

SERVICE ORIENTED BUSINESS PROCESS DEVELOPMENT USING BPEL: A CASE STUDY

Tao Tao¹, Fethi Rabhi², Hairong Yu², & Fan Xiong³

¹School of Computer Science & Engineering, University of
New South Wales, Sydney, NSW, 2052, Australia

²School of Information Systems, Technology and
Management, University of New South Wales, Sydney, NSW,
2052, Australia

³Information Security Institute, Sichuan University, Chengdu,
Sichuan, 610064, China

ABSTRACT: In recent years, we have witnessed a tremendous growth in information and communication technologies that facilitate the design and implementation of complex enterprise business processes. One of the major innovations is the concept of Business Process (BP) which organises services around processes rather than functions. The paper demonstrates a case study which designs a number of BPs in Capital Markets (CM), and implements those BPs using Web services technologies. This paper focuses on adaptability and flexibility, so the system to be easily built and integrated with dynamic changing business environment, with minimized development time and costs.

1 INTRODUCTION

The area of finance has always evolved along side with the development of new technology. Adopting a new technology is often used to gain competitive advantage which is an important requirement for many financial institutions in this industry [AFJMRR97]. In recent years, the concept of defining Business Processes (BPs) in a high level notation has become increasingly popular in financial industry. Besides bringing opportunities to seize global market share, BPs support higher flexibility and maintainability; furthermore, BPs is ideal for applications that fit the definition of true Straight-Through Processing (STP) [DAL05].

This paper focuses on Capital Markets (CM) trading cycle, because this area is characterized by a large number of heterogeneous systems that play an important role in supporting the trading cycle 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 [RYDW05]. The aim of this case study is to design and implement a simplified Broker Service (BS) BP. A common feature of BS is to formulate a Trading Strategy (TS) plan which advise customer on how or when to place an order by identifying some relevant market conditions to maximize profit, and managing customer requests throughout the entire trading cycle (e.g. routing orders, trade settlement).

This paper is organized as follows. Section 2 demonstrates the design of four BPs in order to build BS. Section 3 discusses the some technical details and problems during the implementation of those BPs. The final section concludes the benefits and limitation of using BP.

2 BUSINESS PROCESSES (BPS) DESIGN

We adopt the following terminology when describing BPs. A “service” will refer to the definition of a service in a Service-Oriented Architecture (SOA) which considers software as being made up with autonomous, dynamic, loosely coupled and service-based components [RYDW05]. The SOA approach allows the concept of service to expand to autonomous computing entity (i.e. Basic Business Process (BBP) or basic service), as well as Complex Business Process (CBP) (i.e. composite service) that requires the intervention of a number of other BPs in order to complete the process. A “functionality” will refer to a particular function which is part of the service and can be implemented in code.

We use the Business Process Modelling Notation (BPMN) version 1.0 standard to draw Business Process Diagram (BPD) which shows the functionalities involved and the relationships between them. The objective of BPMN is to support process management by both technical users and business users by providing a notation that is intuitive to business users yet able to represent CBP semantics

[COV05]. Each BBP is shown as a Pool which is a rectangle containing several Lanes. The Lane is used to organize and categorize business activities or system functionalities. Due to the space limits, not all the functionalities or activities are sketched in the BPD.

We choose to design our BS in a bottom-up way. Firstly we will build two BBPs - Trading Engine (TE), and the Trading Data service (TDS) which play an important role in a number of BP scenarios. Then we will build two CBPs - Trading Strategy Simulation (TSS), and Broker Service (BS) which integrated TE and TDS in order to provide client profit management service.

