How to Make It Faster and at Lower Cost? B2B Integration with Semantic Web Services

Silvestre Losada, Dariusz Kłeczek, Richard Benjamins, Jesus Contreras
Intelligent Software Components S.A.
Pedro de Valdivia 10, Madrid, Spain
{slosada, darek, rbenjamins, jcontreras}@isoco.com

Oscar Corcho
The University of Manchester
Kilburn Building, Oxford Road, Manchester, UK
Oscar.Corcho@manchester.ac.uk

Jose Luis Bas, Sergio Bellido
Bankinter, Fundacion de la Innovacion
Paseo Castellana 29, Madrid, Spain
{jlbas, sbellidog}@bankinter.es

Abstract

Web services and service oriented architectures present a new approach to application integration. While it is reasonable inside an enterprise, it has certain deficiencies when applied in a B2B environment. This deficiencies apply to the discovery, invocation and composition phases, which require considerable manual effort. In the paper, we show on example of a mortgage simulator how these deficiencies can be overcome by applying semantic web services. The application is compatible with the Web Services Modelling Ontology and makes use of an execution environment automating the processes of discovery, composition and invocation of semantic web services, enabling faster and cheaper B2B application integration.

1. Introduction

It is common knowledge that application integration is one of the most costly and difficult tasks in software engineering. Several approaches, such as web services and recent service-oriented architectures (SOAs) have been designed to tackle the problem of integration. They follow the principle of loose coupling, which aims at reducing the assumptions that applications which are being integrated make about each other. The business logic of enterprise applications is encapsulated in self-contained services, which can be exposed for example in intranets.

The value of service orientation lies in future proofing of enterprise applications. Once a certain service is implemented and exposed, it can be used by multiple applications, including those that are still to be developed. While this approach offers the desired flexibility and enables maintainable enterprise architectures (for example based on the Enterprise Service Bus concept), it has certain deficiencies when applied in a B2B environment. Firstly, services are not easy to find. They are scattered over different repositories and lack semantic annotations of their interfaces and of their functioning. Secondly, their use normally requires a large amount of manual effort, as the descriptions (WSDL or WSRF) are intended for developers instead of machines. Thirdly, the granularity of services requires that they be composed in order to achieve a non-trivial functionality. In the remainder of this paper we will show how these deficiencies can be overcome with semantic web services (SWS).

In this paper we address the situation of the Spanish financial market. As other competitive business environments, it requires from enterprises that they be innovative and efficient while functioning in collaborative networks. This is the case of Spanish financial market addressed in this paper. Therefore, new services, which can require integrating applications across enterprises, should be developed at low cost and have short time to market. We will show how this can be achieved with SWS on an example of a mortgage simulator developed jointly by iSOCO and Bankinter.

The rest of the paper is organized as follows. In the next section we describe the motivation for implementing a semantically enabled SOA. Section 3 explains the SWS architecture utilized in the mortgage simulator. The following section presents our case study. Section 5 discusses relevant work in the field, and section 6 concludes the paper.

2. Motivation

Bankinter is currently offering a free service which presents data about mortgages from a set of banks in Spain. This data is obtained manually by people, by browsing the web pages (when available) or by calling each bank to gather the information.

The use of SWS technology can optimize this manual process by allowing searching in available registries, so that new web services that have been deployed in the market can be discovered. Besides, these registries provide information about how to invoke the selected Web services so as to include them into other, more complex, services. Hence, the data gathering process is improved since the relevant information can be obtained more easily by means of executing those services.

Consequently, more services (product price comparators, broker information, deposits, etc.) can be offered by banks due to their low cost, since less human interaction is required to discover and invoke new available SWS once the application is launched.

Some of the advantages of SWS over state-of-the-art web service technology are the following:

When facing UDDI with a large number of exported web services, the lookup (discovery) becomes a serious problem. There is no standard for service goal or capabilities in current WSDL which prevents automatic service discovery. For example, a bank offering a mortgage information web service only for fixed interest rates and with a maximum period of 20 years will not be able (or will have many difficulties) to publish such constraints in UDDI registries, so that the external parties looking for services that are compliant with those characteristics will not be able to know in advance whether the service is providing this information according to those constraints or not.

When the discovered services have been defined according to a set of heterogeneous models, discrepancies may occur in the execution of those services. This is summarized as follows by Gartner Research (February 28, 2002): "Lack of technologies and products to dynamically mediate discrepancies in business semantics will limit the adoption of advanced web services for large public communities whose participants have disparate business processes."

Thus the possibilities of better discovery and mediation are the main advantages of SWS technology over current web service technology in the context of the described financial application.

