ISeB (2007) 5:185–200
DOI 10.1007/s10257-006-0041-x

ORIGINAL ARTICLE

A service-oriented architecture for financial business processes

A case study in trading strategy simulation

Fethi A. Rabhi · Hairong Yu · Feras T. Dabous · Sunny Y. Wu

Published online: 5 October 2006

Abstract Recent years have witnessed a tremendous growth in information and communication technologies that facilitate the design and implementation of complex inter-enterprise business processes. One of the major innovations is the concept of service-oriented architectures which considers software systems as being made up with autonomous, dynamic, loosely coupled and service-based components. This paper describes an attempt to automate financial business processes by utilizing a number of basic and composite services. As a case study, the paper describes the implementation of a realistic business process that is related to simulating a trading strategy in capital markets. An evaluation of the appropriateness of service-oriented architectures is conducted taking into account a number of factors such as flexibility, performance and development costs.

Keywords SOA · Trading strategy · Capital market · Web services · Business processes · Financial application

1 Introduction

The area of finance has always evolved along side with the development of new technologies. For instance, utilizing new technologies in trading automation is one of the major factors that contributes to markets efficiency and

F. A. Rabhi (✉) · H. Yu · S. Y. Wu
School of Information Systems, Technology and Management, University of New South Wales, Sydney, NSW 2052, Australia
e-mail: f.rabhi@unsw.edu.au

F. T. Dabous
College of Computer Science and Information Technology, Abu Dhabi University, Abu Dhabi, United Arab Emirates

competitiveness and has had a huge impact on the costs incurred by financial institutions. As a result, financial systems have been undertaking constant evolutions to support new enabled business processes and adapt to drastic technological changes. In the context of this paper, a business process is defined as a series of activities which support the daily operations of an enterprise (Hammer and Champy 1993). Business processes model the behaviour of business interactions required by stakeholders and are defined based on the rules or policies of the business. They often involve cross-department or cross-institutional entities and utilize data from multiple systems. This paper is primarily concerned with the provision of an adequate software infrastructure to support inter-organizational business processes in the finance area.

Related work in this area includes business process automation technologies. Following on the work of standardization bodies such as the workflow management coalition (WfMC) (http://www.aiim.org/wfmc), there are now many workflow management system (WFMS) products available. In practice, they have proved ineffective and hard to evolve when considering the needs of business-to-business interactions (Medjahed et al. 2003) and in particular the utilization of legacy functionalities. Early efforts, such as the definition of a workflow reference model (Hollinsworth 1995) and other standardization efforts [e.g. jointFlow (Workflow management facility 1998), the simple workflow access protocol (SWAP) (Bolcer and Kaiser 1999) and Wf-XML message set (http://www.wfmc.org)], provide little support for inter-enterprise business processes. Although the next generation of inter-enterprise workflow (IEWf) systems promises to interconnect cross-organization business processes while supporting the integration of diverse users, applications, and legacy systems (Yang and Papazoglou 2000), no current approach seems to propose a completely generic solution.

Web services technology is another technique for realizing business process automation. Web services utilizes XML as the fundamental data format to form three underlying standards (Ma 2005)—the messaging format (SOAP), web services description language (WSDL) and universal description, discovery and integration (UDDI). Web Services mostly uses hypertext transfer protocol (HTTP) as the transport of message exchanges of the services. In order to integrate the individual services to form a business process, business process execution language (BPEL) is the most common language to invocate multiple services (http://www.oasis-open.org). Similar to web services, BPEL uses XML as the fundamental data format to define the execution and orchestration of web services through their respective WSDL files and provide a tracking and fault handling mechanism. It supports synchronous/asynchronous, short-lived/long-running and stateful/stateless web services (Pasley 2005). Currently all major vendors offer solutions to the market, such as IBM Service Oriented Architecture (SOA) Foundation (http://www.306.ibm.com/software/solutions/soa), Oracle SOA Suite (http://www.oracle.com/technologies/soa) and webMethods Fabric (http://www.webmethods.com). All of those vendor solutions provide streamlined integration of web service technologies along with higher-level features such as service registry, monitoring and

management, business process orchestration and analytics software. Integrated development suites that offer a number of building blocks for constructing the business processes are also included.

