Monitoring of events in SOA architecture for Real Time Financial Decision System

Dominik Ryżko* and Paweł Gławiński**
*Warsaw University of Technology, **Softman
*Nowowiejska 15/19, 00-665 Warsaw, **Gen. Okulickiego 5F, 05-500 Piaseczno
*d.ryzko@ii.pw.edu.pl, **pawel.glawinski@softman.pl

ABSTRACT

Most of the financial decision systems implemented to day are based on aggregation of transactions in data warehouses or other reporting environments and then generating appropriate reports for the analysts and managers. This makes the decision cycle, as counted from an individual transaction or event, very long. On the other hand modifying functional workflows to embed financial controlling steps imposes technological challenges related to change management and additional delays. The paper proposes an innovative approach to solving the above problems, by implementing a multi-agent system for monitoring of financial systems and providing real-time, flexible and efficient way for decisions. The research was conducted by Warsaw University of Technology and Softman as a part of a project financed by The National Center for Research and Development in Innovative Economy Programme (POIG) measure 1.4 project No: POIG.01.04.00-14-061/12.

KEYWORDS

Service-Oriented Architecture, Mutli-Agent Systems, distributed systems, real time analytics, controlling systems

1 INTRODUCTION

In this Chapter motivation for the research as well as previous approaches found in the literature are described. The business need which is solved by the proposed solution is also analysed.

1.1 Motivation

The Open Group defines formally SOA in the following way:

Service-Oriented Architecture (SOA) is an architectural style that supports service-orientation.
Service-orientation is a way of thinking in terms of services and service-based development and the outcomes of services.
A service:

- *Is a logical representation of a repeatable business activity that has a specified outcome (e.g., check customer credit, provide weather data, consolidate drilling reports)*
- *Is self-contained*
- *May be composed of other services*
- *Is a black box to consumers of the service*

The migration of architectures from monolith to Service Oriented Architectures (SOA) resulted in the change of the way data is exchanged in the systems. Rather than through shared databases, services communicate with the use of well defined APIs [3]. In the full SOA implementation a Enterprise Service Bus (ESB) plays the role of the component responsible for delivery of data according to the defined business processes. The following specific tasks can be handled by the ESB [4]:

- Providing connectivity
- Data transformation
- (Intelligent) routing
- Dealing with security
- Dealing with reliability
- Service management
- Monitoring and logging

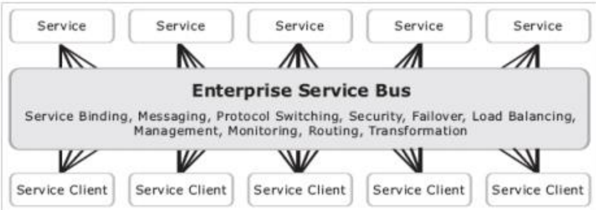

Figure 1

Performing advanced, real time analysis in such a setting is a challenging task. Traditionally this can be done by designing analytical or controlling steps in the business process definition and building specialized services to perform these tasks. However, this approach has several downsides. Firstly, any change in the analytics changes the process and therefore requires retesting to avoid introduction of bugs. Secondly, there is an overhead which affects the performance of the tasks which need to be performed. Finally, it limits the possibility to define analytical rules in a more generic, descriptive way, understandable to non-technical people.

Crucial for the ability to perform the analytical tasks, is proper interpretation of the data. It is not sufficient to capture all the messages exchanged in the system, but they have to be interpreted according to their context and content. Therefore, the issue of semantics is fundamental to this work and is further elaborated.

The motivation for this work is to analyse the possibilities to apply Multi-Agent System paradigms to build analytical functionalities as an additional layer of the system, which can monitor the business processes by plugging into the service bus and intercepting the data from the messages exchanged over it.