3. Semantic Web Services Infrastructure

The mortgage simulator developed at Bankinter is based on the Web Services Modeling Ontology (WSMO) [9] framework, which is one of the conceptual models lying behind SWS. WSMO framework is based on four main elements, namely ontologies, web services, goals and mediators. This is the foundation for formal languages and execution environments, which enable realizing the functionality of SWS.

Ontologies provide formal representation of a certain domain, which consists of its conceptual taxonomy, relations between concepts, instance data and axioms describing logical expressions applicable to other elements of the ontology. All other WSMO elements use ontologies as vocabularies. With respect to web services, WSMO provides a conceptual model for describing their functionality and behavior. This model consists of web service capability expressed in terms of preconditions, assumptions, postconditions and effects specifying the functionality, and choreography and orchestration interfaces, which define web service behavior in terms of communication patterns and internal composition. Goals are specified as a counterpart of web services on the consumer side. They represent users' desires and have similar structure to goals, which enables functional discovery and automatic invocation, composition and execution of web services, possible due to formal logic-based representation of requested and offered functionality. In heterogeneous environments, interoperability is enabled by the definition of mediators resolving mismatches between different representations.

The formal model underlying WSMO eases the tasks of developers concerned with discovering, composing and executing web services. These tasks are delegated to an execution environment compliant with the framework. In the following case study we will describe the execution environment that we created for the case study with implemented discovery and invocation modules.

4. Case Study

4.1. Conceptual Architecture

The mortgage simulator (also referred to as comparator) helps clients of financial institutions in finding and choosing appropriate mortgage. Once a client has found a property she would like to buy, she has to make following calculations:

- What type of mortgage can I deal with?
- What amount of my income should I periodically set aside for the payment of the loan to acquire a property?

Our application helps the customer to answer these questions by allowing simulations. It accesses other financial institutions to obtain data on their offers and presents them in a standardized way to the user, helping her to choose between different offers.

Figure 1 illustrates the structure of the comparator, the services offered by the providers and the end user interfaces. In this use case, the application is the central point of interaction between the customer and other service providers. The comparator aggregates web services of different financial entities that provide mortgage services. A customer uses the comparator service as the entry point for his requests. The response is produced by invoking and aggregating web services offered by several financial entities (service providers). Figure 1 depicts this process.

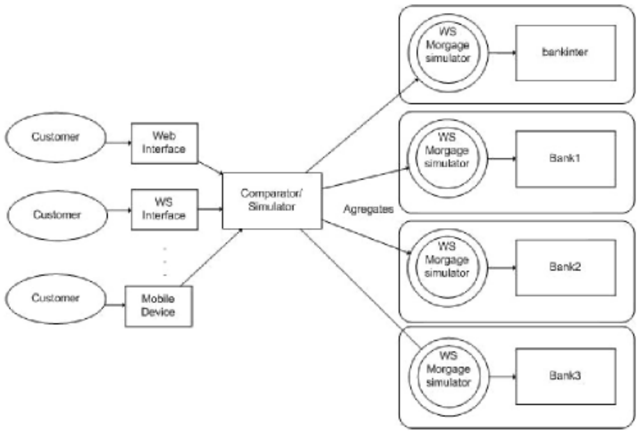

Figure 1. Conceptual architecture

Each time a client wants to know the mortgage market proposals, the application provides her with the actual simulations made online in each bank's WS-based simulator. The results are given back to the user in a human-readable interface so that she is able to compare them.

4.2. System architecture

The basic architecture is shown in Figure 2. It shows components that usually apear in SWS execution environments, in order to provide the functionality of a semantically enabled SOA. The essential functionalities of our architecture are as follows:

1. Providing a user interface for customer interaction: web interface, web service interface, mobile devices interface, etc.
2. Providing the appropriate Web Services for mortgage simulation.
3. Discovering suitable Web Services for a certain user request.
4. Invoking external SWS.

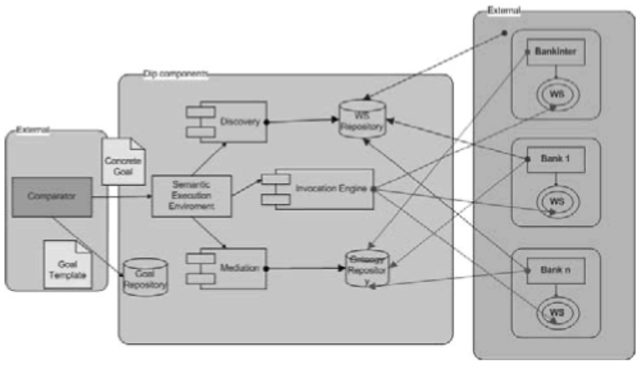

Figure 2. System architecture

5. Providing an execution environment for SWS with control functions, error handling, and support of optional user interaction.
6. Dealing properly with heterogeneous resources, thus providing the suitable mediation facilities (Banks ontologies are expected to be heterogeneous).
7. Registering the providers' semantic descriptions.