Despite these developments, a large and complex application domain such as capital markets presents a number of challenges. Amongst these challenges, we identify the following ones as of major concern to potential developers:

Intellectual complexity A consistent view and understanding of business processes spanning different market segments as well a number of networked computer systems requires multidisciplinary knowledge in areas as diverse as finance, economics, accounting, information systems, networking and computing systems. This results in high development and management costs of the underlying infrastructures.

Difficulty in evaluation Whilst new technologies have clear benefits in terms of technical quality attributes (e.g. better interoperability, higher number of transactions per second), it is hard to relate them to business-related quality attributes such as the performance of an investment strategy, the risk level, the transaction cost etc. A consequence is the difficulty in assessing the impact of introducing a new technology or design, studying trade-offs between different implementation choices etc.

Methodologies that are being used in practice can be classified as either top-down or bottom-up. A top-down strategy works its way from business process specifications (e.g. workflows) down to implementations. A bottom-up strategy usually involves the use of frameworks that guide the process of integrating legacy systems. Examples include the methodological framework of Mercella and Batini (2000) and Mecella and Pernici (2001), the business applications to legacy systems (BALES) methodology (van den Heuvel et al. 1999, 2002) (supports web services development) and the Attachmate's methodology for integrating legacy applications. In recent years, the concept of SOAs (Curbera et al. 2003; Huhns and Singh 2005) has become increasingly popular as it offers an alternative strategy. Given a community of services, it gives the opportunity to make a clear separation between the concerns of service users and those of service providers. It allows the development process to proceed in two directions at the same time. On one hand, business processes can be expressed in an appropriate notation that utilizes services (using BPEL for instance). On the other hand, service wrappers can be developed around legacy systems (for example using web services (http://www.w3.org/TR/ws-arch)). Other researchers have also started to acknowledge the benefits of an SOA in automating financial business processes (Zimmermann et al. 2004; Homann et al. 2004).

However, the literature lacks documented usage of SOAs in this application domain. One major challenge in the use of an SOA is to determine what a service actually is. An adequate trade-off has to be achieved between establishing direct connections between business concepts and existing legacy systems. This can only be achieved through a detailed study of a particular domain, examining its core business processes and the architecture of its existing systems.

This paper's contribution is to provide insights on a particular sub-domain in finance, namely the capital markets trading cycle. This area is characterized by a large number of heterogeneous systems that include information distribution systems (e.g. for real-time quotes, historical trade data), order processing systems (e.g. order routing, presentation, and execution), clearing and settlement systems, and research and analysis systems (Harris 2003). The paper describes our view on the services that operate in this area and explains their roles in the definition and implementation of trading-related business processes. Similar analysis can be conducted on other areas in the finance domain. The rest of this paper is organized as follows. Section 2 states our basic principles and assumptions. Section 3 presents an overview of our SOA that supports the trading cycle in capital markets. Section 4 is a case study which discusses the implementation of two basic services and a composite service that simulates a trading strategy. Section 5 evaluates the proposed SOA. The final section presents the conclusions and future work.

2 Service-oriented concepts

Service-oriented computing (SOC) has become increasingly popular as it offers a technology-independent view of systems. SOC is a design paradigm that utilizes the concept of ''service'' as a fundamental element for developing software. Here the ''service'' is an application logic or underlying computing resource that is exposed through an interface and which can be invoked over a network (Papazoglou 2003). The SOC approach allows the concept of services to expand to include not only an autonomous computing entity (i.e. basic service), but also a complex process (i.e. composite service) that requires the intervention of a number of other services in order to complete required activities like a business process. As a result, SOC provides an abstraction of the definition and modeling of business processes from the actual tools and systems which are required for implementation.

To build the service model, SOC relies on service-oriented architecture (SOA). An SOA is a decentralized way of thinking about composing software. It reorganizes a portfolio of previously developed software and a supporting infrastructure into an interconnected set of services, each of which is accessible through standardized interfaces and messaging protocols, and therefore provisioning interoperability among heterogeneous systems. An SOA also provides a vehicle for enforcing the business managers' perspective during software development by viewing processes as collections of inter-related services. From a recent survey (Quocirca Ltd. SOA 2005), the adoption of SOAs is growing - almost one-fifth of the surveyed companies have already actively adopted SOAs during the development cycle while another quarter is considering adopting in near future.