2.1 Trading Engine (TE)

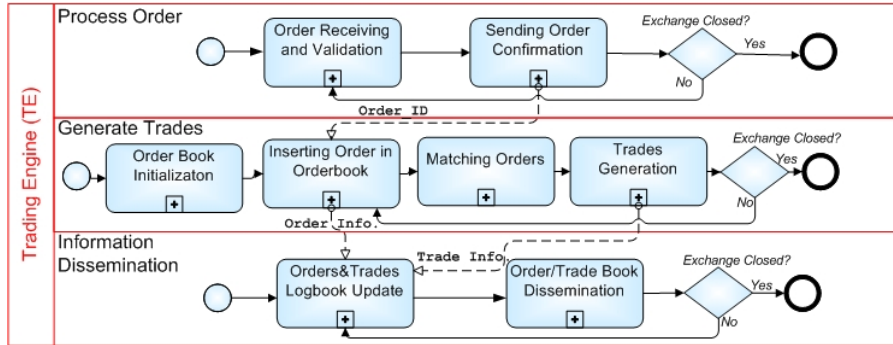


Figure 1. BPD of Trading Engine (TE)

TE simulates functionalities provided by a stock exchange. A TE allows traders to place their orders and performs matching according to the financial market model it supports. After submitting an order, the trader receives a unique confirmed order identification which can be subsequently used in cancellation or amendment requests. Traders can check the position of their orders by inspecting the order book which display all buy and sell orders. The main functionalities of Trading Engine are summarized as follows:

1. **Process Order:** Orders (buy or sell) are processed when they are submitted to the TE 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 date). The orders include cancelled and amended orders, which are occasionally submitted by traders. All the valid orders are collected in a data structure called the order book.
2. **Generate Trades:** Trades are generated by examining the order book 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 Trading Engine (including orders and trades) is stored and disseminated to the market participants through a suitable communication infrastructure (usually based on a publish/subscribe model).

2.2 Trading Data Service (TDS)

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 queries: 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). A frequency (e.g. daily, weekly or monthly) can be associated with the summary and Metrics represents some analysis of the performance or state of a security (e.g. Beta which measures a stock's volatility and VWAP which is a trading benchmark). The main functionalities of TDS are summarized as following:

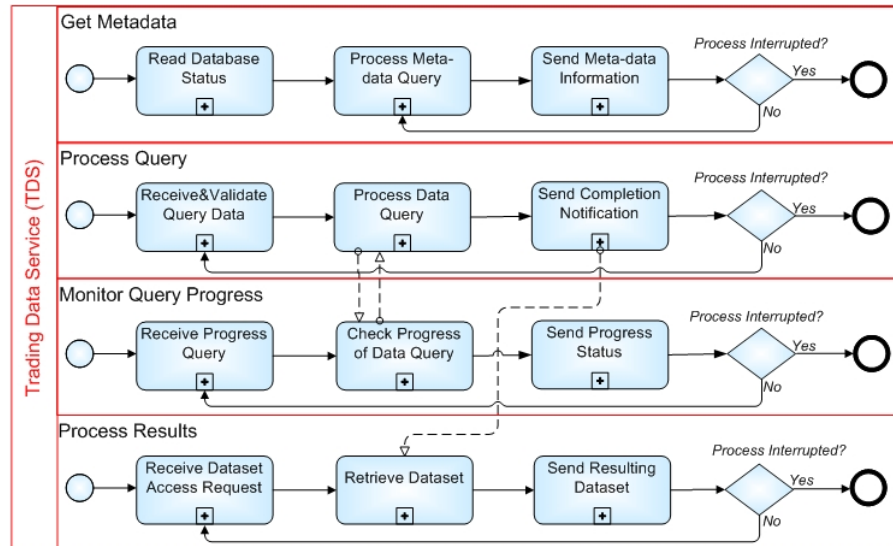


Figure 2. BPD of Trading Data Service (TDS)

1. **Get Metadata:** determines what markets and types of 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).

2.3 Trading Strategy Simulation (TSS)

The first CBP is TSS (shown in the middle of Figure 3). It consists of re-running the market events of a specific previous trading period using a certain TS then evaluate its effectiveness. Recall that an 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 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 BBP TDS.
2. **Best Parameter:** It is responsible for the overall control of the simulation in order to find the best parameter for specific TS. Best Parameter runs a number of different set of parameters over the same historical data in the simulation, and the set of parameter generate maximum profit will be returned to the external BPs (e.g. BS).
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 BBP TE for processing.