In the remainder of this section we will explain our approach with respect to functional requirements of the application.

Discovery The semantic web services used in our scenario have been annotated with f-Logic [5]. We have decided to use our own component based on f-Logic instead of Web Services Modelling Language (WSML) because there was no WSML reasoner capable of supporting our component at the time of its development, as well as no discovery component inside WSMX.

Invocation Semantic web service descriptions are grounded to WSDL descriptions. This allows for automatic invocation of the relevant web services, without manual development effort.

Mediation Mediation is required in the case when one component is not capable of interpreting the content of a message sent by another component. The basic functionality of a data mediation module is to transform messages from source format to a target one, which could require both syntactic and semantic transformation.

Semantic Repository The semantic repository (SEMR) is an ontology server which allows the storage, retrieval, and querying of ontologies and other data (semantic web service descriptions, instance data, etc.)

4.3. Discovery

Our service discovery component uses f-Logic to describe capabilities, goals and to make queries using FLORA-2 reasoner [10] for matchmaking capabilities and goals. In goals, we model the precondition (the state of the information space before executing a web service) and the postcondition (the state of the information space that is desired). We express this by a fact in f-Logic (using FLORA-2 syntax). WSMO defines additionally assumptions and effects, which are designed to model the state of the world outside of the information space of the system, and are not applicable to our case study. Because goals are constructed similarly to web service description, we only provide an example description of a web service, which follows in Listing 1.

Web services, like goals, are modeled with precondition and postcondition (see example in Listing 1). The capability describes in the precondition the state of the information space of the service before its execution and in the postcondition describes the state of the information space of the service after its execution. The service description specifies the constraints of the service, e.g. this service offers simulation for mortgages with interest rate fixed, total term smaller than 320000 euros and the initial quota smaller than 450 euros. The mortgage for which the simulation is performed needs to contract a home insurance. Also, this mortgage has associated an extra opening commission that must be paid when the user asks for a mortgage. This description includes extra information in the interface part to execute the service discovery according to the information specified by the user.

Listing 1

```
capital_if_WSlh_euribor_bankinter:
webService [ interface -> wsInterface[
capability -> capitalWSCapability,
wsdlFile -> "https://aia.ebr.com/
wsDM/MortageService.asmx?WSDL",
operationToInvoke -> "dataInput"] ].
capital_if_h_euribor_bankinter:
capability.
capital_if_h_euribor_bankinter
[precondition]:-
_Mortgg:'MortgageLoan'
[term -> Term,
 intitalQuota-> Quota,
 interestRateType -> Interest2 ],
 Term< 320000,
 Quota< 450,
 Interest2:'ProductRateAplicationFixed'
[interestRateValue -> Rate],
Rate > 3.01.
capital_if_h_euribor_bankinter
[postcondition] :-
_Mortgggg:'MortgageLoan'[
loanCapital -> Any,
openingCommission -> OpCommission,
lifeInsurance -> 'false',
homeInsurance -> 'true' ],
(OpCommission> 1; OpCommission = 1).
```

In the f-Logic approach for discovery we are checking if the capability entails the goal (capability $\sqsubseteq$ goal). Current limitations with respect to available reasoners led to the current modeling, which defines the goal consisting of precondition and postcondition as a fact (which may not be fully specified) and the capability of the web service (also consisting of postcondition and precondition) as a rule. Semantic discovery of web services means discovery of abstract services represented by formal service capabilities, which are part of the semantic description of the web service published in a registry by the service providers. The comparator creates a goal description to describe their service requirements. To create these descriptions a predefined goal template is used by the application to create a concrete goal description based on user data.

A similar approach to discovery has been described in [4].

5. Relevant work

There are several relevant initiatives, which can be classified as semantic web services frameworks and execution environments.

Apart from WSMO, there exist two competitive approaches with respect to semantic web services frameworks. OWL-S [7] follows similar objectives as WSMO, but defines a slightly different conceptual model [6]. Another approach has been developed inside METEOR-S project [8]. It seeks to augment the existing standards, such as WSDL and BPEL with a semantic layer, instead of introducing new standards. There is a need for the competing specifications to merge or be interoperable. Development of independent benchmarks for solutions based on these approaches could help in achieving a consensus.

