Order”, as an alternative to the function “Process Order”. This would provide more information about internal state of TES which surely helps us to keep the BP consistency. On another hand, such finer-grained messages result in increasing network traffic and BPEL work load which will definitely affect performance of the BP. The decision on choosing which level of service granularity often depends on the industry domain characteristic. In this case study, the data need to be passed between TES and TSS such as all the buy/sell orders, trade confirmation. Furthermore real time orderbook information has already cause a heavy network load. We could not afford a finer-grained interface which will slow down the performance tremendously by causing double or triple network load.

2. **Modification on existing systems before deployment to WS.** We can not assume that existing components to act as WS just by creating wrappers and leaving the underlying component untouched. For example, WS has Stateless feature - every time when a Web service is accessed means a new underlying object will be created. In order for all the WS requests referring to a unique TES, we need to modify the underlying business logic of TES. One way of doing this is to make the main class (implemented in Java) of TES to be a static class, this will make sure only one TES instance is created no matter what web services is called.
7. CONCLUSION

The paper has focused on developing Broker Service involving the activities surrounding Capital Markets trading. Methodologies involving analysis, design and development have been discussed. By using WS as service standard, we experience cast of development time and cost even with the existence of heterogeneous systems or technologies. The SOA based BPs are highly maintainable and easy to adapt to market changes. This is extremely important for financial industry since platform-independent adaptability brings in new market opportunities globally which is very difficult for traditional software infrastructure to achieve.

However, there are two limitations in our developed BPs. The first limitation is delayed WS response time comparing to the application-to-application communication. The second limitation is the difficulty to solve inconsistency problem in a distributed environment that is created by the loosely coupled nature of WS. For example, if we experience an exception in a BP, it is very hard to find out which element of the BP is not working properly. It is even harder that we could not identify whether such exception is caused by any services in the BP or the business logic in the BP itself.

The Future Work will concentrate on developing another layer between the WS layer and BP layer to overcome the inconsistency problem. Such layer will be used as a contract by BPs to ensure the Trustworthyness of each WS provider. We will choose to implement such layer in a XML based language based on a WS-Trustworthy framework proposed by Dr Jia Zhang and Dr Liang-Jie Zhang [8], which includes four models: resource model, policy model, validation process model, and management model.

8. ACKNOWLEDGEMENT

We would like to thank the Capital Markets Cooperative Research Centre (CMCRC) (www.cmrcrc.com) for funding this research effort and facilitating access to a number of commercial systems. Also, we would like to acknowledge Sunny Wu for his work in implementing the TDS and Johan Fisher for his help in implementing TES. Finally, we would like to thank Dr Fethi Rabhi and Dr Hairong Yu for supervising T.Tao throughout his thesis research which is the base of this paper.

10. References