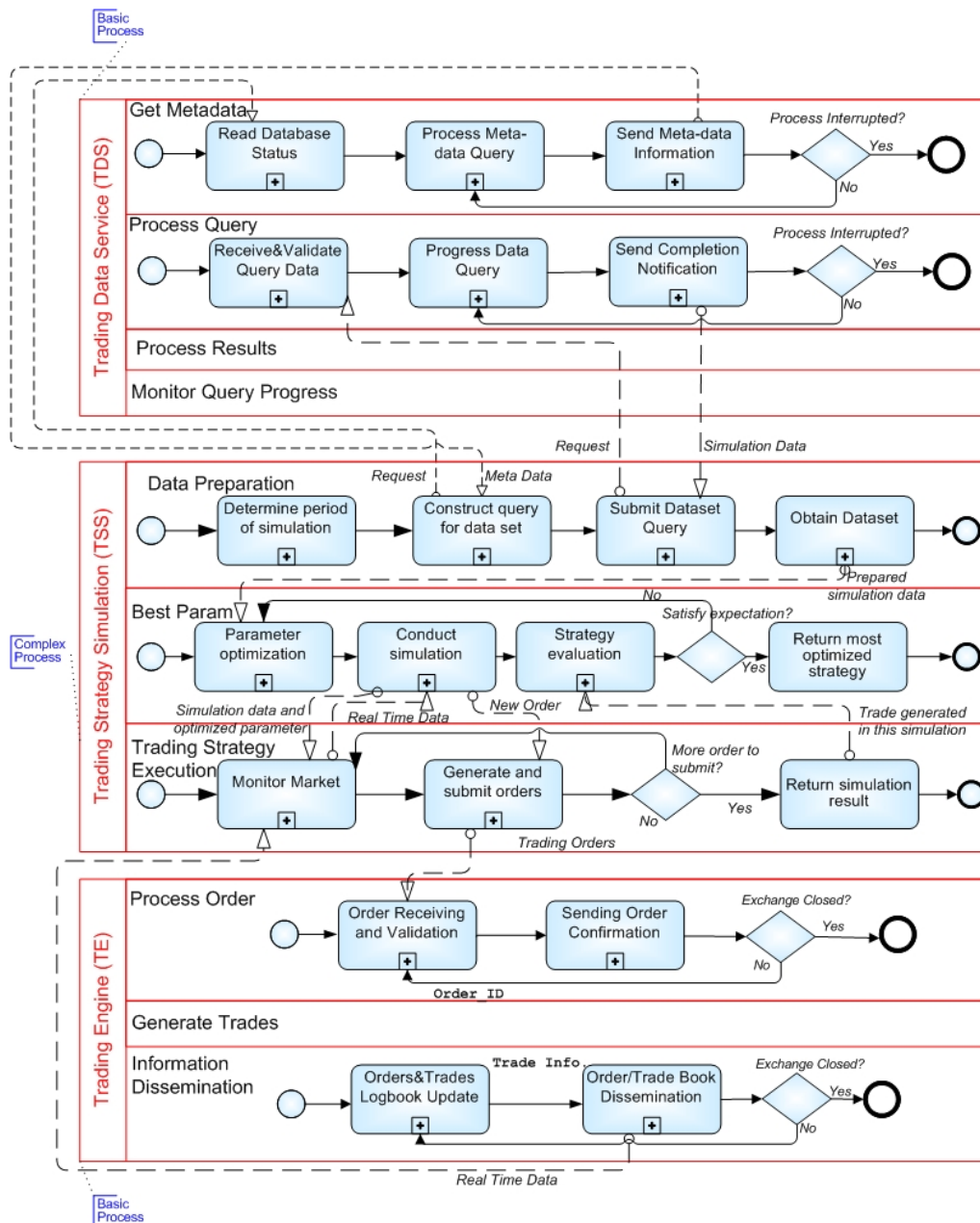


Figure 3. BPD of Trading Strategy Simulation (TSS)

2.4 Broker Service (BS)

The main purpose of the CBP BS is to provide Trading Strategy recommendations to clients in order to maximize their profit. For each stock, the BS runs TSS for a number of different set of TSs over recent historical data overnight, so the best TS generates most profit will be updated in the database daily. Whenever clients want to submit an order, the updated TS recommendation will be provided. The main functionalities of the BS are follows:

1. **Order Execution:** Once an order is received from client by BS, a TS and its related parameter 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.

