A Case Study in Developing Web Services for Capital Markets

Fethi A. Rabhi, Feras T. Dabous, and Hairong Yu
School of Information Systems Technology and Management
University of New South Wales, Sydney NSW 2052 Australia
{f.rabhi,f.dabous,hairong.yu}@unsw.edu.au

Boualem Benatallah and Yun Ki Lee
School of Computer Science and Engineering
University of New South Wales, Sydney NSW 2052 Australia
{boualem,yunki}@cse.unsw.edu.au

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. For instance, utilising new technologies in capital markets trading automation is one of the major factors for the market efficiency and competitiveness as time has a huge impact on the costs incurred by financial institutions [1].

Systems used by financial institutions are often proprietary thus hindering interoperability between potential business partners. While this was not a major problem for decades, the recent widespread use and acceptance of the Internet means that communication between geographically dispersed entities should no longer be a challenge. Businesses can now share information in an established B2B environment, take part in value chains and involve business processes from other business entities. However, technologies that have been inherently in use cause the composition and execution of business processes to be largely adhoc, time consuming and requiring enormous effort in low-level programming and managing the communication environment [3]. The emergence of Web services technologies as a framework for developing interoperable e-services has been endorsed by many development platforms that facilitate automated business processes composition. The lack of realistic large-scale applications that utilise such emergent technologies has motivated this project. Considering capital markets as our case study, this paper investigates the usability of these technologies in implementing business processes that span across a number of legacy applications.

This paper is organised as follows: section 2 describes Web services as emerging technologies that facilitate the composition and execution of distributed business processes. Section 3 presents an overview of a service-oriented architecture for capital market systems (CMSs). This architecture is meant to integrate existing legacy applications and facilitate the automation of trading-related business processes. Section 4 discusses a selected business process scenario and presents it as a composite Web Service called the broker service. Section 5 concludes this paper and presents our ongoing and future work.

2. Web services technologies and applications

Distributed computing over the Internet has been encouraged since the introduction of distributed object technologies such as CORBA and Java RMI. Object technologies are tightly coupled and support well interactions between related components or applications (i.e. within an enterprise). In addition, integration over the Internet has gained momentum since the introduction of the Web services concept. The Web services architecture has been endorsed by both middleware technology vendors and distributed applications development communities especially in the e-businesses domain for its simplicity in integrating arbitrary applications within and across enterprises.

Benatallah [3] distinguishes three key issues when building and executing Web services: fast composition in the rapid competitive market, scalable composition as the number of Web services is potentially increasing, and distributed rather than centralised execution. There are a number of platforms for Web services and business processes composition and execution. These platforms utilise the emerging Web services technologies: WSDL, UDDI, and SOAP [7]. Emerging efforts in this area mainly by Microsoft and IBM such as BPEL4WS [2] have the potential of promoting Web Services as adequate integration platforms that will be able

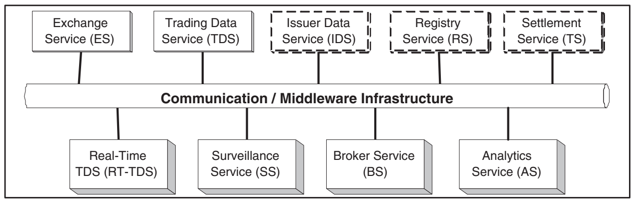

Figure 1. Preliminary Software Architecture for CMSs

to support complex interactions between applications.

Many studies have discussed the trade-offs involved with the introduction of these new technologies and platforms. For example, some studies have emphasised the performance degradation of Web services which could significantly affect response delays in real-time applications. However, few of these studies have been realised in the context of realistic applications. For instance, the throughput of a Web service can be acceptable if every possible realistic usage of the web service would still be within the required throughput.

This paper is concerned with large-scale realistic applications of Web Services. It starts with the definition of a service-oriented architecture that capture business-specific functionalities in the form of services. By describing the implementation of this architecture using Web service technologies, it gives useful insights into a number of development issues such as service composition, performance and security.

3. A service-oriented architecture for capital markets

A preliminary service-based architecture for capital market systems, which is presented in [6], is shown in figure 1. In this architecture, the authors distinguish two types of services which are basic (or elementary) and integrated (or composite). A basic service is entirely processed within one architectural component. The component itself is not necessarily a single application: it could be an entry point for an enterprise workflow or ERP system, but from the architecture's perspective it is viewed as a single component. A basic service is typically implemented on top of a legacy system by including a software wrapper that makes the correspondence between system calls and service calls. An integrated service is a service offered by one component, but it invokes several other components that provide "outsourced" services.

