SOA INTEROPERABILITY, A CASE STUDY

Balázs Simon, Zoltán László, Balázs Goldschmidt
Department of Control Engineering and Information Technology
Budapest University of Technology and Economics, Hungary
balazssimon@gmail.com, laszlo@iit.bme.hu, balage@iit.bme.hu

ABSTRACT

In many places and in different fields of applications continuous experiments are being made to integrate complex systems on a SOA basis. Assuming that the market will remain diversified on the long term, the key to the widespread distribution of SOA is the interoperability among products of different vendors. Authors introduce a demo framework aiming to compare different SOA products and to assess their co-operation. Authors also describe their experiences showing that without the appropriate settings and non-functional requirements seamless interoperability can not be achieved. Authors reveal the causes, and emphasize the necessity for further standardization that might enable the out-of-box interoperability and portability of SOA tools.

KEYWORDS

SOA, interoperability, web-services, WS-* standards.

1. INTRODUCTION

By now all the major vendors of the IT market (e.g. IBM, Microsoft, Sun, Oracle) have come up with their own SOA products, promising a solution to the common problems of distributed systems spreading across multiple organizations (for example e-commerce, e-government). In many places and in different fields of applications (from health care to financials) continuous experiments are being made to integrate complex systems on a SOA basis using one of the tools mentioned above. Assuming that the market will remain diversified on the long term the key to the widespread distribution of SOA is the interoperability among products of different vendors. A few years ago some analysis has been made to predict the future and adoption of web-services [Leavitt 2004, Nezhad et al 2006]. At that time only a few WS-* protocols (WS-Security, WS-Reliability or WS-ReliableMessaging) were supported by the vendors, but by now most of the WS-* standards have been implemented. Within the scope of a research program performed by the Innovation and Knowledge Centre of Information Technology, Budapest University of Technology and Economics, a demo framework was installed and deployed aiming to compare different SOA products and to assess their co-operation based on WS-* standards.

In the present article authors intend to introduce this framework, and share relevant experiences with the reader. Firstly the main characteristics of the applied SOA tools will be briefly described followed by the task to be solved using the framework and the chosen architecture. Afterwards some components of the implemented framework and the connections between them will be introduced in more details – with an emphasis on security, alongside with the business process and its implementation. Finally, the main problems solved will be summarized, and conclusions will be drawn.

2. ARCHITECTURE

The aim of the authors was to create a framework which illustrates the interoperability between the SOA products listed in Table 1 through a real life example. This example implements a regular public administration process by coupling different government systems, in which a person can register a company. The process runs between seven parties (shown in Figure 1) whose names are similar to those in real life however the architectures and technologies applied might not be exactly the same.

SOA product	**OpenESB**	**.NET 3.0**	**Oracle SOA Suite 10g, 11g**	**IBM WebSphere**
Designer	Netbeans 6	Visual Studio 2008	Oracle JDeveloper 10g, 11g	RAD 7 (based on Eclipse)
Application Server	Glassfish	IIS	Oracle AS	WebSphere
Web-services API	JAX-WS	WCF	JAX-WS	JAX-WS
Support of most WS-* standards	yes	yes	in the upcoming version 11g	yes
BPEL support	yes	WF through a converter	yes	yes

Abbreviations: RAD - Rational Application Developer; IIS - Internet Information Services; WCF - Windows Communication Foundation; WF - Windows Workflow Foundation

Table 1. The examined SOA products.

The Customer is the person initiating the registration process. The Clerk is the one who approves the application. They both access the system of public administration through a unified web page, called the Customer Gate. The Customer Gate accepts different forms and starts the corresponding public administration processes. Both the Customer and the Clerk digitally sign their submitted forms. The Registrar of Companies stores the forms of the enterprises and provides a web-service to register a new one. The Tax Authority stores taxation information of the taxpayers, and provides a query to retrieve this information. The Ministry of Justice stores the criminal records and provides access to these. The Process Coordinator is responsible for executing business processes, providing management functions and unifying the web-services offered by the separate parties (Customer Gate, Registrar of Companies, Tax Authority and Ministry of Justice) to each other.

Figure 2 illustrates the technical architecture of the created framework. The Customer and the Clerk access the Customer Gate through a browser (Internet Explorer, Firefox) over an HTTPS connection. The web pages of the Customer Gate are written in ASP.NET 2.0 and the provided web-services are implemented by WCF. Every office communicates directly with the Process Coordinator, which is also responsible for forwarding the messages towards the appropriate destination. Therefore, the Process Coordinator acts as a proxy, as well. The BPEL process is run by an Oracle application server in the Process Coordinator, while its state is stored in an Oracle database. When the process is calling a web-service, it is performed through one of the proxy web-services hosted by the Process Coordinator.