We adopt the following terminology when describing services. A *functionality* will refer to a particular function offered by a service and can be implemented in code. Functionalities will be given names e.g. ''Process

Query'' will refer to the processing of a query for obtaining trade-related data. In every context, there will be a number of existing legacy systems which already implement a set of functionalities.

A business process will be represented by a diagram which shows the functionalities involved and the relationships between them. We use the business process management notation (BPMN) standard (2004) that depicts the process flow dependencies among a number of activities (called business task objects) which are part of a particular functionality. The BPMN standard is promoted by the business process management initiative.

The next section presents an SOA that is concerned with the capital markets trading cycle. The following sections will describe the implementation of selected services using web services technology and their role in expressing complex business processes.

3 A service-oriented architecture for capital markets

This section describes an SOA which addresses the trading cycle in its early phases, based on a preliminary one presented in Rabhi and Benatallah (2002). Experience has shown that an architecture that supports a complete integrated trading cycle (known as straight through processing (STP) in capital markets) is not yet feasible as there are still standardization issues that are far from being resolved. STP is an ideal model that is supposed to automate all transactional processing from initiation to resolution in the same way as supply chain management (SCM) and customer relationship management (CRM) models are used in the manufacturing and the service industry respectively.

Our architecture, which is illustrated in Fig. 1, distinguishes the following types of services:

Exchange services They consist of transaction-based systems that implement market matching functions i.e. matching buyer and seller orders and produce trades. Such services allow the trading of any type of securities such as stocks, bonds, derivatives and currencies.

Trading data services They manage information related to quotes, orders and trade transactions over time. The data could be historical or obtained in real-time from exchanges in different ways e.g. publish-subscribe fashion.

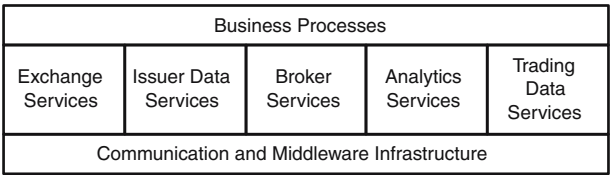

Fig. 1 A service-oriented software architecture for the trading cycle

Issuer data services They manage information about securities issuers (e.g. accounting data, announcements) over time. Data could also be historical or obtained ''on-the-fly''.

Analytics services They refer to data mining and visualization applications which analyze real-time or historical data. They comprise a variety of models for different purposes e.g. offering valuable insights on companies and industry trends for analysts, determining trading opportunities for brokers or checking compliance for regulators.

Broker services Brokerage types vary widely in the quantity and quality of the services they provide for customers. Common features involve formulating a trading strategy based on some objectives (e.g. maximizing returns, risk management), providing information, implementing trading plans, managing customer requests throughout the entire trading cycle (e.g. managing customer databases, routing orders, reporting/accounting, etc.).

These services form core components in realizing a number of trading-related business processes, one of which will be discussed next.

4 Case study

The purpose of the case study is twofold:

- Discuss two basic services (one exchange service and one trading data service) in terms of the functionalities they offer and their web service implementations based on existing legacy systems
- Describe a composite service that implements a business process involving the two basic services using BPMN.

4.1 Exchange service

4.1.1 Service description

The role of the exchange service is to allow traders to place their orders and perform 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 also inspect the orderbook to check the position of their orders. The main functionalities of the exchange service used in the case study are summarized as follows:

- *Process order*. Orders (buy or sell) are processed when they are submitted to the exchange service 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.

- *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 and the market type).
- *Information dissemination*. Information related to all activities of the exchange service (including orders and trades) is stored and disseminated to the market participants through a suitable communication infrastructure (usually based on a publish/subscribe model).

The corresponding activities are illustrated in the BPMN diagram shown in Fig. 2. We only highlight the three main functionalities described earlier. There are many other functionalities such as those that support administrative functions (e.g. authentication, user registration) and market configuration (e.g. opening/closing the market, changing the matching algorithm) that are not shown here for simplicity.

4.1.2 Implementation

A prototype exchange service has been implemented using a web service wrapper on top of a fully-fledged commercial financial market trading system called X-Stream (http://www.computershare.com.au). X-Stream is one of the first generation exchange trading systems that have been designed around a client–server architecture and can be configured to work with different market structures. All configuration information is held in a relational database. X-Stream is a distributed system as it consists of two trading engines (main and backup) and several gateways which can connect trading engine it to broker systems. Most of its code is written in C++ for optimum speed, we adopted a C++ wrapper and used an SOAP development environment called gSOAP. Some experiences on developing the exchange service using gSOAP are reported in Yu et al. (2004).