1.2 Previous Work

In [5] Aggregated Reporting pattern for SOA is described. The pattern is designed to overcome the distribution of data across services by creating a service that gathers immutable copies of data from multiple services for reporting purposes. Figure 2 shows the pattern of the architecture. The service works as follows. Firstly, the data is transferred from the source services into the Raw Data Store. Then it is processed by the transformation backend and put into the reporting store, usually containing joined and aggregated data. Finally, an SQL output endpoint is provided in order to plug in ad-hoc SQL and reporting tools.

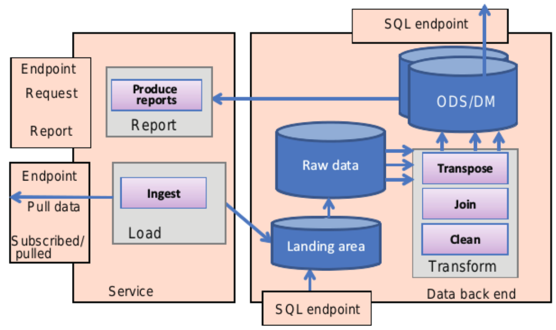

Figure 2

Four different ways of getting the data into the Aggregated Reporting are proposed:

- Actively calling other services - use of other services contracts to get new data
- Passively getting data from services - subscribing to batch data exports or events
- Service SQL push - services export a view of internal data
- ETL SQL push - as in the option above but with the involvement of external ETL tool

The advantage of using Aggregated Reporting include:

- Holding of immutable data with possible versioning if changes are received

Providing single SQL interface for reporting

- Possibility to highly optimize reporting efficiency

As for the disadvantages:

- High complexity of the solution
- Relatively large latency in data access
- Duplication of data

In [8] Schiefer and Seufert propose *sense & respond loops* which enables real-time analytics across corporate business processes. Five distinct stages of data processing are proposed in this approach:

- Sense - Which is the current state of the business environment?
- Interpret - What do the captured data indicate?
- Analyse - Which business opportunities and risks can arise?
- Decide - Which strategy is the best to improve the current situation of the organisation?
- Respond - Who has to implement the decision?

As opposed to this work, the sense & response functionalities are implemented as additional services, which communicate with each other over the ESB. The work leaves out the technological details of the implementation, so it is difficult to asses how it can be applied to systems provided by various vendors.

The use of Multi-Agent systems for monitoring in SOA has been analysed before. Monitoring of performance and SLA was analyzed in [2]. Agents are grouped in clusters assigned to particular Web Services and act as proxies between WS and the client. The approach allows redirection of requests in case of problems, but creates significant overhead and, by becoming part of the flow, agents can generate problems by themselves.

In [1] Intelligent agents located across the system perform asynchronous, distributed measurements separately for each selected service or process. Compliance with required performance measures or SLAs can be defined and appropriate warnings are generated and reported. The approach uses Aspect-Oriented Programming to plug into the application server and overlook the flow of processes in the system.

This work extends the above research by plugging into the service bus via documented APIs, which gives a reliable and sustainable model for integration. Also the monitoring goes much further than in the case of performance monitoring. Here, the actual data has to be accessed, interpreted and processed.

2 Business Need

The risk is a basis of running a business and a condition of market mechanisms working properly. In terms of the business operating on the financial market the risk should be managed. The lack of proper mechanisms defined as risk management processes can lead businesses to substantial losses or even a bankruptcy. Spectacular examples of the lack of the efficient supervision of the risk [6] are connected with using advanced financial instruments. As a consequence, it led to insolvency such businesses like Barrings Bank (the oldest English commercial bank) or LCTM Fund. Less spectacular examples resulting from operational errors or frauds are often concealed by institutions.

The exposure of the business to the risk on the financial market depends on many factors which can be defined by the business profile. Based on research done recently [6], it can be concluded that the operational risk is more and more significant in the banking activities. It is visibly less significant than the credit risk, but its greater importance in the risk portfolio is connected with both the improvement in the quality of credit risk management methods and changes in business model. The distribution of the main risks in the banking activities based on recent research looks as follows:

- credit risk - 60%
- operational risk - 25%
- market risk - 15%

Operational risk will be defined (according to [7]) as the loss risk incurred from internal processes functioning improperly, people, systems and external events. The control of this risk is directly connected with business model used by this business.