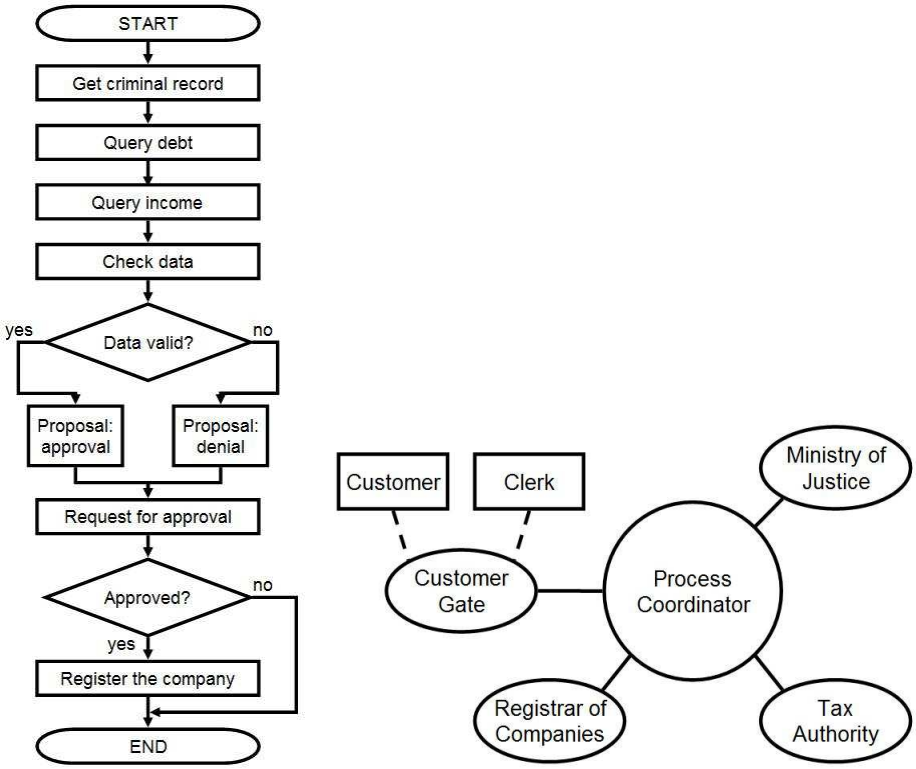

Figure 1. The registration process and the functional architecture

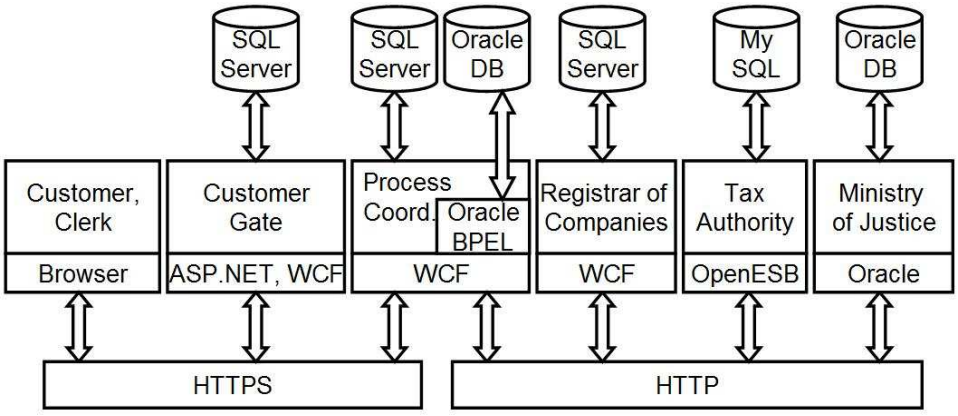

Figure2. Technical architecture

The Customer Gate runs on a Windows Server 2003 operating system, while the Tax Authority uses SuSE Linux 10.3 and the other parties use Windows XP Professional.

From the architecture description it can be inferred, that the framework connects heterogeneous elements both in terms of operating systems and SOA solutions in a way that they can co-operate successfully.

In the following paragraphs we introduce the parties in more detail.