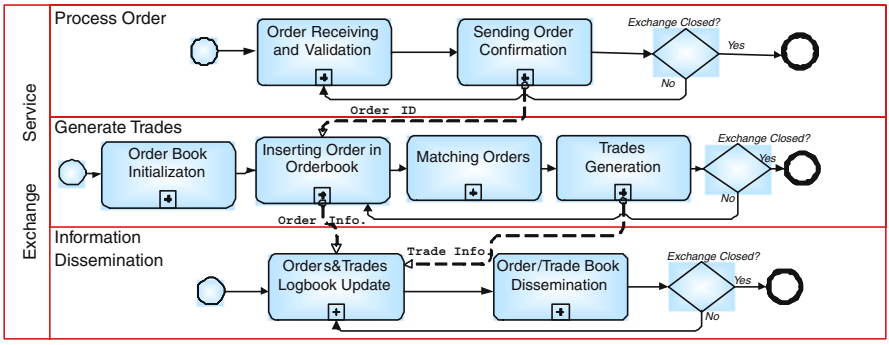

Fig. 2 Functionalities in the exchange service

4.2 Trading data service

4.2.1 Service description

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 1 day). A frequency (e.g. daily, weekly or monthly) can be associated with the summary and several metrics which represent some analysis of the performance or state of a security (e.g. Beta which measures a stock's volatility and volume weighted average price (VWAP) is a typical trading benchmark). The main functionalities of the trading data service used in the case study are summarized as follows (see Fig. 3):

- *Get metadata* determines markets and types of queries that are supported by a particular service implementation.
- *Process query* executes a query formulated according to the metadata and containing the appropriate parameters.
- *Monitor query progress* checks a query's progress in case of a query that involves a large amount of data and a long time to process.
- *Process results* relate to the way query results are accessed (e.g. downloading the data from a network server).

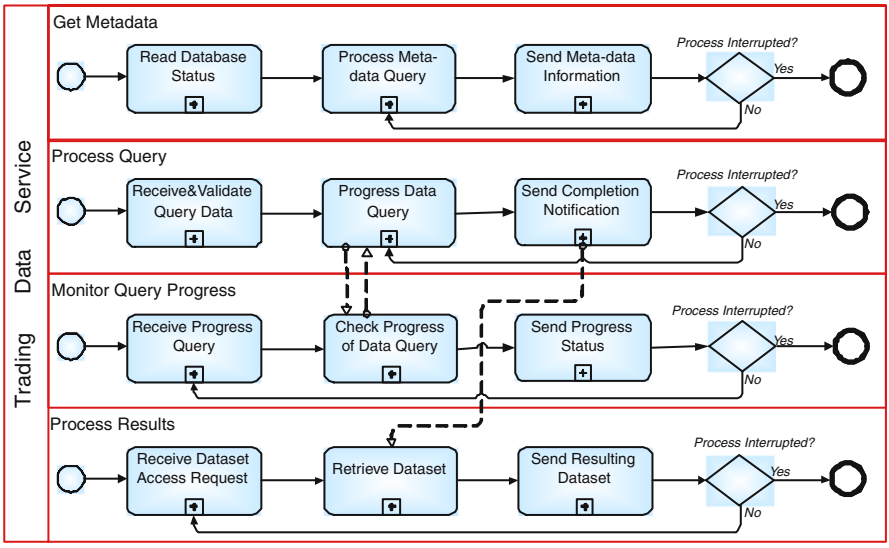

Fig. 3 Functionalities in the trading data service

Though the query types are quite different from each other, they all have a number of common parameters such as Start Date, End Date, Market, Compression (the compression type to be applied to the resulting data file), Output Format (e.g. CSV or XML), WSDLCallBack (the WSDL service end-point which TDS will invoke upon the query process completion), and Email (for sending notifications upon the completion of a query).

4.2.2 Implementation

A prototype has been implemented on top of a number of existing systems that manage large financial data repositories:

- SMARTS surveillance system: (http://www.smarts-systems.com), which is designed to process and manage trade data from a large number of financial markets worldwide. SMARTS stores all data in a proprietary binary format and provides some basic access functions in C to the data through a customized application programming interface (API).
- ASPECT accounting database: (http://www.aspectfinancial.com.au), which is a comprehensive source for listed companies on the Australian and New Zealand Stock Exchange.
- IBES financial analyst forecast database: (http://www.thomson.com/financial), which is a source for analysts' forecast data, research reports and tools for institutional money managers.