Let's assume the process of granting financial instruments is conducted on a 24/7 basis with the instant time of decision and immediate disbursement and the business process itself is highly or fully automated. Then, it requires the control and the risk management to be conducted in the automated way and it concerns both the transaction and the appropriately defined portfolio of these transactions.

The crucial elements connected with the risk management of the above example of granting the credit instrument are integrated directly with a fundamental business process supported by the IT system. The list of crucial elements includes:

- Verification of the customer in debtor databases, credit information bureaus, as well as blocked identification documents bases. These verification is performed in the context of managing the operational risk.
- Scoring performed by the IT system for the purpose of the credit risk management.

An automatic model of granting a credit, the use of services provided by different external suppliers in the business process (e.g. based on SOA architecture) and most importantly, the use of Internet channels of communication with the customer make this particular business model more vulnerable to frauds and software errors.

If we implement the above described business process based on the chosen engine of the business processes and we integrate all steps of the process based on SOA architecture, we should supplement this model with control mechanisms over the implementation of particular business processes instances as well as sets of these instances. Control mechanisms should enable changes of particular procedures to improve them and create new forms of control without changes in the basic business process. Control mechanisms should be based on information provided dynamically as part of the communication in the business process.

If we assume that complete information in the process consists of e-documents and the process messages then the control mechanism implementation model can be based on the analysis of these forms of information.

The suggested solution is based on capturing and analysing information of the business process in the context of this process instance and the defined time window.

3 SYSTEM ARCHITECTURE

3.1 Data Interception

Possible technologies to capture the data communication over ESB include:

- Use fo APIs
- Proxy
- Aspect Oriented Programming

In the project feasibility of applying these techniques for various environments was tested. Namely Oracle, IBM and JBoss service buss were analysed. The conclusions show that there is no single mechanism, which can be used in each scenario.

APIs provide the most controlled environment and are supported by the supplier of the technology, but are usually limited with respect to the functionality and the scope of data. There no simple way to fix this.

Proxy allows the most generic approach, in which we capture all communication and process it, extracting only the data which is needed. The downside, is that we are prone to the changes in protocols and data formats, something which can be transparent if we use APIs. A single message can be also stripped of some context, which is provided to the service by another mechanism, thus semantics can become ambiguous.

Aspect Oriented Programming (AOP) is probably the most advanced of the methods. It uses Load-Time Weaving to execute a code at join points (eg. method execution) [7]. It allows to introduce some computation at the locations where it was not prepared by the vendor, so e.g.

can reach the data not provided through the API. The major challenge is finding the right join points and also to avoid introduction of new bugs to the system.
Since the analysis showed no single good way of extracting the data, we propose to create dedicated plugins for particular products, which can use any of the technologies described above. The plugins should be able only to extract the data and transform it into a common format. This allows us to abstract from the particular technology and build further common components of the system.
E.g. for Oracle OSB - Oracle Web Service Manager (OWSM) policies have been chosen as the best mechanism. OWSM provides capabilities to build, enforce, run and monitor Web service policies, such as security, reliable messaging, MTOM, and addressing policies. It is possible to declaratively attach policies to particular WSs. They also provide wide possibilities for defining parameters (constants, global and service level).
For JBoss, Message Store or Pipeline interceptors can be used. The first one seems to be more efficient, but is more memory heavy since entire messages are serialized. In the case of Pipeline Interceptors, it is possible to select specific fields to be captured and therefore to save on the data size.
IBM Integration Bus provides two native mechanisms for message monitoring and capturing. It is also possible to modify the message flow and redirect a message to a different application.
It is also possible to perform data capture and analysis in case of a loosely-coupled service architecture without the ESB. In such a case services communicate directly with each other when needed. In this case there is no ready mechanism to do so. Therefore, using a proxy to capture the messages is possible. The most important property of all the approach to produce dedicated plugins using the mechanisms described above, is that as an output of each of them we can define a common XML format with specified data fields, which will be processed in the further analytics. Therefore a common administration mechanism has been created, which allows specification of the message fields and metadata, which should be captured by the plugins.