2.1 Customer, Clerk

Both the Customer and the Clerk must authenticate themselves with a PKI (Public Key Infrastructure) certificate when entering the Customer Gate, thus eliminating the need for the less secure username/password login. Both of them access a web-interface tailored to their roles. The Customer can initiate a registration process; the Clerk can approve an already initiated process. To perform these activities both persons have to upload a digitally signed form. The digital signature is created by a smart-card based application. After submitting the documents both the Customer and the Clerk receive a digitally signed acknowledgment by e-mail from the Customer Gate.

2.2 Customer Gate

Since the Customer Gate was written in ASP.NET 2.0 the database storing user data was created accordingly. Different certificates are used for user authentication and for digital signatures. Therefore the thumbprints of these certificates are stored for every user as well. The web page is hosted by an IIS server which is set to be only accessible through HTTPS and also to demand client-side certificates. Upon downloading a page the common name (CN) of the certificate provided in the browser is mapped into a username and the thumbprint of the certificate is compared to the one stored for authentication. If all data is valid the client can enter the web page and access functions according to his ASP.NET roles.

The Customer Gate provides two web-services for the Process Coordinator. One of these services starts the approval as a sub-process. This means that Clerks with the appropriate assignment will see a new approval request in their work list. When a Clerk completes the request this sub-process finishes. The other web-service is called in the final step of the main BPEL process to close the registration process. The web-services can only be called trough an HTTPS channel and only by providing the certificate of the Process Coordinator.

2.3 Registrar of Companies

The Registrar of Companies provides a single web-service for registering a company. Messages are encrypted and signed according to the WS-SecureConversation protocol.

For further security reasons the Registrar of Companies also employs an STS (Security Token Service). WCF does not contain an STS implementation therefore it was created based on internet sources [Gudgin 2006]. The web-service can only be called providing a valid SAML token issued by the STS. The declarative method of [Bustamante 2007] is applied to facilitate granting user access. For each accepted client certificate the database of the STS contains the tokens to be issued. The tokens required for calling the web-service mentioned above are listed only for the certificate of the Process Coordinator, thus the service can only be called by the Process Coordinator and by no external parties.

To increase the reliability of message delivery the WS-ReliableMessaging protocol is also applied.

2.4 Tax Authority

The taxpayers' accumulated debt and income for a given financial year can be retrieved from a web-service provided by the Tax Authority.

Again, messages are encrypted and signed according to the WS-SecureConversation protocol. Making this work was one of the biggest challenges of this project. The custom format of the Java keystore and bugs in the continuously changing OpenESB versions caused several problems which took considerable time to resolve. Appropriately setting all the parameters of the algorithms used between WCF and OpenESB also proved to be a difficult task. Even though the certificates, algorithms, and key lengths are the same on both sides, due to the different configuration solutions the settings were not that obvious. Moreover, co-operation also requires the system clocks of the two parties to be in sync within a fixed limit.

2.5 Ministry of Justice

The Ministry of Justice provides a single web-service for retrieving the criminal record of an individual.

Configuring encrypted data transfer at message level for this web-service is still in progress. While several other tools support the WS-Policy standard to provide parameter synchronization for the various WS-* standards, the lack of WS-Policy support in Oracle 10g results in defining parameter values in an empirical way.

The web-service provided by this office is also implemented in IBM WebSphere and is capable of replacing the Oracle version.

2.6 Process Coordinator

The Process Coordinator itself provides two web-services. One of these initiates the registration process (see the flow-chart in Figure 1), while the other is waiting for the Clerk's acknowledgement. The latter is called by the Customer Gate upon the Clerk approving or refusing a request.

In addition, the Process Coordinator acts as a proxy wrapping the services of the other parties. The Process Coordinator always uses the appropriate protocol towards a given party, but shows a unified interface to the outside. It unifies the various data formats, logs every call, solves addressing problems, and stores the status of the running BPEL processes. Due to these features the description of BPEL processes becomes significantly easier both in terms of addressing and status handling.

The implemented registration process runs in an Oracle Application Server. The BPEL description follows the standard, the extensions (e.g. for addressing) provided by Oracle are not utilized, thus the description is expected to be portable. This assumption will be tested in the near future.

3. SECURITY

During the creation of the framework particular attention was paid to the correct handling of security features. Authors' aim was to use a wide variety of security protocols in the system. Table 2 summarizes the security factors in each connection.