In this study, three basic services have been implemented. These services are:

- the Exchange Service (ES): allows traders to place, cancel and amend their orders on the market. The Exchange Service has been built on top of a fully-fledged commercial financial market trading system called X-Stream[9] which has been developed over many years [12]. All configuration information is held in a relational database so that the system can be configured to work with different market structures.
- the Trading Data service (TDS): allows subscribers to access historic financial trade data. TDS unifies access to data from heterogeneous sources through a Web interface that processes user queries. TDS contains also metadata that provide information to users about data availability and the kind of queries that can be formed [5].
- the Surveillance Service (SS) which receives real-time transactions (orders and trades) and check s against them illegal trading behaviour rules, then issues and disseminates alerts (if any) to market analysts. Our implementation is based on a commercial system called SMARTS [11]. SMARTS employs a special purpose built-in alerting and analysing language called ALICE and therefore can adapt itself to new market rules and regulations by simply updating the corresponding ALICE program.

These services will be used in a number of business process scenarios one of which will be discussed in the next section.

4. Towards Integrating Business Processes as Composite Services

The main advantage of providing basic services integrated with realistic applications is to enable composite services to be defined.

4.1. Examples of composite services

Here are two examples of composite services:

- **Analytics service**: before submitting orders to the market, brokers must pay attention to the size of the order. Since large orders impact the market, they are typically broken into smaller orders (called a *trading plan*) to place on the exchange. In our case, the Analytics Service is responsible for generating trading plans for large block orders so that they can be placed on the exchange without substantial market impact. Given a security code and an order size, an invocation of this service returns a list of order sizes that can be placed onto the exchange at regular intervals. Analytics is a composite service because it requires access to historical trade data information through Trade Data Service (TDS).

- **Broker service** : its role is to take a large order and execute it as a series of smaller orders submitted to the Exchange Service (ES). It is a composite service that invokes the operations of Analytics (another composite service) and Exchange services.

4.2. Example usage scenario

As typical usage scenario for a composite Web service, consider the following example. Bill works for an investment bank and has been told to sell 10 million BHP shares. However, once this order is submitted to the trading engine, it could have an impact on the market price for BHP. Traditionally, Bill would have had to decide by himself how to split this order into smaller ones and when to place these orders in the market.

With all financial services exposed as Web services, Bill can place his order by contacting a Broker Web service stating that he wishes to sell 10 million BHP shares. The following list shows the interactions occuring between different Web services in order to perform the broker scenario described earlier.

1. Bill contacts the Broker composite service intending to sell 10 million BHP shares.
2. The Analytics service is invoked with these parameters.
3. The Analytics service itself is a composite service. It requires trend information regarding BHP and contacts a Trading Data service.
4. The trend information is calculated and a trading plan is generated and returned to Bill for confirmation.
5. Bill checks the plan and approves it.
6. The Broker service submits orders according to the plan to the Exchange service.
7. Each order that is placed on the Exchange service successfully generates a receipt which is returned to Bill.
8. The Surveillance service monitors each order and the generated trades. It detects possible illegal actions such as trading based on insider information.

4.3. Tool support for developing composite services

Some development platforms are emerging from leading vendors such as IBM and Microsoft. IBM WEBSHPERE studio provides a single consistent programming environment based on open standards. WEBSHPERE Business Integration (WBI) Modeller, Server, and Monitor support a unique visual environment for defining, analysing, simulating, deploying, and monitoring of business processes. Microsoft .NET also supports a very similar platform for business process integration through the provision of a number of associated tools. These competing tools still in their infancy in supporting business process integration requirements.

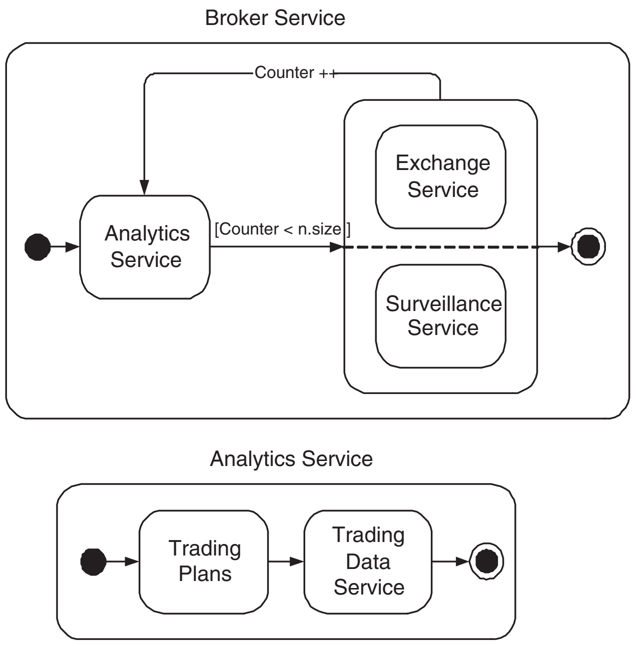

Figure 2. Statechart for the broker composite service

We selected an experimental platform called SELF-SERV for service composition because it supports a clear distinction between basic and composite services [4]. SELF-SERV has a visual it Composer for composite services that uses the concept of statechart. Figure 2 depicts the statechart for the Broker composite service that shows the interactions described earlier. It clearly shows that after the invocation of the Analytics service, the Exchange and Surveillance services will be executed in parallel. Each order on the list returned by the Analytics service is submitted to the Exchange and Surveillance services.