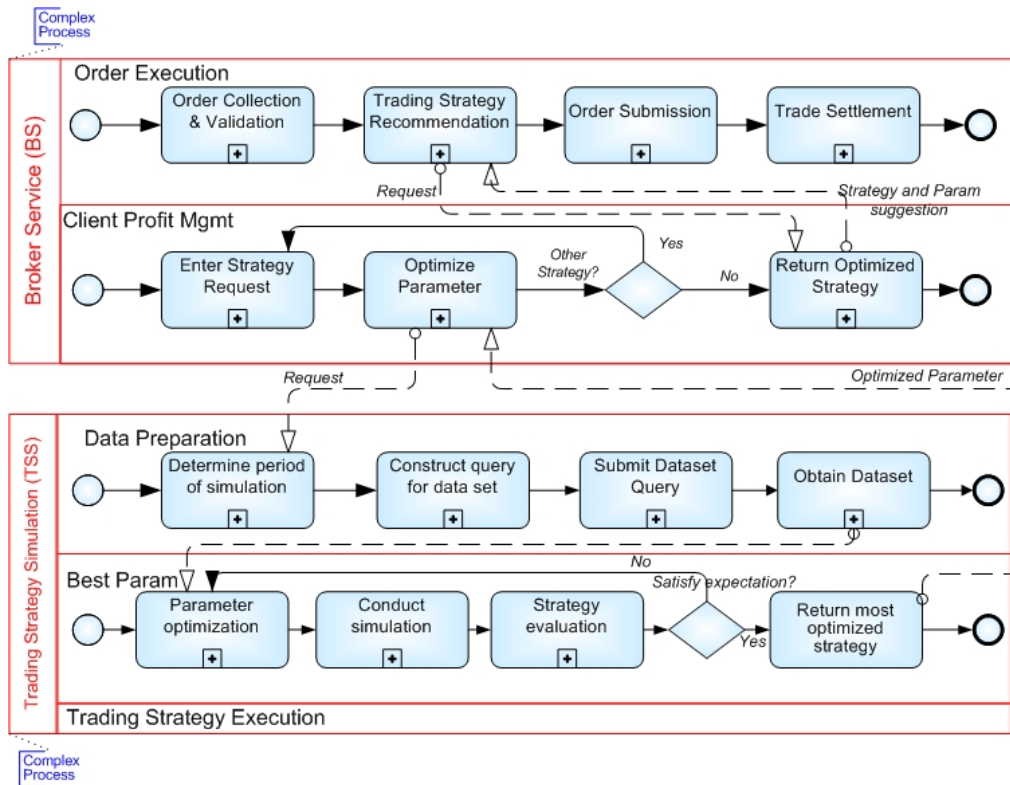


Figure 4. BPD of Broker Service (BS)

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

3 BUSINESS PROCESSES (BPS) IMPLEMENTATION

3.1 Basic Business Processes (BBPs) implementation

After design the BPs, we first implement BBPs TE (Figure 1) and TDS (Figure 2) which will be the basis for CBPs implementation described in next section.

All BBPs are implemented as autonomous entities using Enterprise Java Beans (EJB) components, and deployed on the Tomcat J2EE-based application server. The Web Service [W3C] 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 open sourced software and widely supported.

Each Pool in previous BPD is corresponding to a Web service with a description in WSDL format. Each Lane in those Pools will be represented by one function provided by Web service. For example, Figure 5 shows part of WSDL code for TE (BPD in Figure 1).

```
<wsdl:message name="processOrderRequest">
  <wsdl:part name="args" type="impl:ArrayOf_xsd_string" />
</wsdl:message>
<wsdl:message name="generateTradesRequest">
  <wsdl:part name="transaction" type="xsd:string" />
</wsdl:message>
<wsdl:message name="informationDisseminationRequest">
  <wsdl:part name="transaction_id" type="xsd:long" />
</wsdl:message>
```

Figure 5. Part of Trading Engine's interface in WSDL format

3.2 Complex Business Processes (CBPs) implementation

There are two major steps when implementing CBPs BS and TSS. First step is to build necessary internal BPs for CBPs, this is in similar way how we implement BBPs in section 3.1. Second step is to combine those internal BPs with external CBPs by accessing their Web services using a BP orchestration tool.