For Customers (CST) and Clerks (CLRK) the Customer Gate (CG) can only be accessed through HTTPS and a valid certificate is required to log in. The web-interface available for the user is determined by the user's ASP.NET role. The web-services between the Customer Gate and the Process Coordinator (PC) can only be called through HTTPS, meaning that messages are encrypted at the transport level. Between the Process Coordinator and the Registrar of Companies (RC), and the Process Coordinator and the Tax Authority (TA) the WS-SecureConversation protocol provides SOAP level encryption and integrity. Moreover, there is a simple STS at the Registrar of Companies. This service issues time stamped and digitally signed SAML tokens ensuring that the web-services provided by the RC can only be accessed by the authorized clients (PC in our case).

Connection	Identification	Authentication	Authorization	Integrity	Encryption	Non-repudiation
CST-CG	certificate	SSL	ASP.NET role	SSL	SSL	digital signature
CLRK-CG	certificate	SSL	ASP.NET role	SSL	SSL	digital signature
CG-PC	certificate	SSL	Automatic	SSL	SSL	-
PC-RC	certificate	WS-Security	WS-Trust (STS)	WS-Security	WS-Security	WS-Security
PC-TA	certificate	WS-Security	Automatic	WS-Security	WS-Security	WS-Security
PC-JUST	-	-	-	-	-	-

Table 2. Security features between the communicating parties

4. EXPERIENCES AND RESULTS

The first version of the framework is completed and it was introduced fully functioning to a limited, professional publicity. Preparations for a demonstration to a wider audience are currently in progress.

The major challenge during the creation of the system was the appropriate configuration of the computers, operating systems, application servers and SOA-tools used by the parties. According to the usual routine, at first handbook and sample settings were applied, which in turn usually proved to be completely useless in the configurations used herein. Mailing lists and very often intuition provided significant help during discovering the activities (e.g. changing settings, reinstalling components in a different order) required to create a working version. In the following paragraphs a few problems encountered and the solutions thereof will be briefly discussed.

IIS is only able to run the HTTPS protocol on a Windows Server operating system. Special care must be taken when setting user rights, since IIS makes operations on behalf of three users, none of them being the one developing the web-site.

Most of the certificates used are OpenSSL generated self-signing certificates. These must be installed in the corresponding certificate library of Windows, and the three IIS users need special access to the private keys. This is solved by the assistance program *WinHttpCertCfg*[1]. Java has a special file format for storing certificates but the keytool provided in the JDK is not capable of importing PKCS12 certificates with private keys into this keystore. Again, an assistance tool (*pkcs12import*[2]) is required for this task. Due to the export restrictions of the USA Java programs by default are not allowed to use long encryption keys unless some jar files in the JDK are replaced by the Unlimited Strength Jurisdiction Policy Files.

The detailed logs of the WCF – which provide significant help when developing web-services – cannot be accessed while the IIS is running, thus the server must be stopped before retrieving the log files.

Even though ASP.NET 2.0 has built-in support for username/password login, certificate-based authentication requires a special solution.

BPEL processes match the asynchronous request and response messages based on correlation: however in Oracle JDeveloper providing the correct BPEL description itself is not sufficient, in the project containing the process a special flag must be set in order to use correlations instead of WS-Addressing. An unexpected error occurs when the otherwise standard BPEL process is deployed with version number 1.0 because in this case the application server is unable to match the SOAP actions with the operations. Even the graphical editor has its limitations: XSD types using inheritance are not handled properly, thus upon using these types the XML source of the BPEL must be edited.

5. CONCLUSION

Most SOA products provide a uniform but vendor-specific interface to interconnect different components (e.g. applications, databases etc.). This interface hides the implementation details of the underlying layers therefore the application does not have to bother with the format and routing of the calls. Each vendor has developed its own protocol to realize the communication layer making it impossible to interconnect different SOA products directly (see Figure 3). Today the simplest way to achieve this interoperability is using WS-* standards. The specifications of the WS-* protocols are defined by messages instead of interfaces. Therefore, the modules (denoted by WS-* implementation in Figure 3) responsible for producing and consuming these messages are different in every SOA product not only regarding their implementation but in most cases their API, too. This diversity is reflected in the code of the applications built on top of these modules as well as in the format of the configuration files. The latter means that the modules have to be parameterized differently in order to produce messages having the appropriate format and content, and to consume the received

---

[1] WinHttpCertCfg tool. *http://www.microsoft.com/Downloads/details.aspx?familyid%=C42E27AC-3409-40E9-8667-C748E422833F\&displaylang=en.*

[2] Pkcs12Import tool. *https://xwss.dev.java.net/servlets/ProjectDocument-List?f%olderID=6645\&expand-Folder=6645\&folderID=6645.*

messages as intended. This dependency of the parameters can be eased by using WS-Policy standards (assuming the module supports these) which make it possible to set them in a uniform and automatic way. However, some elements in the configuration (e.g. security certificates) may require unique solutions. These can differ even between JAX-WS implementations although JAX-WS intends to unify the web-services interface for Java applications.