In this prototype implementation, both queries and metadata are expressed in XML so that clients can use any third-party applications (e.g. Extensible Stylesheet Language Transformation or XSLT engines) to manipulate them. TDS's business logic is implemented using Enterprise Java Beans (EJB) (http://www.oss.org) components and deployed on the JBoss J2EE-based application server. The implementation of most components involves making calls to the SMARTS API. The Web Service is supported by the Axis SOAP engine (http://www.apache.org/soap) and a MySQL relational database (http://www.mysql.com) is used for storing administrative information such as security permissions and submitted query details.

4.3 Trading strategy simulation service

We now illustrate the implementation of a composite service as a combination of several basic services. The selected service is part of a trading strategy simulation suite tools which is available for traders to test the effectiveness of a particular trading strategy. Each simulation consists of re-running the market events of a specific trading period using a particular trading strategy and then evaluating its effectiveness. 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.

The BPMN business process diagram that corresponds to this service is presented in the middle of Fig. 4 (the top and bottom part of the figure are reproductions of the two basic services described previously). We can see that there are three main functionalities involved:

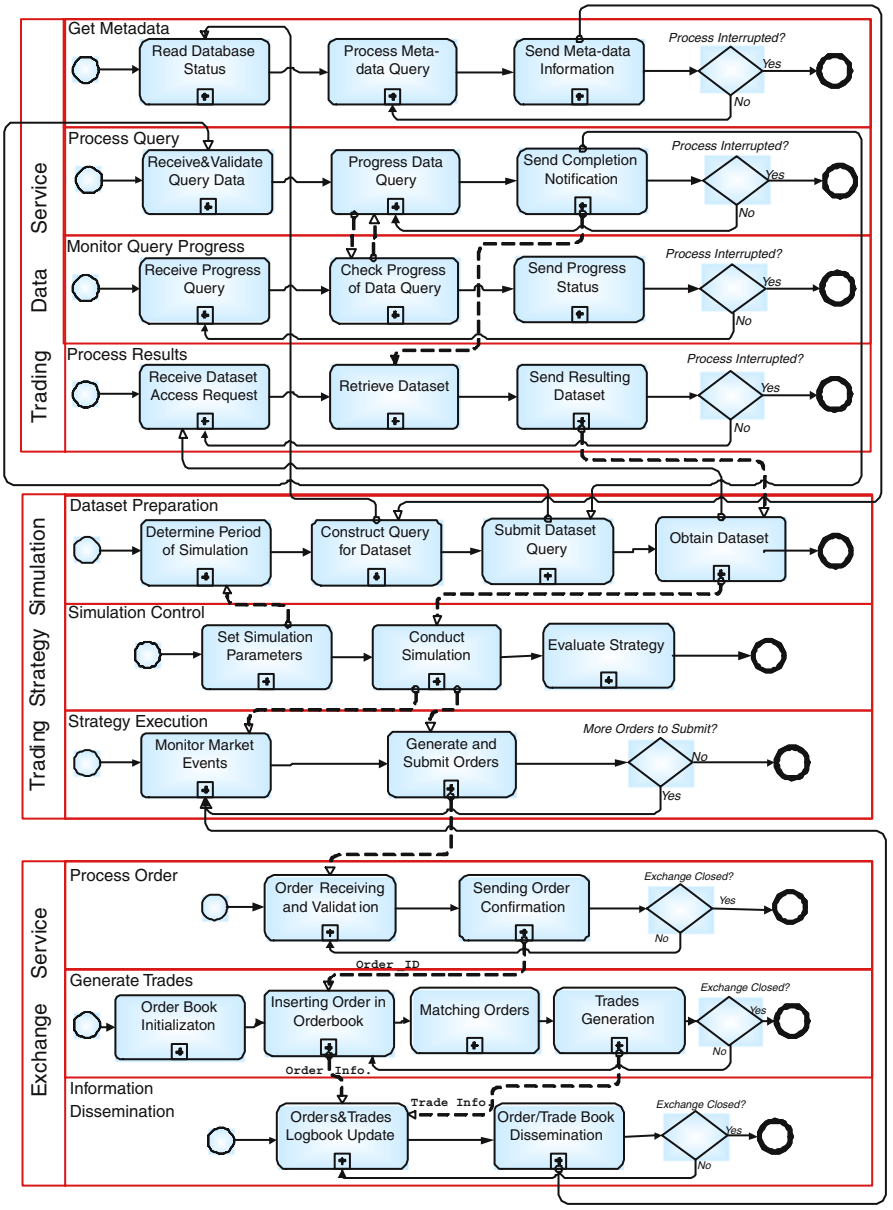

Fig. 4 Trading strategy simulation business process

- *Dataset preparation*. It involves 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 an external service (TDS shown on top).
- *Strategy execution* implements the strategy being evaluated by monitoring the market events (looking for trading signals) and generating orders according to the market conditions being recreated and the strategy being evaluated (i.e. new orders generated from trading signals). Orders are submitted to an external service (ES shown at the bottom) for processing.
- *Simulation control* is responsible for the overall control of the simulation. It reads the simulation parameters launches the generation of the historic dataset and controls the strategy execution according to these parameters. Finally, an evaluation will be undertaken and the performance of the strategy will be determined. This strategy evaluation part may access TDS to obtain the artificial trade data created during the simulation (this not shown in the figure for simplicity).

We can see that most functionalities used in this business process are either internal to the business process or ''outsourced'' to basic services like the exchange service (ES) or the trading data service (TDS). Some internal operations can themselves be made into new services and reused as part of other business processes. For example, ''Monitor Market Events'' in strategy execution can be used in business processes related to market surveillance. Also, a composite service could be made into a basic service provided there is an existing system that provides the required functionalities. This is a characteristic of this application domain which has a large number of systems providing overlapping functionalities. A consequence is that there are many alternatives in expressing business processes hence the need for flexibility through the use of an SOA and business process modeling tools.