There are a number of standards that support BP orchestration which assembling a set of discrete services into an end-to-end process flow [PAP03]. Widely supported by large organizations such as Oracle, IBM, Microsoft and BEA, Business Process Execution Language (BPEL) is totally compatible with BPMN which provides a smoother and quicker transformation between design and implementation phase. Therefore, we select BPEL as BPs orchestration standard. 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 BPs for CBP. Taking the implementation of the CBP TSS as an example, Figure 6 shows part of TSS implementation corresponding to BPD in Figure 3. The first step is to implement business logics for “Conduct Simulation (CS)” and “Generate and submit orders (GSO)” and deploy them into Web services as internal BPs. The second step is to connect CS and GSO with external BBPs TE and TDS in order to perform as a whole CBP TSS.

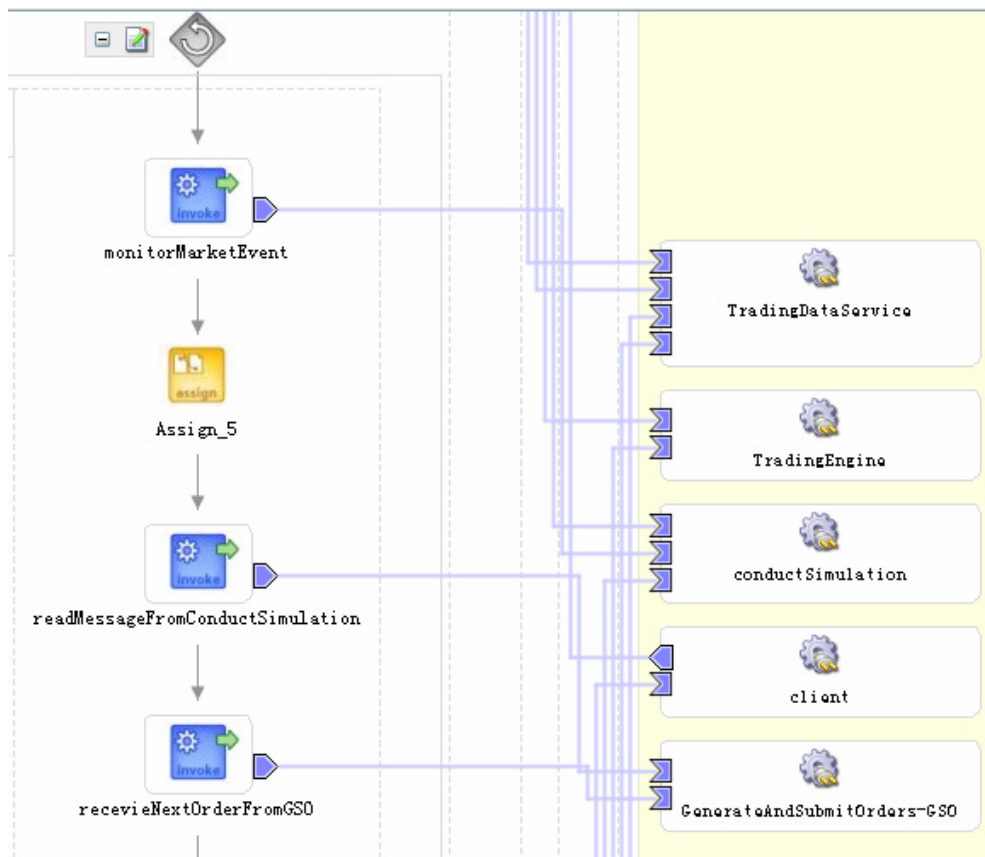


Figure 6. Part of graphic BPEL code for TSS

It is worth mentioning that Oracle BPEL console provides HTML interfaces to facilitate internal testing for CBPs besides WSDL interfaces, which makes it much more convenient for developer to revise the process. Shown in Figure 7 is a HTML interface for TSS (shown in Figure 3).