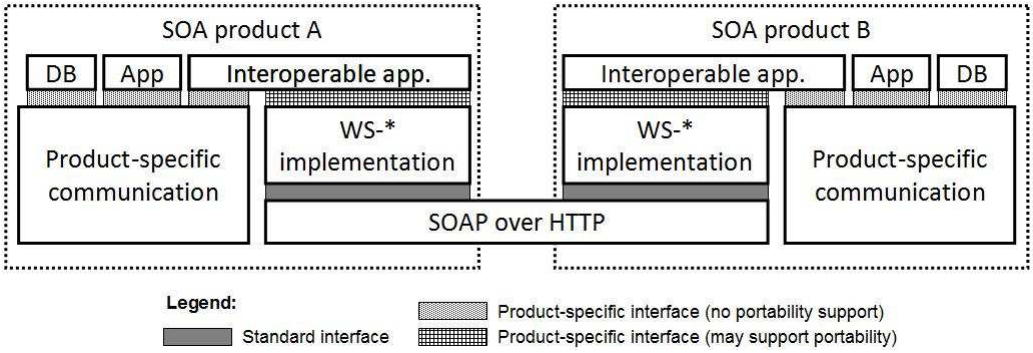

Figure 3. A model of connecting different SOA products.

The most commonly used transport layers are HTTP and HTTPS since these are the only protocols having standard WSDL bindings. Authors have made successful experiments between WCF and Axis2 using JMS [Monson-Haefel & Chapell 2001] as transport but the lack of standard bindings and policies resulted in empirical configurations. Connecting different JMS providers and the fact that vendors may not always use the same message format can cause further complications. (For example Oracle can only process ObjectMessages while IBM utilizes TextMessage and BytesMessage formats for web-services.) It can be therefore stated that JMS or any other message queue system is unsuitable for interoperability and the use of WS-* standards through HTTP is recommended for this purpose.

Authors have built a heterogeneous system from SOA products of different vendors to investigate if co-operation between the components can be achieved. According to their experiences - and based on the SOA products used - it can be stated that with the appropriate settings and assistance programs; and by occasionally loosening security requirements interoperability is feasible. This reveals that further standardizations are needed for seamless interoperability and portability. Without that higher level integration among companies, government organizations and inter-governmental organizations – choosing their SOA solutions from a wide range of products by different vendors – might be at risk. Authors are currently working on the integration of the products of other vendors, following the continuously changing versions, and improving the framework to meet higher security requirements.

Authors have made some performance analysis to compare the WS-* implementation modules of different products in terms of efficiency. These measurements were performed on simple web-services over HTTP transport without any WS-* protocols. In the future authors intend to make additional measurements to determine how enabling WS-* protocols affects these results.

ACKNOWLEDGEMENT

The research-development work behind the current article was carried out by the BME Innovation and Knowledge Centre of Information Technology within the scope of the "Pázmány Péter Program" (RET-06/2005.) supported by the National Office for Research and Technology (NKTH).

REFERENCES

Michele Leroux Bustamante, 2007. Building a Claims-Based Security Model in WCF. http://www.theserverside.net/tt/articles/showarticle.tss%?id=ClaimsBasedSecurityModel2.

Martin Gudgin, 2006. *STS implementation in WCF*. http://www.pluralsight.com/blogs/mgudgin/archive/2006/06%/19/28503.aspx.

Matjaz B. Juric, Benny Mathew, and Poornachandra Sarang, 2006. *Business Process Execution Language for Web Services.* Packt Publishing, second edition edition.

N. Leavitt, 2004. Are web services finally ready to deliver? *IEEE Computer*, 37(11).

Richard Monson-Haefel and David A. Chappell, 2001. *Java Message Service.* O'Reilly, first edition edition.

Craig McMurtry, Marc Mercuri, Nigel Watling, and Matt Winkler, 2007. *Windows Communication Foundation Unleashed.* Sams.

H.R.M. Nezhad, B. Benatallah, F. Casati, and F. Toumani, 2006. Web services interoperability specifications. *IEEE Computer*, 39(5).