5 Evaluation

The Service-oriented architecture needs to be evaluated in the way it addresses the two challenges outlined in the introduction: ability to handle the design of complex systems from a business perspective and the possibility to reason about its quality attributes, study trade-offs between different implementation choices, etc.

5.1 Expressing complex business processes

Concerning the first aspect, the case study described in the previous section has demonstrated the capability of describing a complex business process as several services interacting with each other visually. It also illustrated how the

trading strategy simulation business process makes use of existing facilities, e.g. here the exchange service is provided by a legacy system in a way that is transparent to the user. The use of a BPMN modeling tool (http://www.itp-commerce.com) has allowed direct connections to be made to the two prototype web services (exchange service and trading data service). Other tools e.g. acting as a business process orchestration engine (Oracle BPEL process manager available at http://www.oracle.com/technology/products/ias/bpel), have also allowed the business process to be simulated, modified and results to be checked. This is important considering that there are many variations in conducting trading strategy simulations, depending on the type of the market structure (e.g. type of orders and matching algorithms), the information available (e.g. type of market events being monitored), how evaluation is conducted etc.

This verifies the claim that an SOA and its supporting technologies allow business processes to be constructed, analyzed and modified much more easily. Opportunities for the reuse of services in several business processes are also apparent. For example, the TDS can be called by many other services from anywhere over a network for completing either do-it-yourself or outsourcing business processes.

Despite the facilities offered, we found that the existing business process description language shows limitations when more realistic situations are being modelled. In these cases, diagrams become quickly cluttered and their meaning is more obscure due to the difficulties in visually monitoring a large number of activities and connections.

5.2 Quality-related aspects

Concerning the second aspect which is the ability to reason about the quality of the architecture at a high level. We have selected three important quality criteria that play important roles in many real-life projects and for which various trade-offs exist:

- Performance: many business processes require efficient implementations to maintain an adequate response time, process high volumes of data, etc.
- Development costs: this requirement originates from business stakeholders, for which the desirable project's cost is strictly specified. This would impact other critical features such as quick deployment when the stakeholders share to maintain some competitive advantage, exploit some market opportunities, etc.
- Maintenance costs: this requirement is usually mandated by technical staff, as they would have to bear the brunt of maintaining the system.

We now present some of the results of our evaluation with respect to these three quality attributes. Additional details related to the evaluation process and quality models used are available from Dabous (2005).