StrategyName	<input type="text"/>	string
BrokerAccount	<input type="text"/>	string
BrokerPassword	<input type="text"/>	string
SecurityId	<input type="text"/>	string
HistoricalDataSpec	<input type="text"/>	string
MaximumVolumeToTrade	<input type="text"/>	int
PercentAboveToSell	<input type="text"/>	double
PercentBelowToBuy	<input type="text"/>	double

Figure 7. The HTML page for TSS process

3.3 Technical Problems

When we build BBPs, we need to be careful to modify our underlying business logic to be compatible with Web services. For example, Web services are Stateless [BRP04], every time when a Web service is accessed means a new underlying object will be created. As a result, it causes problems when we want a Web service to be Stateful to remember previous transactions. For instance, in our case study, when we run TSS, we want all the orders to be sent to a unique TE, and also it should remember all the transactions it has previously done. This problem can be solved by making the underlying Java or C++ to be static, by doing this we can enforce all the operation related to the specific class refer to one object.

When we build CBPs, the most challenge issue is the integration between Oracle BPEL and Axis. We will give out two examples as follows:

```
<typeMapping
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
qname="ns1:string"
serializer="org.apache.axis.encoding.ser.SimpleSerializerFactory"
deserializer="org.apache.axis.encoding.ser.SimpleDeserializerFactory"
type="java:java.lang.String"
xmlns:ns1="http://www.w3.org/2001/XMLSchema"
/>
```

Figure 8. Part of code in Axis deployment descriptor to force type mapping.

1. The main problem with OracleBPEL that it is not totally compatible with some SOAP standards provided by Axis. For example, BPEL is not compatible with "soapenv:string" which is Axis' default deployed string for Java String, and this causes error such "Exception: Problem with xml schema". We have map "Java.lang.String" to "xsd:string" which is supported by OracleBPEL in the deployment descriptor. We do this by adding the whole expression as shown in Figure 8 after the <service> element in the Axis deployment descriptor.
2. Oracle BPEL does not support complex object data type (e.g. classes in Java or C++) yet. Hence we have to insure the Web services provide simple data type (e.g. int, string) message for BPEL's accessing purpose.

4 CONCLUSIONS

The paper has focused on developing Broker Service involving activities surrounding Capital Markets trading. Benefited from Web service technologies, we experience less development time and cost; the service client applications can be as heterogeneous as possible since there is no restriction placed on the clients' architecture, platform and applications as long as it can make WS calls. This is extremely important for financial industry since it brings in new market opportunities globally which is very difficult for traditional software infrastructure to achieve.

5 DISCUSSION

However, there are two limitations in our developed BPs. The first limitation is that Web services provide delayed response time comparing to the application-to-application talk. The second limitation is less control problems are created by Web services' loosely coupled nature. For example, if we experience an exception in a CBP, it is very hard to find out which BBP of the CBP is not working properly. In the future we are going to optimize our BPs in order to have a better performance such as quicker response time and business process monitoring.

ACKNOWLEDGEMENTS

We would like to thank the Capital Markets Cooperative Research Centre (CMCRC) (www.cmcrc.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 TE.

REFERENCES

- [AFJMRR97] M. Aitken, A. Frino, E. Jarnecic, M. McCorry, R. Segara, and R. Winn. The microstructure of Australian stock exchange: An introduction. Securities Industry Research Centre of Asia-Pacific (SIRCA), 1997.
- [All03] R. Allen, Workflow: An Introduction, www.wfmc.org/information/Introduction.pdf, 2003
- [BRP04] B. Benatallah, F.A. Rabhi and H. Paik, E-Commerce Enabling Technologies, 4th Edition, Pearson Education, 2004
- [COV05] R. Cover, reports on the release of Business Process Modeling Notation (BPMN) Version 1.0. <http://xml.coverpages.org/ni2003-08-29-a.html>
- [DAL05] Mark Daley, Pragmatic New Business STP, White Paper, 2005. http://whitepapers.telecomasia.net/detail/RES/1119018985_430.html
- [PAP03] M. P. PAPAZOGLU, Web Services and Business Transactions, World Wide Web: Internet and Web Information Systems, 6, 49–91, 2003
- [RD04] F.A. Rabhi, F. Dabous, H. Yu, B. Benatallah, and Y.K. 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.
- [RYDW05] F.A. Rabhi, H. Yu, F. Dabous and S. Wu, A Service-Oriented Architecture for Financial Business Processes, FiananceCom05 international conference, 2005
- [W3C] Web Service Definition, <http://www.w3.org/>