There exist two implemented execution environments compliant with WSMO: Web Services Execution Environment (WSMX) [3] and IRS-III [2]. In our application, we use the architecture provided by WSMX, but with functional components developed inside iSOCO, as at the time of developing the application there did not exist any components which could satisfy our requirements. There is however ongoing work on implementing choreography, orchestration, discovery and invocation components inside WSMX project [1] and we may be using them in the future if they can be adapted to our needs.

6. Conclusions

We have presented the complete design process of a B2B application with SWS as the underlying technology. Based on this experience, we can formulate lessons learned with regard to choosing the application field, implementing the technology and analyzing benefits of the SWS approach.

The e-Banking application field described in this paper can be characterized by three features, which in our opinion rationalize the additional effort required by a solution based on SWS. Firstly, the environment is distributed. There are many actors involved in the process of offering a mortgage to the final customer and many information sources influencing the decision taking process.This feature ensures that the potential of semantically-enhanced discovery and composition can be utilized.

Secondly, the market is dynamic. The circumstances change, the products evolve and the partners come and go. This feature of the environment ensures that a one-time investment will pay off in the long term. The loose coupling enforced by separating goals and web service descriptions enables SWS-based applications to continue working even if some of the web services used in application stops working provided that an alternative web service can be dynamically discovered and invoked.

Thirdly, our application field is profitable, but not mission critical. These two characteristics should be a general guideline for introducing new technologies, as the initial costs of adopting new technologies are influenced by the need of training the personnel, which can pay off in the longer term. New technologies bear however risks, which can be minimized when they are deployed in an iterative process, starting with low-risk areas.

With respect to the characteristics of markets described above, we have shown how SWS can bring added value in B2B application integration scenarios. In our case, the benefits centered on the automatic discovery and invocation capabilities provided by SWS. We expect that with the ongoing work on these functionalities, as well as composition and mediation, these benefits will be further boosted, so that SWS will become a common artifact in enterprise IT landscapes.

Acknowledgements This material is based upon work supported by the EU funding under the DIP project (FP6 -507483).

References

[1] C. Bussler, D. Fensel, H. Lausen, and E. Oren, editors. *Proceedings of the WIW 2004 Workshop on WSMO Implementations*, Sept. 2004.

[2] J. Domingue, S. Galizia, and L. Cabral. The choreography model for irs-iii. In *HICSS*. IEEE Computer Society, 2006.

[3] A. Haller, E. Cimpian, A. Mocan, E. Oren, and C. Bussler. Wsmx - a semantic service-oriented architecture. In *ICWS*, pages 321–328. IEEE Computer Society, 2005.

[4] M. Kifer, R. Lara, A. Polleres, C. Zhao, U. Keller, H. Lausen, and D. Fensel. A logical framework for web service discovery. In *ISWC 2004 Workshop on Semantic Web Services: Preparing to Meet the World of Business Applications*, volume 119, Hiroshima, Japan, 2004. CEUR Workshop Proceedings.

[5] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *J. ACM*, 42(4):741–843, 1995.

[6] R. Lara, D. Roman, A. Polleres, and D. Fensel. A conceptual comparison of wsmo and owl-s. In L.-J. Zhang, editor, *ECOWS*, volume 3250 of *Lecture Notes in Computer Science*, pages 254–269. Springer, 2004.

[7] D. L. Martin, M. Paolucci, S. A. McIlraith, M. H. Burstein, D. V. McDermott, D. L. McGuinness, B. Parsia, T. R. Payne, M. Sabou, M. Solanki, N. Srinivasan, and K. P. Sycara. Bringing semantics to web services: The owl-s approach. In J. Cardoso and A. P. Sheth, editors, *SWSWPC*, volume 3387 of *Lecture Notes in Computer Science*, pages 26–42. Springer, 2004.

[8] A. A. Patil, S. A. Oundhakar, A. P. Sheth, and K. Verma. Meteor-s web service annotation framework. In S. I. Feldman, M. Uretsky, M. Najork, and C. E. Wills, editors, *WWW*, pages 553–562. ACM, 2004.

[9] D. Roman, U. Keller, H. Lausen, R. L. Jos de Bruijn, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel. Web service modeling ontology. *Applied Ontology*, 1(1):77–106, 2005.

[10] G. Yang, M. Kifer, and C. Zhao. Flora-2: A rule-based knowledge representation and inference infrastructure for the semantic web. In R. Meersman, Z. Tari, and D. C. Schmidt, editors, *CoopIS/DOA/ODBASE*, volume 2888 of *Lecture Notes in Computer Science*, pages 671–688. Springer, 2003.