5.2.1 Performance

The main difference between SOA and non-SOA implementations is that an SOA imposes additional penalties in accessing functionalities particularly those supported by legacy systems. Exposing a legacy functionality by direct invocation (i.e. through an API access method) has much better performance than a service-oriented wrapper that normally uses text-based protocols. In order to estimate the performance degradation caused by service wrappers, we have measured the access time for our prototype exchange service with and without a web service wrapper. Table 1 shows that the maximum degradation in performance is around 10%. The performance overhead is generally defined as a ratio of absolute performance difference over the combined unit performance in percent. Taking the example of one client case in Table 1, its overhead is always as (42.5 – 40.1)(orders/s)/40.1(orders/s) = 5.99%. This is based on messages containing a single transaction. In practice, existing exchanges are optimized to handle transactions in batches where the transactions in each batch share a common opening and ending. Overall the results show that the exchange service does not have much negative impact on the performance when comparing it with the original legacy system. More experimental data can be found in Yu et al. (2004).

In conclusion, service-based wrappers inevitably result in some performance degradation. Such penalties are not important in running simulation scenarios such as trading strategy simulation. In other business processes which operate in real-time, a fraction of a second can be significant as it could result in a lost opportunity to conduct a trade for example.

5.2.2 Development effort

The development effort required for implementing a business process is an accumulation of the effort that is needed in:

- developing the required functionalities and creating the wrappers for remotely invoking these functionalities: one advantage of leveraging legacy functionality is that there is no effort needed for developing that functionality. Development effort associated with a wrapper depends on the corresponding access type. For instance, the development effort for a service-oriented wrapper is much more than that of direct access method like RPC.

Table 1 Transaction rates (orders/s) for the exchange service

	Without wrapper	With wrapper	Degradation (%)
1 client	42.5	40.1	5.99
4 clients	153.1	147.0	4.15
8 clients	166.7	151.0	10.40
12 clients	180.0	169.5	6.19
16 clients	190.0	182.6	4.05

- implementing the business process logic (enactment) and invocations to the required functionalities (through their wrappers): the code required to invoke service-oriented wrappers is much simpler to develop than that of direct access methods.

In the case of the majority of functionalities being provided by legacy systems, service-oriented wrappers impose an additional development burden which is only reduced when the number of business processes that utilize such wrappers is large. For example, the functionality ''Get Metadata'' in our case study is supported by an existing system (i.e. SMARTS). Therefore, reusing such a functionality through a service-oriented wrapper can significantly reduce the development effort of other domain business processes such as brokerage business processes.

Our experience has also shown that developing wrappers for new functionalities should always be service-oriented as the development effort is not significantly affected (but it reduces the maintenance effort as we will see in the next subsection).

5.2.3 Maintenance effort

Based on the assumption that access rights to the code of legacy systems are very restricted, we are adopting the view that whenever there is a change request for the code of a legacy functionality, then the whole functionality encompassing the new change needs to be redeveloped. The new functionality would then be accessible through a new wrapper and links to the old legacy system re-routed to the new service.

One consequence is that, despite savings in development efforts, the maintenance costs of leveraging legacy functionality are very high in the long term. Providing service-oriented wrappers to either legacy or new functionalities has the advantage that code to access these functionalities will be stable and require little maintenance effort in the long term. In conclusion, when an implementation uses an SOA, the overall maintenance effort is inevitably reduced.

6 Conclusions

This work is motivated by the lack of realistic large-scale service-based applications reported in the literature. Despite their promises, SOAs and their enabling technologies (e.g. web services) still need to live up their claims as powerful means of integrating business processes that can span across a number of large distributed commercial applications.

Our study focused on financial business processes especially those involving activities surrounding trading in capital markets. We adopted an SOA which addresses the shortcomings of both top-down and bottom-up approaches. It allows business processes to be expressed in an appropriate notation at the

same time as service implementations can be realized (using web services in our case study).

Results of the case study show that it is possible to handle business processes that are much more complex than those reported in the literature. However, a service-oriented approach often assumes the existence of well-specified business processes (i.e. that can be represented using BPMN diagrams). We have found out that most business processes operating in the financial domain are not explicitly identified and documented, unlike in other application domains such as supply-chain-management (where standardization bodies are very active).