Some current experiences of using SELF-SERV are described next:

- Web services can deployed within Web application servers with a SOAP engine. Therefore there is no need to have coordinating classes which must be downloaded to be run as separate Java programs that listen on their own port. Deploying such classes as part of the Web application allows these classes to be protected by the same security policies as the Web container. Another advantage is that no extra ports need to be opened at the company firewall.

- Trivial condition classes that perform tasks such as checking whether strings are equal or return values are within some range need to implemented by Self-serv's Service Composer. These condition classes are required by the statecharts to direct flow of control between different states. While implementing these classes are trivial and do not take substantial time, it does not promote code reuse and extra code is being produced.

- Currently SELF-SERV's selection algorithms are unsuitable for selecting between the two services when requests arrive. Its current selection algorithms are based on technical criteria such as response time and availability. In the financial industry, the criteria for placing orders is the price and quantity at the given price. Therefore a new Web service that queries the prices at each exchange in the service community needs to be created.

5. Conclusion and Future Work

The paper is motivated by the lack of realistic large-scale e-service applications reported in the litterature. Despite their promises, new technologies such as Web services still need to live up their claims as a powerful means of integrating business processes that span across a number of large distributed commercial applications. Our study is concerned with financial applications especially those involved in the activities surrounding capital markets trading.

We adopted a step-by-step approach based on sound software engineering principles. First, we defined a service-oriented architecture for capital market systems (CMSs). Next, we presented three major services (i.e. Exchange, Surveillance, and Trade Data) and reported our experiences in implementing these services using Web service technologies. We provided feedback on a number of aspects including functionality, development tools, performance and security. Finally, we presented some business process scenarios and discussed their implementation as compositions of basic services supported by software tools.

Future work will be conducted on several fronts. Firstly, existing services need to be enhanced both functionally and qualitatively. For example, the Trade Data Service needs to get access to more data sources such as Reuters and at the same time maintain an acceptable response time. Secondly, the initial architecture needs to be extended with other important basic services that play an important role in capital markets trading such as those required for settlement (to settle funds transfers) and registry (to keep track of the legal ownership of securities). Finally, we need to develop more complex business process scenarios and leverage the existing services into composite services. Throughout these activities, interoperability, performance and security will be significant research focus points as they have been identified as the major areas of concern by our industrial partners.

Acknowledgments

The authors would like to thank Capital Markets Cooperative Research Center (CMCRC) [8] for its support and facilitating access to a number of commercial CMSs such as SMARTS [11] , X-STREAM [9], and SIRCA [10] trading data. The authors would also like to sincerely acknowledge Sunny Wu, Anthony Cheung, Stanley Yip, Joshua Mok, Johan Fischer and Anne-Laure Mazon for their contribution in prototyping.

References

[1] 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.

[2] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Lui, D. R. D. Smith, S. Thatte, I. Trickovic, and S. weerawarana. Business process execution language for web services (BPEL4WS) specification. *http://www.siebel.com/bpel*, 2003.

[3] B. Benatallah, M. Dumas, and Q. Z. Sheng. Declarative composition and peer-to-peer provisioning of dynamic web services. In *The 18th International Conference on Data Engineering*, 2002.

[4] B. Bwnatallah, Q. Sheng, and M. Dumas. The self-serv environmet for web services composision. *IEEE Internet Computing*, Jan/Feb 2003.

[5] T.-H. A. Cheung and S. Y.-H. Wu. Trade data service for capital markets. Honors, school of Computer Science and Engineering, UNSW, Nov 2003.

[6] F. A. Rabhi and B. Benatallah. An integrated service architecture for managing capital market systems. *IEEE Networks*, 16:15–19, 2002.

[7] U. Wahli, M. Drobnic, C. Gerber, G. G. Ochoa, and M. Schramm. *WebSphere Version 5 Web Services Handbook*. IBM Redbooks, 1st edition, March 2003.

[8] Capital market cooperative research centre (CMCRC). *http://www.cmcrc.com*.

[9] Computershare X-STREAM system. *http://www.computershare.com.au*.

[10] SIRCA research center. *http://www.sirca.com.au*.

[11] SMARTS surveillance system. *http://www.smarts.com.au*.

[12] Trading technology survey of exchange technologies. *http://www.tradingTechnology.com*, 2003.