3.2. Overall architecture

The overall architecture of the data extraction system is depicted in Figure below.

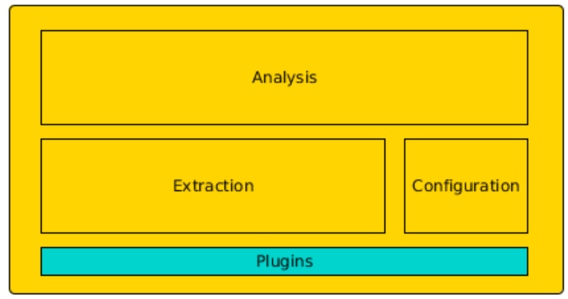

Figure 3

Data can be captured from various systems by the plugins according to the predefined configuration. Then it is physically extracted and enriched with metadata, such as time, subsystem etc. Such prepared data is transferred to the analytical modules.

3.3 System performance

Performance requirements for the system were estimated based on the analysis of business processes on the financial services market. The maximum number of financial instruments of the credit type granted by leading companies on the financial market doesn't exceed one thousand. Conclusion of such a number of agreements depending on the availability and attraction of the product can be associated with the need for substantial amount of operational processes. We have assumed that 10 000 processes of granting the credit instrument will be initiated. The process consists of around 25 steps connected to web services. A scope of business and operating information is on average 50 elements. For the daily after-sales service we should adopt the same information requirements as for sale processes. To sum up, we create one

ISBN: 978-1-941968-38-3©2016 SDIWC

million of XML messages daily, each of the average content of 50 information elements with simple data (like number, date, content). The system was design to handle such load and to scale up for potentially more complex scenarios.

4. CONCLUSIONS

The paper describes a system for monitoring of events in SOA architecture for Real Time Financial Decision System. Means for intercepting of the data have been analyzed and a plugin layer was introduced to manage this task. The captured data is enriched with metadata and passed to a multi-agent system for further analysis.

The overall approach is flexible and robust, imposes no computational overhead and its technology dependence is reduced to minimum. It allows separation of controlling steps from the core business processes, which results in minimizing the re-testing need and therefore simplifies the change management.

REFERENCES

1. D.Ryżko and A.Ihnatowicz. Multi-agent Approach to Monitoring of Systems in SOA Architecture. In *New Challenges for Intelligent Information and Database Systems* (pp. 309-318). Springer Berlin Heidelberg, 2011
2. A.Miede, N.Repp, J.Eckert, R.Steinmetz, Cooperation Mechanisms for Minitoring Agents in Service-oriented Architectures. *Tagungsband der 9. internationalen Tagung Wirtschaftsinformatik* 2009, vol. 1, p. 749-758, sterreichische Computer Gesellschaft, February 2009
3. M. P. Papazoglou. Service-oriented computing: Concepts, characteristics and directions, Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on , pages 312. IEEE, 2003.
4. N. M. Josuttis. SOA in practice: the art of distributed system design. O'Reilly Media, Inc., 2007.
5. A. Rotem-Gal-Oz, E. Bruno, and U. Dahan. Co, SOA Patterns. Manning Publications 2012
6. D. Gątarek, R. Maksymiuk, M. Krysiak, Ł. Witkowski, Nowoczesne metody zarządzania ryzykiem finansowym, WIG-Press, Warszawa 2001
7. K. Jajuga, *Zarządzanie ryzykiem*, PWN Warszawa 2009
8. J. Schiefer and A. Seufert, Management and Controlling of Time-Sensitive Business Processes with Sense & Respond. In *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)* (Vol. 1, pp. 77-82). IEEE, (2005, November).

ISBN: 978-1-941968-38-3©2016 SDIWC