Through our implementation experiences, we have also evaluated the pros and cons of using an SOA. We have determined that the following forces will push towards the creation of a service:

- the functionalities offered by the service are supported by an underlying legacy system, so a service interface is a way to ''open up'' a legacy system without compromising its underlying intellectual property.
- the functionalities offered by the services are reused across several business processes associated with the application domain.

Against the creation of a service would be the following factors:

- the performance penalties incurred by SOA-enabling technologies.
- the extra development time associated with developing the service wrappers that can be significant when such a service in used by only one business process.

The results of this evaluation are conformant to another study that considers five other business processes in the same domain (Dabous 2005). This shows that although service-oriented technologies have the potential to bridge the business-technical divide, their effective usage still depends on the domain context taking into account the nature and number of its business processes, existing legacy systems and the quality factors that are of most concern to stakeholders.

References

Bolcer G, Kaiser G (1999) SWAP: leveraging the web to manage workflow. IEEE Internet Comput 3(1):55–88

Business Process Management Notation (BPMN). Business process management initiative (BPMI.org), May 2004. Version 1.0, available at http://www.bpmn.org/Documents

Curbera F, Khalaf R, Mukhi N, Tai S, Weerawarana S (2003) The next step in web services. Commun ACM 46(10):29–34

Dabous FT (2005) Pattern-based approach for the architectural design of e-business applications. Phd thesis, School of Information Systems, Technology and Management, The University of New South Wales, Australia

Hammer M, Champy J (1993) Reengineering the corporation: a manifesto for business revolution. HarperBusiness, New York

Harris L (2003) Trading and exchanges: market microstructure for practitioners. Oxford University Press, New York
van den Heuvel W, Papazoglou MP, Jeusfeld MA (1999) Configuring business objects from legacy systems. In: Proceedings of 11th International Conference CAiSE'99. Springer, Heidelberg, Germany, pp 41–56
van den Heuvel W, van Hillegersberg J, Papazoglou M (2002) A methodology to support web-services development using legacy systems. In: IFIP TC8 / WG8.1 Working Conference on Engineering Information Systems in the Internet Context, pp 81–103
Hollinsworth D (1995) The workflow reference model. Technical Report TC00-1003, Workflow Managment Coalition. http://www.wfmc.org/standards/docs/tc003v11.pdf
Homann U, Rill M, Wimmer A (2004) Flexible value structures in banking. Commun ACM 47(5):34–36
Huhns MN, Singh MP (2005) Service-oriented computing: key concepts and principles. Internet Comput IEEE 9(1):75–81
Ma KJ (2005) Web services: what's real and what's not? IT Prof 7(2):14–21
Mecella M, Batini C (2000) Cooperation of heterogeneous legacy information systems: a methodological framework. In: Proceedings of the 4th International Enterprise Distributed Object Computing Conference (EDOM'00). Makuhari, Japan, pp 216–225
Mecella M, Pernici B (2001) Designing wrapper components for e-services in integrating hetrogeneous systems. VLDB J 10(1):2–15
Medjahed B, Benatallah B, Bouguettaya A, Ngu AHH, Elmagarmid AK (2003) B2B interactions: issues and enabling technologies. VLDB J:59–85
Papazoglou M (2003) Service-oriented computing: concepts, characteristics and directions. In: Proceedings of the 4th International Conference on Web Information Systems engineering (WISE'03). Rome, Italy, pp 3–12
Pasley J (2005) How BPEL and SOA are changing web services development. Internet Comput IEEE 9(3):60–67
Quocirca Ltd. SOA (2005) Substance or hype—the IT professional verdict on service oriented architecture. Technical report, Quocirca Insight Report, 2005
Rabhi FA, Benatallah B (2002) A service-based architecture for capital markets systems. IEEE Netw 16(1):15–19
Workflow management facility, revised submission, July 1998. ftp://ftp.omg.org/pub/docs/bom/98-06-07.pdf
Yang J, Papazoglou M (2000) Interoperation support for electronic business. Commun ACM 43(6):39–47
Yu H, Rabhi FA, Dabous FT (2004) An exchange service for financial markets. In :Proceedings of the 6th International Conference on Enterprise Information Systems (ICEIS'04). Porto, Portugal, pp 403–410
Zimmermann O, Milinski S, Craes M, Oellermann F (2004) Second generation web services-oriented architecture in production in the finance industry. In: OOPSLA '04: companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications. ACM Press, New York, pp 283–289